Group no:-4

1.Pratibha Khandagale 17

2.Aishwarya Chatake 18

3.Utkarsha Deshmukh 19

Manual For talend Workshop

Topic:-Logs and Errors

1.How to Write an assertive statement to evaluate the status of job execution ?

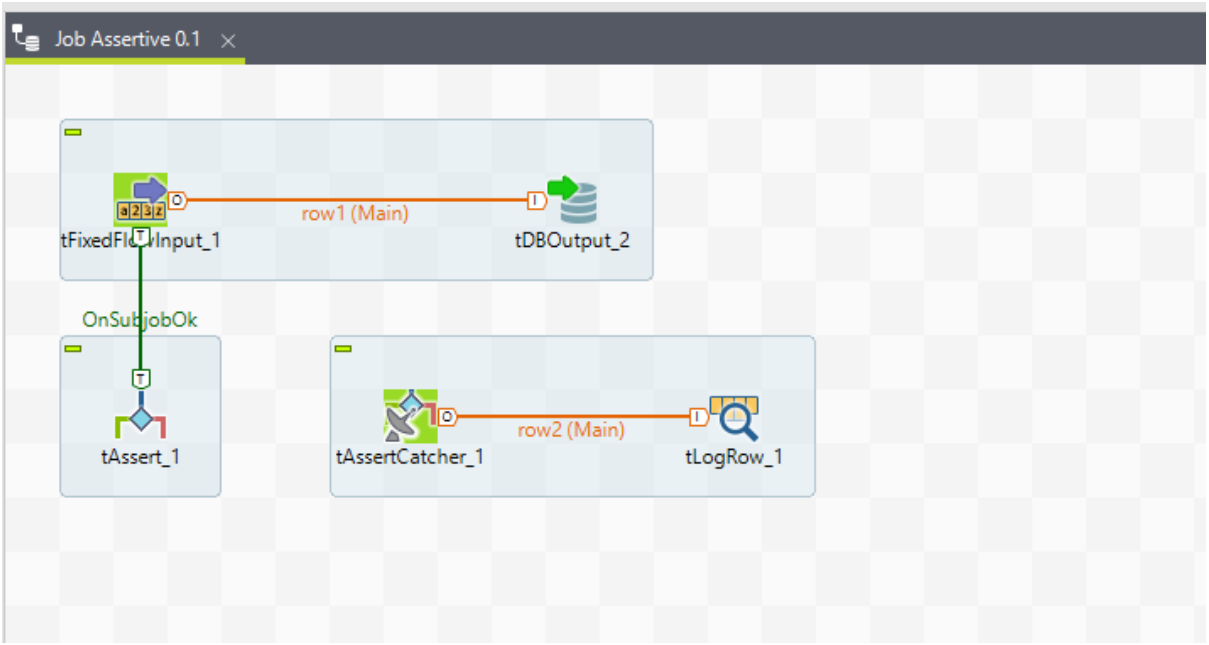
example :- checking status of job execution regarding orders

This scenario allows you to insert the orders information into a database table and to evaluate the orders status (every day once scheduled to run) by using **tAssert** to compare the orders against a fixed number and **tAssertCatcher** to indicate the results. In this case, **Ok** is returned if the number of orders is greater than 20 and **Failed** is returned if the number of orders is less than 20.

In practice, this Job can be scheduled to run every day for the daily orders report and **tFixedFlowInput** as well as **tLogRow** are replaced by input and output components in the **Database**/**File** families.

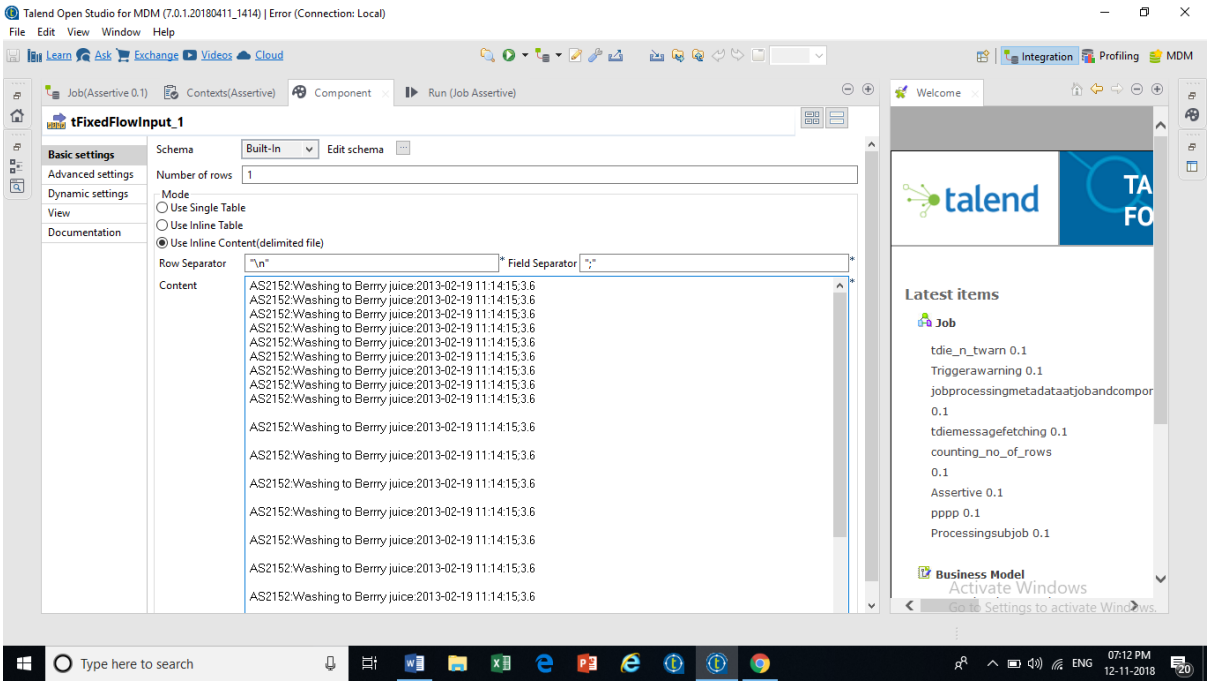
Procedure

1. Drop **tFixedFlowInput**, **tMysqlOutput**, **tAssert**, **tAssertCatcher**, and **tLogRow** onto the workspace.
2. Rename **tFixedFlowInput** as **orders**, **tAssert** as **orders >=20**, **tAssertCatcher** as **catch comparison result** and **tLogRow** as **ok or failed**.
3. Link **tFixedFlowInput** to **tMysqlOutput** using a **Row** > **Main** connection.
4. Link **tFixedFlowInput** to **tAssert** using the **Trigger** > **On Subjob OK** connection.
5. Link **tAssertCatcher** to **tLogRow** using a **Row** > **Main** connection.



Configuring the components

Double-click **tFixedFlowInput** to open its **Basic settings** view



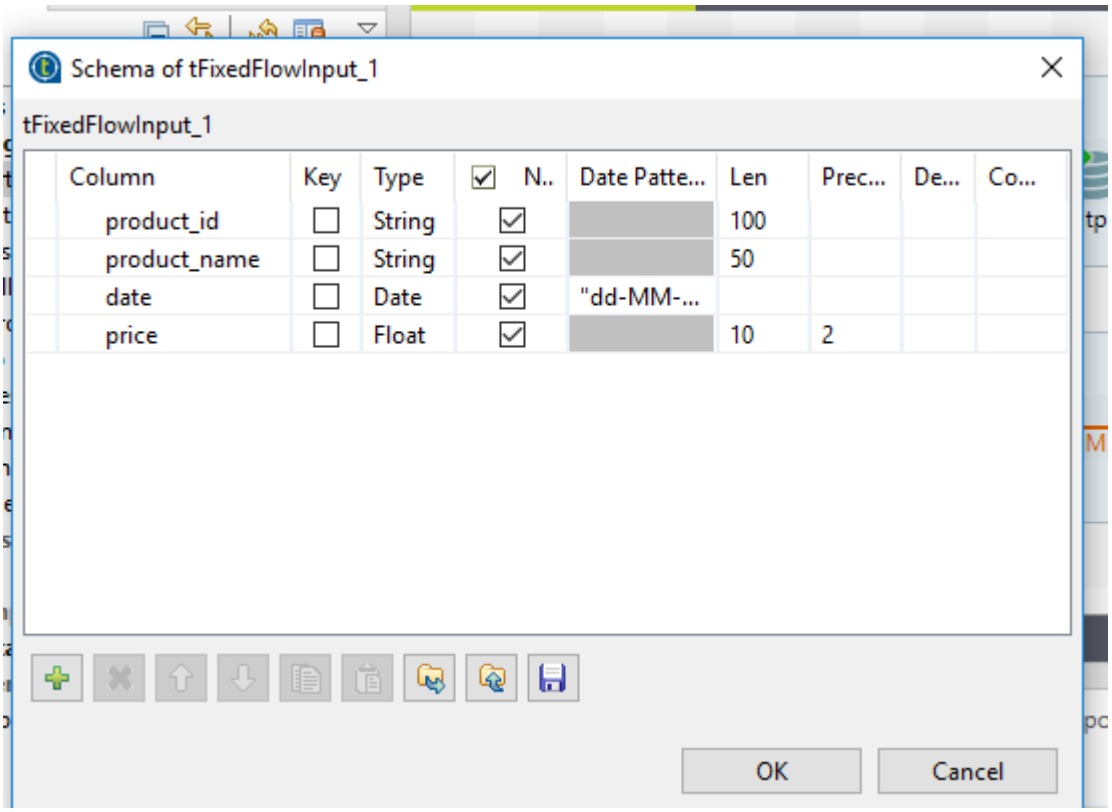
2. Select **Use Inline Content (delimited file)** in the **Mode** area.

In the **Content** field, enter the data to write to the Mysql database, for example:

```
AS2152;Washingto Berry Juice;2013-02-19 11:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 13:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 14:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washingto Berry Juice;2013-02-19 12:14:15;3.6
```

Note that the orders listed are just for illustration of how **tAssert** functions and the number here is less than 20.

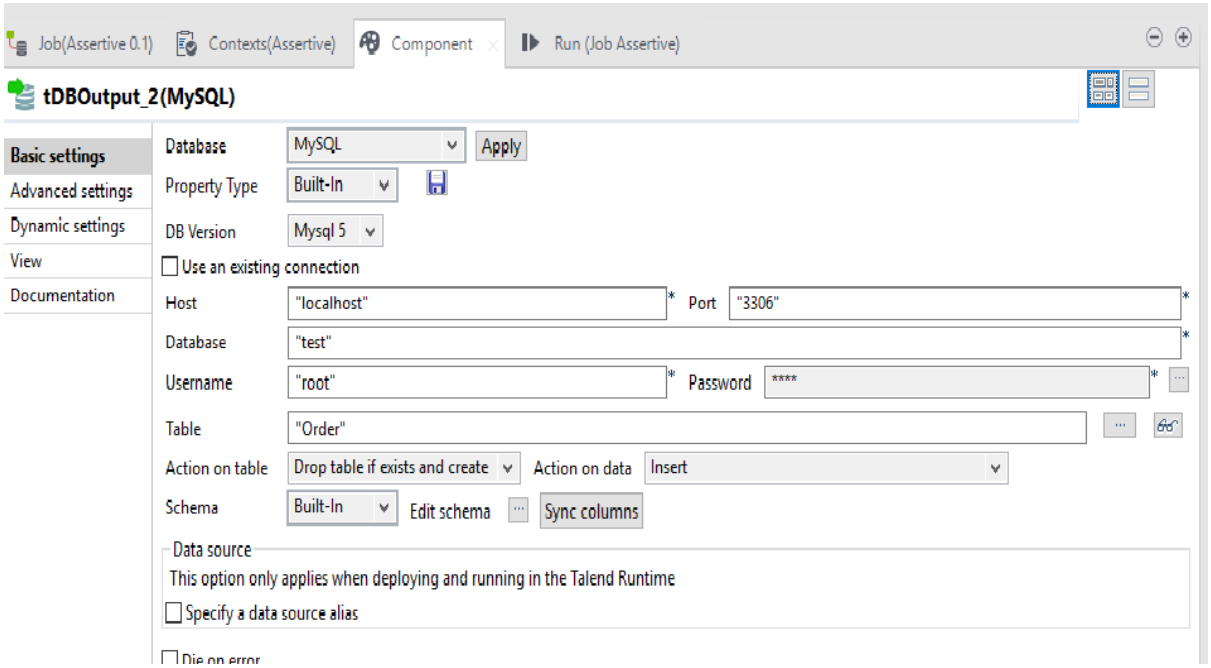
2. Click the **Edit schema** button to open the schema editor.



1. Click the **[+]** button to add four columns,
   namely *product_id*, *product_name*, *date* and *price*, of the **String**, **Date**, **Float** types respectively.

2.Click **OK** to validate the setup and close the editor.

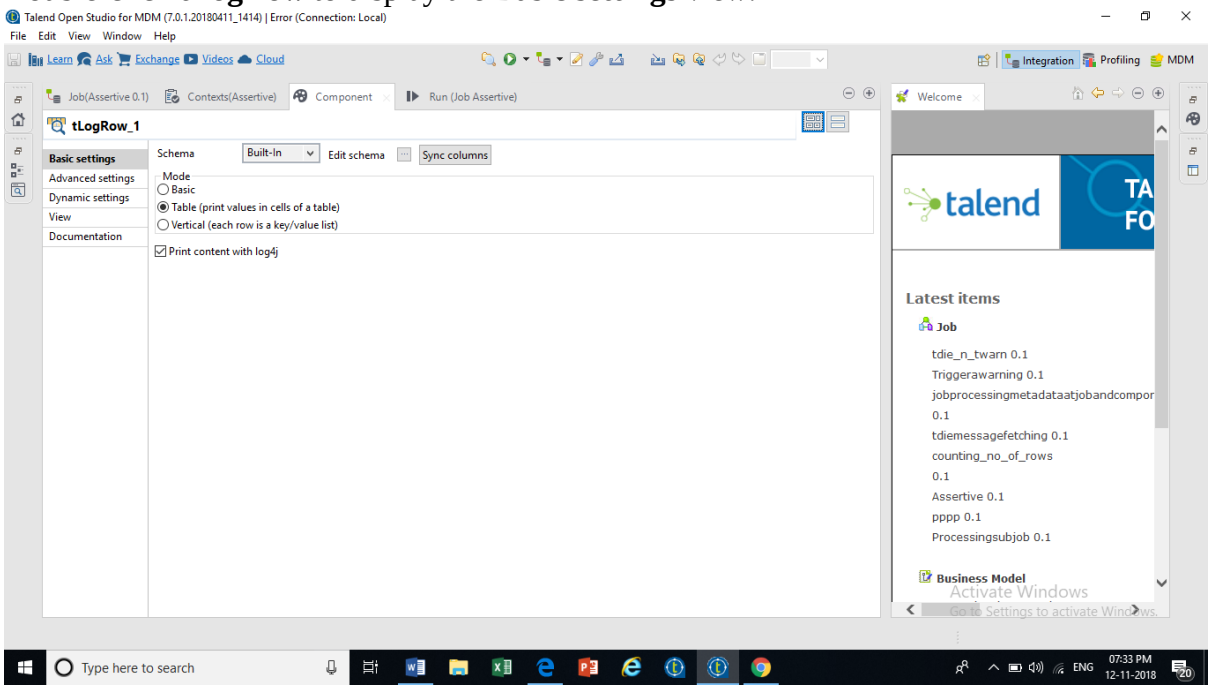
3.Double-click **tMysqlOutput** to display the **Basic settings** view.

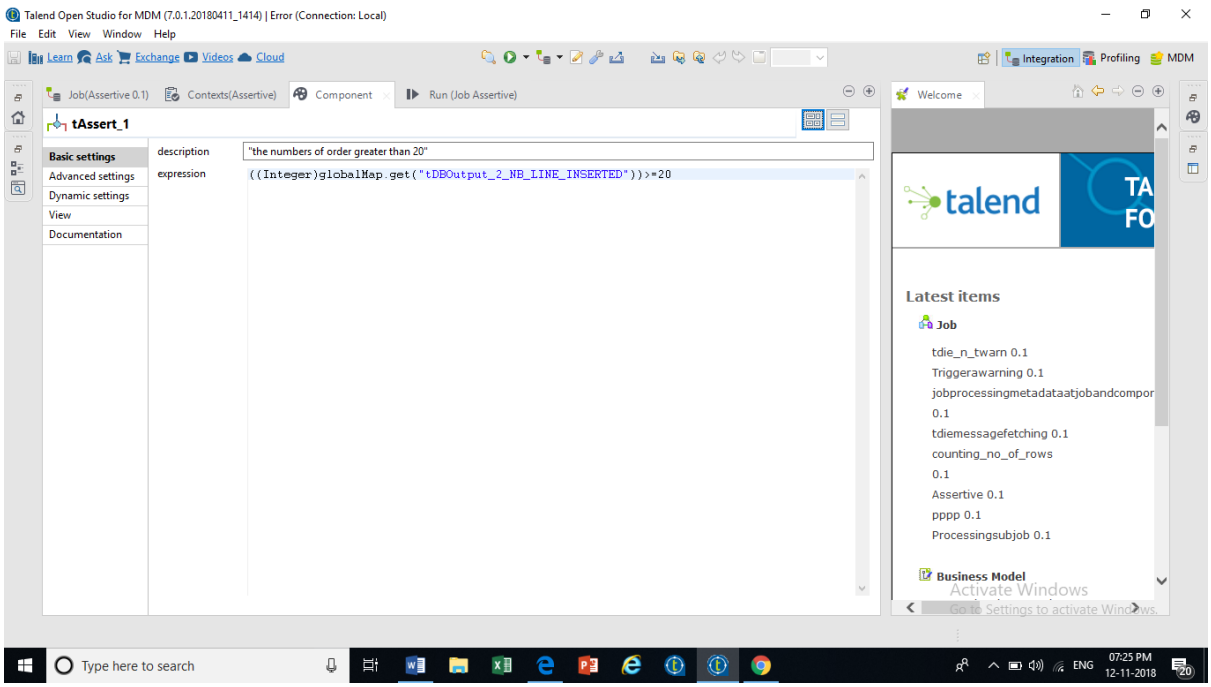


1. In the **Host**, **Port**, **Database**, **Username** and **Password** fields, enter the connection details and the authentication credentials.
2. In the **Table** field, enter the name of the table, for example *order*.
3. In the **Action on table** list, select the option **Drop table if exists and create**.
4. In the **Action on data** list, select the option **Insert**.
5. In the **Host**, **Port**, **Database**, **Username** and **Password** fields, enter the connection details and the authentication credentials.
6. In the **Table** field, enter the name of the table, for example *order*.
7. In the **Action on table** list, select the option **Drop table if exists and create**.
8. In the **Action on data** list, select the option **Insert**.
9. Double-click **tAssert** to display the **Basic settings** view.
10. In the **description** field, enter the descriptive information for the purpose of **tAssert** in this case.
11. In the **expression** field, enter the expression allowing you to compare the data to a fixed number:

```
((Integer)globalMap.get("tMysqlOutput_1_NB_LINE_INSERTED"))>=20
```

12. Double-click **tLogRow** to display the **Basic settings** view.

13. 



13.In the **Mode** area, select **Table (print values in cells of a table)** for a better display.

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.

2.How to Catch assertive logs generated by tassert

The **tAssertCatcher** component is part of the Logs & Errors family of components, and listens for *non-blocking* messages from tAssert and other *die* commands.

T assert catcher catches the comparsion from tassert and display it in tLogRow Shown as above in this way it can catch assertive logs generated by tassert.

3..Measure processing time of one or more subjob

This scenario is a subjob that does the following in a sequence:

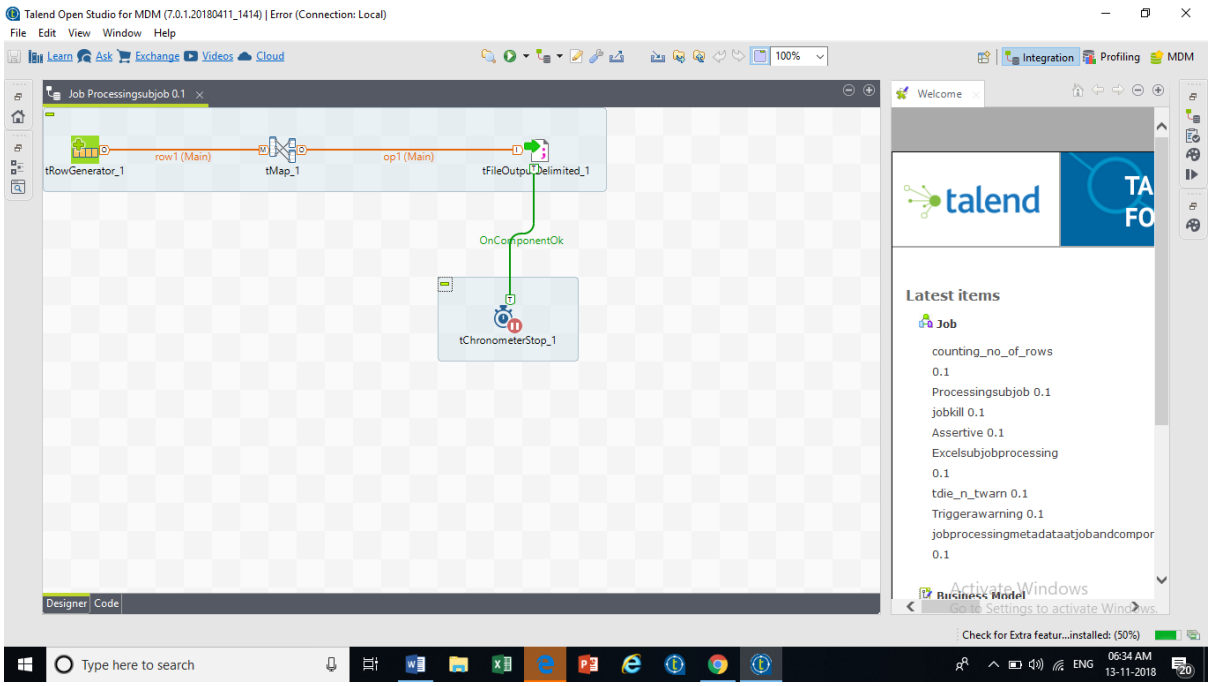
- generates 1000 000 rows of first and last names,
- gathers first names with their corresponding last names,
- stores the output data in a delimited file,
- measures the duration of the subjob as a whole,
- measures the duration of the name replacement operation,
- displays the gathered information about the processing time on the **Run** log console.

To measure the processing time of the subjob:

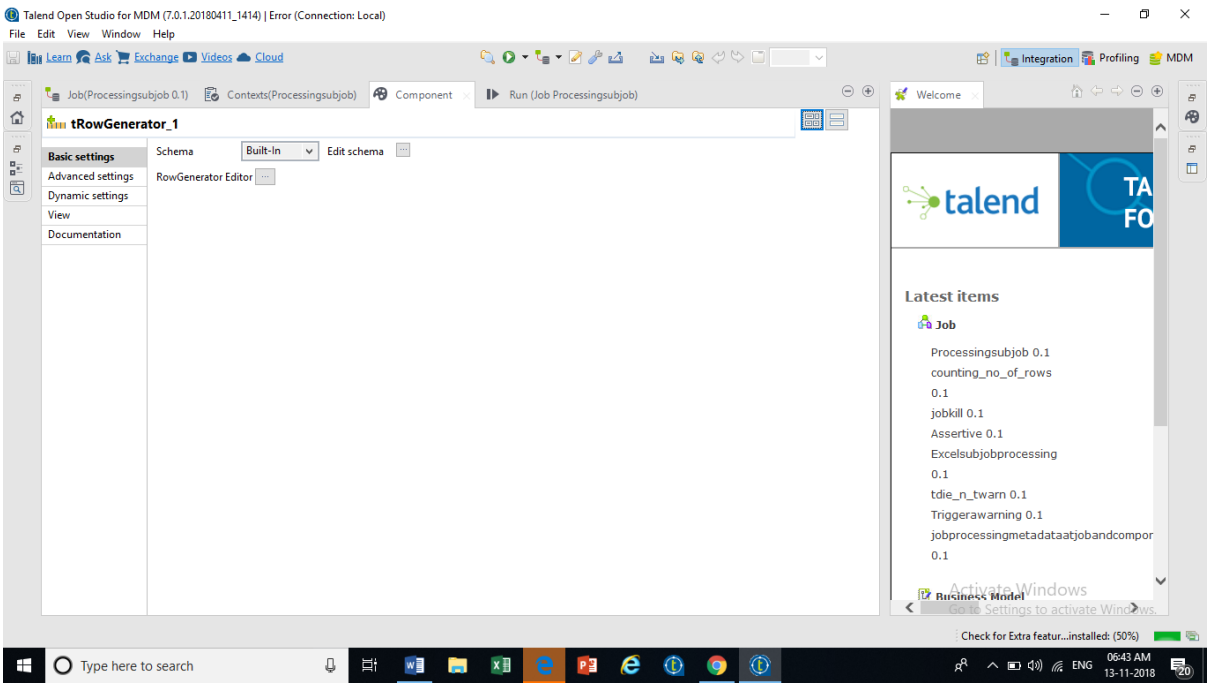
- Drop the following components from the **Palette** onto the design workspace: **tRowGenerator**, **tMap**, **tFileOutputDelimited**, and **tChronometerStop**.
- Connect the first three components using **Main Row** links.

Note

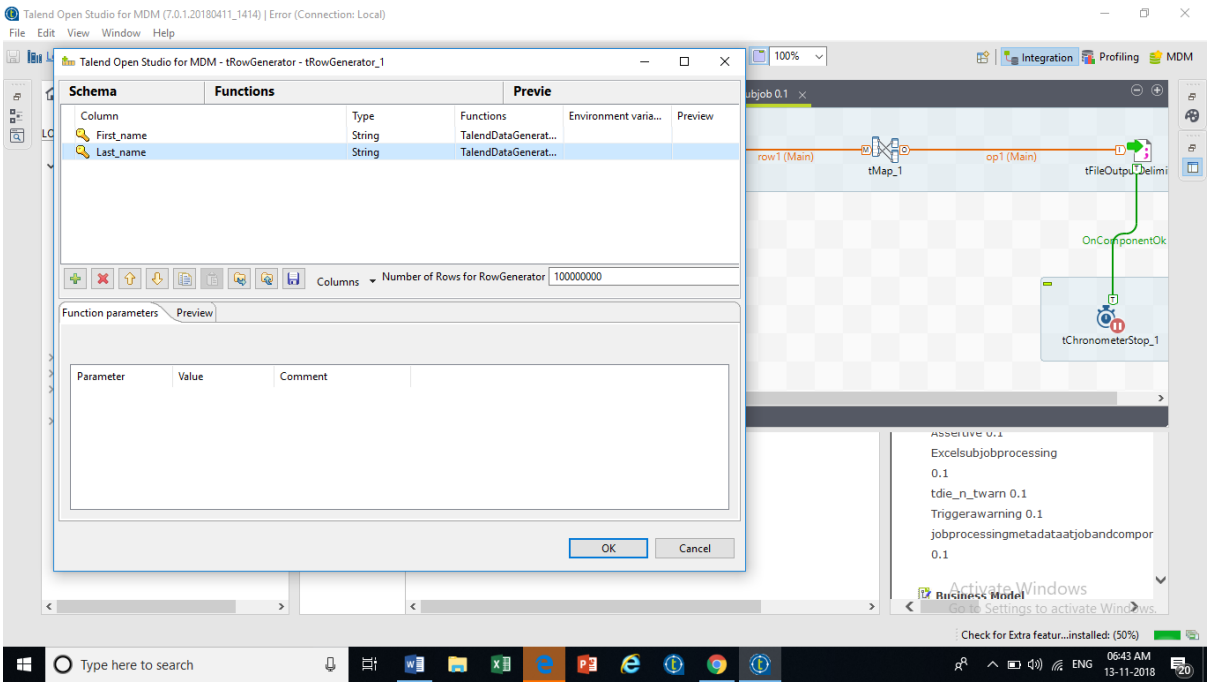
When connecting **tMap** to **tFileOutputDelimited**, you will be prompted to name the output table. The name used in this example is "new_order".



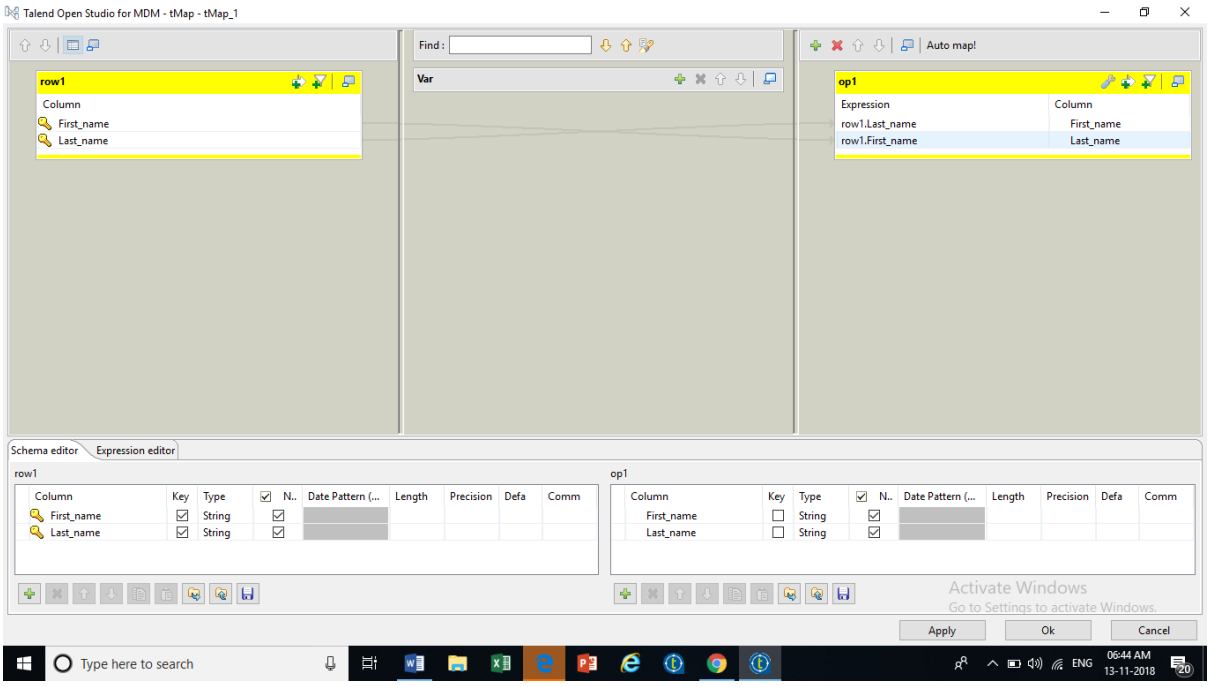
- Connect **tFileOutputDelimited** to **tChronometerStop** using an **OnComponentOk** link.
- Select **tRowGenerator** and click the **Component** tab to display the component view.
- In the **component** view, click **Basic settings**. The **Component** tab opens on the **Basic settings** view by default.



- Click **Edit schema** to define the schema of the **tRowGenerator**. For this Job, the schema is composed of two columns: *First_Name* and*Last_Name*, so click twice the **[+]** button to add two columns and rename them.
- Click the **RowGenerator Editor** three-dot button to open the editor and define the data to be generated.

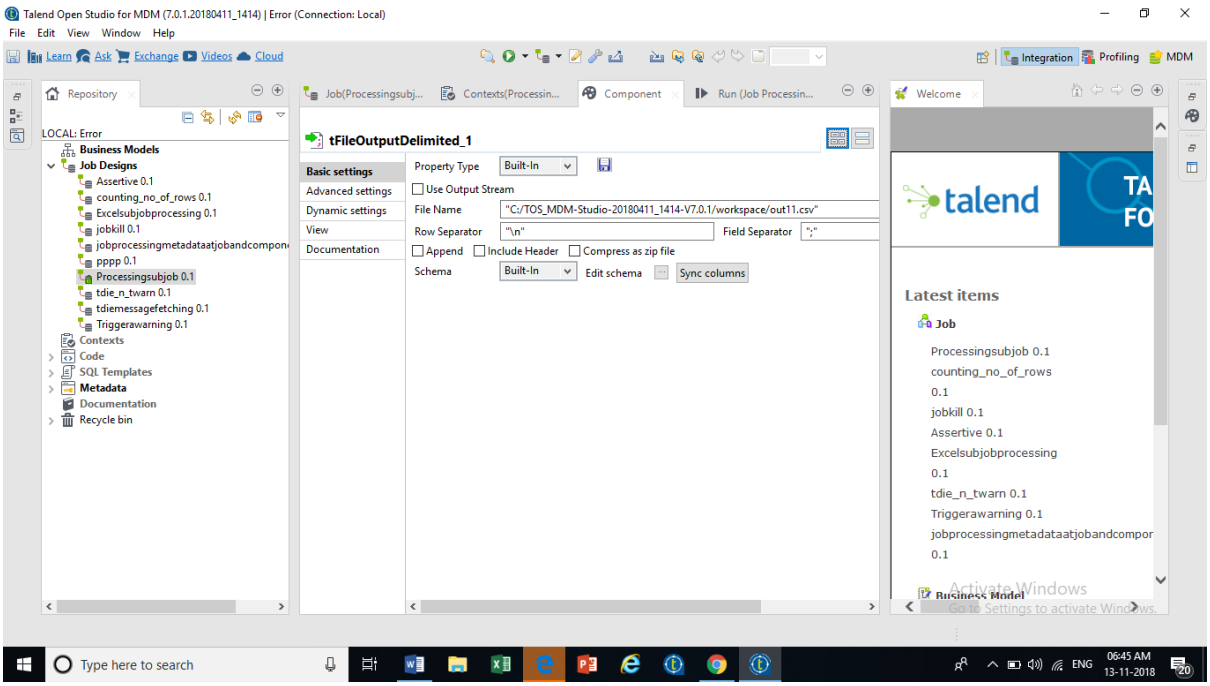


- In the **RowGenerator Editor**, specify the number of rows to be generated in the **Number of Rows for RowGenerator** field and click **OK**. The **RowGenerator Editor** closes.
- You will be prompted to propagate changes. Click **Yes** in the popup message.
- Double-click on the **tMap** component to open the Map editor. The Map editor opens displaying the input metadata of the **tRowGenerator**component.

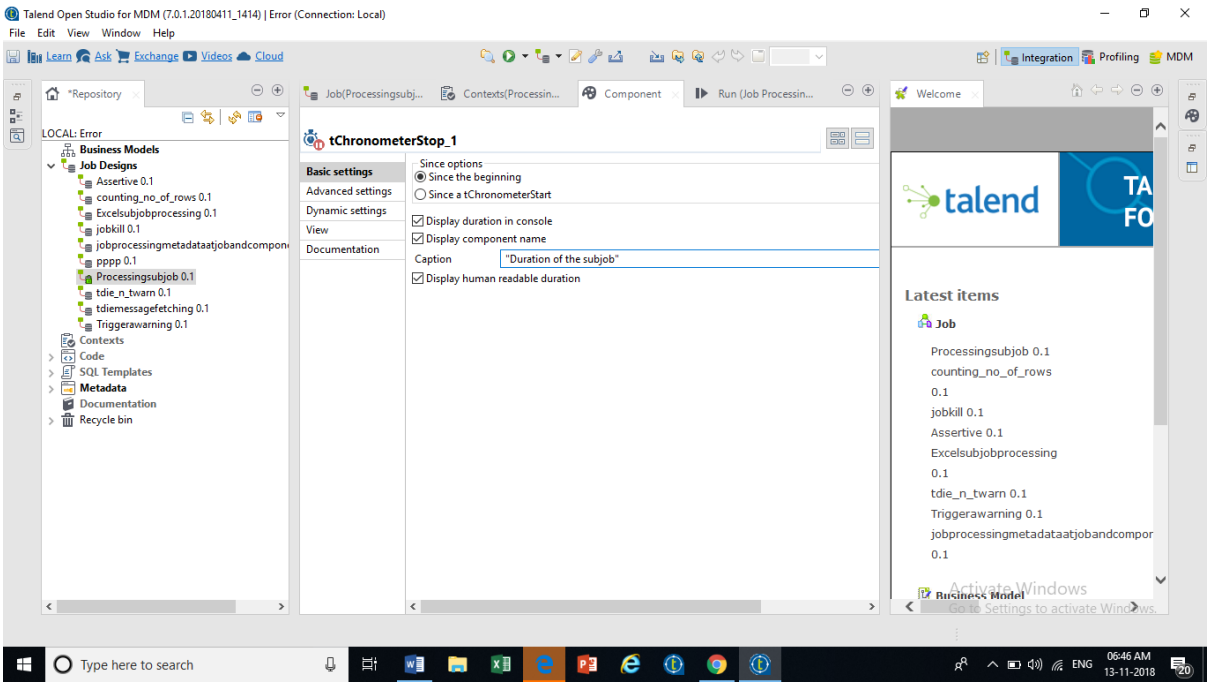


- In the **Schema editor** panel of the Map editor, click the plus button of the output table to add two rows and define them.

- In the Map editor, drag the *First_Name* row from the input table to the *Last_Name* row in the output table and drag the *Last_Name* row from the input table to the *First_Name* row in the output table.
- Click **Apply** to save changes.
- You will be prompted to propagate changes. Click **Yes** in the popup message.
- Click **OK** to close the editor.
- Select **tFileOutputDelimited** and click the **Component** tab to display the component view.
- In the **Basic settings** view, set **tFileOutputDelimited** properties as needed.



- Select **tChronometerStop** and click the **Component** tab to display the component view.
- In the **Since options** panel of the **Basic settings** view**,** select **Since the beginning** option to measure the duration of the subjob as a whole.



- Select/clear the other check boxes as needed. In this scenario, we want to display the subjob duration on the console preceded by the component name.
- If needed, enter a text in the **Caption** field.
- Save your Job and press **F6** to execute it.

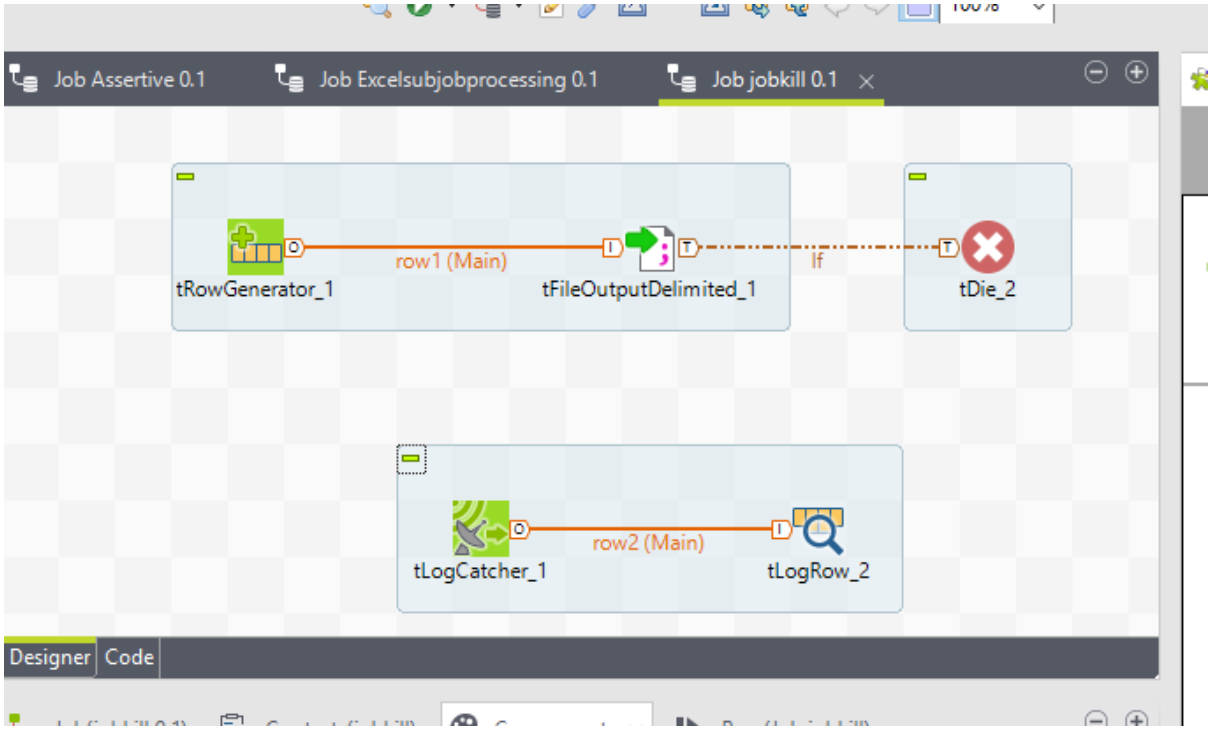
```
Starting job Processingsubjob at 06:47 13/11/2018.

[statistics] connecting to socket on port 3518
[statistics] connected
  [ tChronometerStop_1 ]      76seconds    Duration of the subjob   76564
milliseconds
 [statistics] disconnected

Job Processingsubjob ended at 06:49 13/11/2018. [exit code=0]
```

4.How to measure processing time of subjob from start to a point

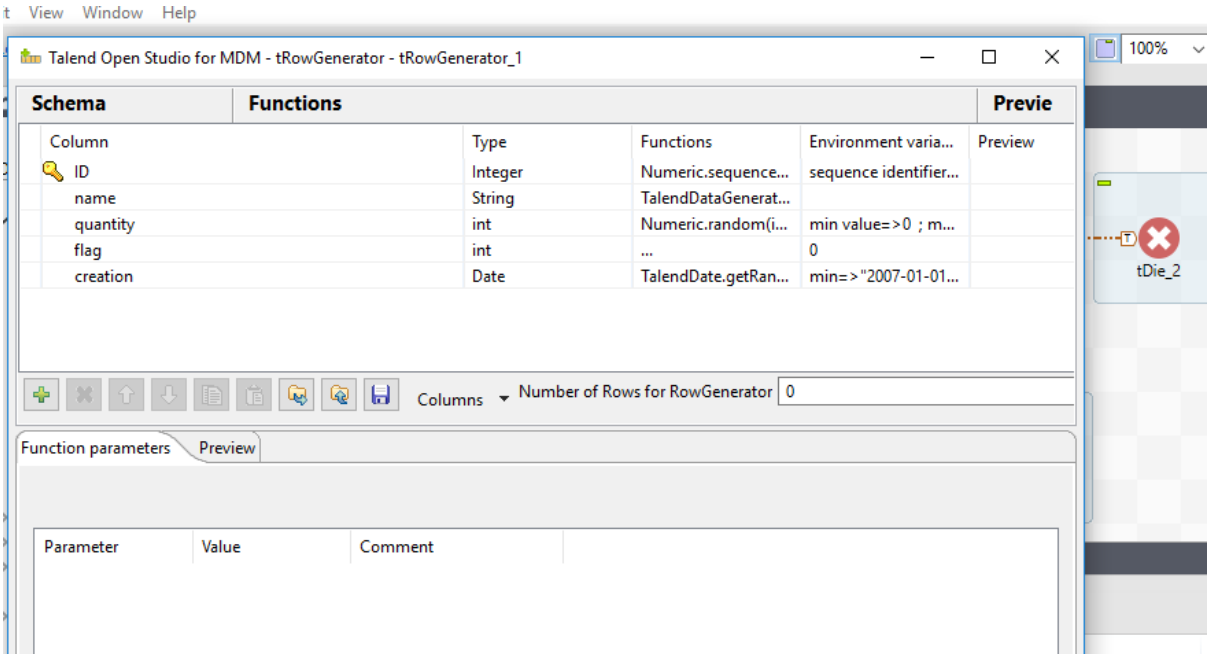
In this we have to use two tChromometerstart first will calculate the time of chromometer start then process till chromometer stop

6.How to kill a job and throw an error

This scenario uses a **tLogCatcher** and a **tDie** component. A **tRowGenerator** is connected to a **tFileOutputDelimited** using a Row link. On error, the **tDie**triggers the catcher subjob which displays the log data content on the **Run** console.



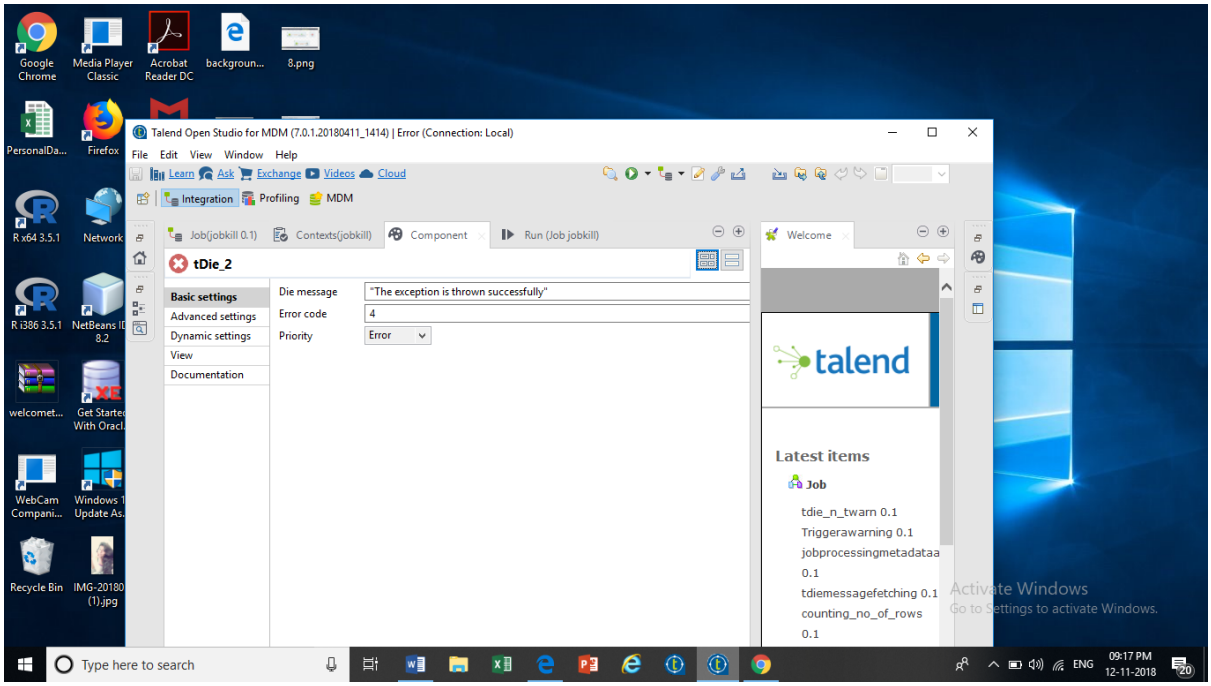
- Drop all required components from various folders of the **Palette** to the design workspace: **tRowGenerator**, **tFileOutputDelimited**, **tDie**,**tLogCatcher**, **tLogRow**.
- On the **tRowGenerator Component** view, define the setting of the input entries to be handled.



- Edit the schema and define the following columns as random input examples: *id, name, quantity, flag and creation*.
- Set the **Number of rows** onto 0. This will constitute the error which the Die operation is based on.
- On the **Values** table, define the functions to feed the input flow.
- Define the **tFileOutputDelimited** to hold the possible output data.
  The **row** connection from the **tRowGenerator** feeds automatically the output schema. The separator is a simple semi-colon.
- Connect this output component to the **tDie** using a **Trigger > If** connection. Double-click on the newly created connection to define the if:

```
((Integer)globalMap.get("tRowGenerator_1_NB_LINE")) <=0
```

- Then double-click to select and define the **Basic settings** of the **tDie** component.

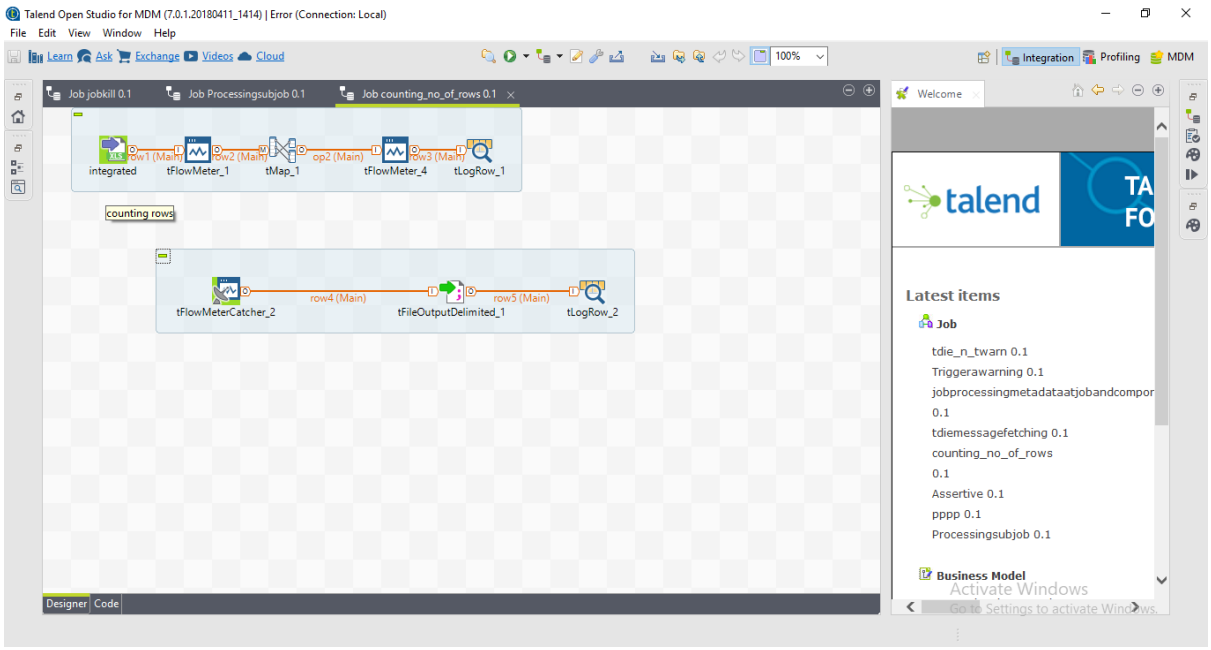


- Enter your **Die** message to be transmitted to the **tLogCatcher** before the actual kill-job operation happens.
- Next to the Job but not physically connected to it, drop a **tLogCatcher** from the **Palette** to the design workspace and connect it to a **tLogRow**component.
- Define the **tLogCatcher Basic settings**. Make sure the **tDie** box is selected in order to add the Die message to the Log information transmitted to the final component.

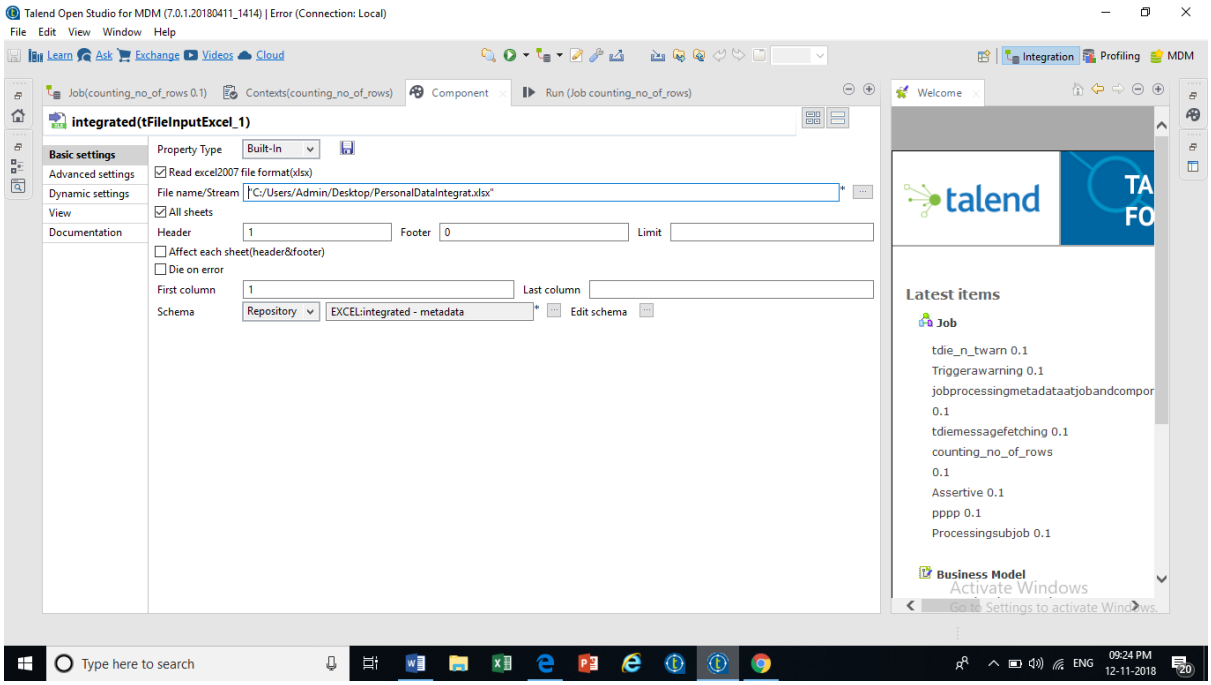
- *Starting job jobkill at 21:19 12/11/2018.*
-
- [statistics] connecting to socket on port 3352
- [statistics] connected
- The exception is thrown successfully
- moment: 2018-11-12 21:19:36|pid: lO0cFz|root_pid: lO0cFz|father_pid: lO0cFz|project: ERROR|job: jobkill|context: Default|priority: 5|type: tDie|origin: tDie_2|message: The exception is thrown successfully|code: 4
- [statistics] disconnected
-
- *Job jobkill ended at 21:19 12/11/2018. [exit code=4]*

4. How to count the rows processed in a flow

The following basic Job aims at catching the number of rows being passed in the flow processed. The measures are taken twice, once after the input component, that is, before the filtering step and once right after the filtering step, that is, before the output component.

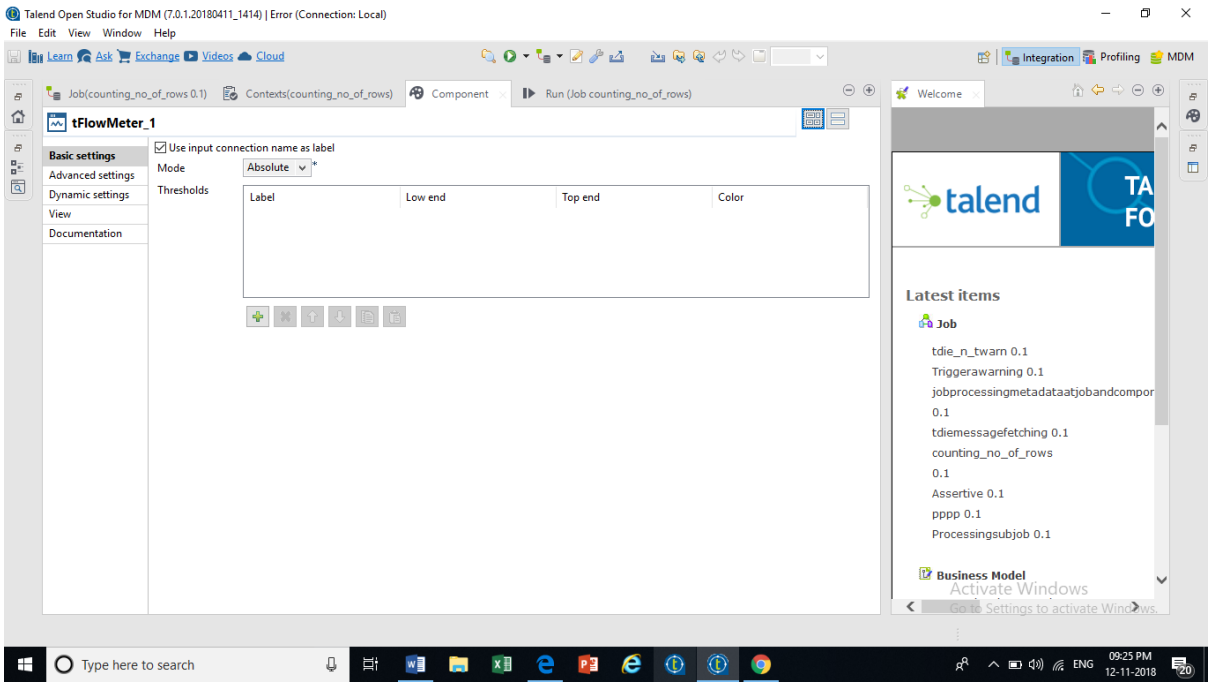


- Drop the following components from the **Palette** to the design workspace: **Excel inputfile**, **tFlowMeter** (x2), **tMap**, **tLogRow**, **tFlowMeterCatcher** and**tFileOutputDelimited**.
- Link components using row main connections and click on the label to give consistent name throughout the Job, such as *names* from the input component and *filtered_names* for the output from the **tMap** component, for example.
- Link the **tFlowMeterCatcher** to the **tFileOutputDelimited** component using a row main link also as data is passed.



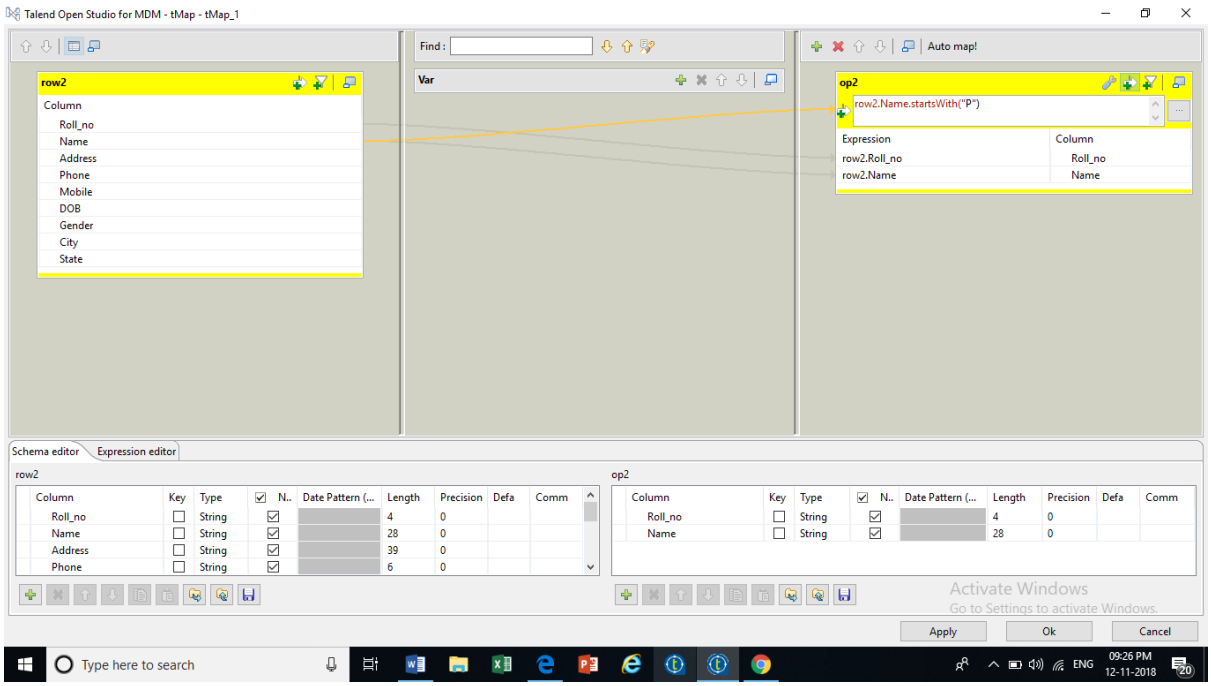
- Select the relevant **encoding type** on the Advanced settings vertical tab.

- Then select the following component which is a **tFlowMeter** and set its properties.

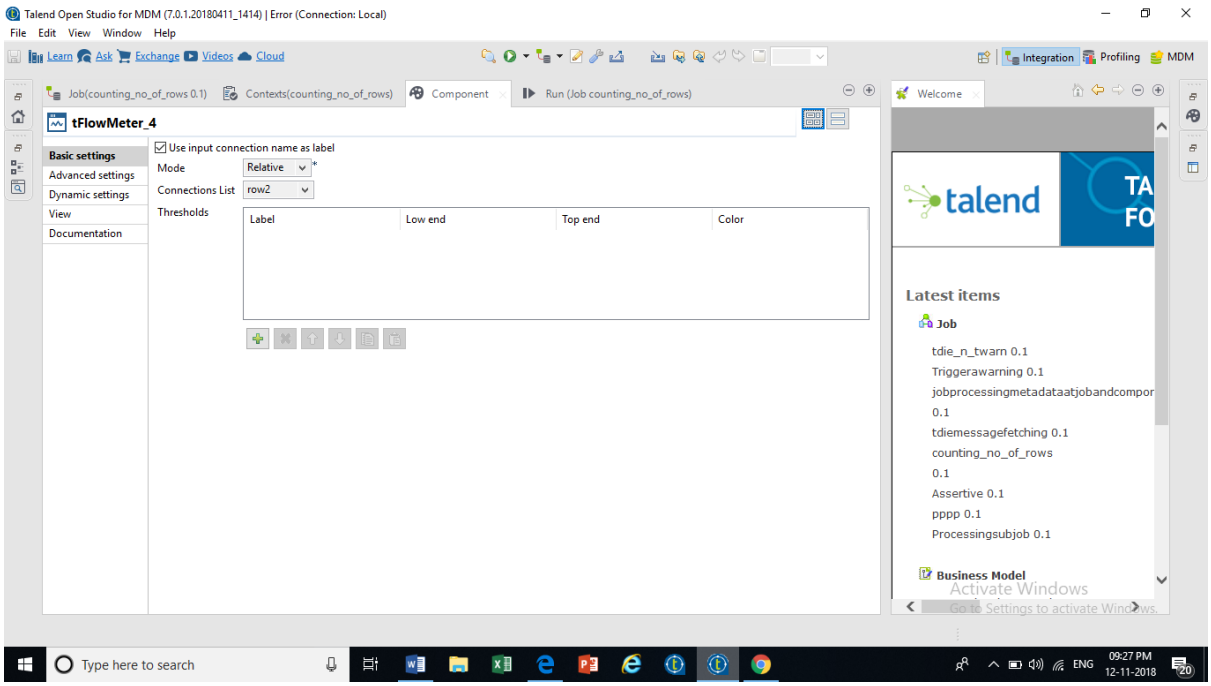


- Select the check box **Use input connection name as label**, in order to reuse the label you chose in the log output file (**tFileOutputDelimited**).
- The mode is **Absolute** as there is no reference flow to meter against, also no **Threshold** is to be set for this example.

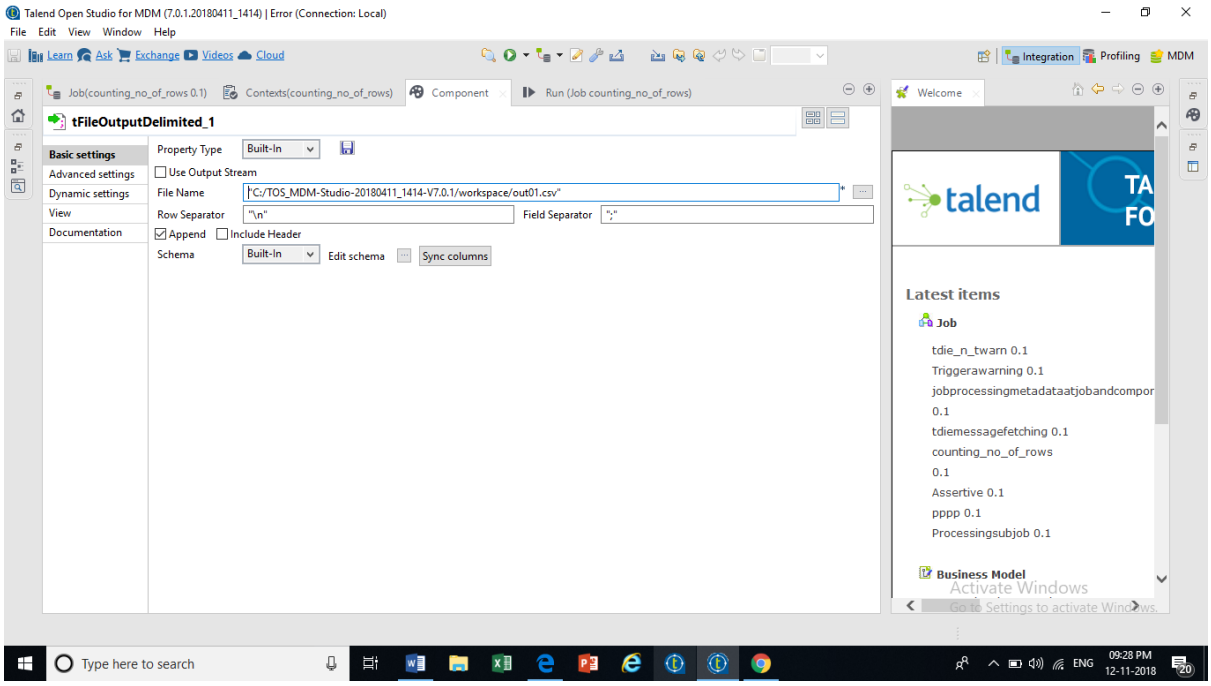
- Then launch the **tMap** editor to set the filtering properties.
- For this use case, drag and drop the ID and State columns from the Input area of the **tMap** towards the Output area. No variable is used in this example.



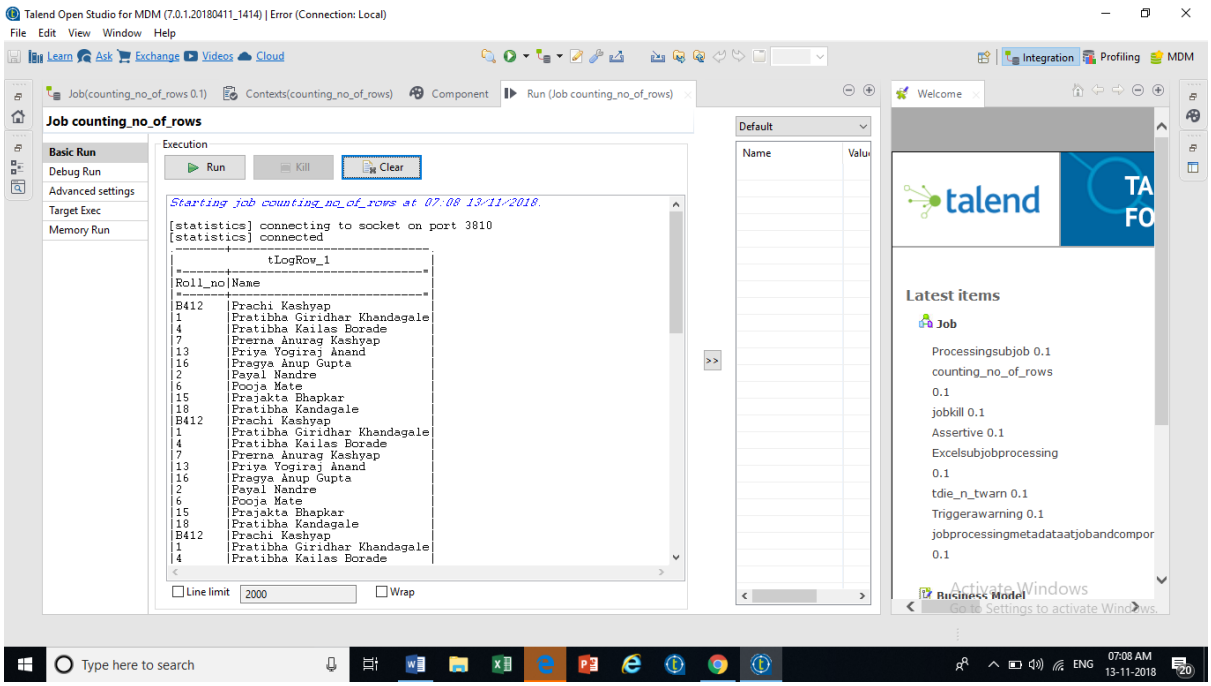
- On the Output flow area (labelled *filtered_states* in this example), click the arrow & plus button to activate the expression filter field.
- Drag the *State* column from the Input area (*row2*) towards the expression filter field and type in the rest of the expression in order to filter the state labels starting with the letter *M*. The final expression looks like: `row2.State.startsWith("M")`
- Click **OK** to validate the setting.
- Then select the second **tFlowMeter** component and set its properties.

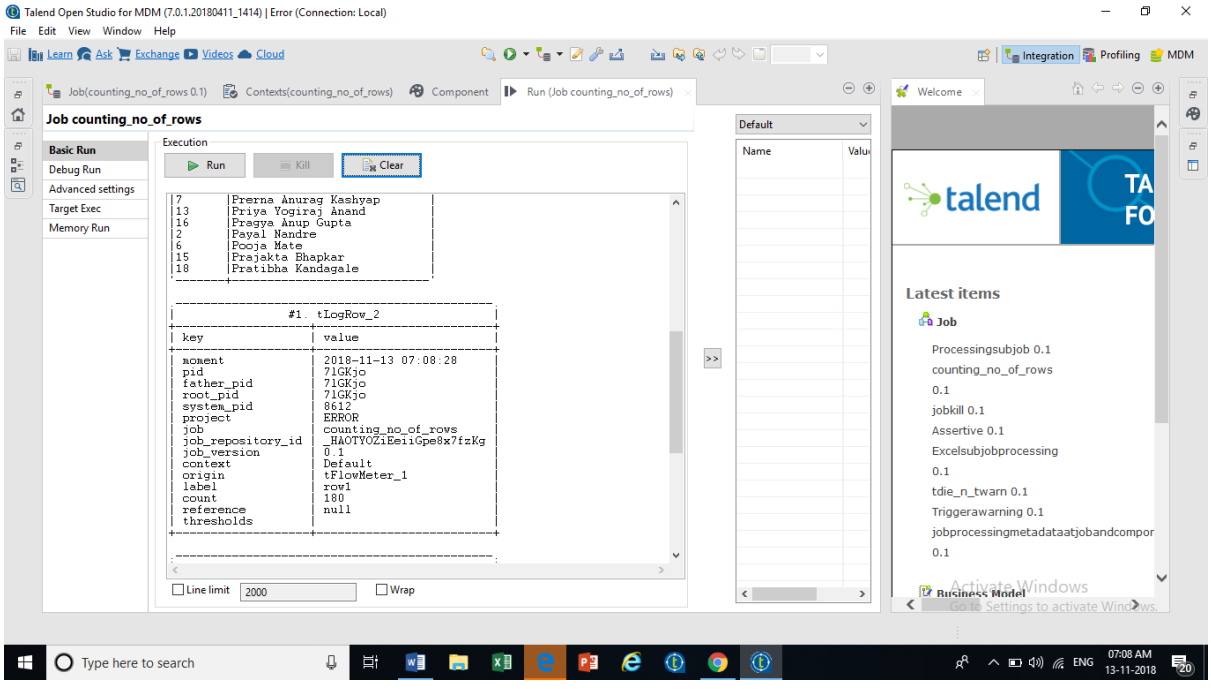


- Select the check box **Use input connection name as label**.
- Select **Relative** as **Mode** and in the **Reference connections** list, select *Excel* as reference to be measured against.
- Once again, no threshold is used for this use case.
- No particular setting is required in the **tLogRow**.
- Neither does the **tFlowMeterCatcher** as this component's properties are limited to a preset schema which includes typical log information.
- So eventually set the log output component (**tFileOutputDelimited**).



- Select the **Append** check box in order to log all **tFlowMeter** measures.
- Then save your Job and press **F6** to execute it.





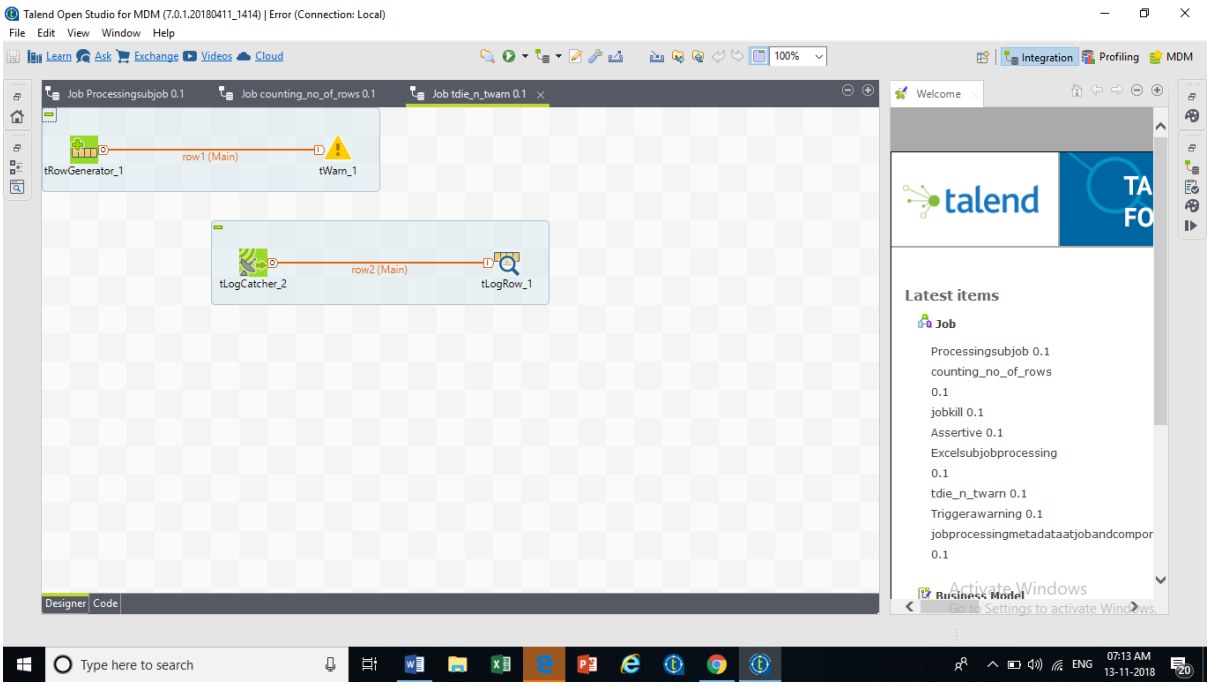
7.How to fetch messages from tDie and tWarn components

1. Catching messages triggered by a tWarn component

This example shows you how to use the **tLogCatcher** component to catch the messages triggered by a **tWarn** component.

Creating a Job for catching messages triggered by a tWarn component

Create a Job to trigger some messages using the **tWarn** component, then catch the messages using the **tLogCatcher** component and display the messages on the console.

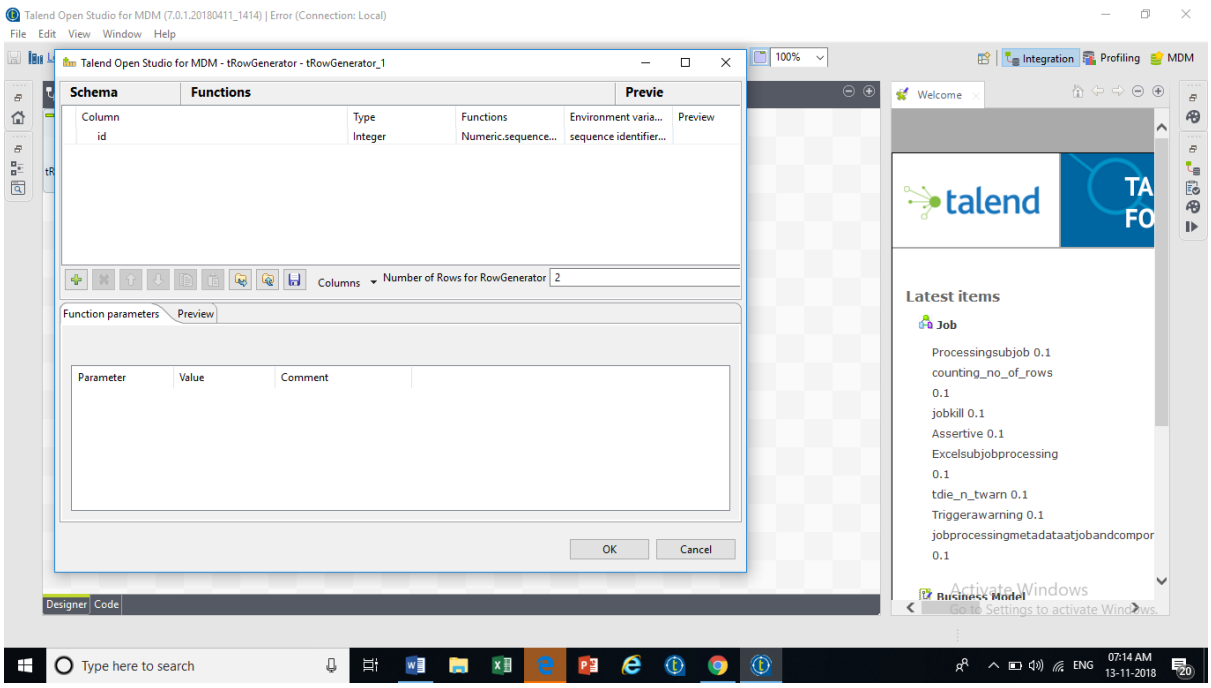


1. Create a new Job and add a **tRowGenerator** component, a **tWarn** component, a **tLogCatcher** component, and a **tLogRow** component by typing their names in the design workspace or dropping them from the **Palette**.
2. Link the **tRowGenerator** component to the **tWarn** component using a **Row** > **Main** connection.
3. Link the **tLogCatcher** component to the **tLogRow** component using a **Row** > **Main** connection.

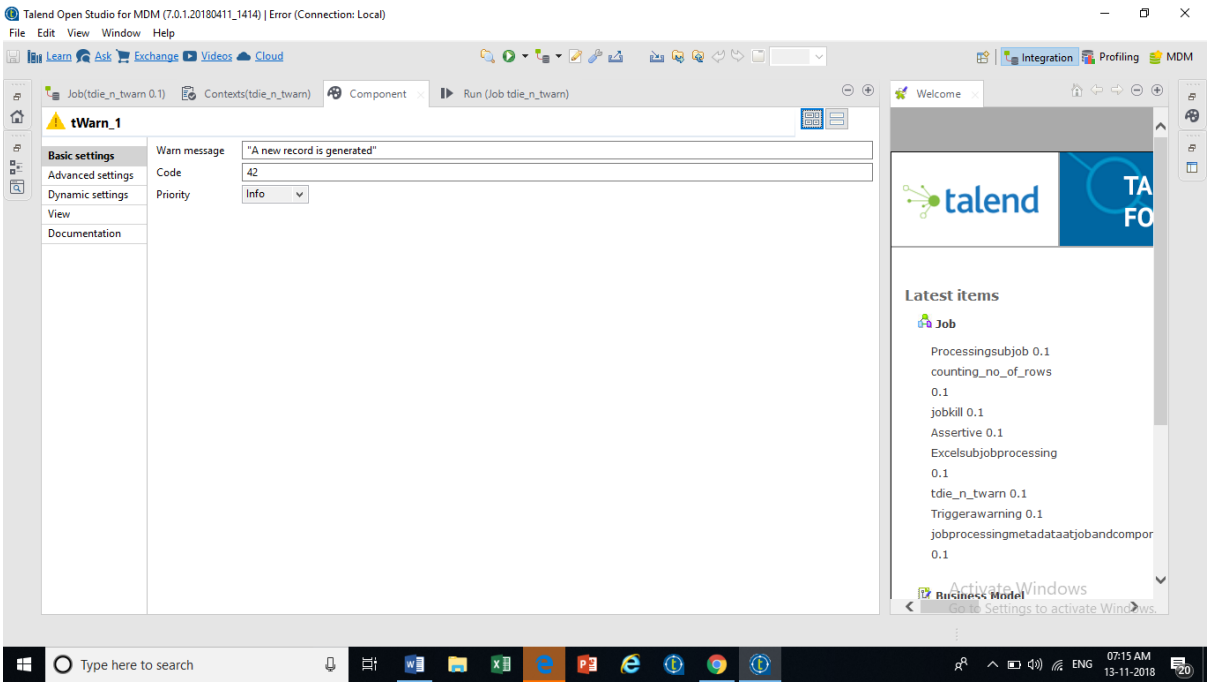
Configuring the Job for catching messages triggered by the tWarn component

Configure the components used in the Job that catches the messages triggered by the **tWarn** component and then displays the messages on the console.

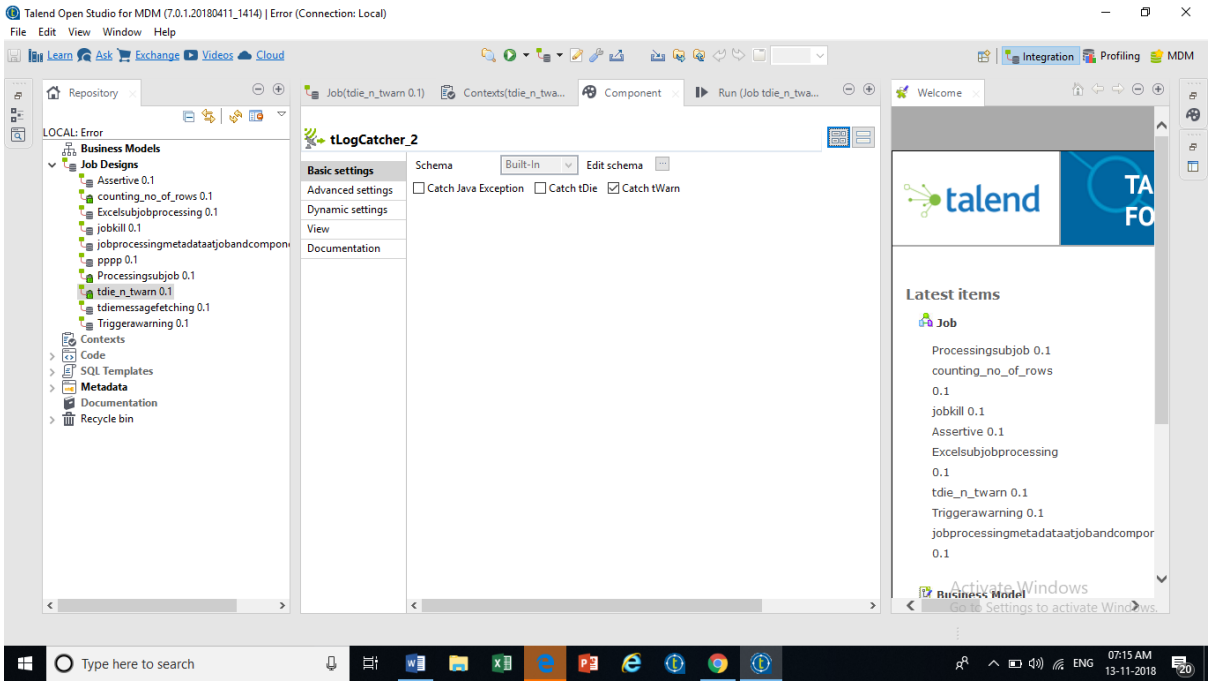
1.Double-click the **tRowGenerator** component to open its row generator editor.



1.
2. Define the schema by adding one column *id* of Integer type, and select the predefined function *Numeric.sequence(String,int,int)* in the **Functions**column.
3. Enter the number of records to be generated in the **Number of Rows for RowGenerator** field, *3* in this example. When done, click **OK** to close the dialog box.
4. Double-click the **tWarn** component to open its **Basic settings** view.

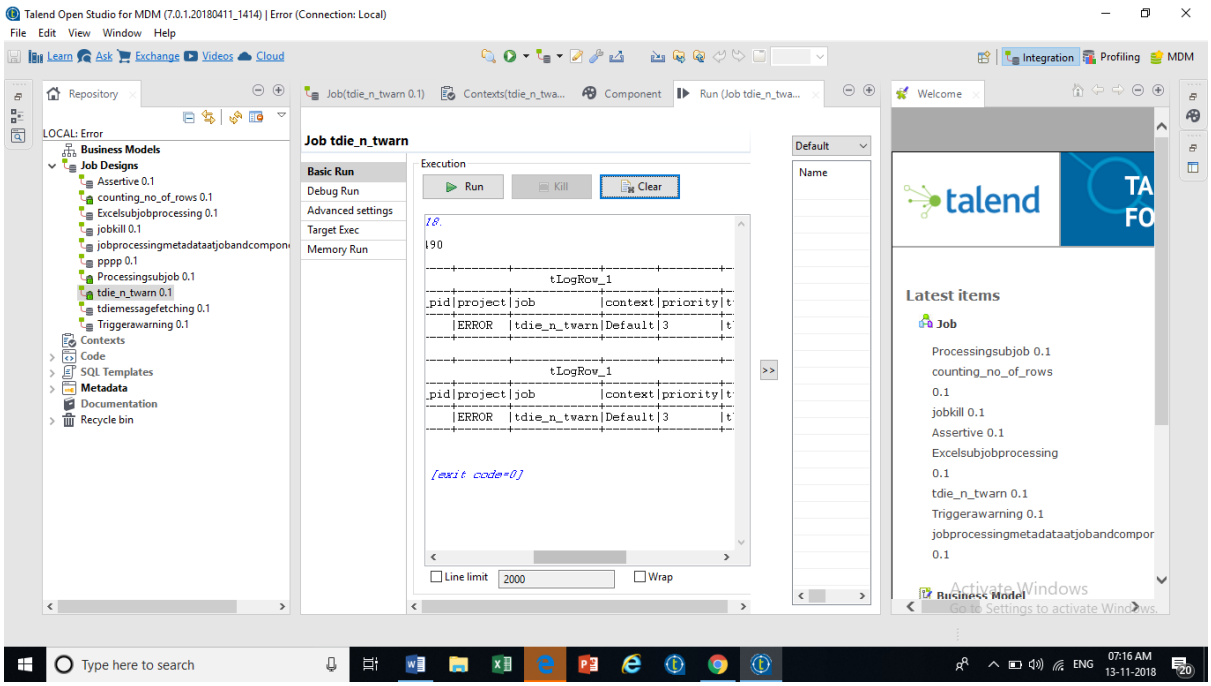


1. Select *Info* from the **Priority** drop-down list.
2. In the **Warn message** field, enter the message to be triggered when a new record is generated, *a new record is generated* in this example.
3. Double-click the **tLogCatcher** component to open its **Basic settings** view.



Executing the Job to catch messages triggered by a tWarn component

After setting up the Job and configuring the components used in the Job for catching messages triggered by the **tWarn** component, you can then execute the Job and verify the Job execution result.

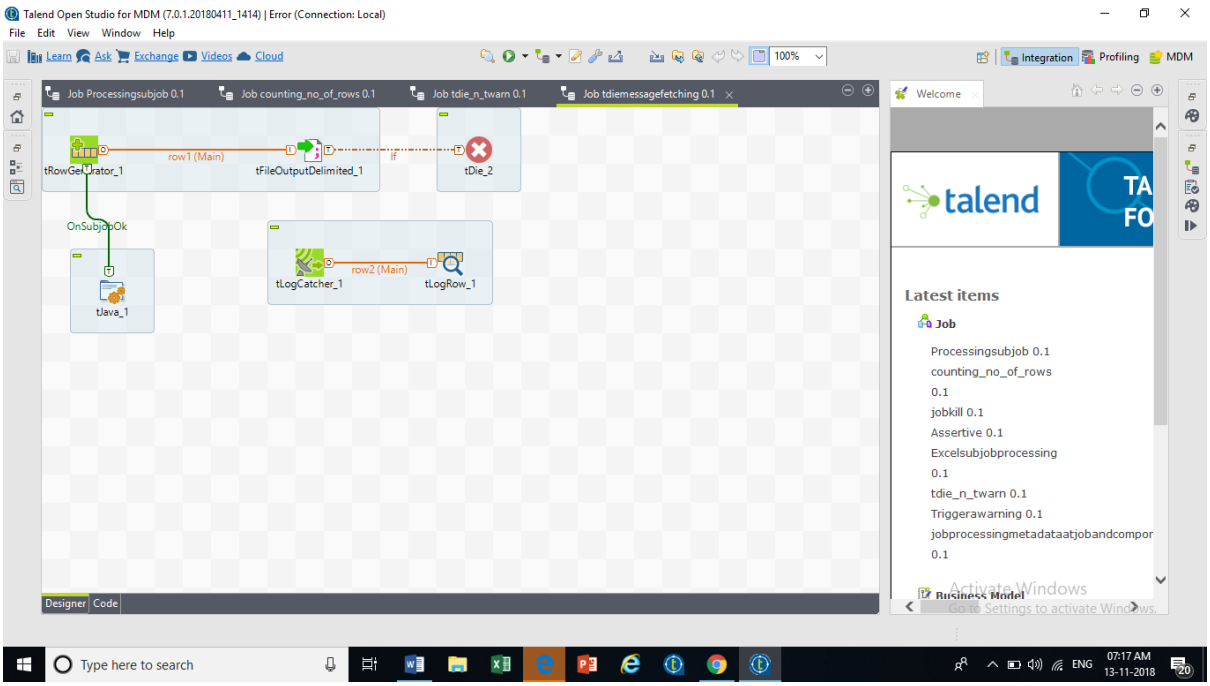


Catching the message triggered by a tDie component

This example shows you how to use the tLogCatcher component to catch the message triggered by a tDie component.

Creating a Job for catching the message triggered by a tDie component

Create a Job to trigger a message using the tDie component, then catch the message using the tLogCatcher component and display the message on the console.



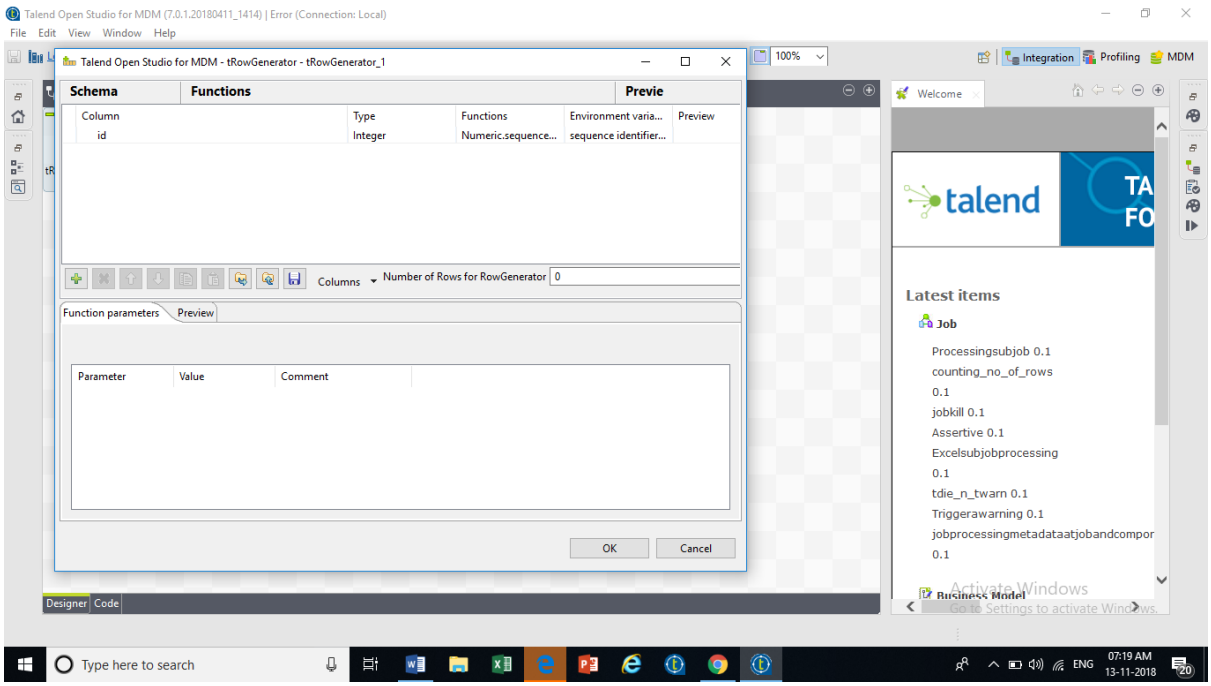
1.Create a new Job and add a **tRowGenerator** component, a **tFileOutputDelimited** component, a **tDie** component, a **tJava** component, a **tLogCatcher** component, and a **tLogRow** component by typing their names in the design workspace or dropping them from the **Palette**.

1. Link the **tRowGenerator** component to the **tFileOutputDelimited** component using a **Row** > **Main** connection.
2. Link the **tFileOutputDelimited** component to the **tDie** component using a **Trigger** > **Run if** connection.
3. Link the **tRowGenerator** component to the **tJava** component using a **Trigger** > **On Subjob Ok** connection.
4. Link the **tLogCatcher** component to the **tLogRow** component using a **Row** > **Main** connection.

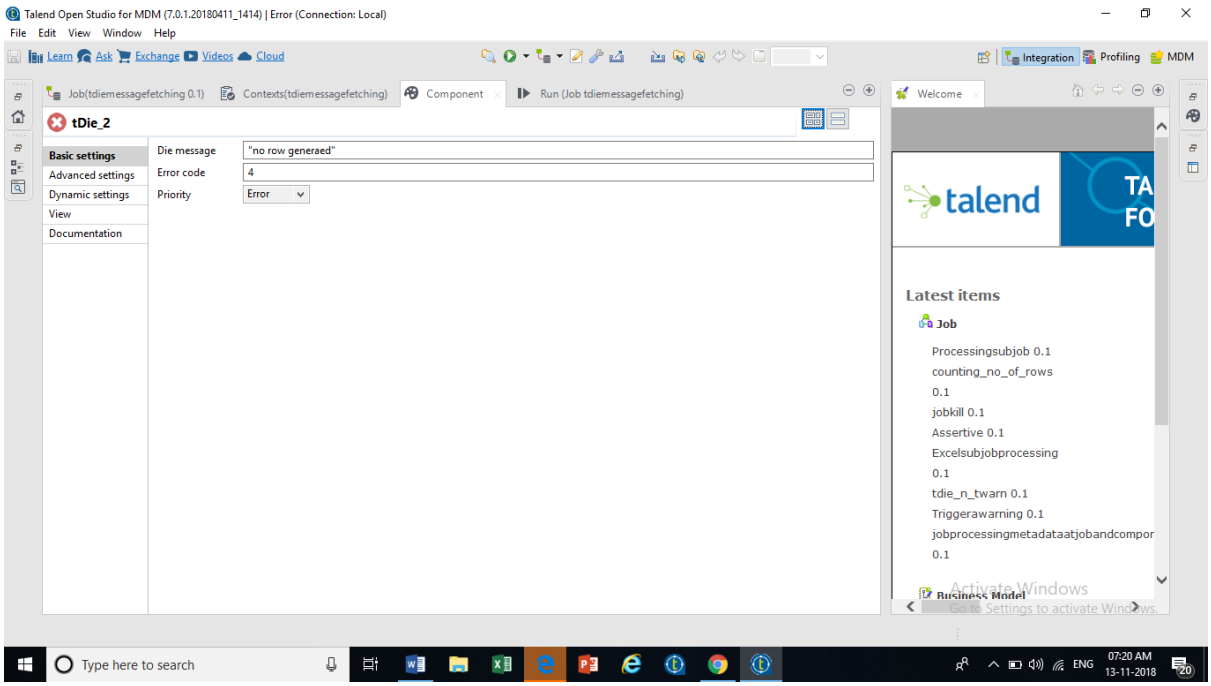
Configuring the Job for catching the message triggered by the tDie component

Configure the components used in the Job that catches the message triggered by the **tDie** component and then displays the message on the console.

1. Double-click the **tRowGenerator** component to open its row generator editor.



1. Define the schema by adding one column *id* of Integer type, and select the predefined function *Numeric.sequence(String,int,int)* in the **Functions**column.
2. Enter the number of rows to be generated in the **Number of Rows for RowGenerator** field, *0* in this example. When done, click **OK** to close the dialog box.
3. Double-click the **tFileOutputDelimited** component to open its **Basic settings** view, and in the **File Name** field, specify the path to the file that will hold the data to be generated.
4. Click the **If** connection, and in the **Condition** field on the **Basic settings** view, specify the condition based on which the **tDie** component will be triggered. In this example, it is *((Integer)globalMap.get("tRowGenerator_1_NB_LINE")) <=0*, which means the **tDie** component will be triggered when the number of rows to be generated is less than or equal to zero.
5. Double-click the **tDie** component to open its **Basic settings** view, and in the **Die message** field, enter the message to be triggered before the Job is killed. In this example, it is *no row generated*.

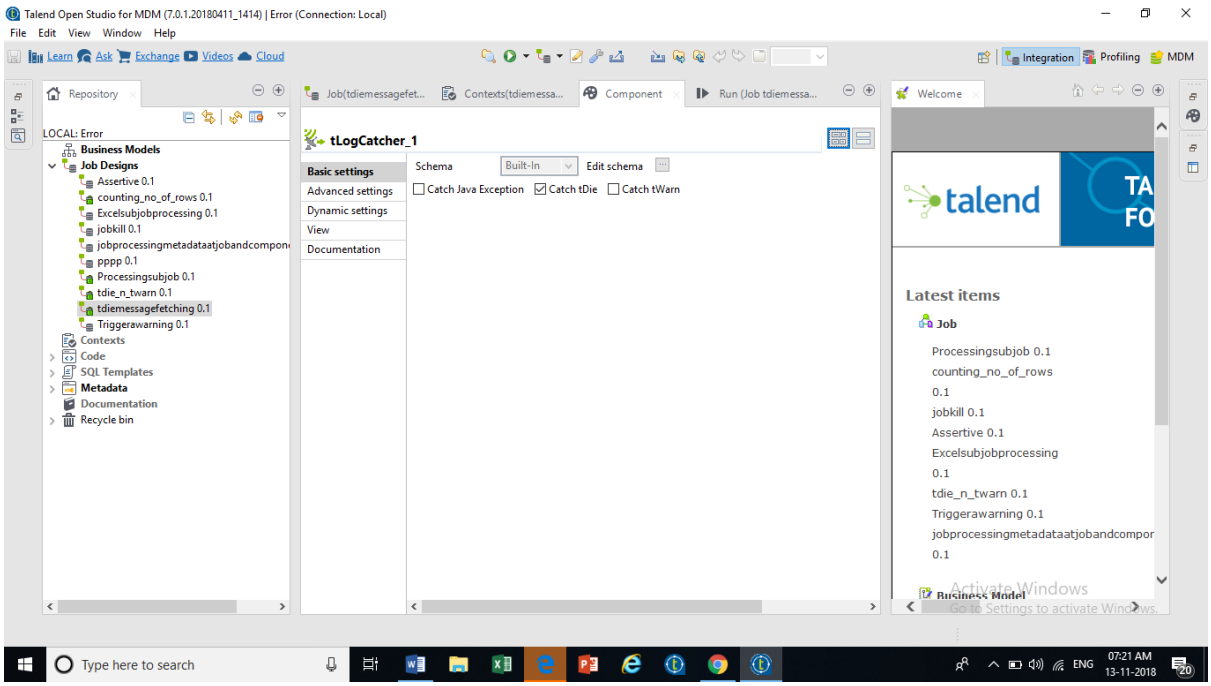


1. Double-click the **tJava** component to open its **Basic settings** view, and in the **Code** field, enter

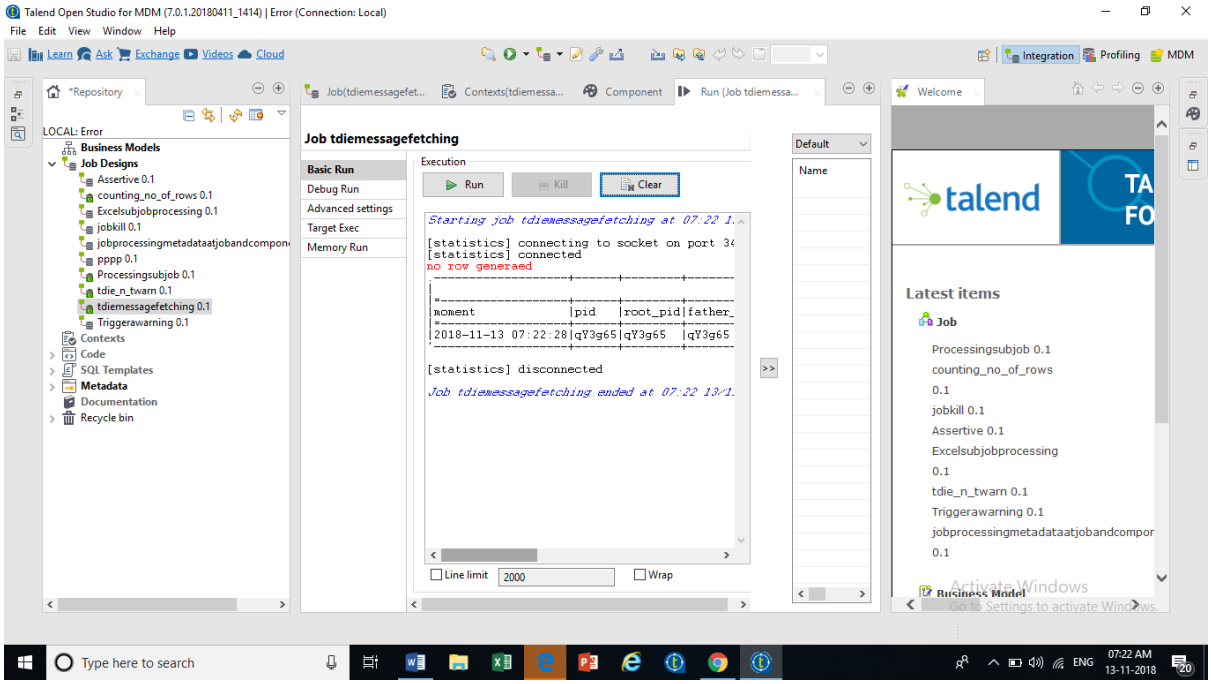
   ```
   System.out.println("The number of rows generated is " + ((Integer)globalMap.get("tRowGenerator_1_NB_LINE")) + ". #This message will not be displayed if no row is generated.");
   ```

   Note that this message will be displayed only when the number of rows generated is greater than zero. In this example, the number of rows to be generated is *0*, so the Job will be killed and this message will not be displayed.

2. Double-click the **tLogCatcher** component to open its **Basic settings** view and select the **Catch tDie** check box to catch the message triggered by the **tDie** component.



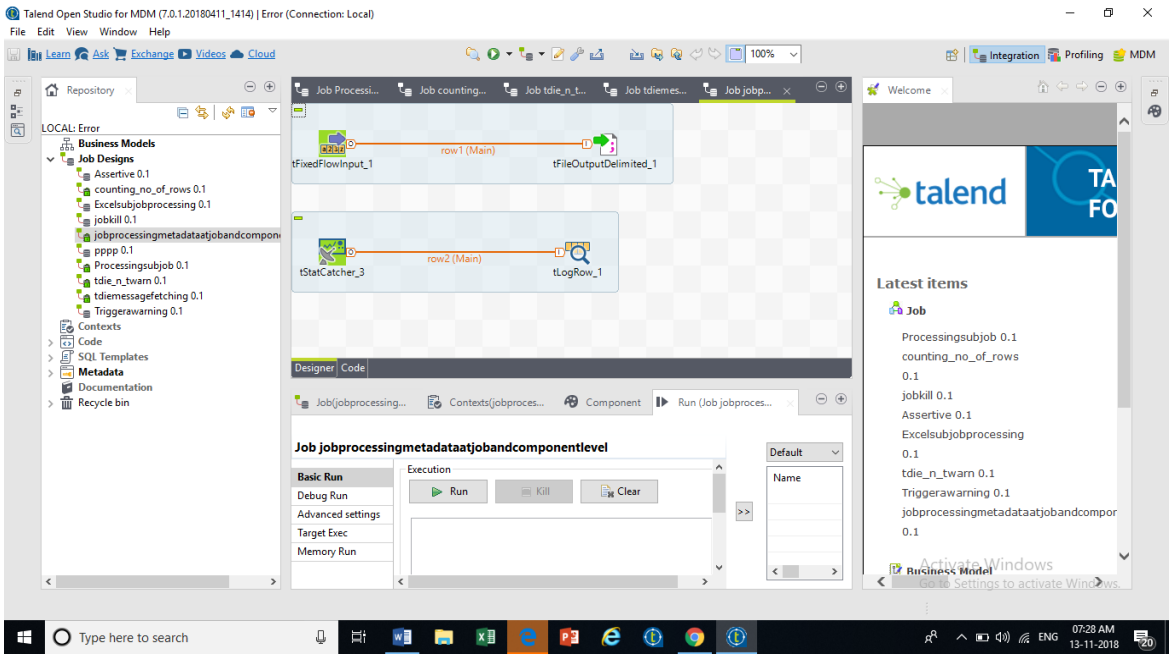
After setting up the Job and configuring the components used in the Job for catching the message triggered by the **tDie** component, you can then execute the Job and verify the Job execution result.



8.How to gather job processing data at job level and component level

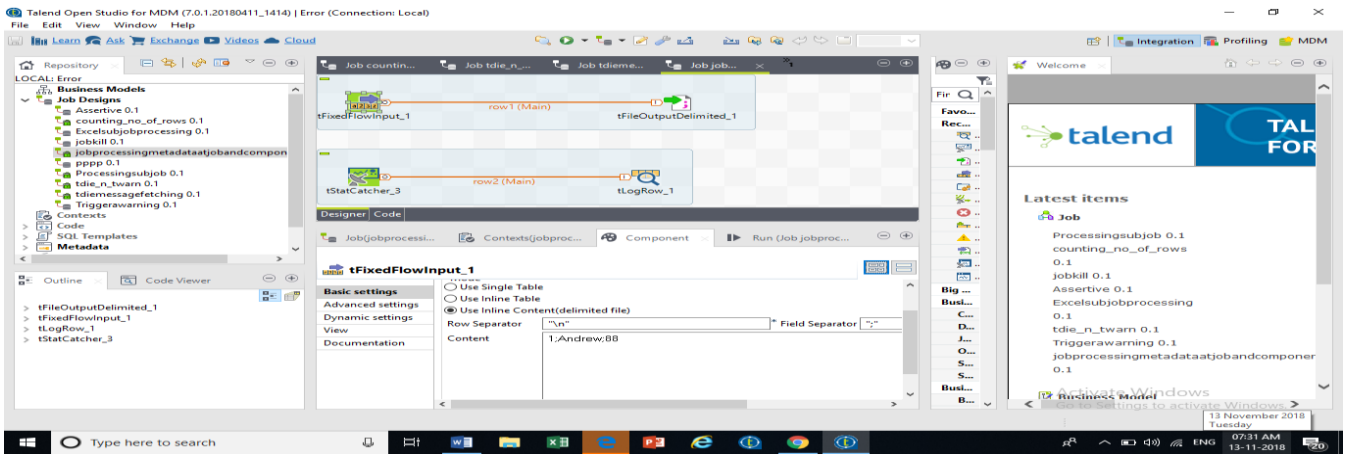
This scenario collects the statistics log for the Job execution and displays it on the **Run** console. Note that, since the tStatCatcher Statistics check box is not selected for the components, the statistics log applies solely to this specific Job.

1. Drop **tFixedFlowInput**, **tFileOutputDelimited**, **tStatCatcher** and **tLogRow** onto the workspace.
2. Link **tFixedFlowInput** to **tFileOutputDelimited** using a **Row** > **Main** connection.
3. Link **tStatCatcher** to **tLogRow** using a **Row** > **Main** connection.

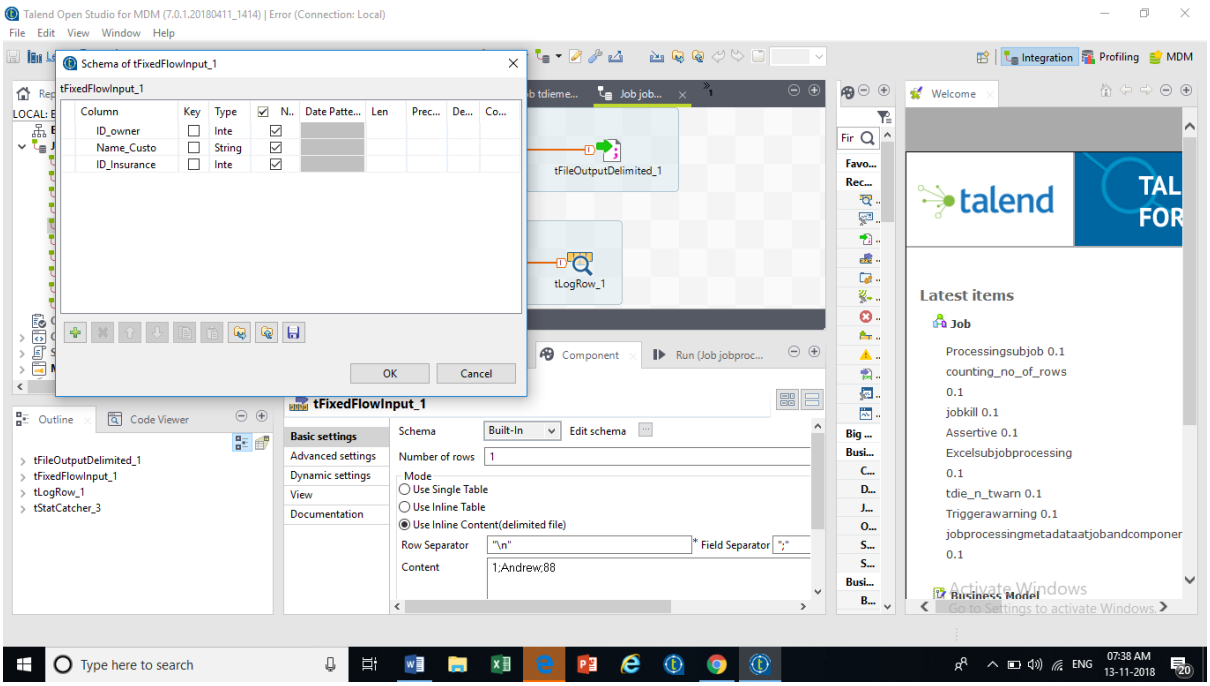


Procedure:

1.Double-click **tFixedFlowInput** to open its **Basic settings** view.

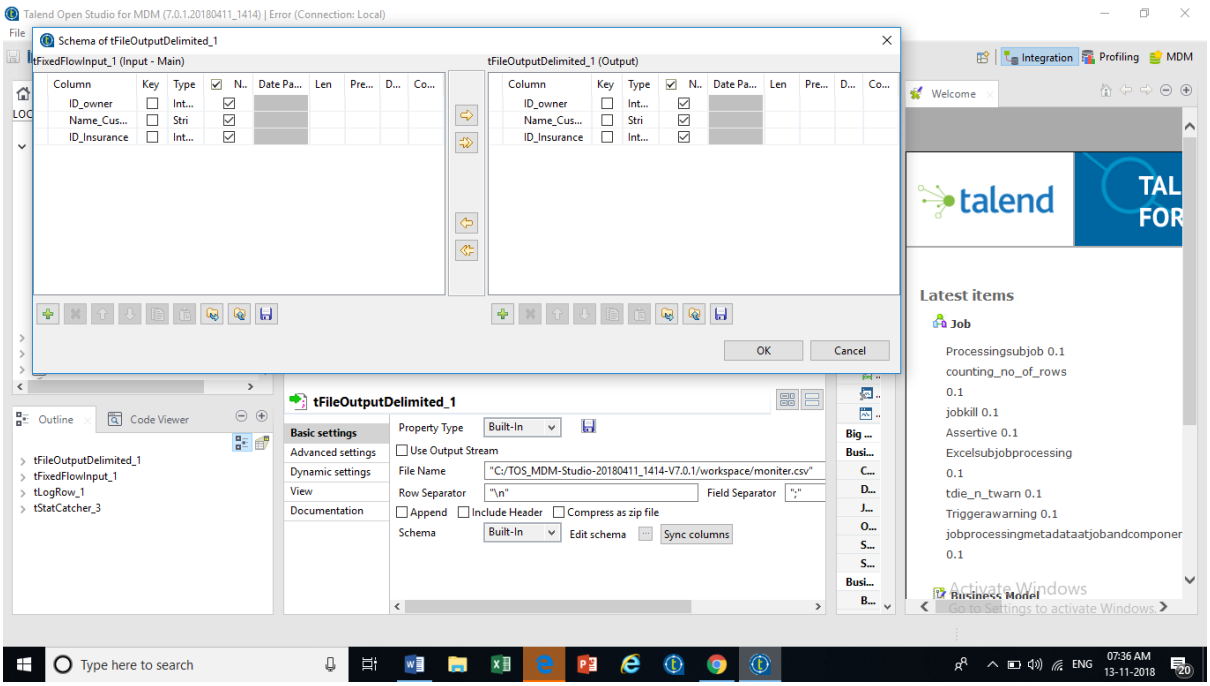


1. Click the **Edit schema** button to open the schema editor.



2. Click the **[+]** button to add three columns,
   namely *ID_Owners*, *Name_Customer* and *ID_Insurance*, of the Integer and String types respectively.
3. Click **Ok** to validate the setup and close the editor.
4. In the dialog box that appears, click **Yes** to propagate the changes to the subsequent component.

5. Select the **Use Inline Content (delimited file)** option.
6. In the **Content** box, enter 1;Andrew;888.
7. Double-click **tFileOutputDelimited** to open its **Basic settings** view.

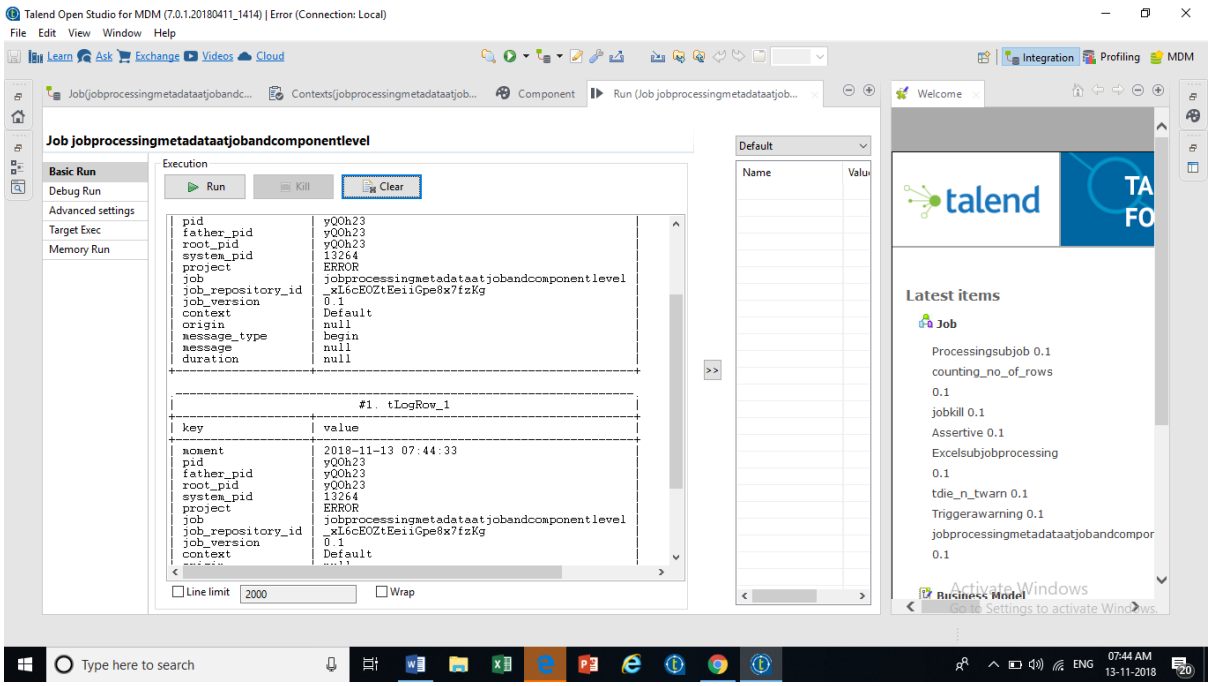


8. In the **File Name** field, enter the full name of the file to save the statistics data.
9. Double-click **tLogRow** to open its **Basic settings** view.

10. Select **Vertical (each row is a key/value list)** for a better display of the results

1. Press **Ctrl + S** to save the Job.

2. Press **F6** to run the Job.


Talend Open Studio for MDM (7.0.1.20180411_1414) | Error (Connection: Local)
File Edit View Window Help
Learn Ask Exchange Videos Cloud
Integration Profiling MDM
Job(jobprocessingmetadataatjobandc...
Contexts(jobprocessingmetadataatjob...
Component
Run (Job jobprocessingmetadataatjob...
Welcome
Job jobprocessingmetadataatjobandcomponentlevel
Basic Run
Debug Run
Advanced settings
Target Exec
Memory Run
Execution
Run
Kill
Clear
pid	yQOh23
father_pid	yQOh23
root_pid	yQOh23
system_pid	13264
project	ERROR
job	jobprocessingmetadataatjobandcomponentlevel
job_repository_id	_xL6cEOZtEeiiGpe8x7fzKg
job_version	0.1
context	Default
origin	null
message_type	begin
message	null
duration	null
#1. tLogRow_1
key	value
moment	2018-11-13 07:44:33
pid	yQOh23
father_pid	yQOh23
root_pid	yQOh23
system_pid	13264
project	ERROR
job	jobprocessingmetadataatjobandcomponentlevel
job_repository_id	_xL6cEOZtEeiiGpe8x7fzKg
job_version	0.1
context	Default
Line limit 2000
Wrap
Default
Name
talend
Latest items
Job
Processingsubjob 0.1
counting_no_of_rows 0.1
jobkill 0.1
Assertive 0.1
Excelsubjobprocessing 0.1
tdie_n_twarn 0.1
Triggerawarning 0.1
jobprocessingmetadataatjobandcompor 0.1
Business Model
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
07:44 AM
13-11-2018

9.How to moniter data processed in Run console

Data processed in run console can be managed with the help of tLogRow this will help us to manage run console as shown in above example.

10.How to Trigger a warning

Warning can be triggerd by two methods

1.tDie

2.tWarn

These methods are shown above