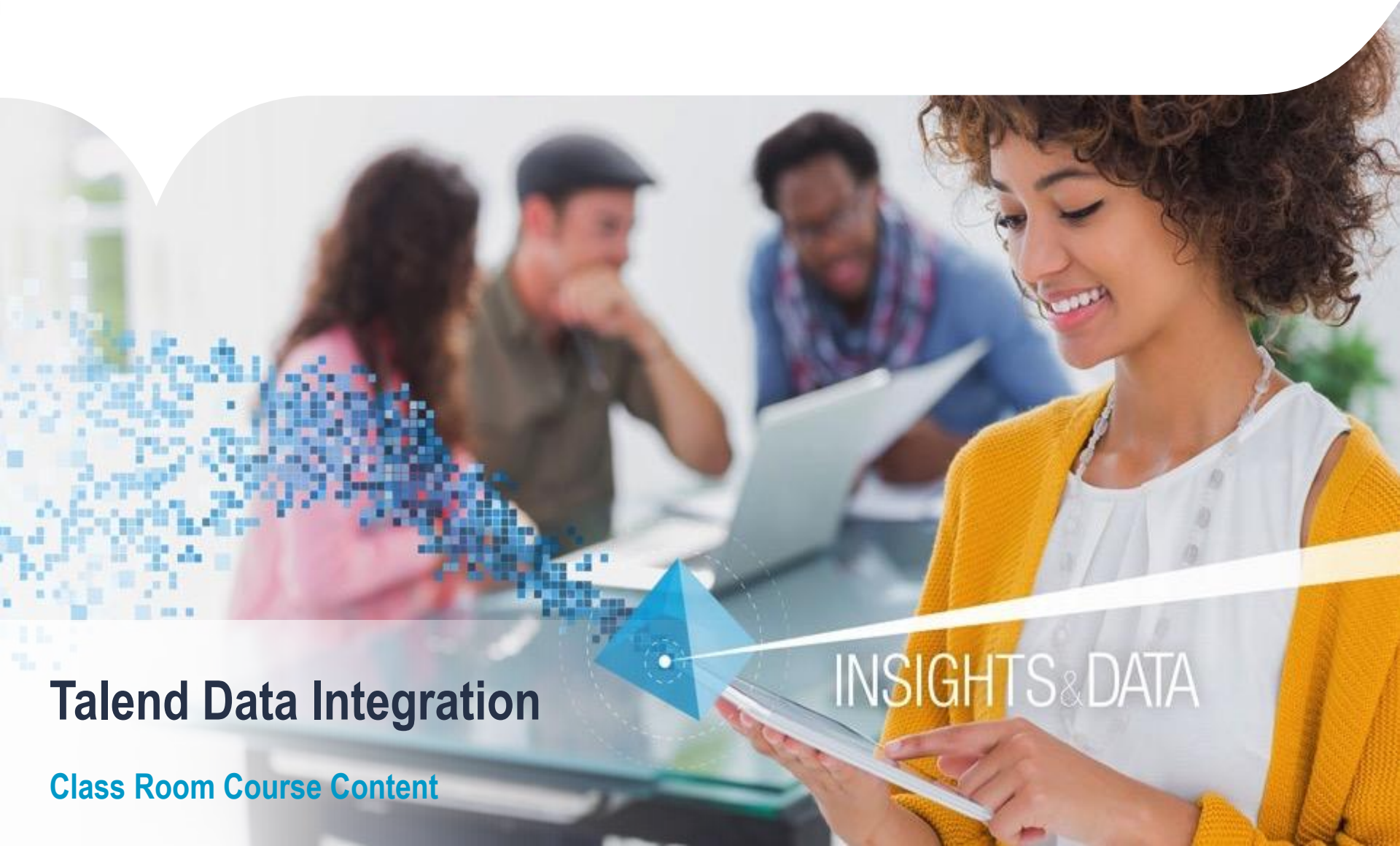
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.



Talend Data Integration

Class Room Course Content

Course Outline

- Talend Introduction and Architecture
- Talend Data Integration Installation
- Talend Data Integration Overview
- Building First Job in Talend
- Source and Target Connectivity
- TDI – Processing and Transformation Components
- Working With Master Jobs
- Building Standalone jobs
- Talend Best Practices
- ERROR – Handling
- How to catch information about your jobs' executions (Additional topic)
- How to use a Multi Schema component (Additional topic)
- Talend Big Data DI and Hadoop Components

Talend Introduction and Architecture

Module Outline

- Talend Introduction
  - Introduction
  - Scope
  - History
  - Products
- Talend Architecture
  - Architecture

Talend - Introduction

- Talend is the leading open source integration software provider to data-driven enterprises.
- It's modern data platform and open source approach simplifies
  - ✓the development process
  - ✓reduces the learning curve
  - ✓decreases total cost of ownership
  - ✓to connect more data
- Talend connects at big data scale, 5x faster and at 1/5th the cost.

Scope

- Talend addresses all of an organization's data integration needs:
  - ✓Synchronization or replication of databases
  - ✓Right-time or batch exchanges of data
  - ✓ETL (Extract Transform Load) for BI or analytics
  - ✓Data migration
  - ✓Complex data transformation and loading
  - ✓Basic data quality
  - ✓Big Data.

History

- Talend was founded in 2005 by Bertrand Diard and Fabrice Bonan. Headquartered in Redwood City, California.
- The company's first product. Talend Open Studio for Data Integration was launched in October 2006.
- Currently Talend has more than 4000 paying customers including eBay, Sony Online Entertainment ,Disney etc.
- Current version available is 6.2.0.
- More than 900+ built-in connectors that allow you to easily link a wide array of sources and targets.

Products

Big Data Integration

Application Integration

Data Integration

Data Preparation

Cloud Integration

Master Data Management

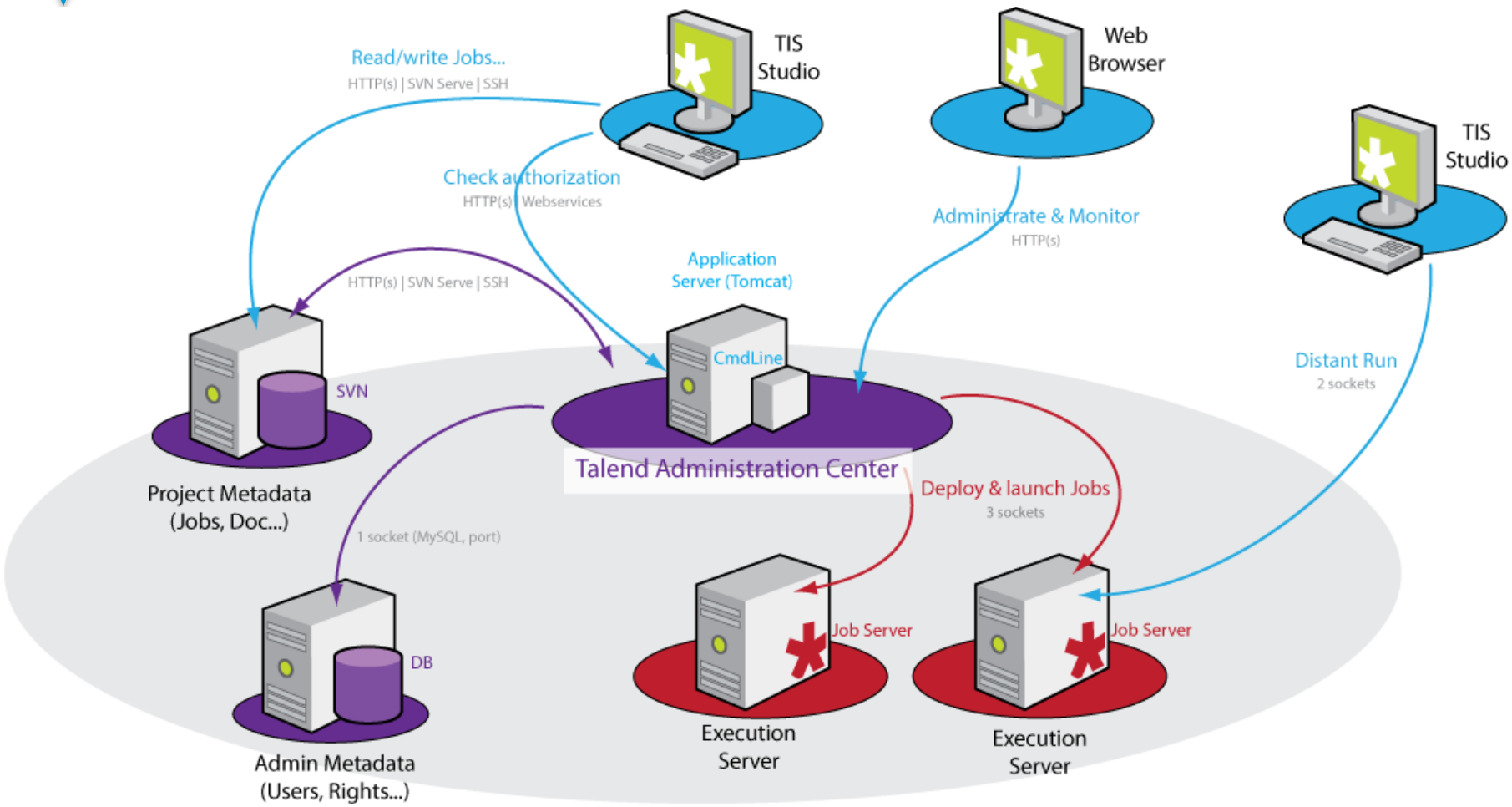
Products (Contd..)

- Big data integration - Realize the speed and scale of Big Data without coding
- Data integration - Respond to business needs for integrated data
- Cloud integration - Connect all your data in the cloud and on premises
- Application integration - Deliver agile real-time integration of applications
- Data preparation - Fast and easy self-service data prep for everyone
- Master data management - Empower business with a single view of the truth

Talend - Architecture

- The operating principles of the Talend products could be summarized as briefly as the following topics:
  - ✓ building technical or business-related processes,
  - ✓ administrating users, projects, access rights and processes and their dependencies,
  - ✓ deploying and executing technical processes,
  - ✓ monitoring the execution of technical processes.
- Each of the above topics can be isolated in different functional blocks and the different types of blocks and their interoperability can be described as in the following architecture diagram

Talend – Architecture (Contd..)



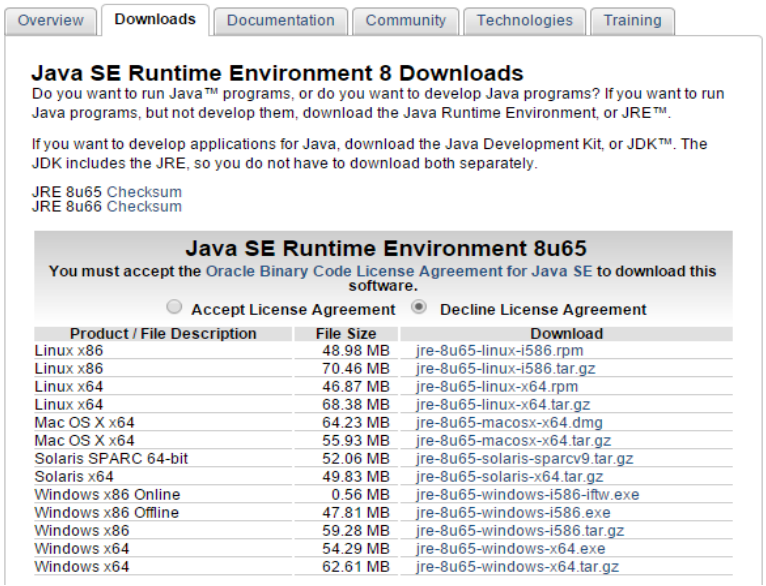
Talend Data Integration Installation

Module Outline

- Talend Data Integration Installation
  - Pre-requisites to Run TDI/Studio
  - Talend Studio - Installation steps
  - Welcome to TalendStudio?

Pre-requisites to Run TDI/Studio

- **Follow the steps below to download Java JRE 8. (Talendonly supports 64 bit):**
- 1.From the Java SE Downloads Page, click on**JRE Download.**
- 2.Scroll down to the latest **“Java SE Runtime Environment”.**
- 3.Click on the radio button to “**Accept License Agreement”.**
- 4.Select the appropriate download for your Operating System.
- 5.Follow Oracle steps to install.



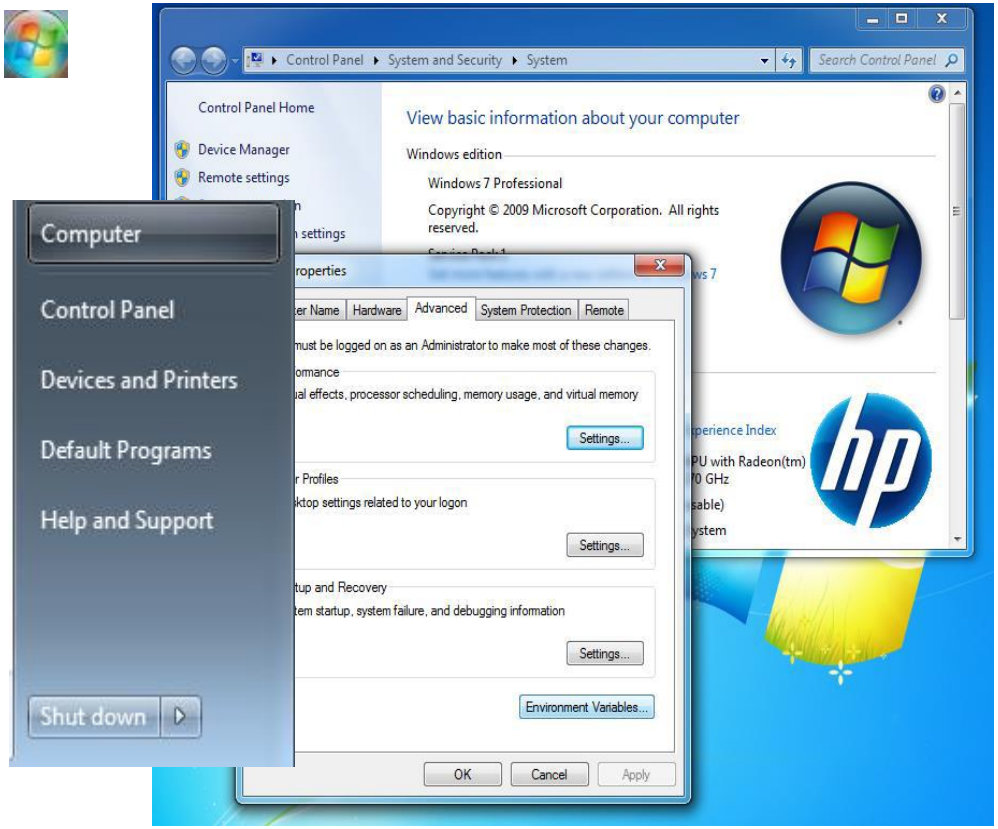
How do I set-up Java JRE for Windows? Cont.

Prior to installing TalendStudio you will have to set the JAVA_HOME and JRE_HOME environment variables:

1. Go to the **Start Menu and Right-Click on "Computer" then select "properties".**

2. In the left Pane, click on "**Advanced system settings".**

3. In the popup, click on "**Environment Variables".**



Talend Studio - Installation

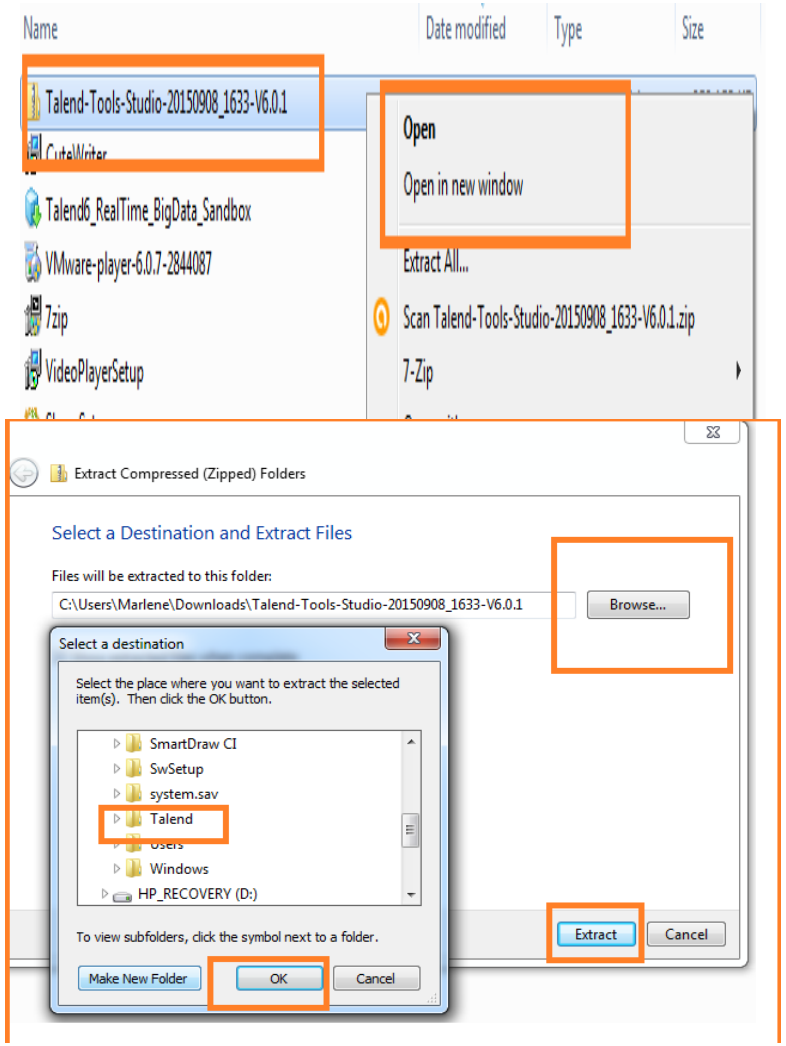
- **Follow the steps below to download TalendStudio:**

1. The top of the screen will display a message **“Download will start in a few seconds”**
2. A message window will appear asking “what do you want to do with Talend-Tools-Studio…zip”, Click on **Save as and save to your local drive C:\TalendDemo**
3. A message window will display when installation file download is complete. Next click on **open.**

How do I Install TalendStudio?

- **To install TalendStudio, follow the steps below:**

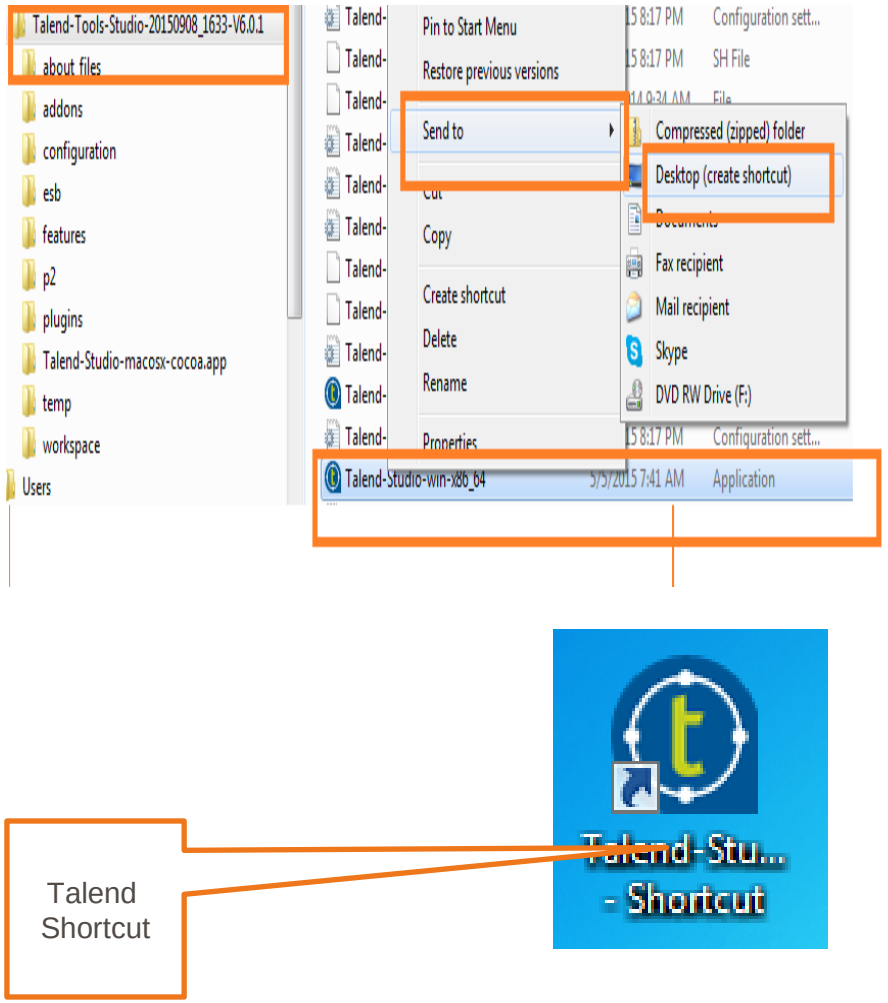
1. Navigate to your local folder, locate and unzip the TalendStudio zip file by right clicking on the compressed file and select, **Extract All.**
2. Click on **Browseand navigate to the C:drive. Select, Make new folderand Name the folder "Talend". Then click OK. Click on "Extract" to begin the installation.**



How do I Install TalendStudio?

- **After all files are extracted you are now ready to open Studio:**

1. Navigate to C:\Talend\Talend-Tools-Studio... and locate the Application file that corresponds to your operating system.
2. Once you locate the Application file, right-click on the file and "Send to> Desktop" to create a shortcut on your desktop. Navigate to your desktop and "double-click" the Studio Icon to start Studio.

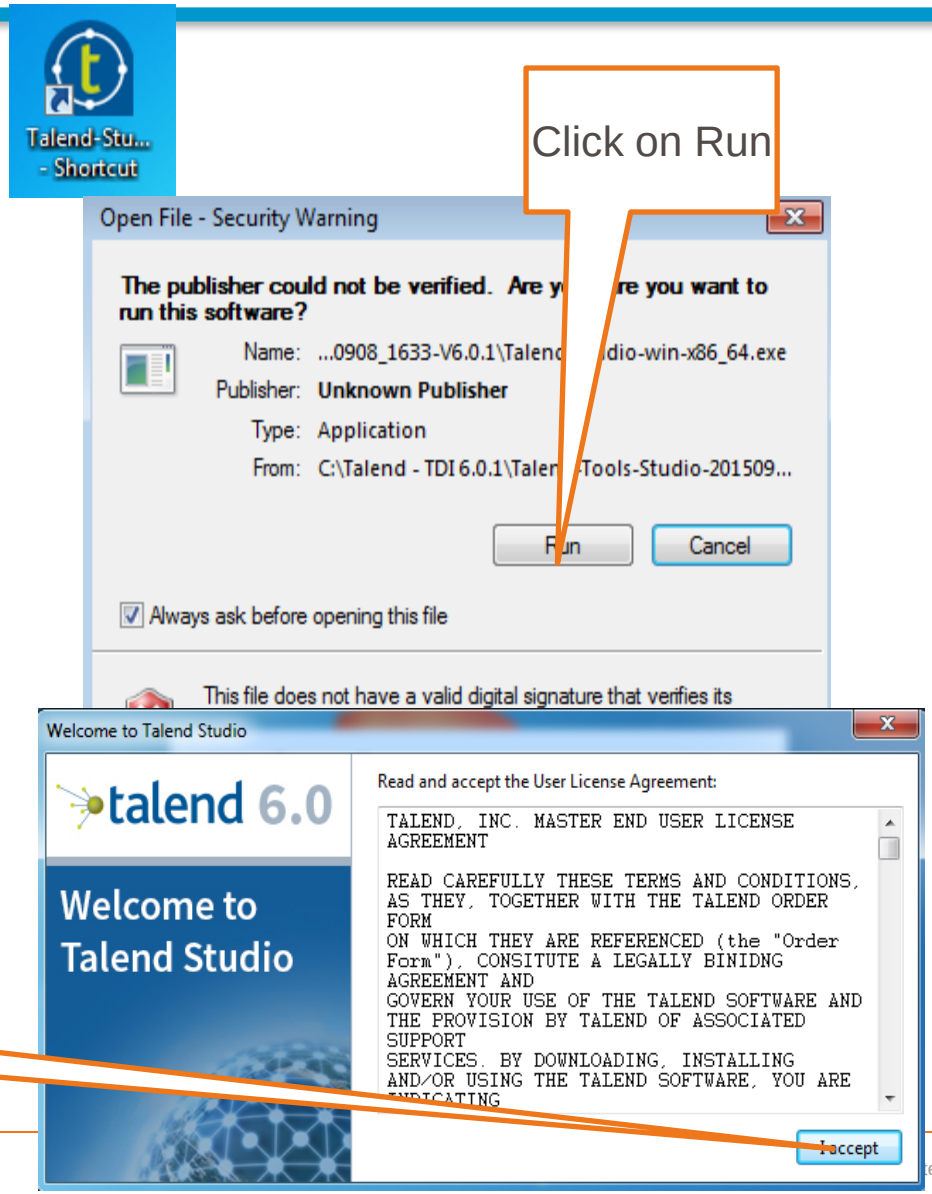


How do I Install TalendStudio? (Cont.…)

- **To install TalendStudio, follow the steps below:**

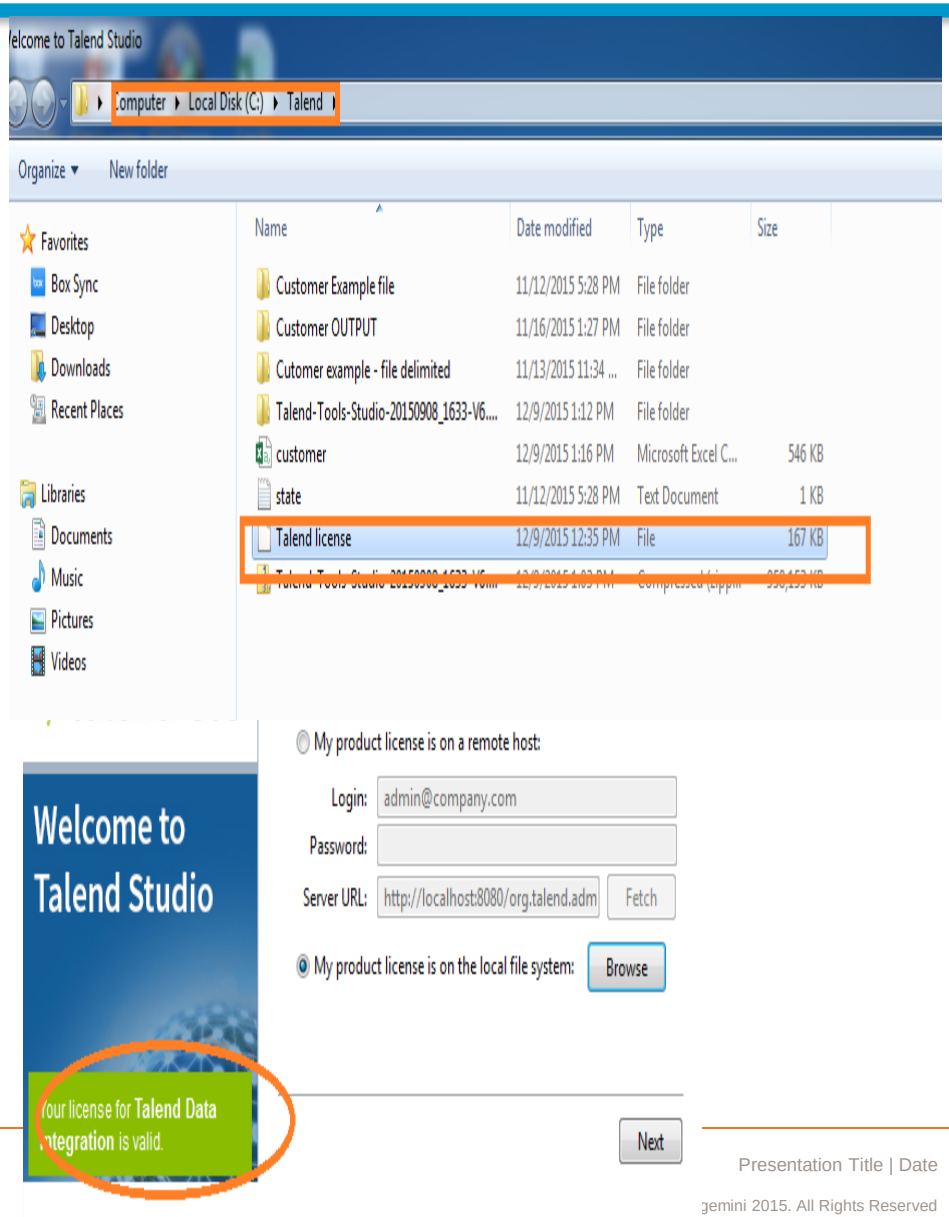
3. In the popup, uncheck the "Always run before opening the file," Click **Run.**

4. Read and accept the User license agreement. Click on**I accept.**



How do I Install TalendStudio?

1. In the Welcome window, Select radio button, “my product license is on the local file system”, then click **browse.**
2. Find and select the license key on your local C drive, then click on **open.**
3. A message will display in green box stating “your license for TalendData Integration is valid”, click **next to load your license.**

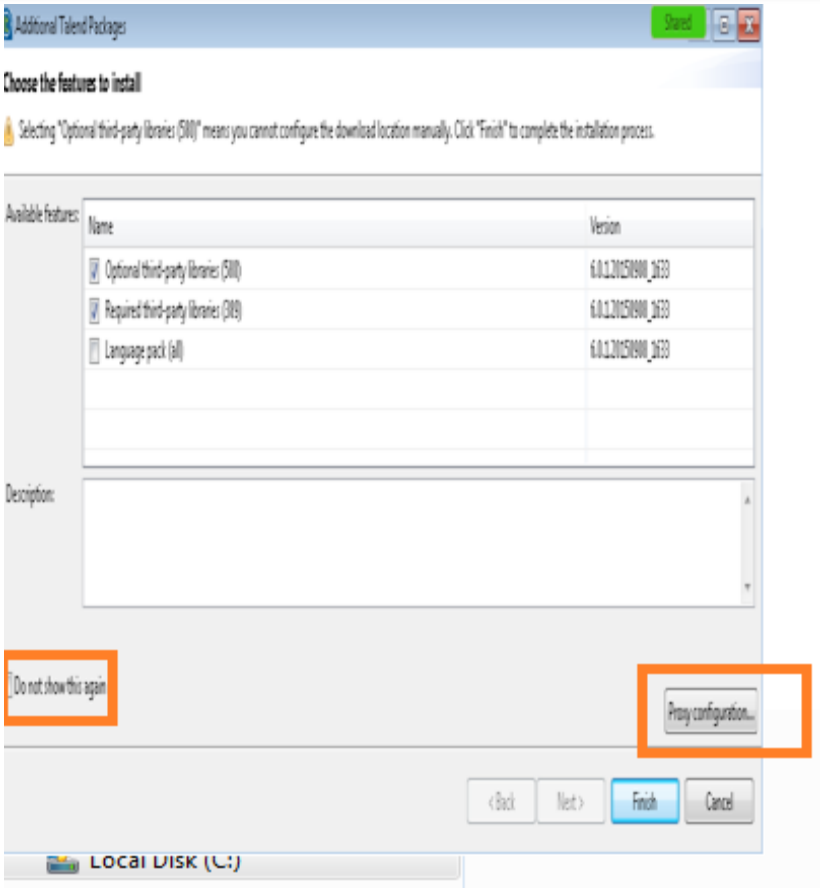


Welcome to TalendStudio?

- **You are now ready to set up your project. In the TalendStudio Login window Select an option to define your project:**
- 1.Select **Create a new Project,name it** “TDI_Cookbook” specify a project name and click Finish. (project name can not have spaces)
- **Other options:**
- Select **Import a demo project and click Finish** to import a demo project that includes numerous samples of ready-to-use Jobs.
- Select **Import an existing project and click Finish** to import an existing projects stored locally.

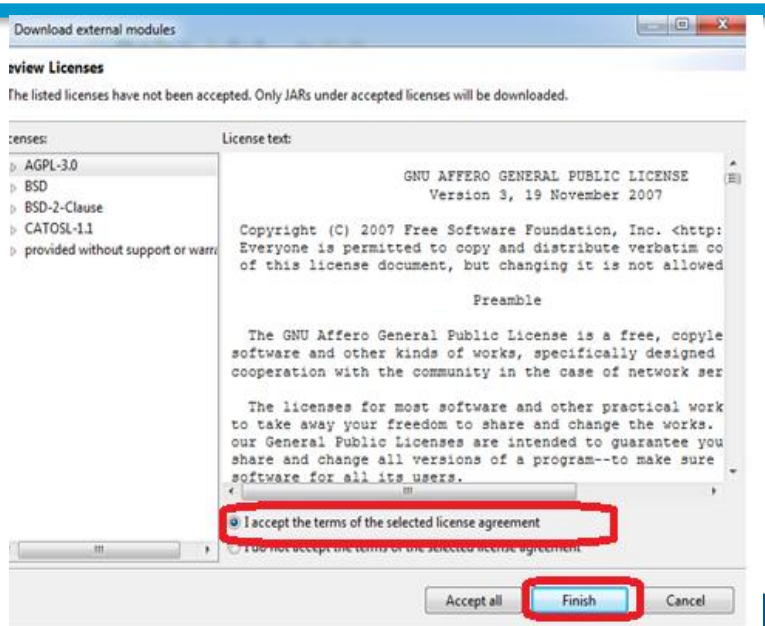
Welcome to TalendStudio?

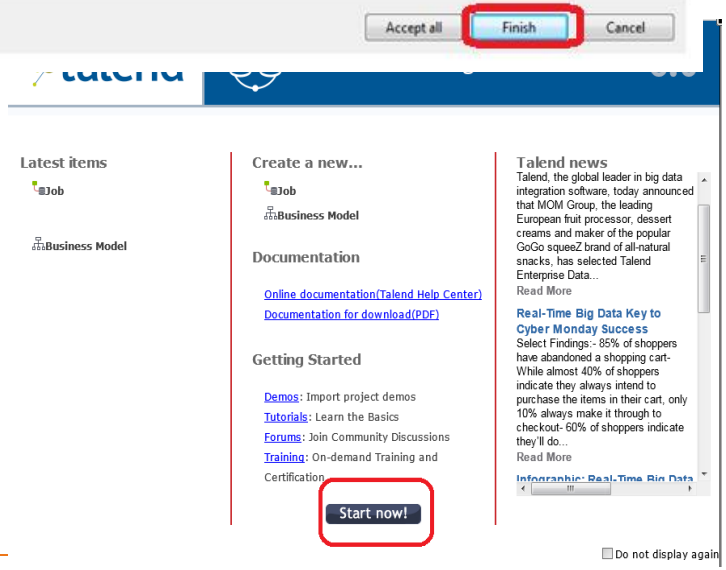
- When you first log into TalendStudio you will receive messages about “Additional TalendPackages”, the packages recommended are selected by default.
- **(optional)**
- Talend offers various languages, place a check in the box next to Language pack to install a language other than English.
- 1.Place a check in the box next to "Do not show this again“.
- 2.Click**Finish to install all features.**



Welcome to TalendStudio?

- Click on radio button "**I accept the terms of the selected license agreement**"
- 2.Next click **Accept All.**
- The TalendStudio will open to a **Welcome Page, which you can use to quickly launch new Jobs, analyses, or Business Models.**
- 3.Click on **Start Now.**





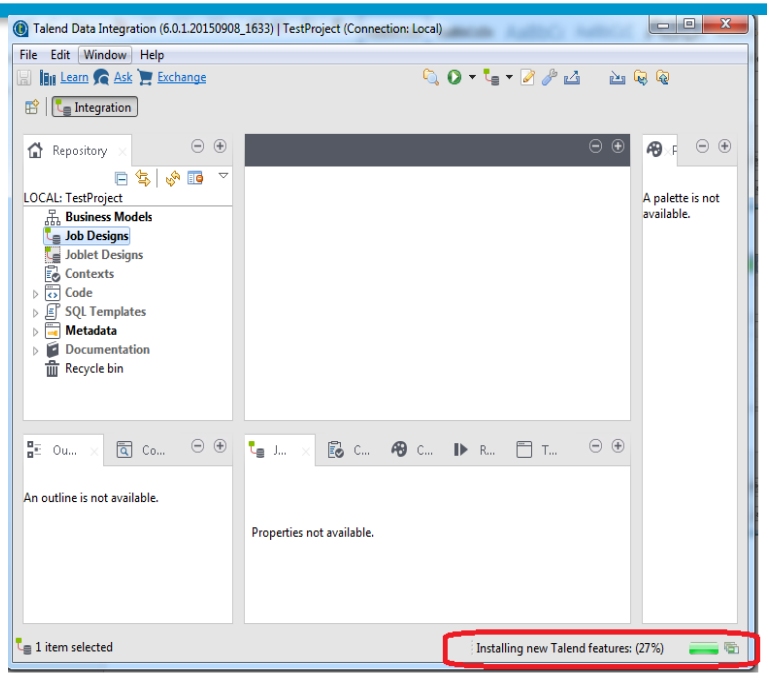
Talend Data Integration Overview

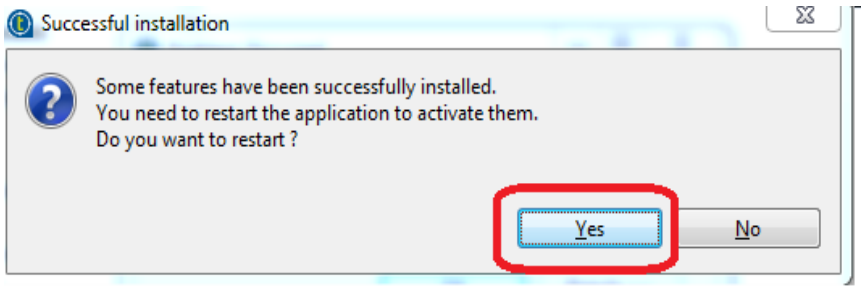
Module Outline

- Talend Studio Integration Overview
  - Welcome to TalendStudio?
  - TalendStudio –Repository
  - TalendStudio –Component Palette
  - TalendStudio –Design Window
  - TalendStudio –Component Configuration
  - TalendStudio –Run Window
  - TalendStudio –Metadata Repository

Welcome to TalendStudio?

- The bottom right corner of the page will show a progress bar indicating that Talendis installing new features.
- Wait for installation to complete prior to starting a job design.
- Once installation is complete you will receive a “successful installation” message. You will need to restart the Studio application by clicking, **Yes.**





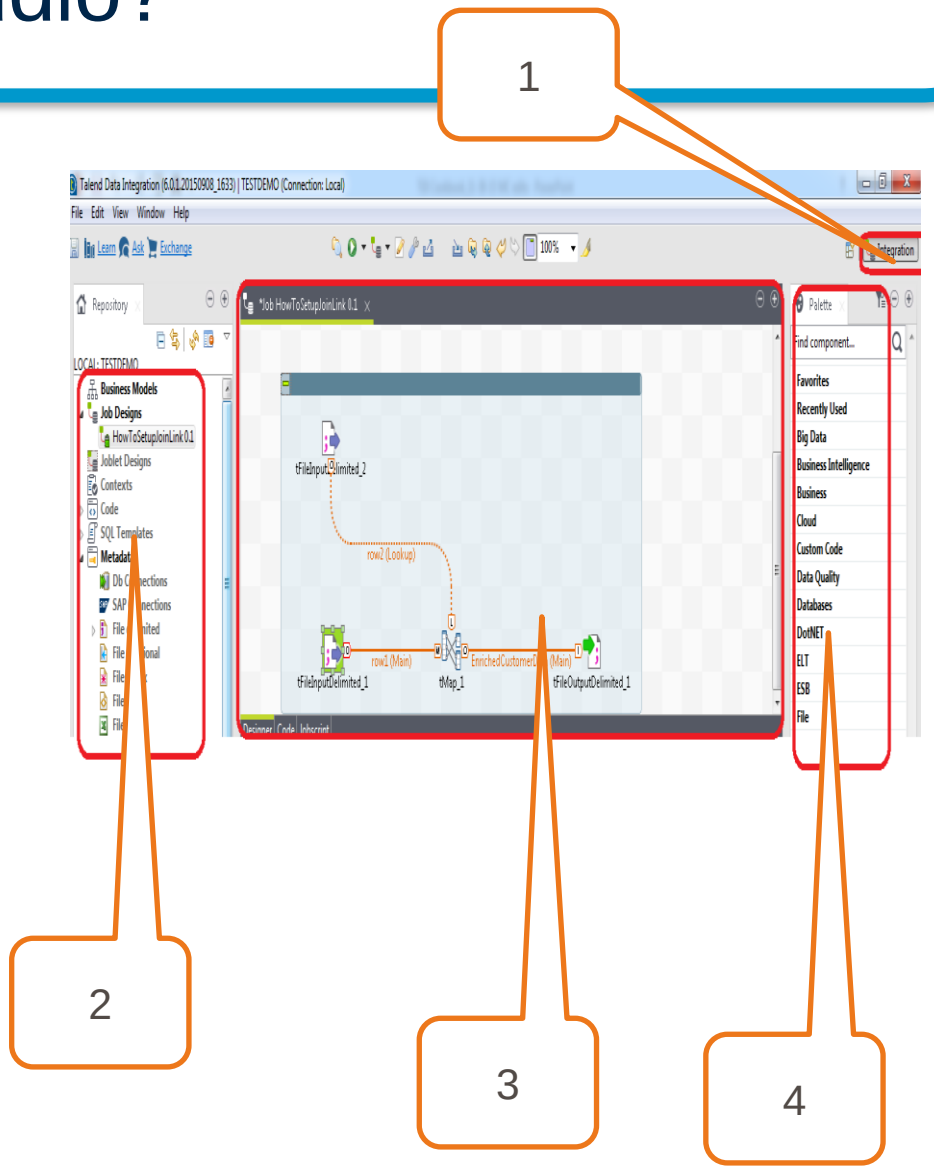
Welcome to TalendStudio?

1. If the Studio does not look like the image displayed here make sure to click on the **Integration perspectivetab in the upper right hand corner.** TalendStudio consists of **three major parts:**

2. **Repository-listing all the DI artifacts in TalendStudio.**

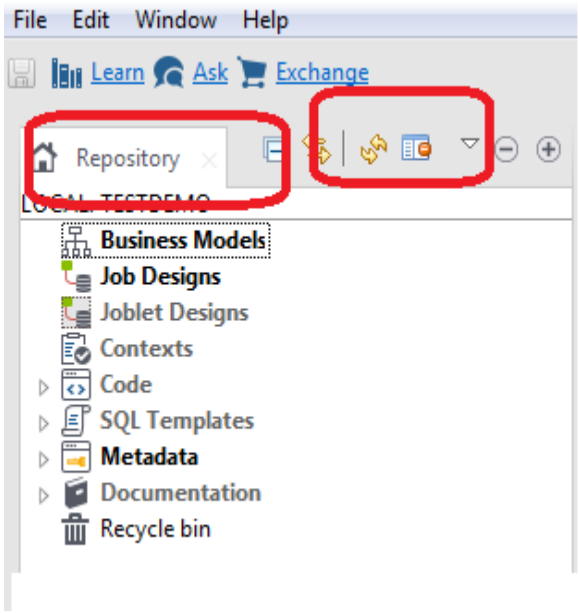
3. **Job Designer -where Jobs are designed and components are configured.**

4. **Component Palette–providing 800+ components and connectors**



TalendStudio –Repository

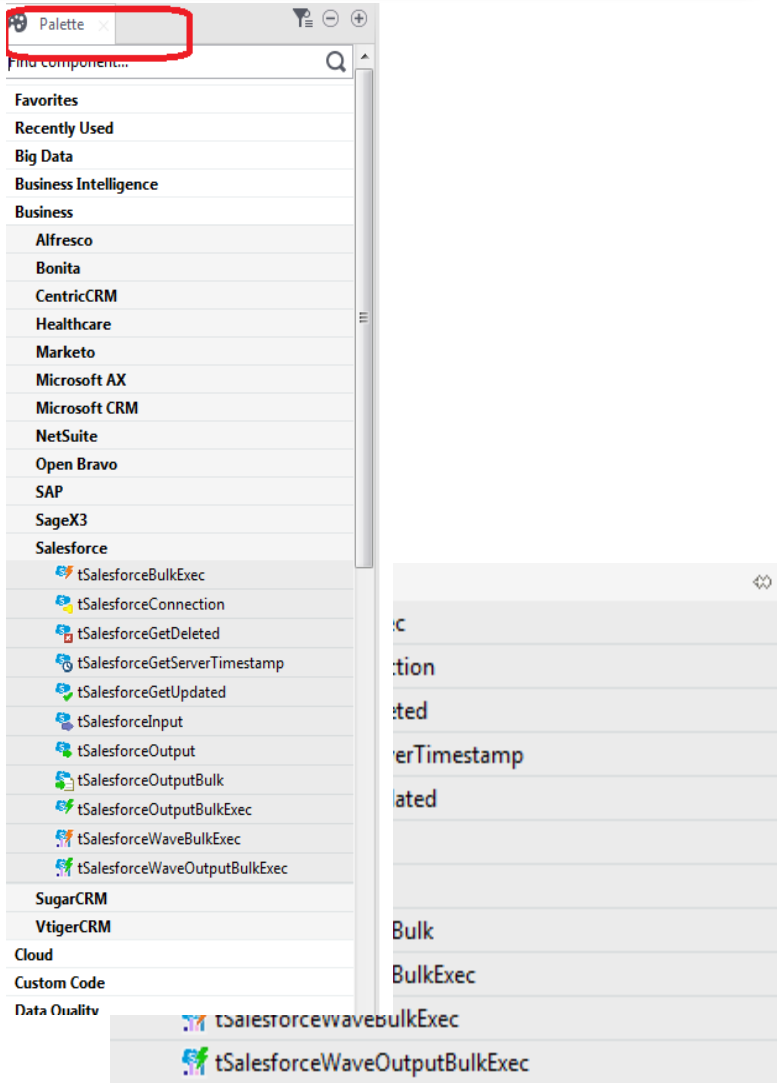
- The Repository tree view gathers all the technical items that can be used either to describe business models or to design Jobs. It gives access to any item including Business Models, Job Designs, as well as reusable routines or documentation.
- The Repository centralizes and stores all necessary elements for any Job design and business modeling contained in a project.
- This display illustrates the elements stored in the Repository.
- The Refresh button allows you to update the tree view with the last changes made
- The Activate filter button allows you to open the filter settings view so as to configure the display of the Repository view.
- The Switch branch button is displayed when your Studio is connected to a remote project. It allows you to switch across project branches without the need of restarting your Studio. For further information, see the Getting Started Guide.
- The Repository review stores all your data (Business, Jobs, Joblets) and metadata (Routines, DB/File connections, any meaningful Documentation and so on).



TalendStudio –Component Palette

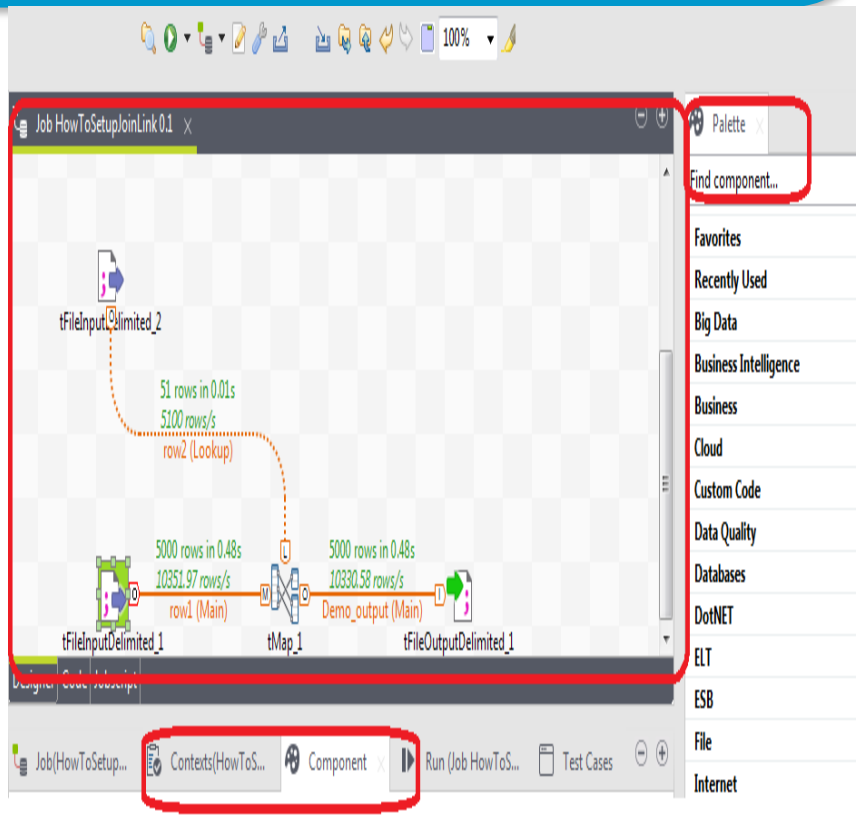
- From the **Palette, depending on whether you are designing a Job or modeling a Business Model, you can drop technical components or shapes, branches and notes to the design workspace for Job design or business modeling.**

800+ Components



TalendStudio –Design Window

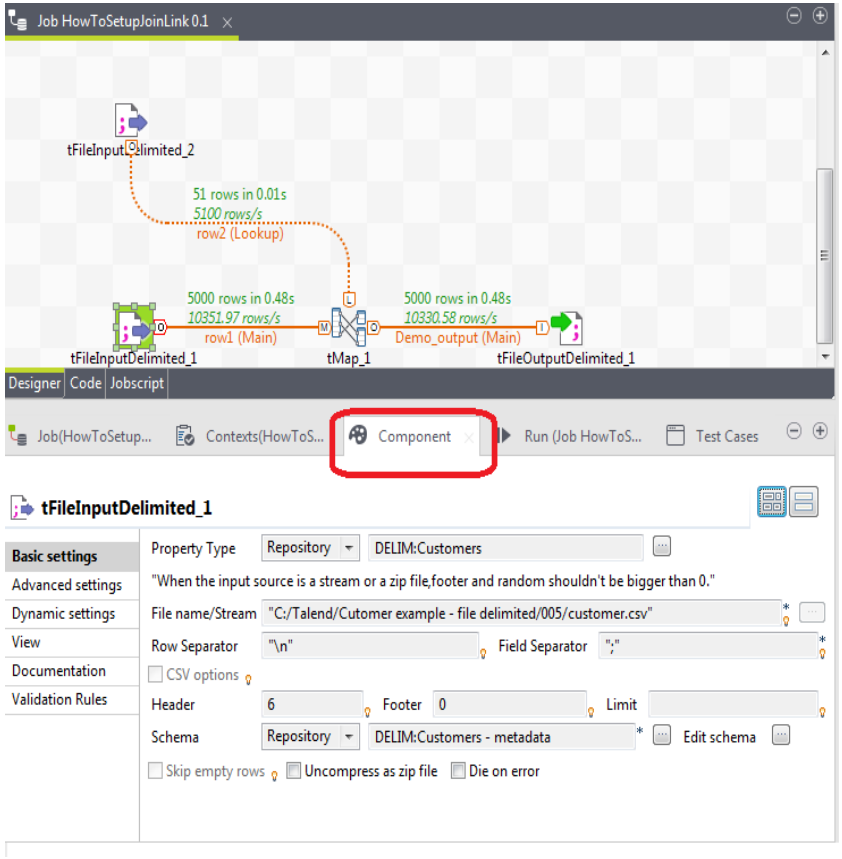
- In the *TalendStudio's design workspace, both Business Models and Job Designs can be laid out.*
- For both Business Models and Job Designs: active designs display in a easily accessible tab system above this workspace.
- For Job Designs only. Under this workspace, you can access several other tabs:
- the **Designertab.** It opens by default when creating a Job. It displays the Job in a graphical mode.
- the **Codetab.** It enables you to visualize the code and highlights the possible language errors.
- the **Jobscriptenables you to visualize and edit the Jobscript**



- A **Paletteis docked at the top of the design workspace to help you draw the model corresponding to your workflow needs.**

TalendStudio –Component Configuration

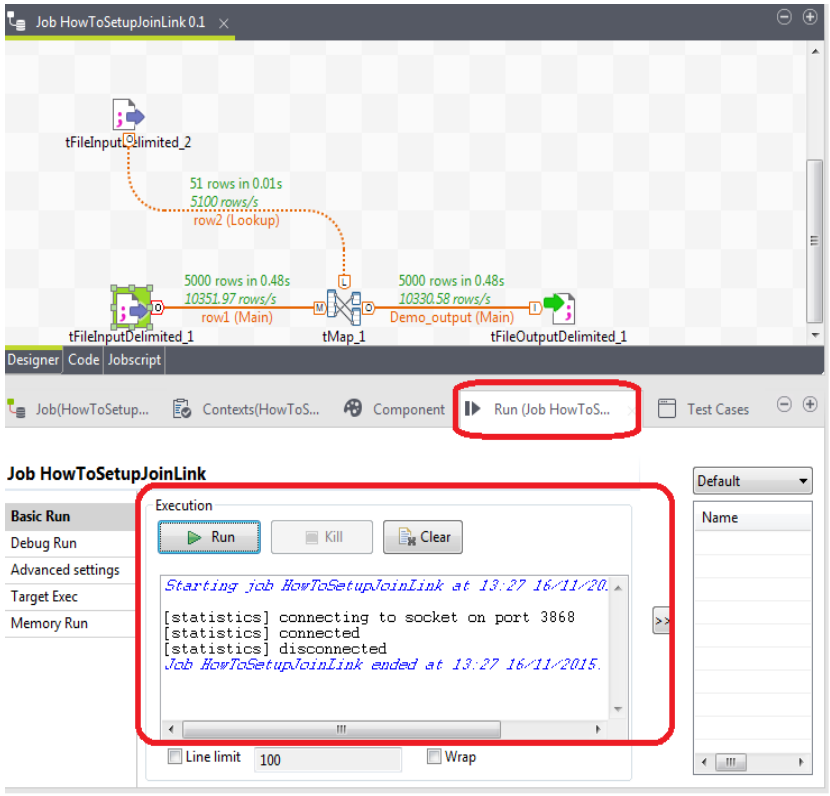
- The configuration tabs are located in the lower half of the design workspace. Each tab opens a view that displays the properties of the selected element in the design workspace. These properties can be edited to change or set the parameters related to a particular component or to the Job as a whole.
- The **Component** view gathers all information relative to the graphical elements selected in the design workspace.



TalendStudio –Run Window

- The **Run view** is used to execute your job after all components are configured.

- The **Execution window** within the run view displays the progress of the execution. The log includes any error message as well as start and end messages. It also shows the Job output in case of a tLogRowcomponent is used in the Job design.



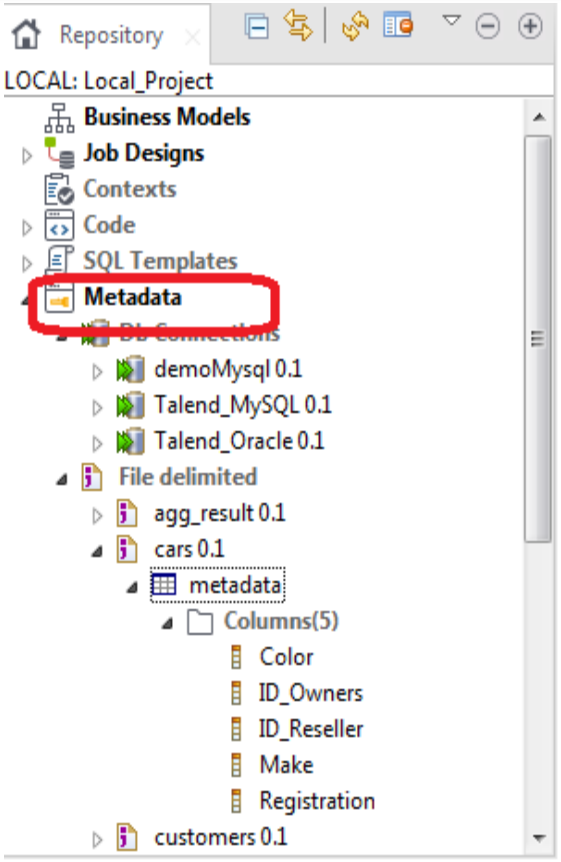
TalendStudio –Metadata Repository

- Click **Metadatain** the Repositorytree view to expand the folder tree. Each of the connection nodes will gather the various connections and schemas you have set up.

- You can create and manage various metadata items in the **Repository** that can be used in all your Job designs**.**

- The **Metadata** folder in the Repository tree view stores reusable information on files, databases, and/or systems that you need to create your Jobs.

- Various corresponding wizards help you store these pieces of information that can be used later to set the connection parameters of the relevant input or output components and the data description called "schemas" in a centralized manner in *TalendStudio.*



Building First Job in Talend

Module Outline

- Building First Job in Talend
- Data Integration Basics
- Building First Job – Demo steps
- Talend Data Integration Demo – Metadata Creation
- Job Design – Linking from source to map to target
- Configuring Map component
- Configuring Output
- Running Job
- Job Documentation

Data Integration Basics

- The fastest and the most cost effective way to connect data.
- Robust data integration in an open and scalable architecture
- Provides unified tools to integrate, cleanse, mask and profile all of the data
- Over 900+ prebuilt components to connect various data sources
- Offers collaboration and management tools
- Synchronize metadata
- Deliver self-service data preparation to everyone

Building First Job – Demo steps

Step 1: Read Source - See how the "Delimited File" wizard within Talend Studio can help you deal with complex file formats. You can create specific Schemas for all your needs.

Step 2: Enrich and Transform Data : The tMapcomponent will be used to enrich your source data with lookups and data transformations.

Step 3: Write Target: Output your enriched data to any number of targets.

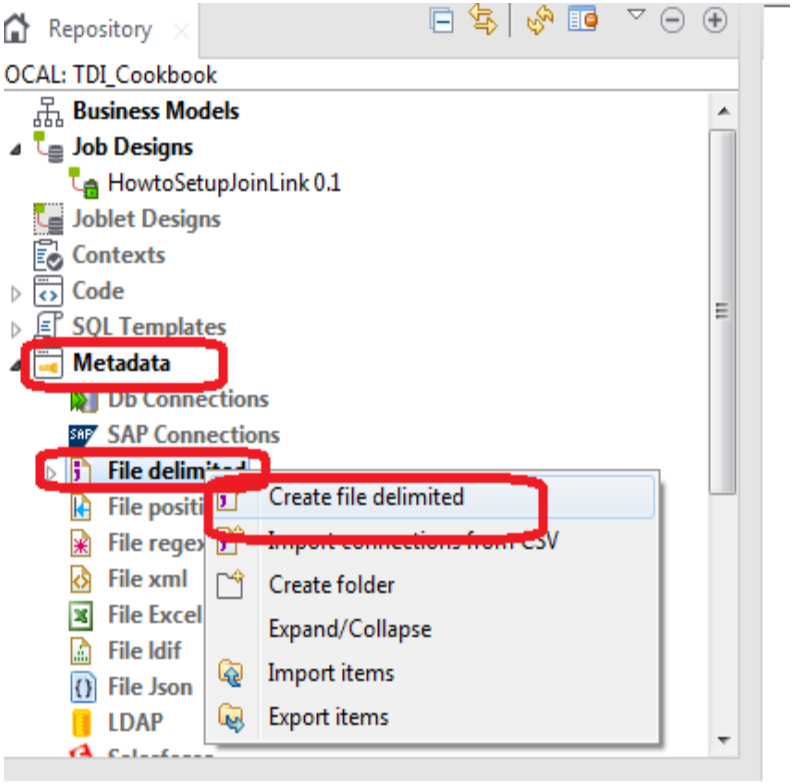
Talend Data Integration Demo – Metadata Creation

- **In the Repository on the left of the TalendStudio main screen:**

1. Expand the **Metadatanode.**

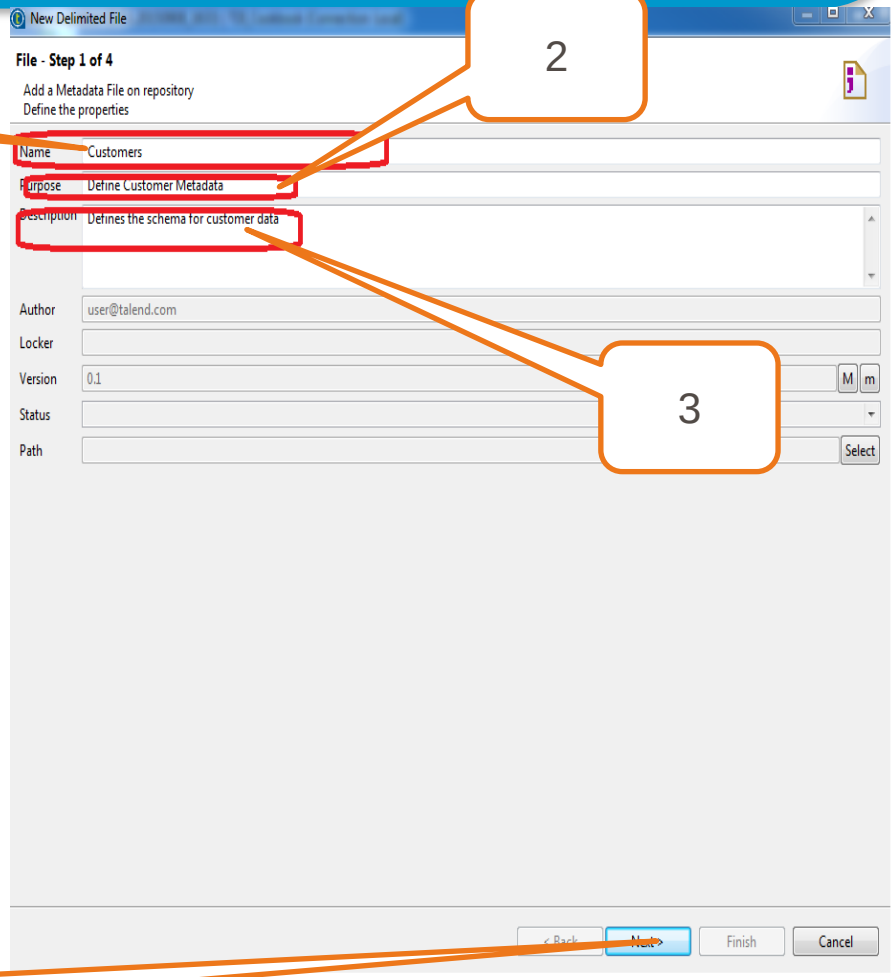
2. Right-click on **File delimited.**

3. In the menu, click **Create file delimited to open the New Delimited Filewizard.**



Talend Data Integration Demo – Metadata Creation (Cont…)

1. In the **Name field, name the metadata *customers.***
2. Add a **Purpose. Example: "Define Customer Metadata"**
3. Add a **Description. Example: "Defines the schema for customer data"**
4. Click **Nextto continue.**

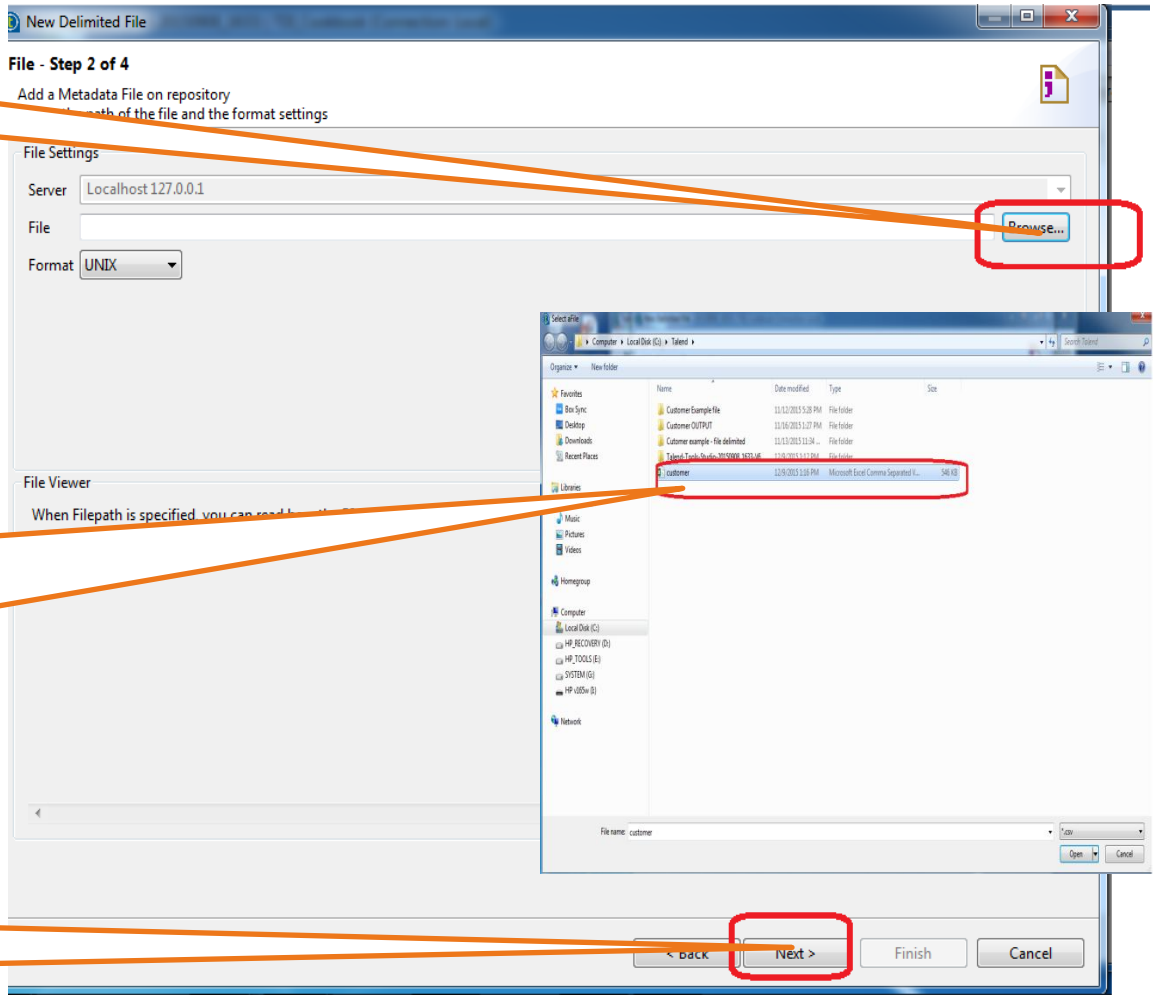


Talend Data Integration Demo – Metadata Creation (Cont…)

Click **Browse**

Select the ***customer.csvfile*** from your computers local drive and Open.

Click on Next

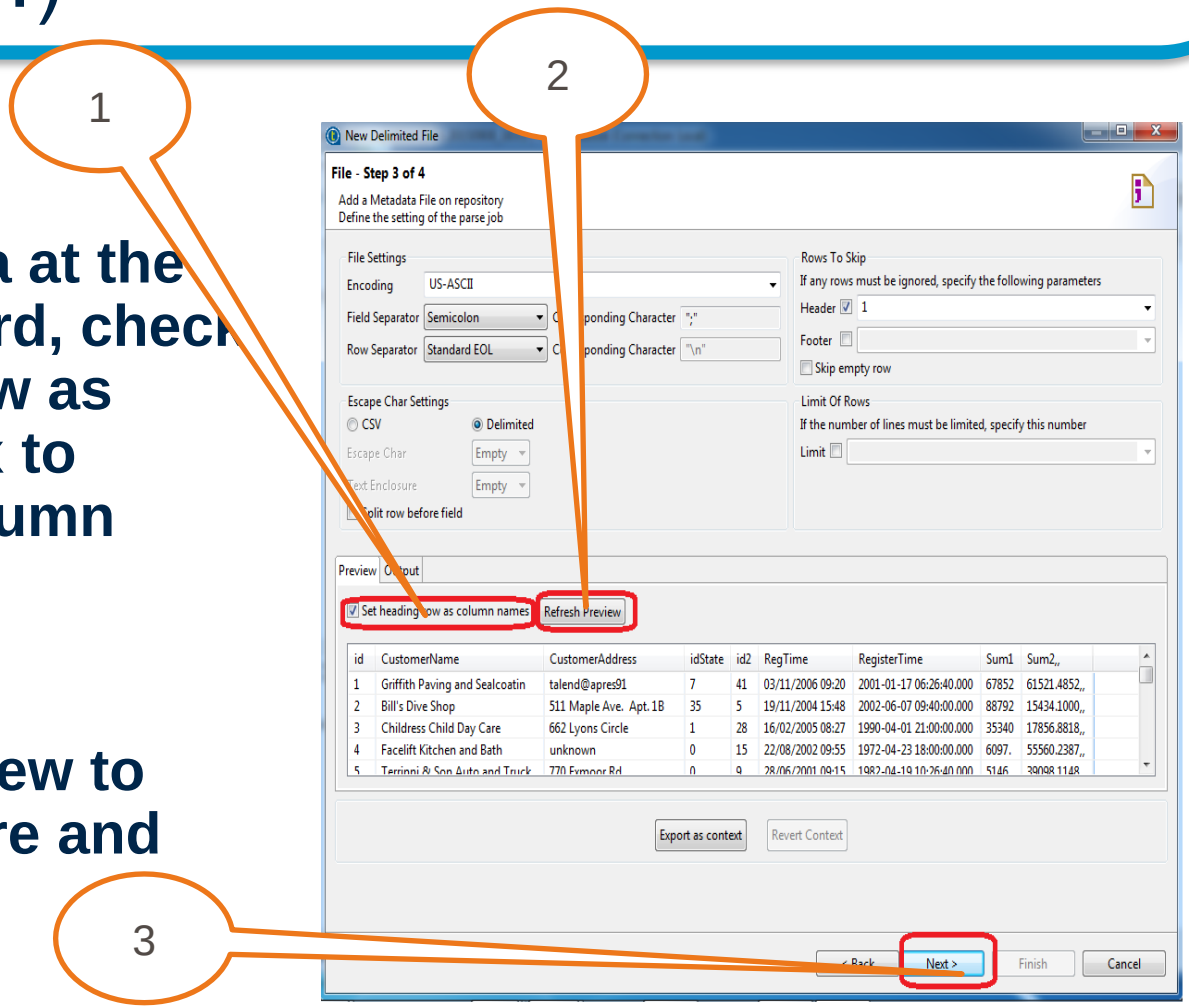


Talend Data Integration Demo – Metadata Creation (Cont…)

1. In the **Preview area at the bottom of the wizard, check the Set heading row as column names box to retrieve the file column names.**

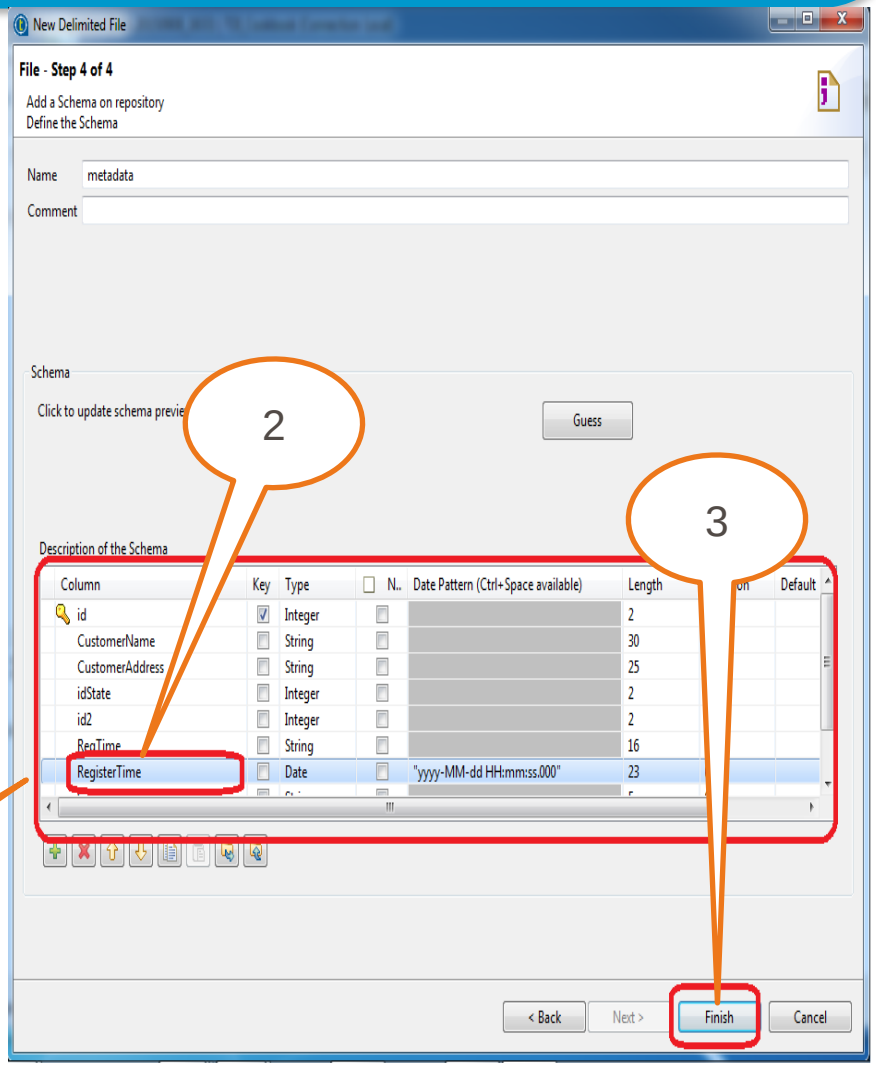
2. Click **Refresh Preview to update the structure and data preview.**

3. Click **Next.**



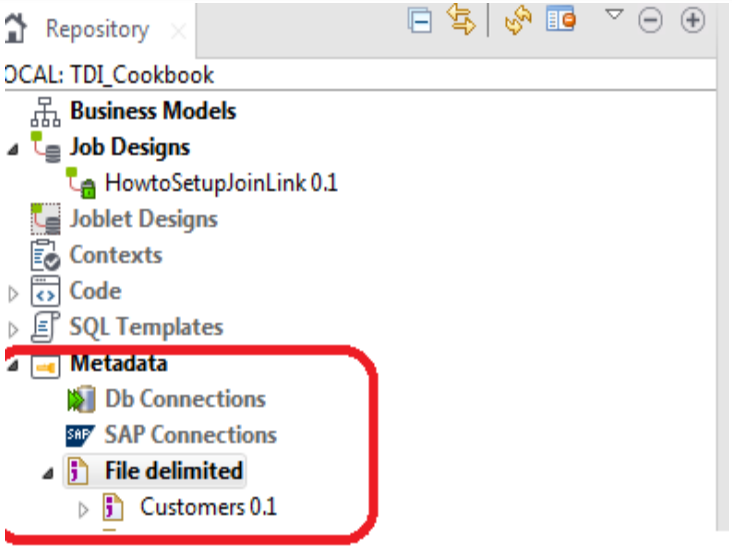
Talend Data Integration Demo – Metadata Creation (Cont…)

1. In the **Description of the Schema table,** set the columns as shown in the screenshot.
2. In the **RegisterTime** Column, verify the date pattern matches the date format as specified in the file. Date format should be specified as: “yyyy-MM-ddHH:mm:ss.000” (quotes should be included)
3. Click **Finish to** close the wizard.



Talend Data Integration Demo – Metadata Creation (Cont…)

- The *customersmetadata is displayed in the* ***Metadata> File delimited node.***

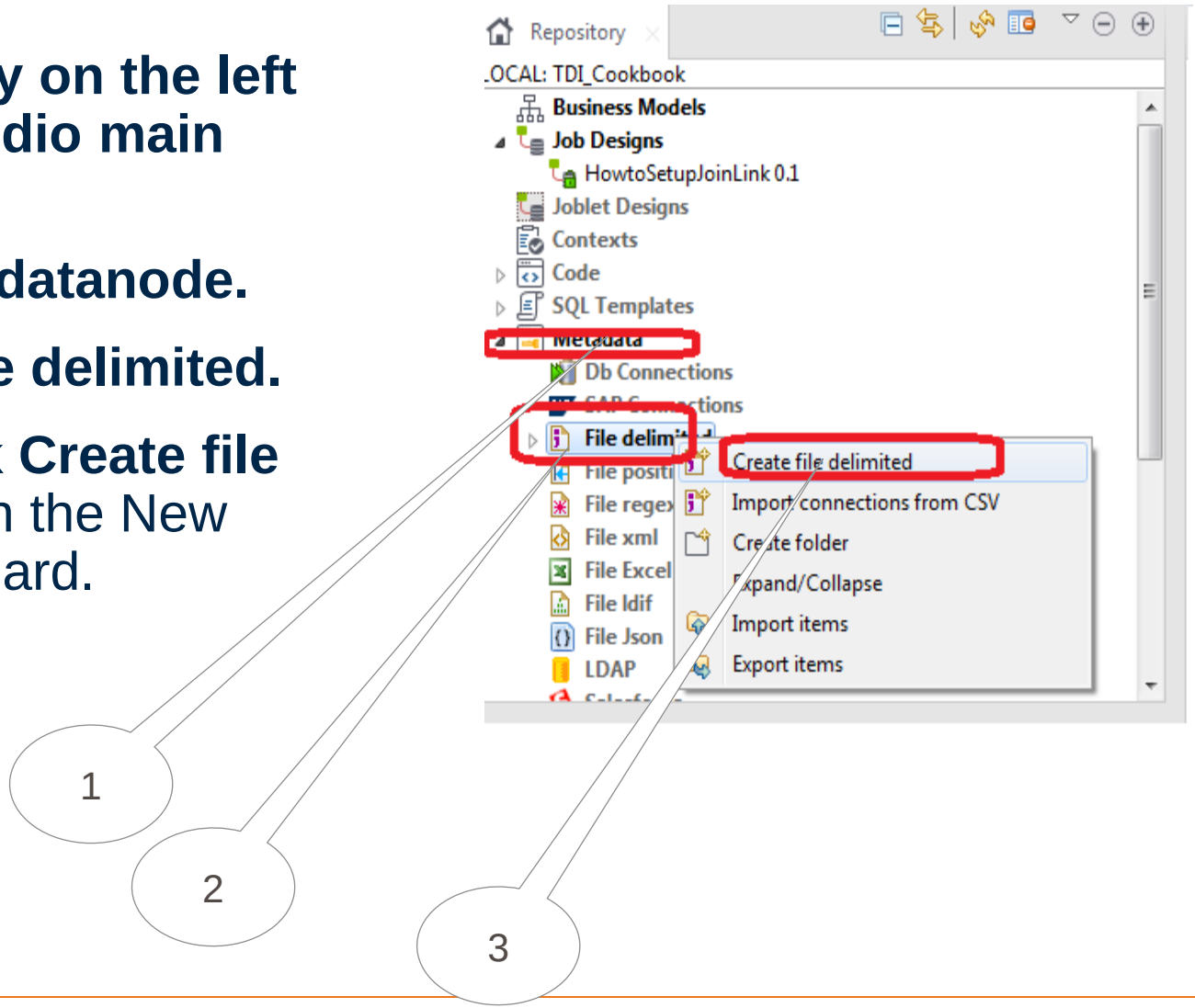


- The **customers metadata is created!**

Talend Data Integration Demo – Metadata Creation (Cont…)

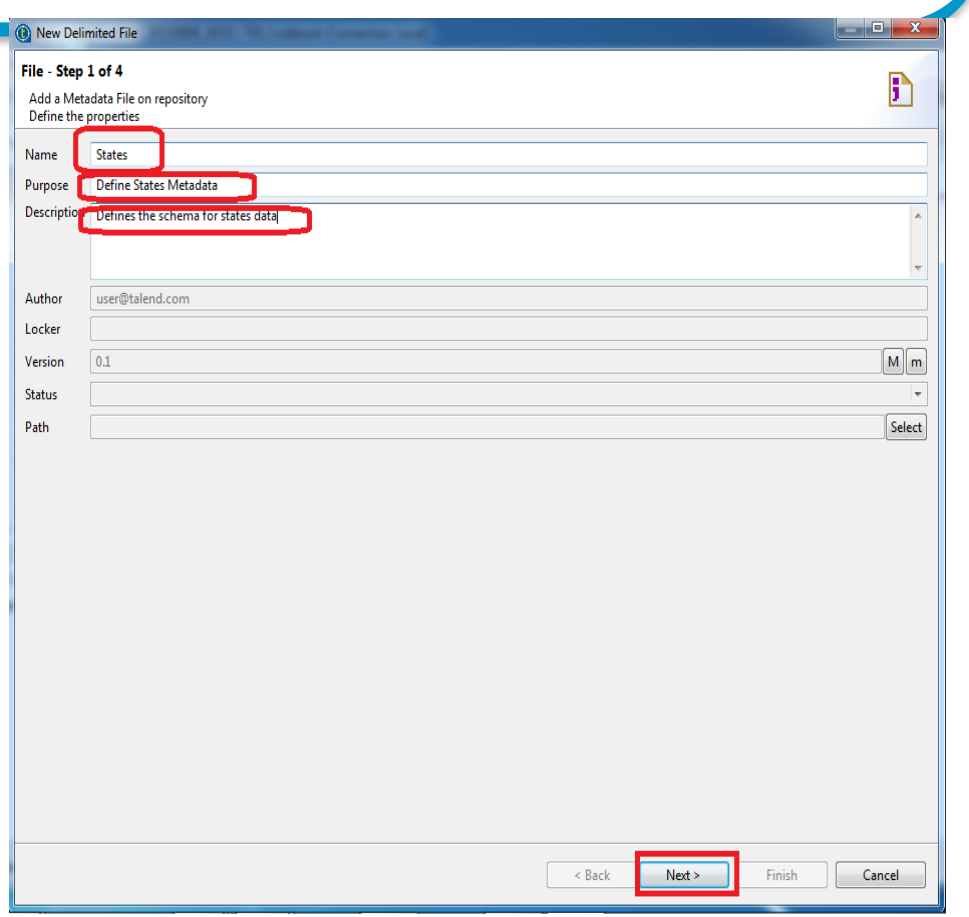
- **In the Repository on the left of the TalendStudio main screen:**

1. Expand the **Metadatanode.**
2. Right-click on **File delimited.**
3. In the menu, click **Create file delimited** to open the New Delimited File wizard.



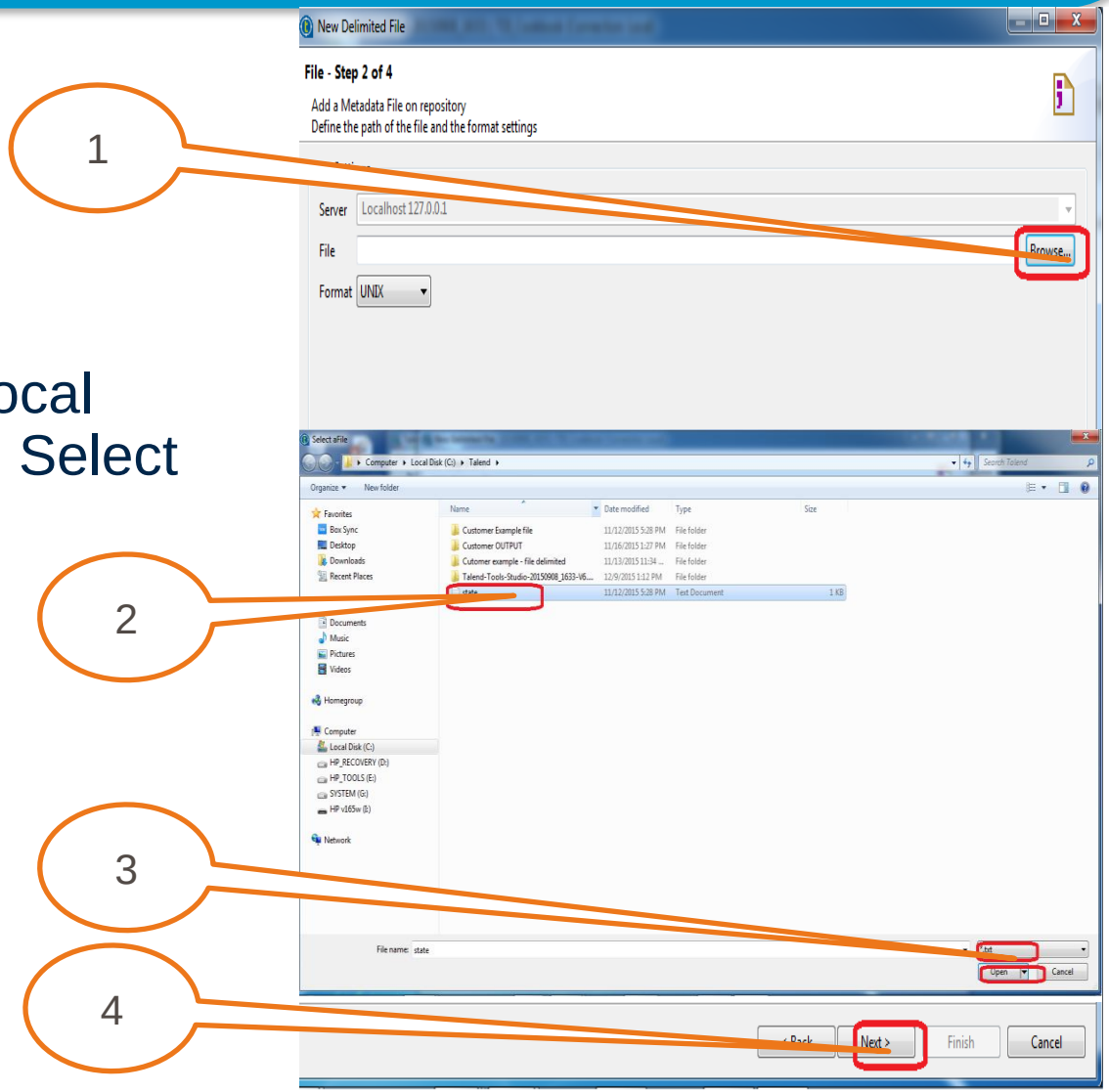
Talend Data Integration Demo – Metadata Creation (Cont…)

1. In the **Namefield, name the metadata *states.***
2. Add a **Purpose.** Example: "Define States Metadata"
3. Add a **Description.** Example: "Defines the schema for states data“



Talend Data Integration Demo – Metadata Creation (Cont…)

1. Click **Browse**
2. From your computers local drive (C:\TalendDemo), Select the ***state.txtfile.***
3. Click **Open.**
4. Click **Next.**

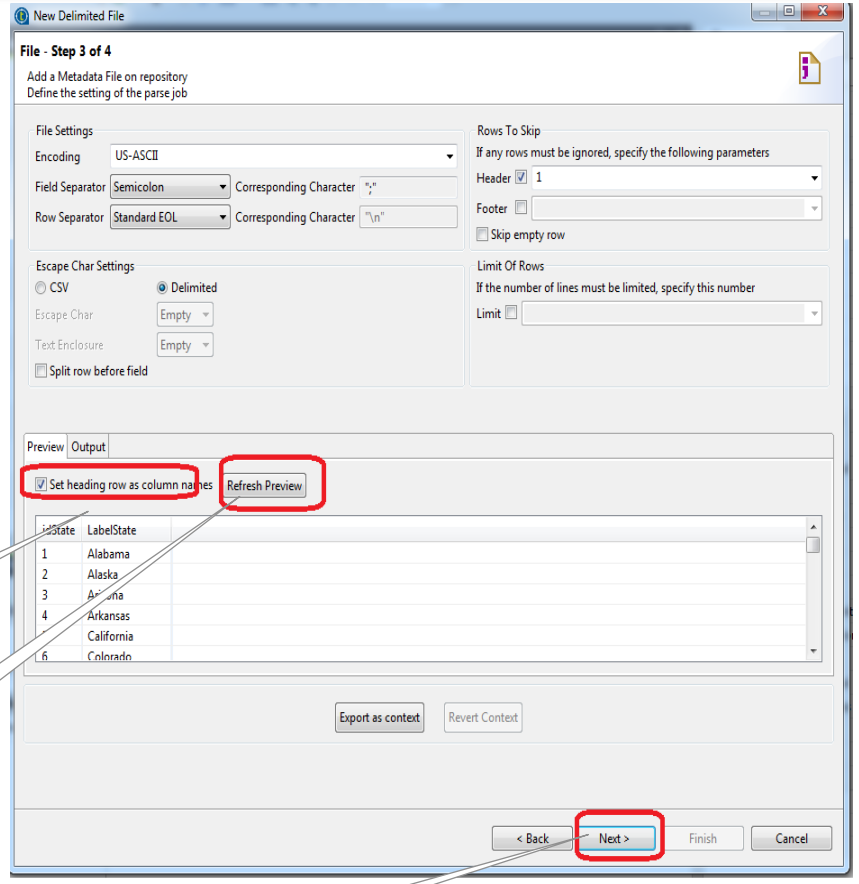


Talend Data Integration Demo – Metadata Creation (Cont…)

1. In the **Preview** area at the bottom of the wizard, check the **Set heading row as column names** box to retrieve the file column names.

2. Click **Refresh Preview** to update the structure and data preview**.**

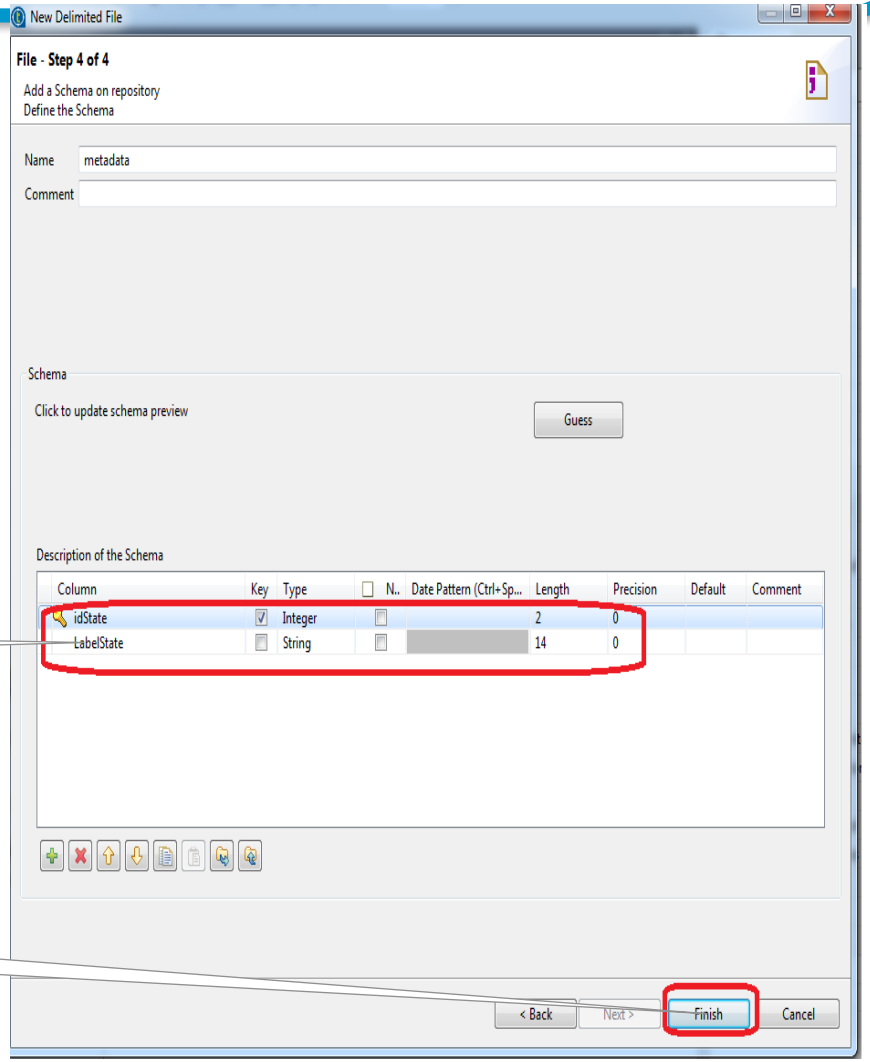
3. Click **Next.**



Talend Data Integration Demo – Metadata Creation (Cont…)

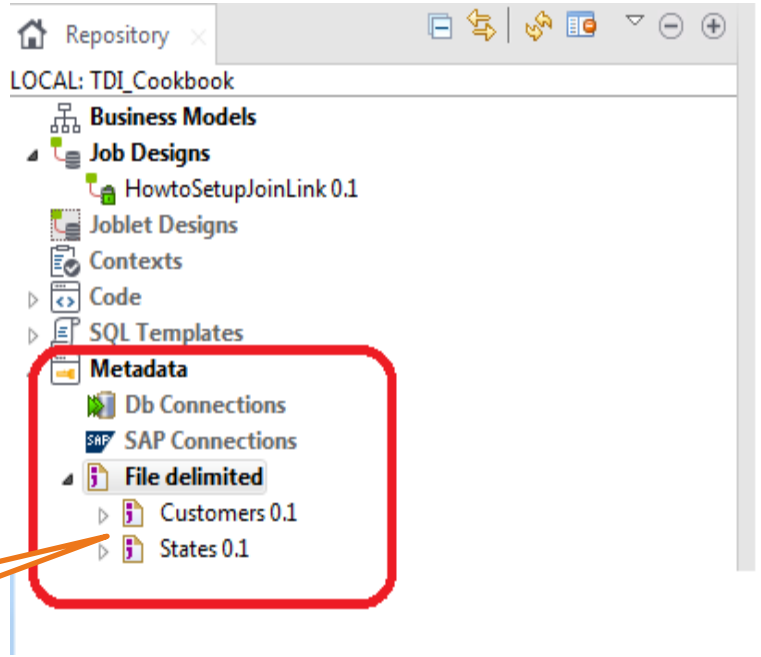
1. In the **Description of the Schema** table, set the columns as shown in the screenshot.

2. Click **Finish** to close the wizard.



Talend Data Integration Demo – Metadata Creation (Cont…)

- The *statesme tadata is displayed in the* ***Metadata> File delimited***



The two metadata files now exist,
and can be used in a Job.

Job Design

1. Right-click on **Job Designs**

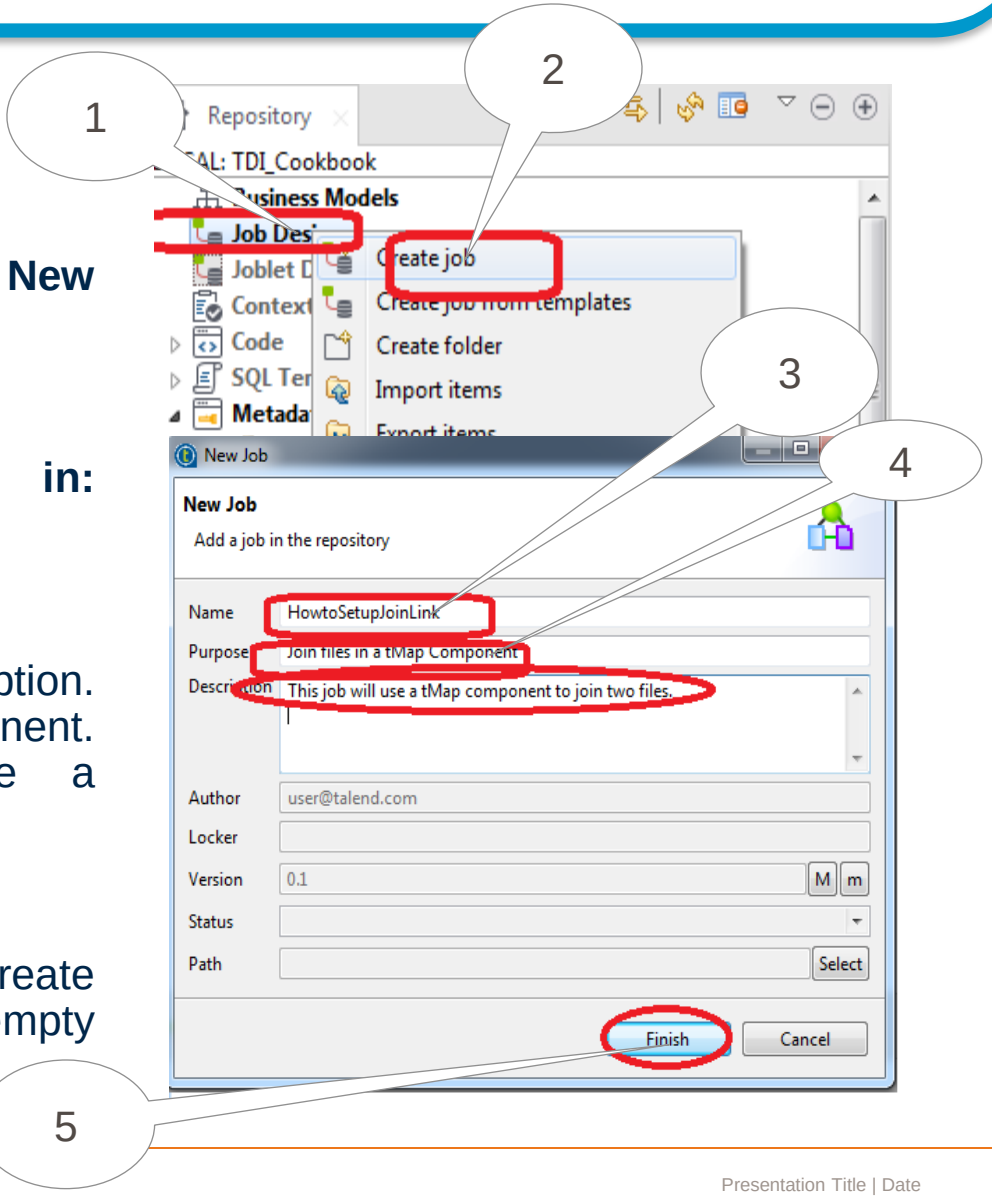
2.Next click **Create Jobto open the New Jobwizard.**

3.In the **Namefield, fill in:** ***HowtoSetupJoinLink***

.

4. Add an appropriate Purpose and Description. **Purpose:** Join files in a tMapComponent. **Description:** This job will use a tMapcomponent to join two files.

5.Click **Finish** to close the wizard and create your Job. The Job Designer opens an empty Job.

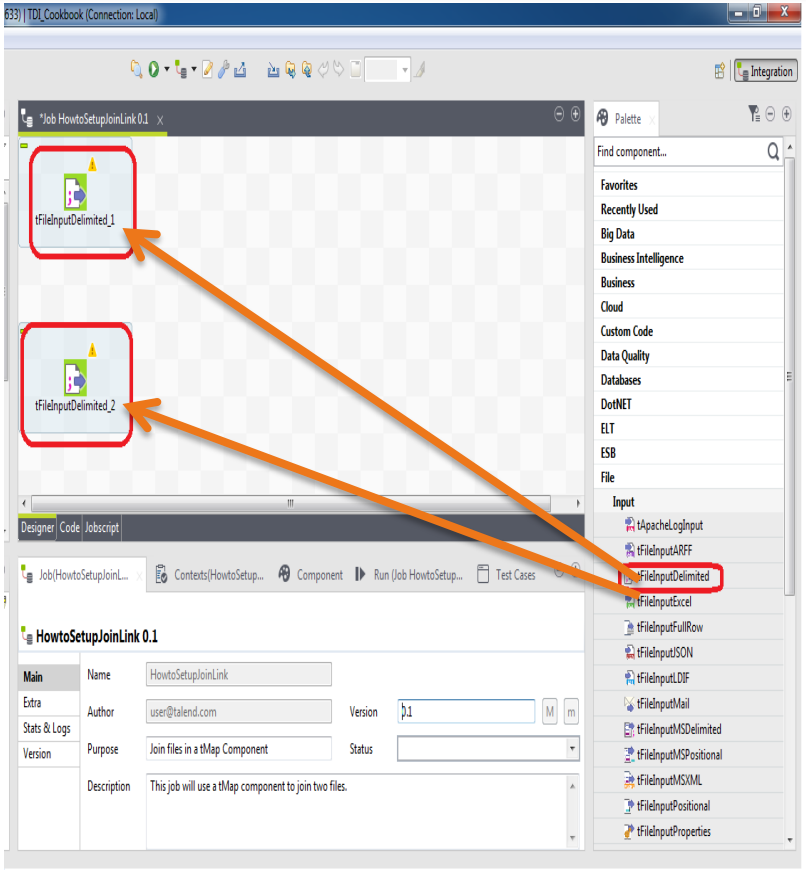


Job Design (Cont…)

1. the **File family and the Input sub-family.**

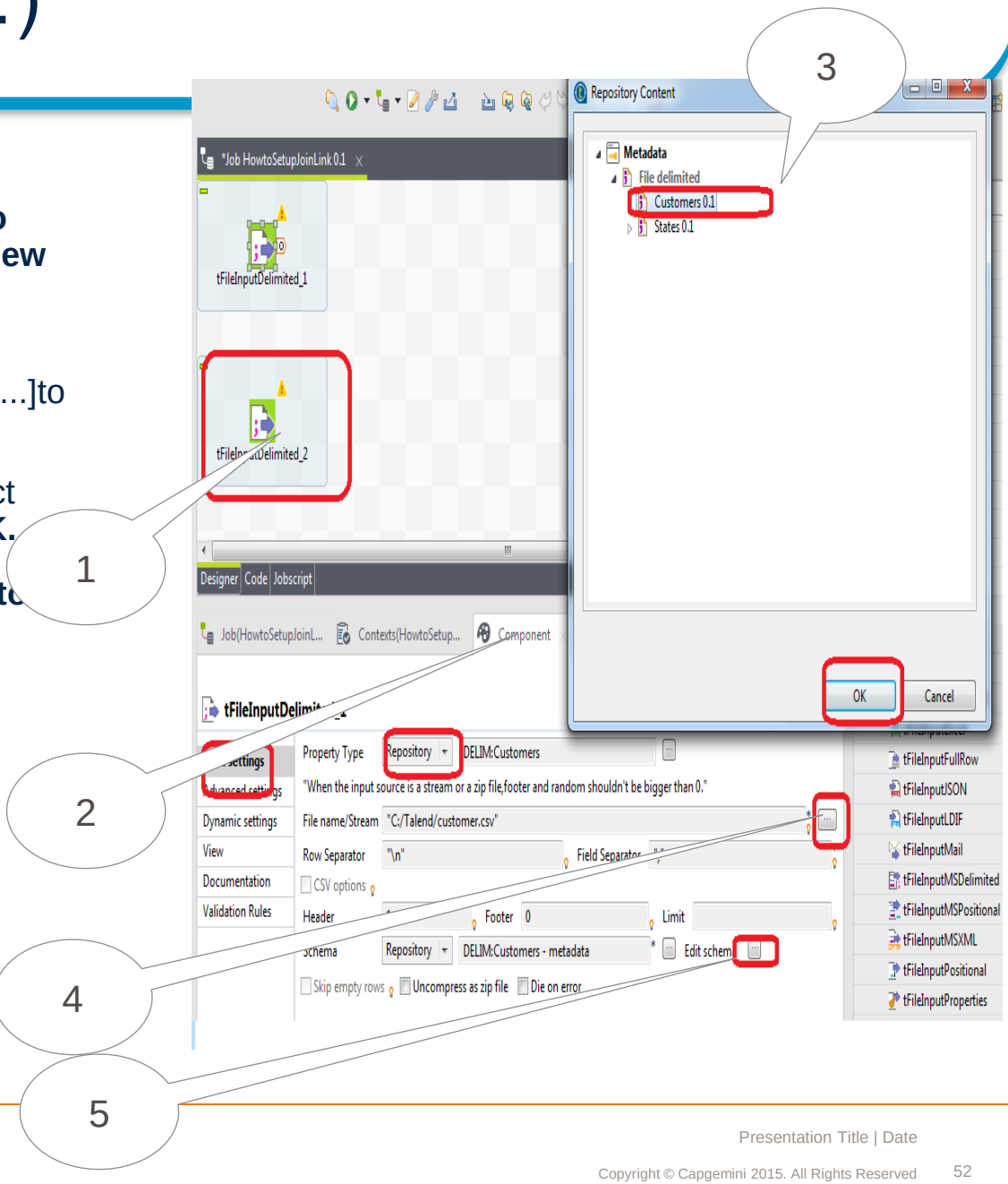
2. Click on the **tFileInputDelimited** component, drag and drop it on the Job Designer.

3. Do the same to add a second **tFileInputDelimited** component. (drag and drop)



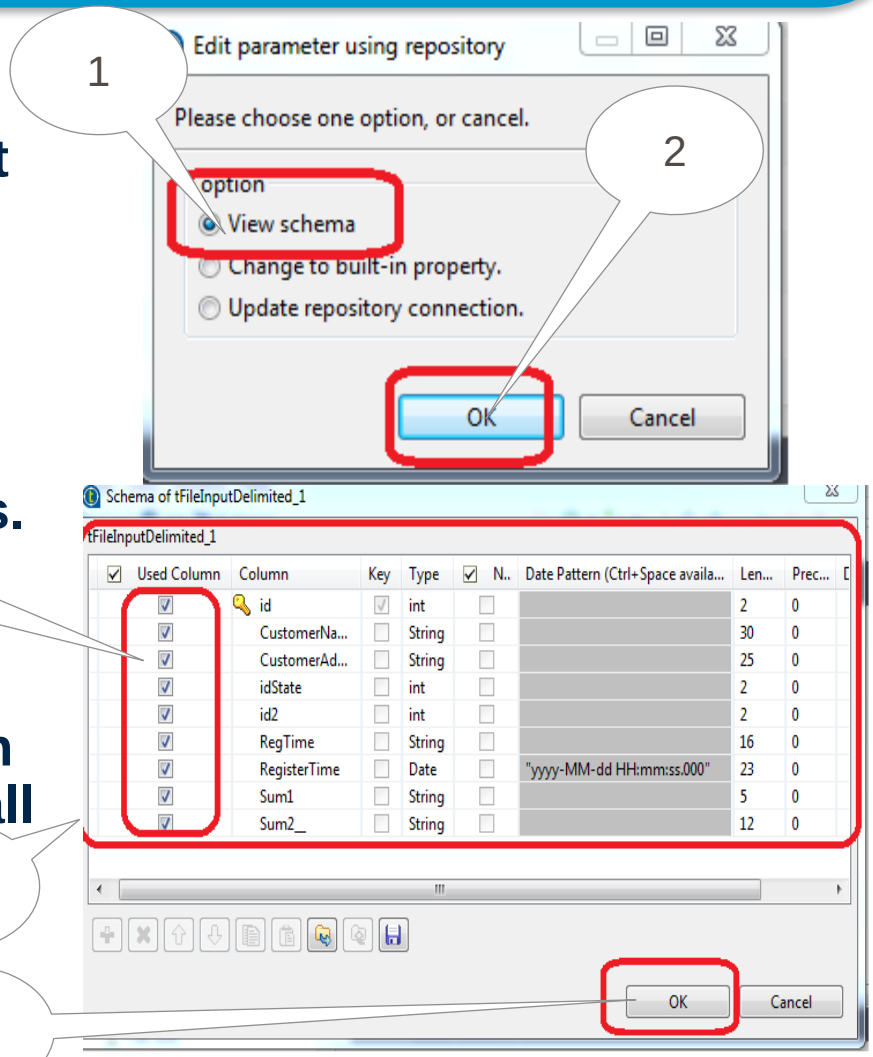
Job Design (Cont…)

- 1. Double-click **tFileInputDelimited_1 to show the corresponding Componentview to define its Basic settings.**
- **2.In the Component** view:Select Repositoryin the Property Typelist, Click [...]to select Customers metadata.
- 3.In the Repository content window, select **Customers metadatafile then. Click OK.**
- 4.Click [...] next to the **Edit schemafield to check the file schema.**
- 5. The **Edit parameter using repositorywizard opens.**



Job Design (Cont…)

1. Select **View schemain the optionlist as you only want to verify the schema.**
2. Click **OK.**
3. The **Schema of tFileInputDelimited_1 wizard opens.**
4. The schema is the same as the one you created in the **Repositoryview in Metadata> File delimited. Be sure all boxes are checked in “Used Columns”.**
5. Click **OK.**



Job Design (Cont…)

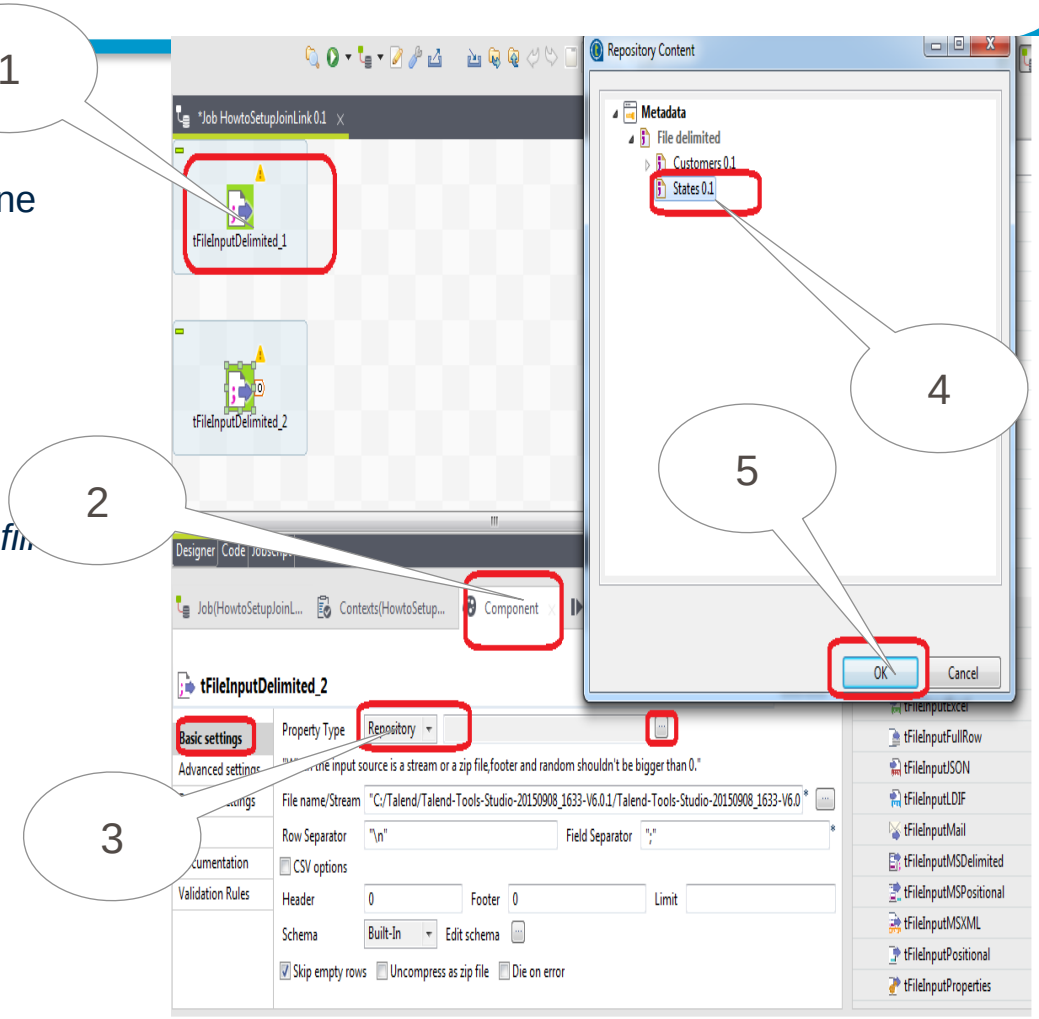
Double-click **tFileInputDelimited_2** to show the corresponding Componentview to define its Basic settings.

2.In the **Componentview:** Select Repositoryfrom the Property Typelist and click [...].

3.The **Repository Content wizard opens.**

4.Select the *statesmetadata to automatically fi in the* ***tFileInputDelimited_2Basic settingswith the state.txtfile properties.***

5.Click **OK.**



Job Design (Cont…)

1. Click [...] next to the **Edit schema field to check the file schema.**

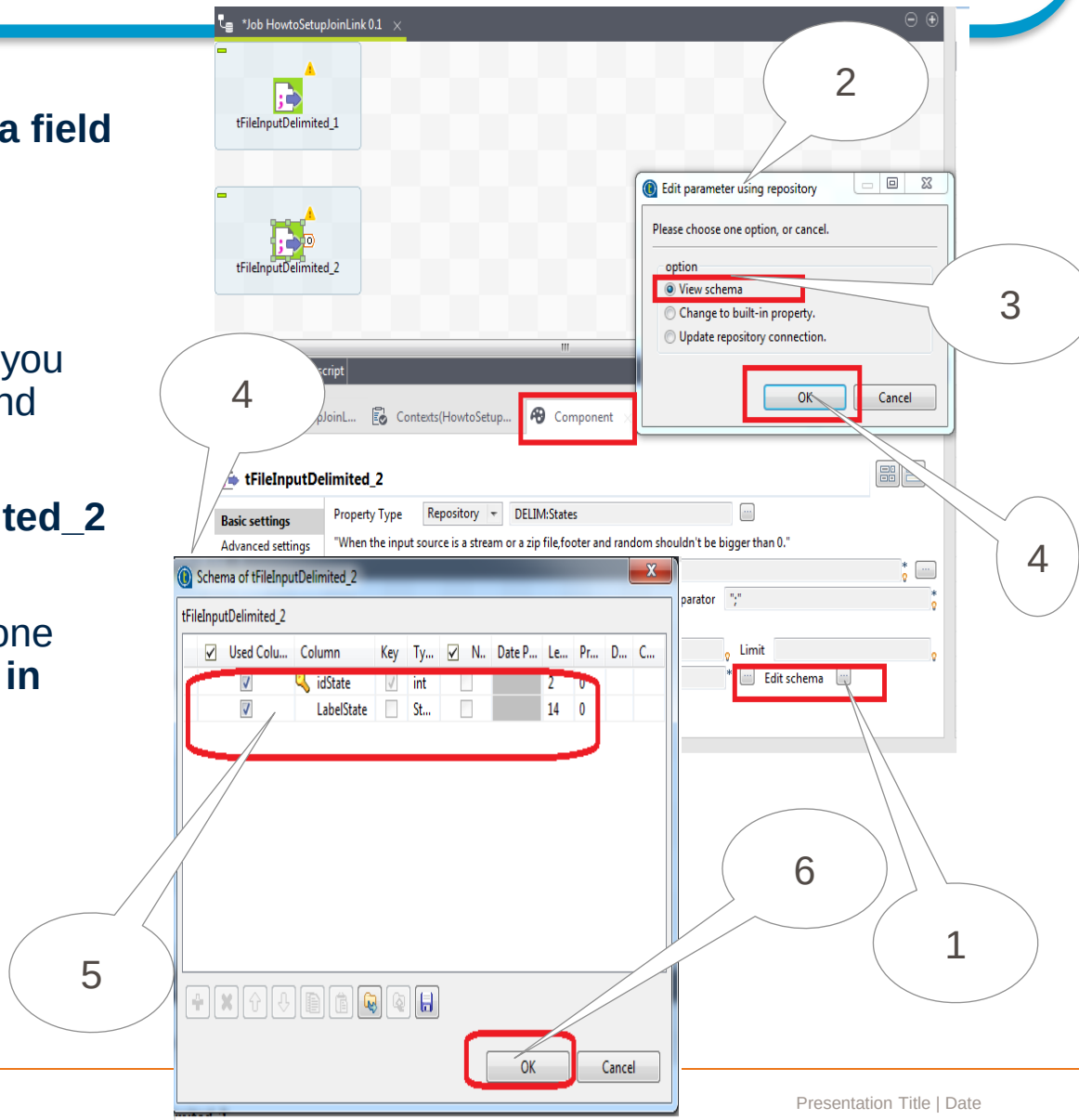
2. The **Edit parameter using repositorywizard opens.**

3. Keep **View schema selected** as you only want to verify the schema and click **OK.**

4. The **Schema of tFileInputDelimited_2 wizard opens.**

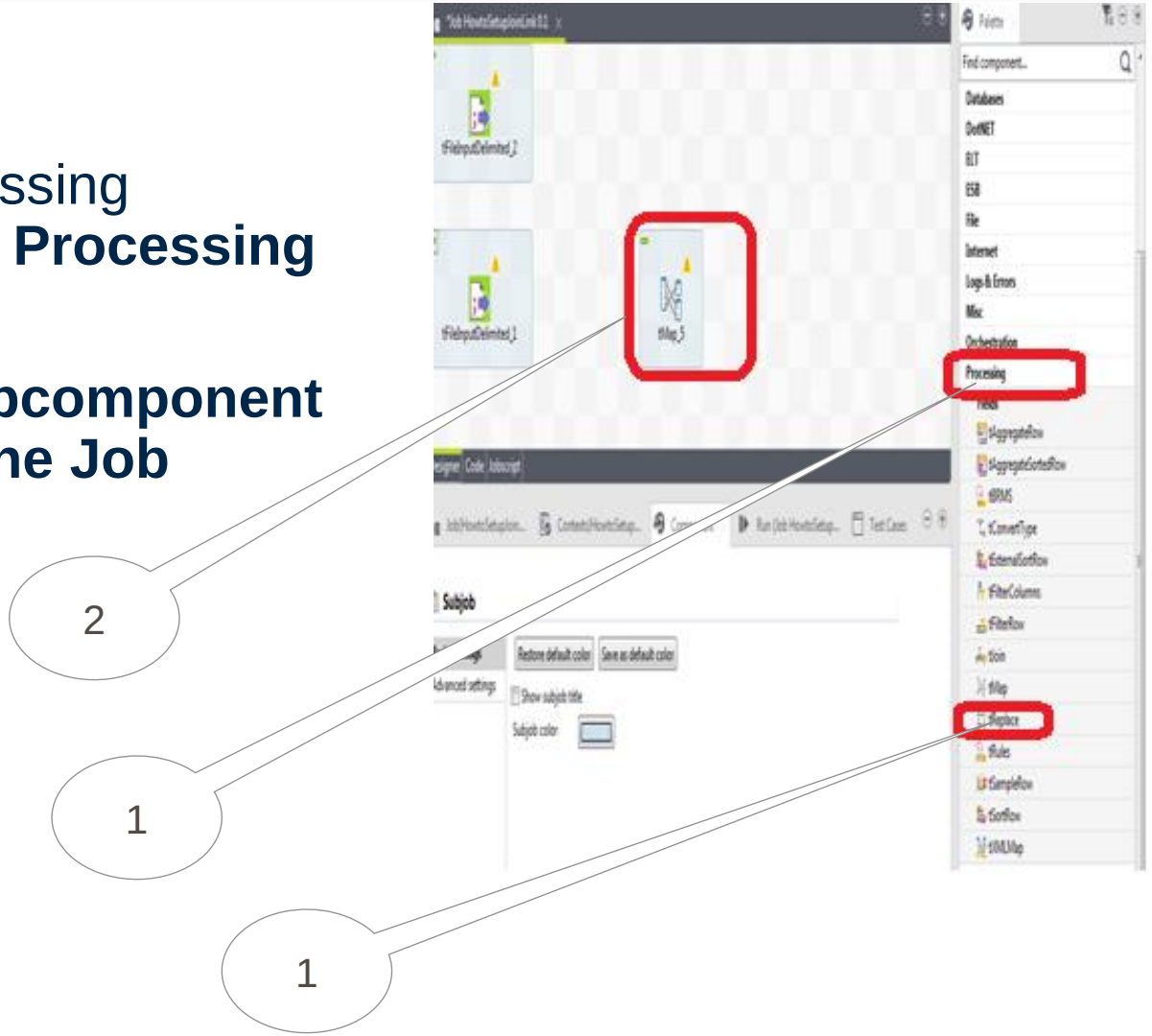
5. The schema is the same as the one available in the **Repositoryview in Metadata> File delimited.**

6. Click **OK**



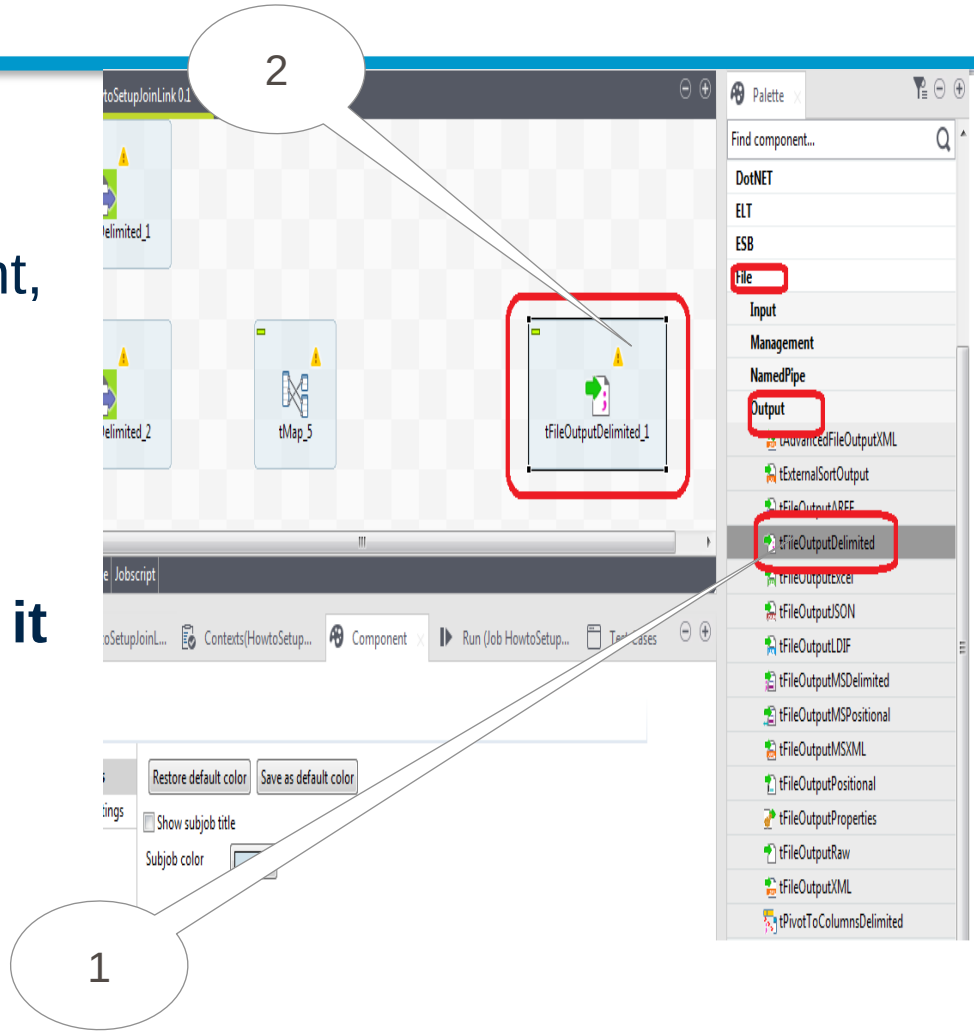
Job Design (Cont…)

1. To add the processing component, click **Processing family.**
2. Click on the **tMapcomponent and drop it on the Job Designer.**



Job Design – Output file

1. To add the output component, click **Filefamily and then Outputfamily.**
2. Click on the **tFileOutputDelimited component, drag and drop it on the Job Designer.**

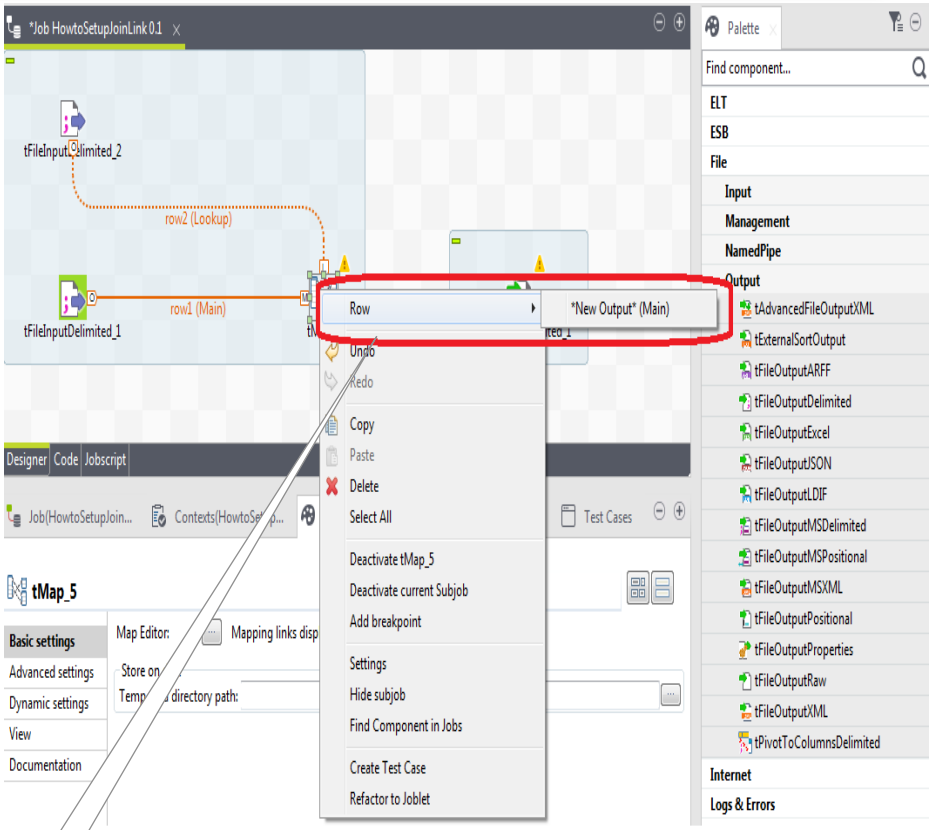


Job Design – Linking from source to map to target

1. To link the components together, right-click on **tFileInputDelimited_1, hold and drag it to the tMap.**

2. Do the same to link the **tFileInputDelimited_2 to the tMap**

3. To link the **tMapcomponent to the tFileOutputDelimited right click on the tMapand select row> *New Output*(Main) and connect it to tFileOutputDelimited**

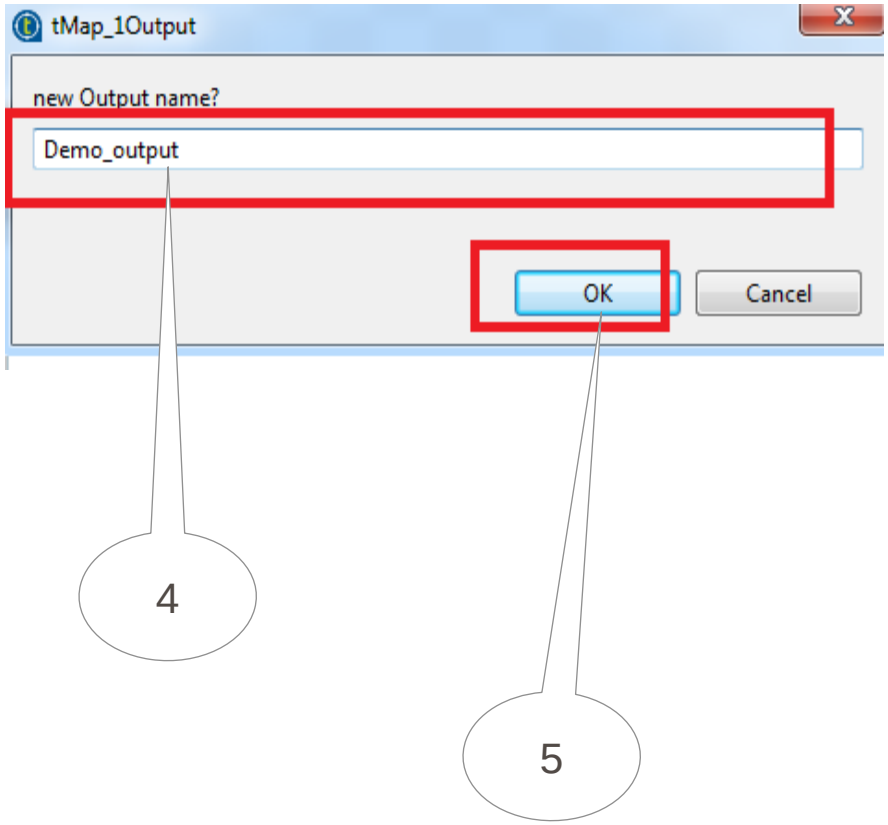


3

Job Design – Linking from source to map to target (Cont…)

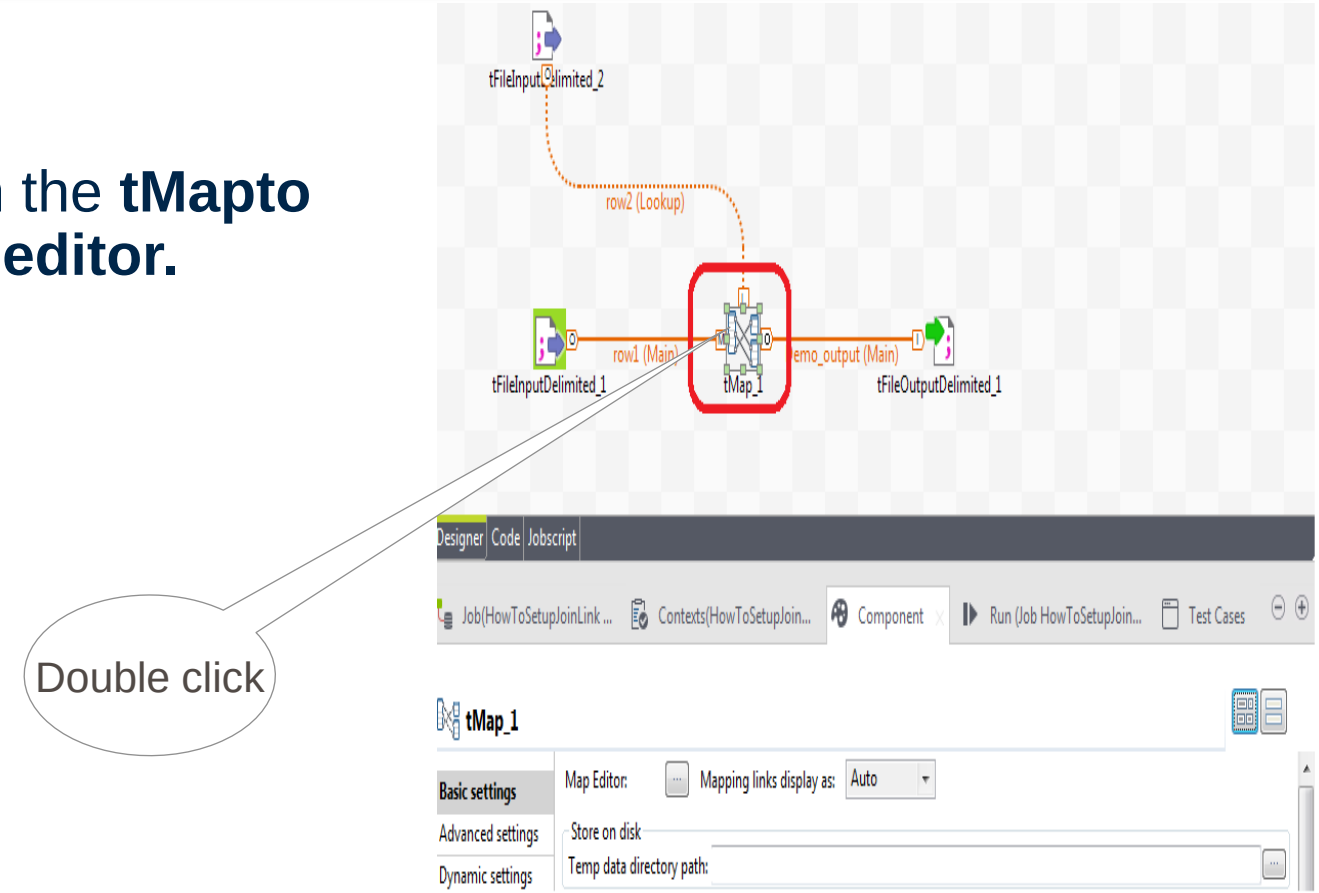
4. In the **tMap_1 Outputwizard, *name the link between the tMapand the* tFileOutputDelimited**

5. Click **OK,** a message window will appear, “Do you want to get the schema of the target component? Click Yes



Configuring Map component

- Double-click on the **tMapto open the tMapeditor.**



Configuring Map component (cont..)

1. In the row1 table, select the *id, CustomerName and CustomerAddresscolumns and drag them to the* ***outputtable.***

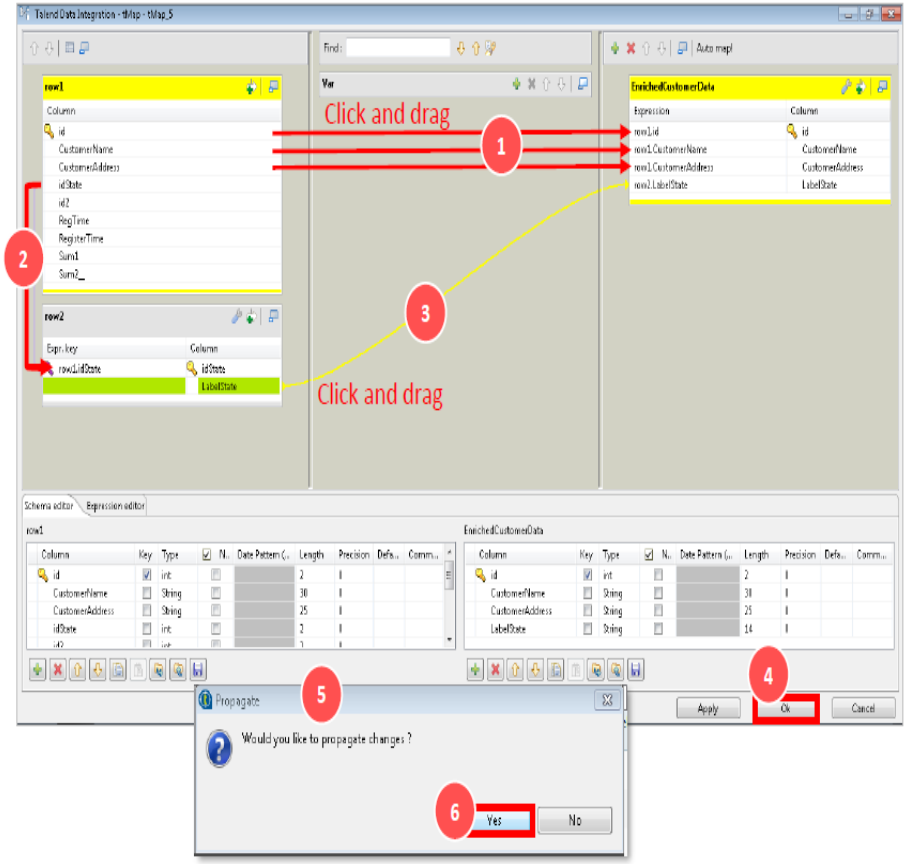
2. Then select the *idStatecolumn of the* ***row 1*** *table and drag it to the idStatecolumn of the* ***row2*** *table. This creates a join between the two tables.*

3. In the **row2** table, select the *LabelStatecolumn and drag it in the outputtable.*

4. Click **OK.**

5. The **Propagate message** box opens.

6. Click **Yes** to propagate the schema you defined in the tMapeditor to the next component.



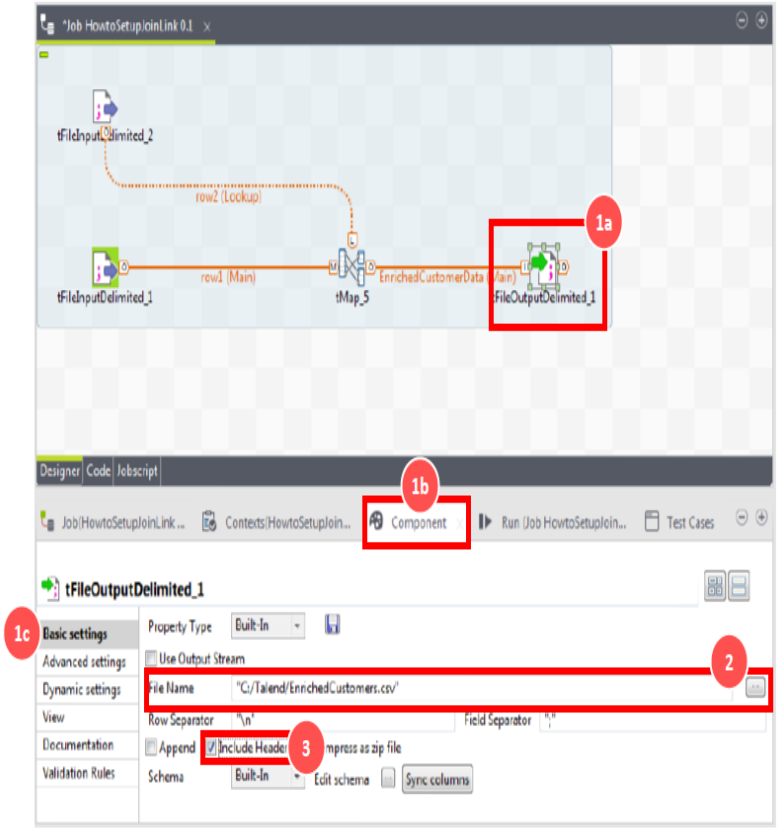
Configuring Output

1. Double-click on the **tFileOutputDelimitedto show the corresponding Component view to define its Basic settings.**

- **In the Component view:**

2. Click[...] next to the **File Name** field to specify the path of the file to be created. Navigate to you local drive C:/Talend and name the output file *EnrichedCustomers.csvthen click open.*

3. Check the **Include Header box to include the column headings in the output file.**

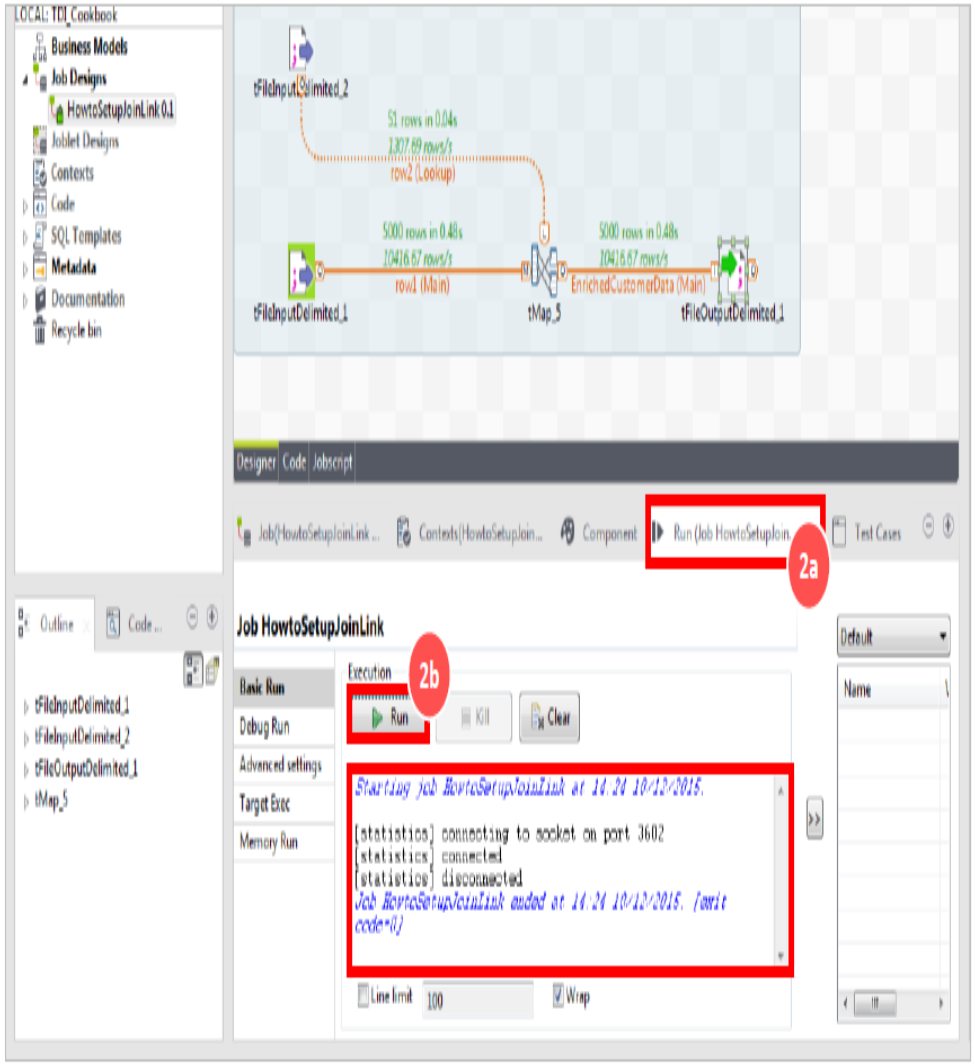


Running Job

Run the Job

1. Press **Ctrl+S to** save the Job.
2. In the run tab, Click on **Run** to execute the job.

The **Run view displays** at the bottom of TalendStudio and the execution window follows the Job execution.



Job Documentation

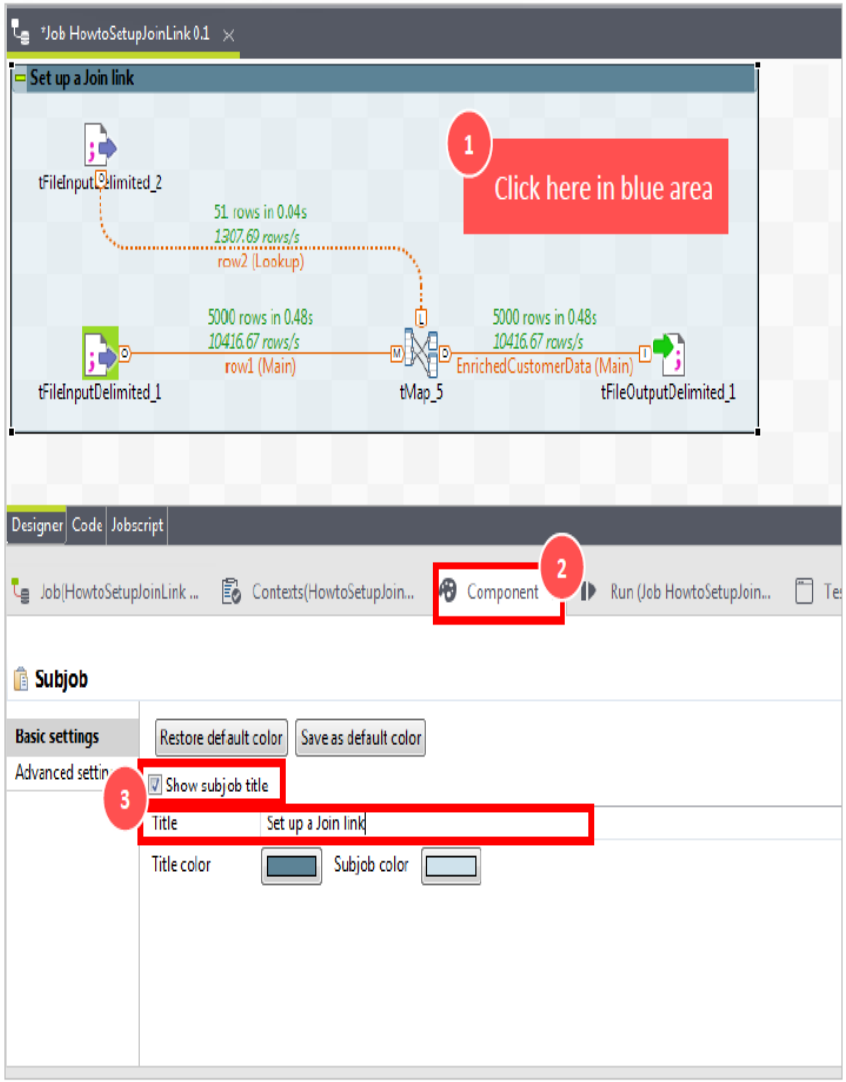
To document your Job, give it a title.

1. Click in the blue area around it.

2. Click the **Component view.**

3. Check the **Show subjobtitle** check box and in the Titlefield, fill in the corresponding title: *Set up a Join link.*

4. Save your Job again.



Source and Target Connectivity

Module Outline

- Connect to a database from a Talend Job
- Use a component to create a database table
- Write to and read from a database table
- Filter unique data rows
- Perform aggregation
- Write data to an XML file

Working with Database Components

- These include connectors for the most popular and traditional databases.
- These connectors cover various needs, including:
  - ✓ opening connections,
  - ✓ reading and writing tables,
  - ✓ committing transactions as a whole,
  - ✓ performing rollback for error handling.
- Over 40 RDBMS are supported. Other types of database connectors, such as connectors for Appliance/DW databases and database management.
- There are about 346 components under the Database component category.

Database Component (Contd..)

Databases - Traditional components

tAccessBulkExec	tDB2Commit	tInformixSP	tMSSqlTableList	tOleDbRow	tPostgresqlOutput
tAccessClose	tDB2Connection	tMemSQLClose - Databases	tMysqlBulkExec	tOracleBulkExec	tPostgresqlOutputBulk
tAccessCommit	tDB2Input	tMemSQLConnection - Databases	tMysqlClose	tOracleClose	tPostgresqlOutputBulkExec
tAccessConnection	tDB2Output	tMemSQLInput - Databases	tMysqlColumnList	tOracleCommit	tPostgresqlRollback
tAccessInput	tDB2Rollback	tMemSQLOutput - Databases	tMysqlCommit	tOracleConnection	tPostgresqlRow
tAccessOutput	tDB2Row	tMemSQLRow - Databases	tMysqlConnection	tOracleInput	tPostgresqlSCD - Databases
tAccessOutputBulk	tDB2SCD - Databases	tMSSqlBulkExec	tMysqlInput	tOracleOutput	tPostgresqlSCDELT - Databases
tAccessOutputBulkExec	tDB2SCDELT - Databases	tMSSqlColumnList	tMysqlLastInsertId	tOracleOutputBulk	tSybaseBulkExec
tAccessRollback	tDB2SP	tMSSqlClose	tMysqlLookupInput	tOracleOutputBulkExec	tSybaseClose
tAccessRow	tInformixBulkExec	tMSSqlCommit	tMysqlOutput	tOracleRollback	tSybaseCommit
tAS400Close	tInformixClose	tMSSqlConnection	tMysqlOutputBulk	tOracleRow	tSybaseConnection
tAS400Commit	tInformixCommit	tMSSqlInput	tMysqlOutputBulkExec	tOracleSCD - Databases	tSybaseInput
tAS400Connection	tInformixConnection	tMSSqlLastInsertId	tMysqlRollback	tOracleSCDELT - Databases	tSybaseIQBulkExec

Database Component (Contd..)

Databases - Appliance/Datawarehouse components

tGreenplumBulkExec	tIngresBulkExec	tNetezzaClose	tParAccelConnection	tRedshiftInput	tTeradataFastLoadUtility
tGreenplumClose	tIngresClose	tNetezzaCommit	tParAccelInput	tRedshiftOutput	tTeradataInput
tGreenplumCommit	tIngresCommit	tNetezzaConnection	tParAccelOutput	tRedshiftOutputBulk	tTeradataMultiLoad
tGreenplumConnection	tIngresConnection	tNetezzaInput	tParAccelOutputBulk	tRedshiftOutputBulkExec	tTeradataOutput
tGreenplumGPLoad	tIngresInput	tNetezzaNzLoad	tParAccelOutputBulkExec	tRedshiftRollback	tTeradataRollback
tGreenplumInput	tIngresOutput	tNetezzaOutput	tParAccelRollback	tRedshiftRow	tTeradataRow
tGreenplumOutput	tIngresOutputBulk	tNetezzaRollback	tParAccelRow	tRedshiftUnload	tTeradataSCD - Databases
tGreenplumOutputBulk	tIngresOutputBulkExec	tNetezzaRow	tParAccelSCD - Databases	tTeradataClose	tTeradataSCDELT - Databases
tGreenplumOutputBulkExec	tIngresRollback	tNetezzaSCD - Databases	tRedshiftBulkExec	tTeradataCommit	tTeradataTPTExec
tGreenplumRollback	tIngresRow	tParAccelBulkExec	tRedshiftClose	tTeradataConnection	tTeradataTPTUtility
tGreenplumRow	tIngresSCD - Databases	tParAccelClose	tRedshiftCommit	tTeradataFastExport	tTeradataTPump
tGreenplumSCD - Databases	tNetezzaBulkExec	tParAccelCommit	tRedshiftConnection	tTeradataFastLoad	tVectorWiseCommit

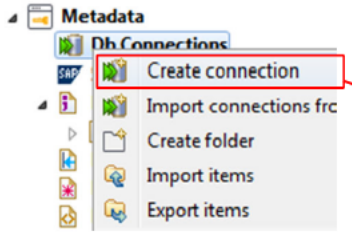
Database Component (Contd..)

Databases - Other components

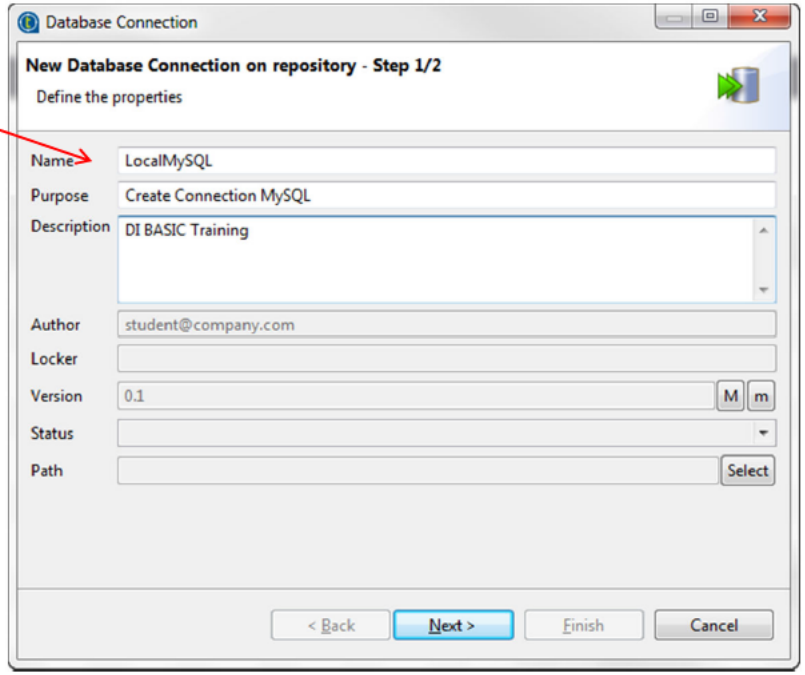
tCassandraBulkExec - Databases	tCouchDBClose - Databases	tEXAInput	tFirebirdCommit	tHiveCreateTable - Databases	tInterbaseRollback
tCassandraClose - Databases	tCouchDBConnection - Databases	tEXAOutput	tFirebirdConnection	tHiveInput - Databases	tInterbaseRow
tCassandraConnection - Databases	tCouchDBInput - Databases	tEXARollback	tFirebirdInput	tHiveLoad - Databases	tJavaDBInput
tCassandraInput - Databases	tCouchDBOutput - Databases	tEXARow	tFirebirdOutput	tHiveRow - Databases	tJavaDBOutput
tCassandraOutput - Databases	tCreateTable	tEXistConnection	tFirebirdRollback	tHSQLDbInput	tJavaDBRow
tCassandraOutputBulk - Databases	tDBInput	tEXistDelete	tFirebirdRow	tHSQLDbOutput	tJDBCColumnList
tCassandraOutputBulkExec - Databases	tDBOutput	tEXistGet	tHBaseClose - Databases	tHSQLDbRow	tJDBCClose
tCassandraRow - Databases	tDBSQLRow	tEXistList	tHBaseConnection - Databases	tInterbaseClose	tJDBCCommit
tCouchbaseClose - Databases	tEXABulkExec	tEXistPut	tHBaseInput - Databases	tInterbaseCommit	tJDBCConnection
tCouchbaseConnection - Databases	tEXAClose	tEXistXQuery	tHBaseOutput - Databases	tInterbaseConnection	tJDBCInput
tCouchbaseInput - Databases	tEXACommit	tEXistXUpdate	tHiveClose - Databases	tInterbaseInput	tJDBCOutput
tCouchbaseOutput - Databases	tEXAConnection	tFirebirdClose	tHiveConnection - Databases	tInterbaseOutput	tJDBCRollback

Database Connection- Creating Metadata for database

- Database connection metadata can be found under
- Repository>Metadata>Db connections

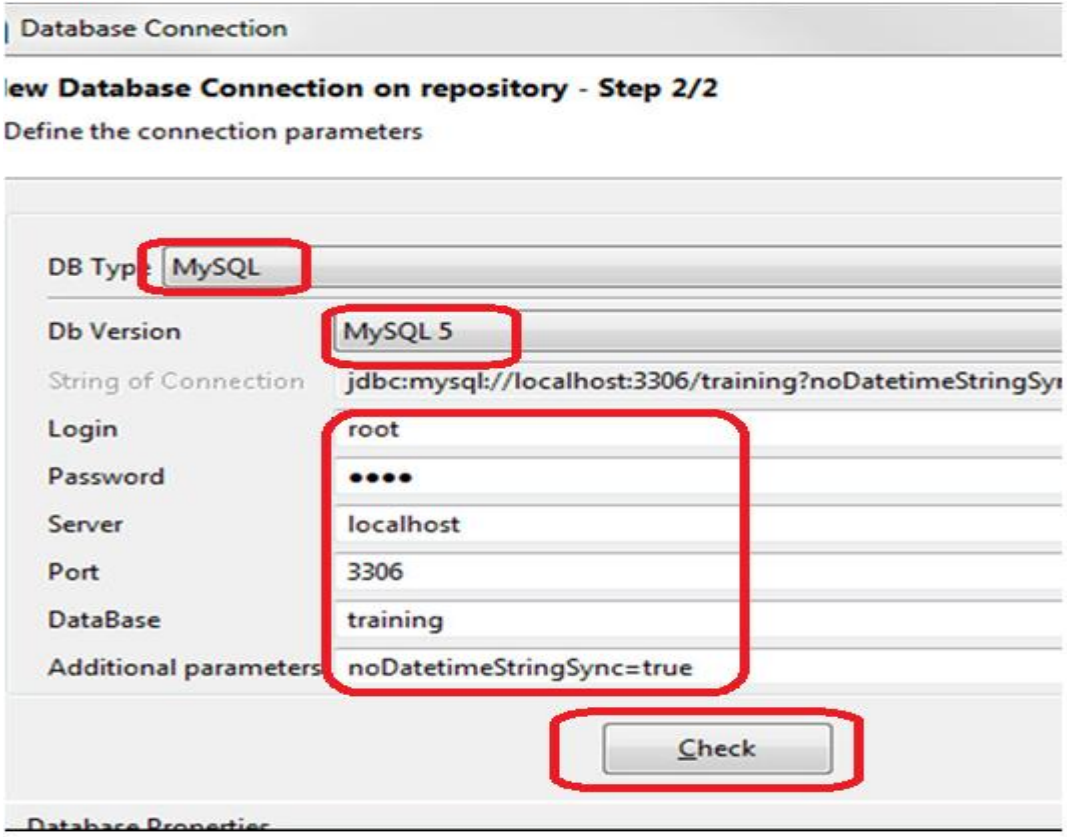


Provide Name, Purpose, and Description.



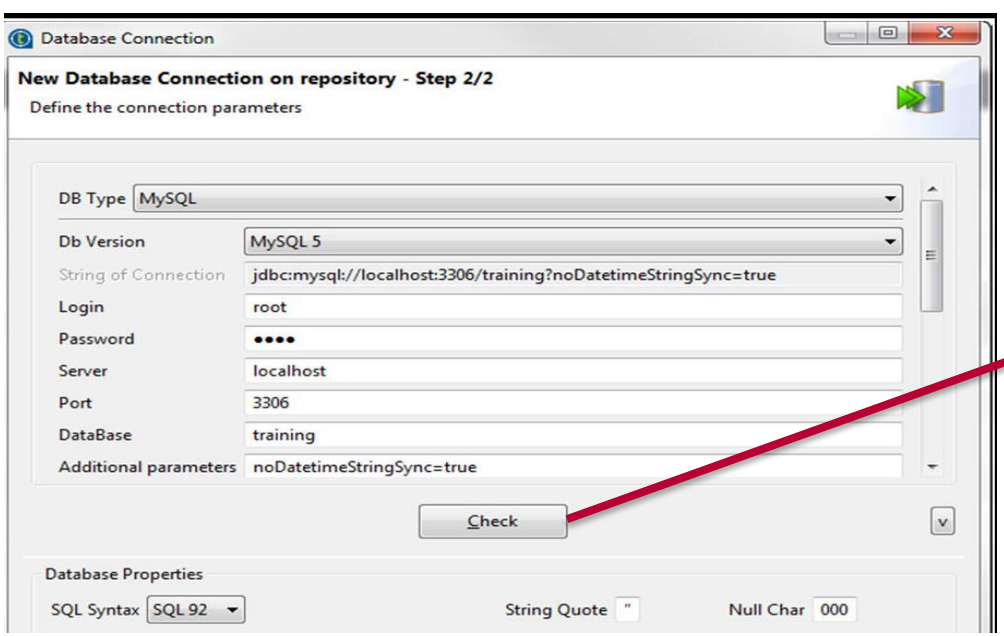
Database Connection- Creating Metadata for database

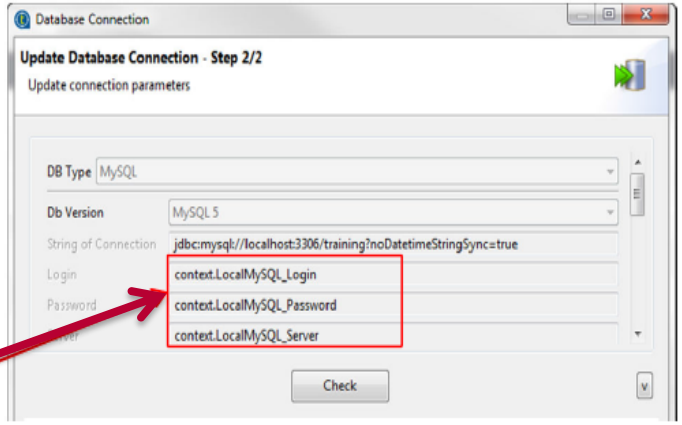
- Specify key connection information and check your connection.



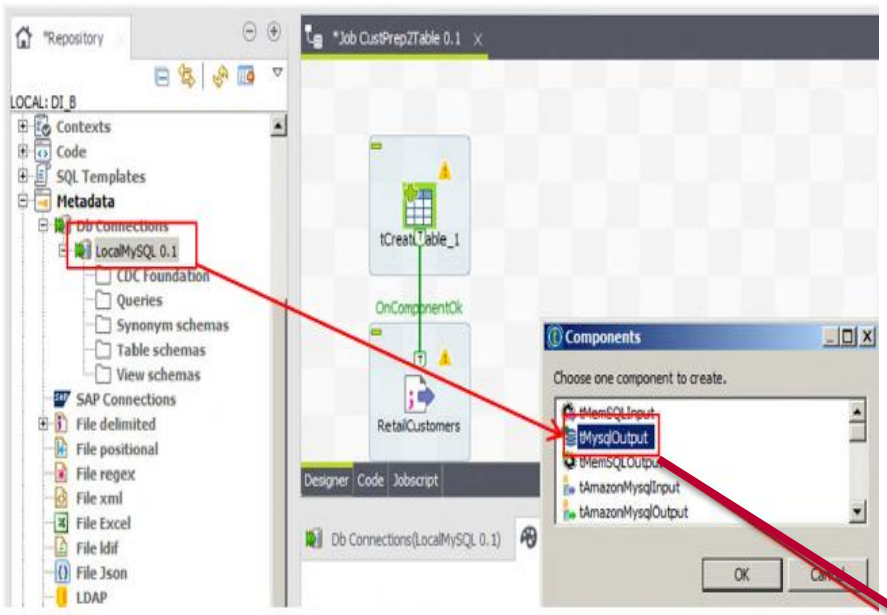
Export as Context

Save connection info as context variables

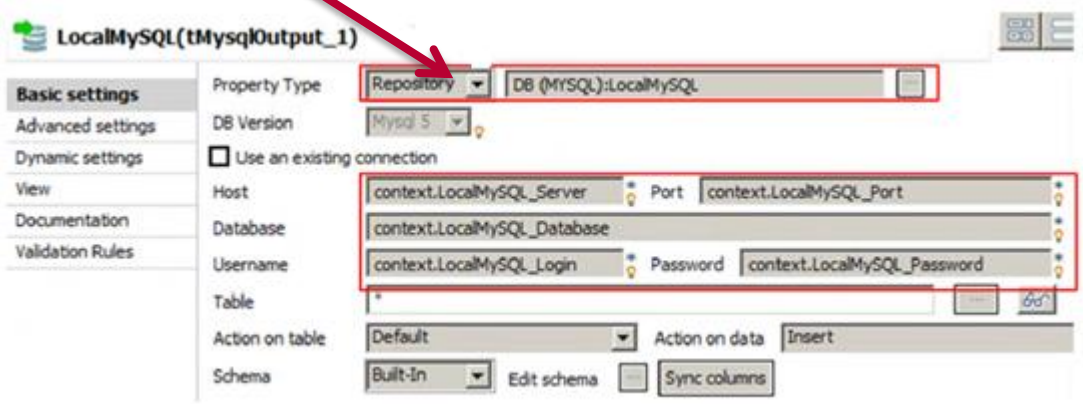




Using Metadata

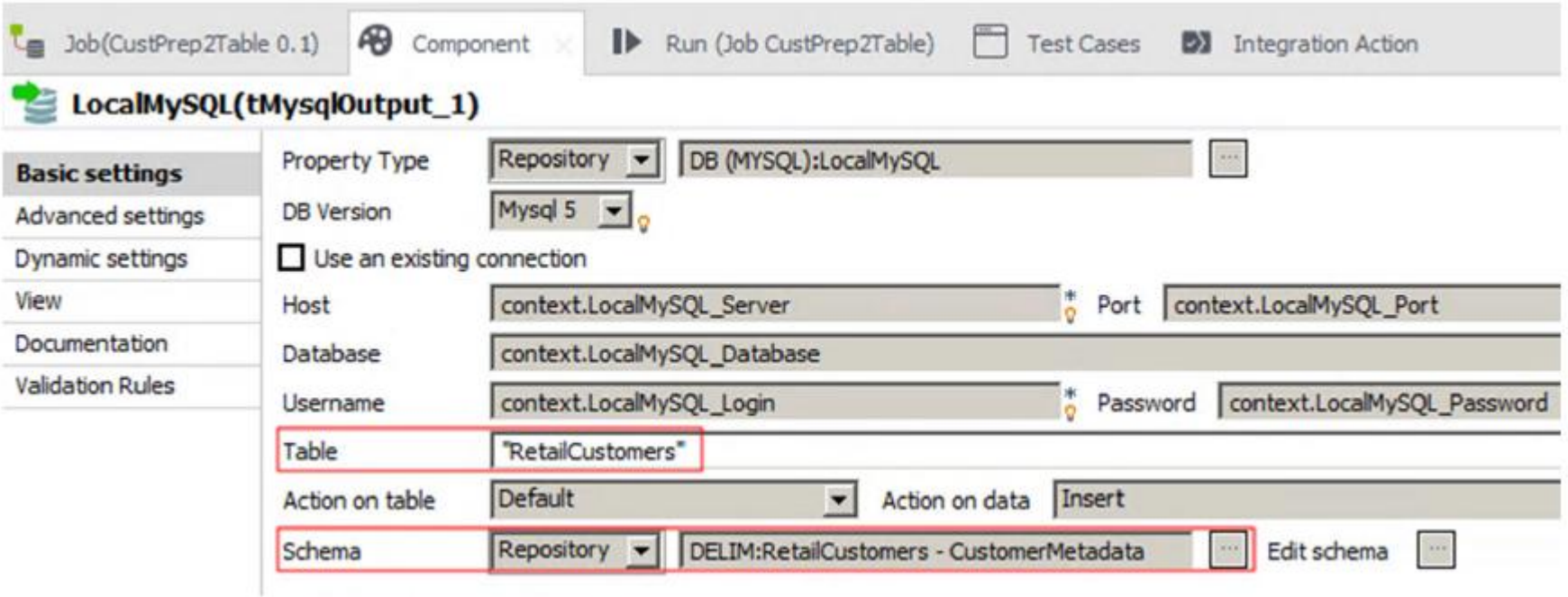


- Drag the database connection metadata to the Designer
- Select a component in the list.
- The component is configured with the connection information metadata.



Writing Data To a Database- tMysqlOutput

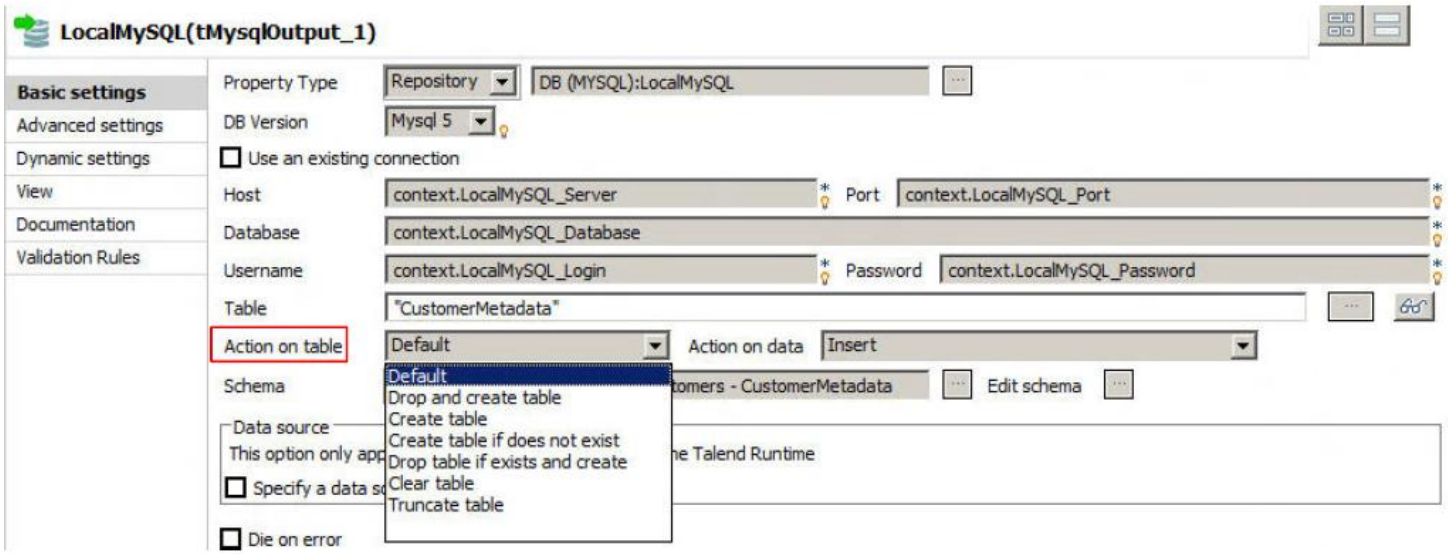
- To write data to a MySQL database, you can use the tMysqlOutput component.



- You should also provide the Table Name and the Schema.

Writing Data To a Database - tMysqlOutput – Action on table

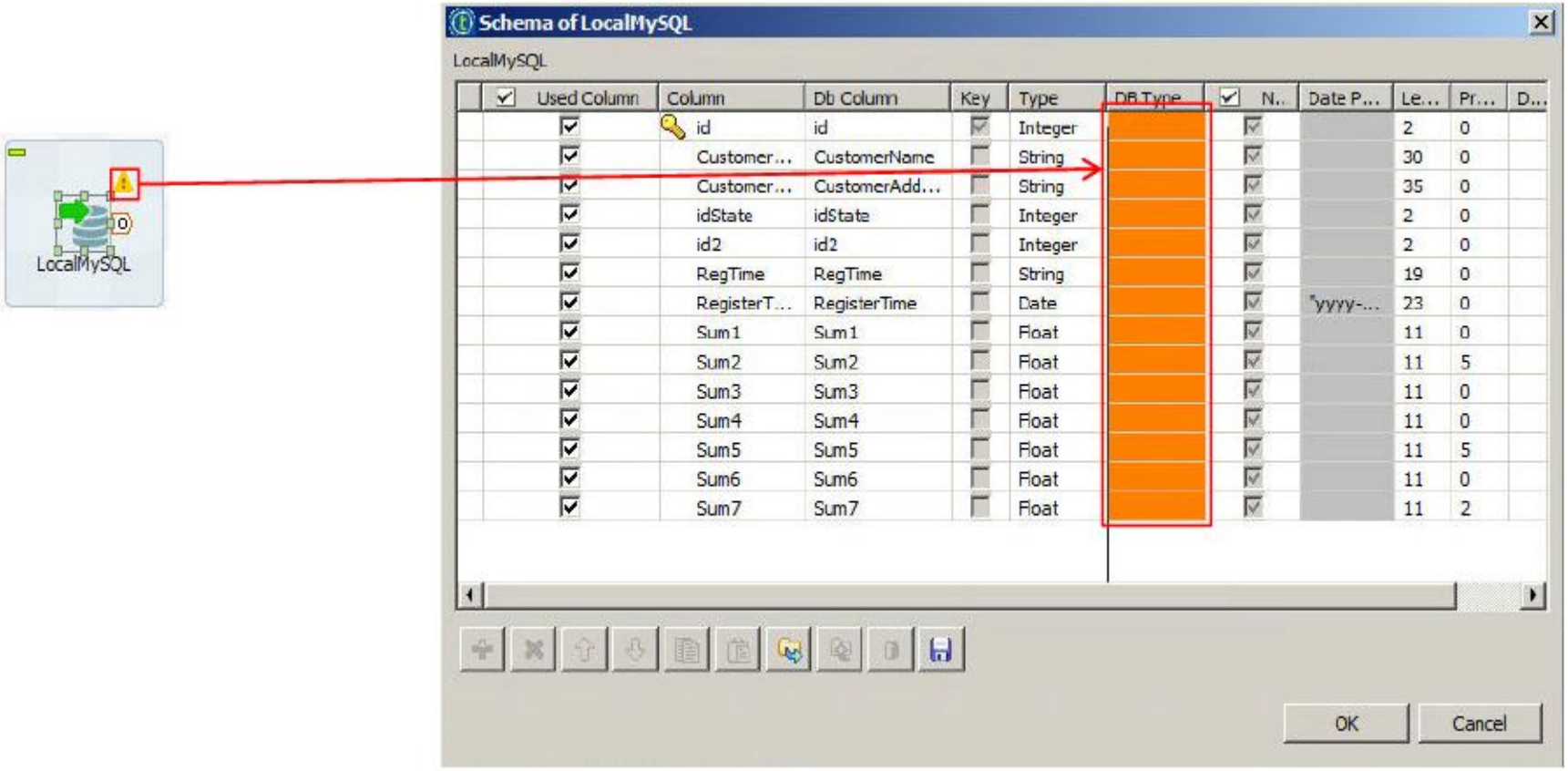
- Select the Action on table in the drop down list:



- Note: tMysqlOutput allows you to create a table and then write to it.

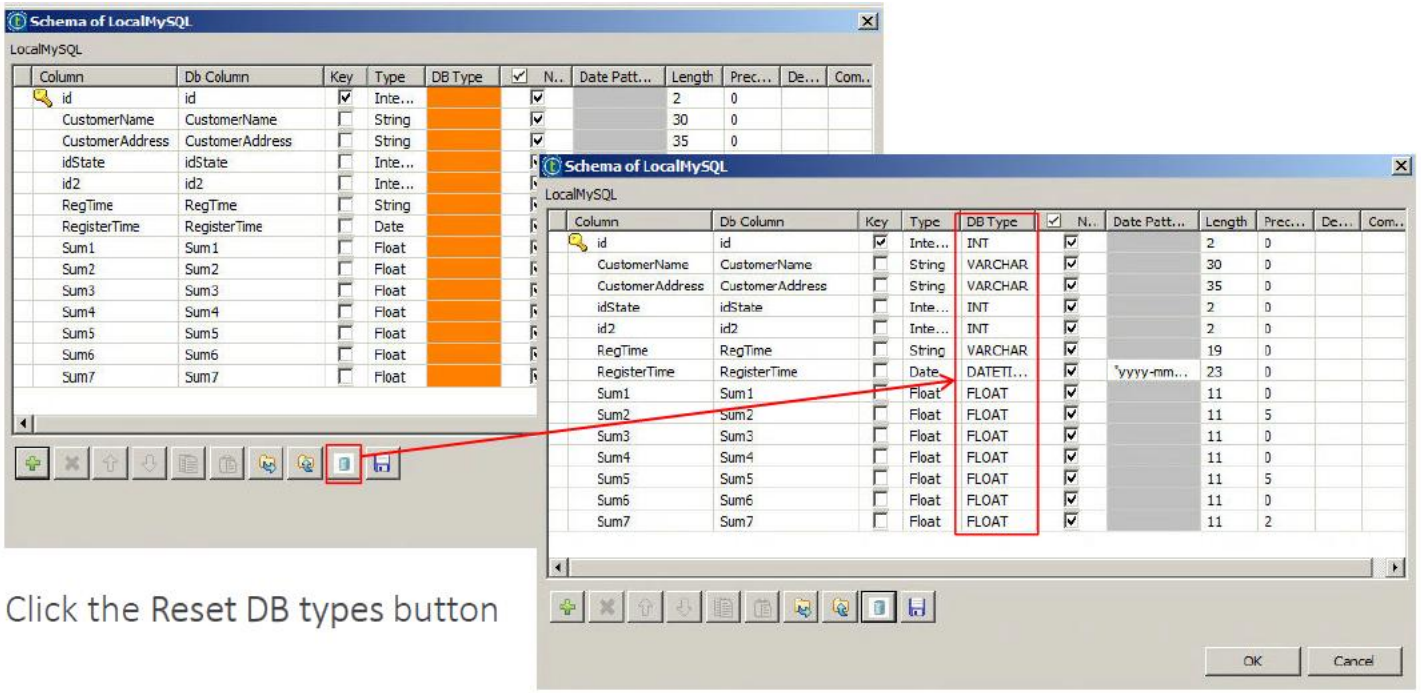
Writing Data To a Database - tMysqlOutput – Fix schema

- The schema in a tMysqlOutput component requires Database column names and types:



Writing Data To a Database - tMysqlOutput – Fix schema

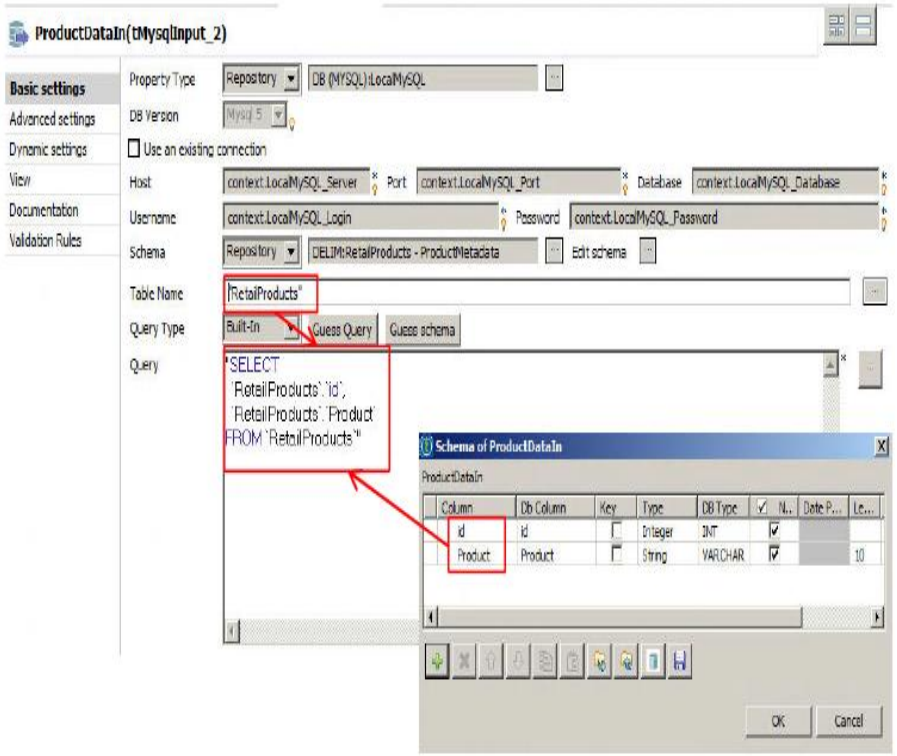
- The schema in a tMysqlOutput component requires Database column names and types:



Reading From a Database- tMysql Input

- To read data from a MySQL database, you can use the tMysqlInput component. This component requires a SQL Query to read the datafrom the MySQL database.

Click Guess Query button to build the SQL Query based on the Table Name and the Schema.



Creating DB Connection from CSV file

1. To create a new database connection within the repository, *right-click* Repository->Metadata->DB Connections and select the option Create connection. This will display the Database Connection dialog.

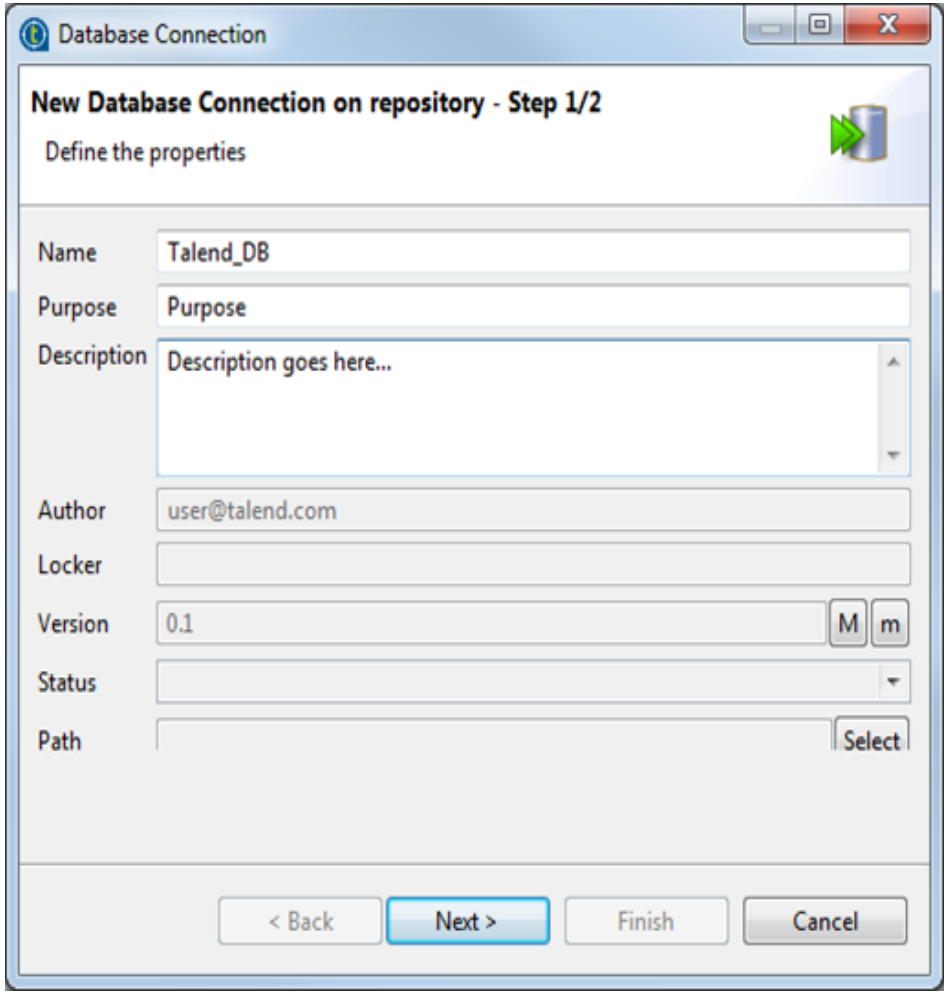
2. Enter a Name, Purpose and Description for your database connection.

- ✓ Use a Name that suggests its purpose, for example, *Development Accounts* indicating that this connection is for the *development* instance of your *Accounts* database.
- ✓ Always establish these connections to a *development* or *test* database and never to *production*
- ✓ Should be consistent in the databases that you connect to and the naming conventions you use.

Creating DB Connection (Contd..)

3. When you've completed step 1 of this dialog,

   press Next, as shown in the screenshot below.



Database Connection Parameters

1. Step 2 of the Database Connection dialog allows you to enter the connection parameters of the database that you want to connect to.

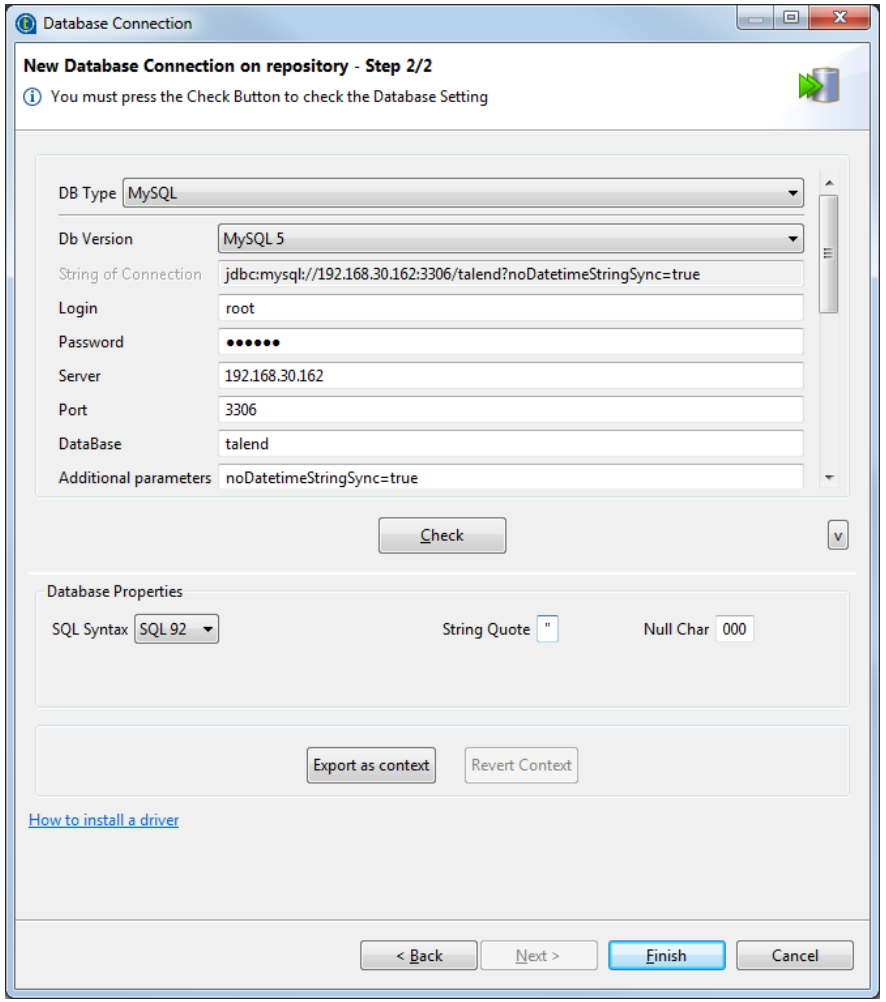
2. Select the DB Type that you want to connect to.

- ✓ For some database vendors, there may be multiple types that you may select, for example, Oracle connections are supported using *OCI*, RAC, *SID* and *Service Name*.

- ✓ Once you have selected a DB type, the remainder of the dialog will be preconfigured for the database vendor and connection type that you selected. As shown below, for MySQL, we have some basic setings that we need to make for a connection to a MySQL 5database.

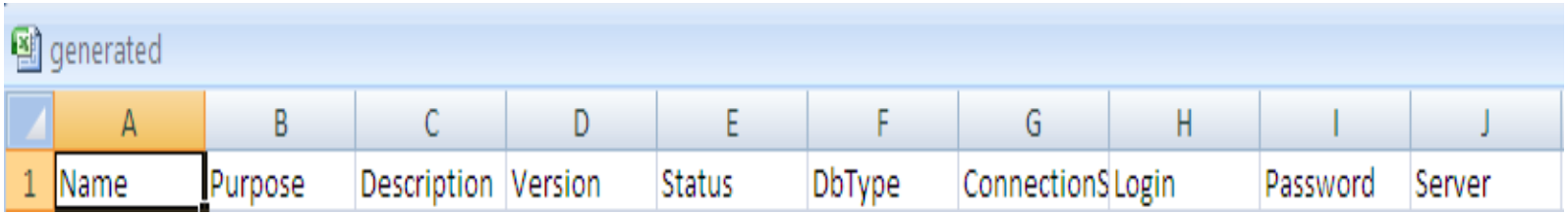
Database Connection Parameters (Contd..)

3. When you have completed this page of the dialog, you can Check your connection and then hit Finish.



Importing tables from DB

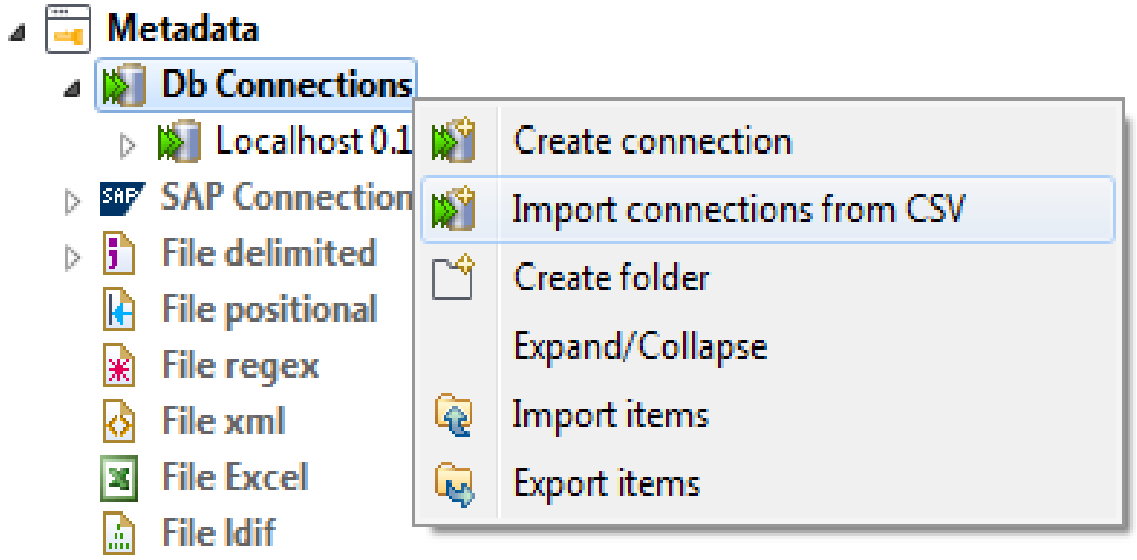
- Before importing database connection metadata from a CSV file, make sure that your CSV file format is valid. The file columns should be filled as follows:
  - ✓ Name; Purpose; Description; Version; Status; Db Type; Connection String; Login; Password; Server; Port; Database; DB Schema; Data source; File; DB Root; Table Name; Original Table Name; Label; Original Label; Comment; Default; Key; Length; Null able; Pattern; Precision; Talend Type; DB Type.
  - ✓ It is recommended to use either Talend Type or DB Type, not both.
  - ✓ Table Name is the name displayed in Talend Studio, original Table Name is the original table name in the database. (You can choose to fill only the original Table Name).
  - ✓ Label is the column name used in Talend Studio; original Label is the column name in the table. (You can choose to fill only the original Label).



Importing tables from DB (Contd..)

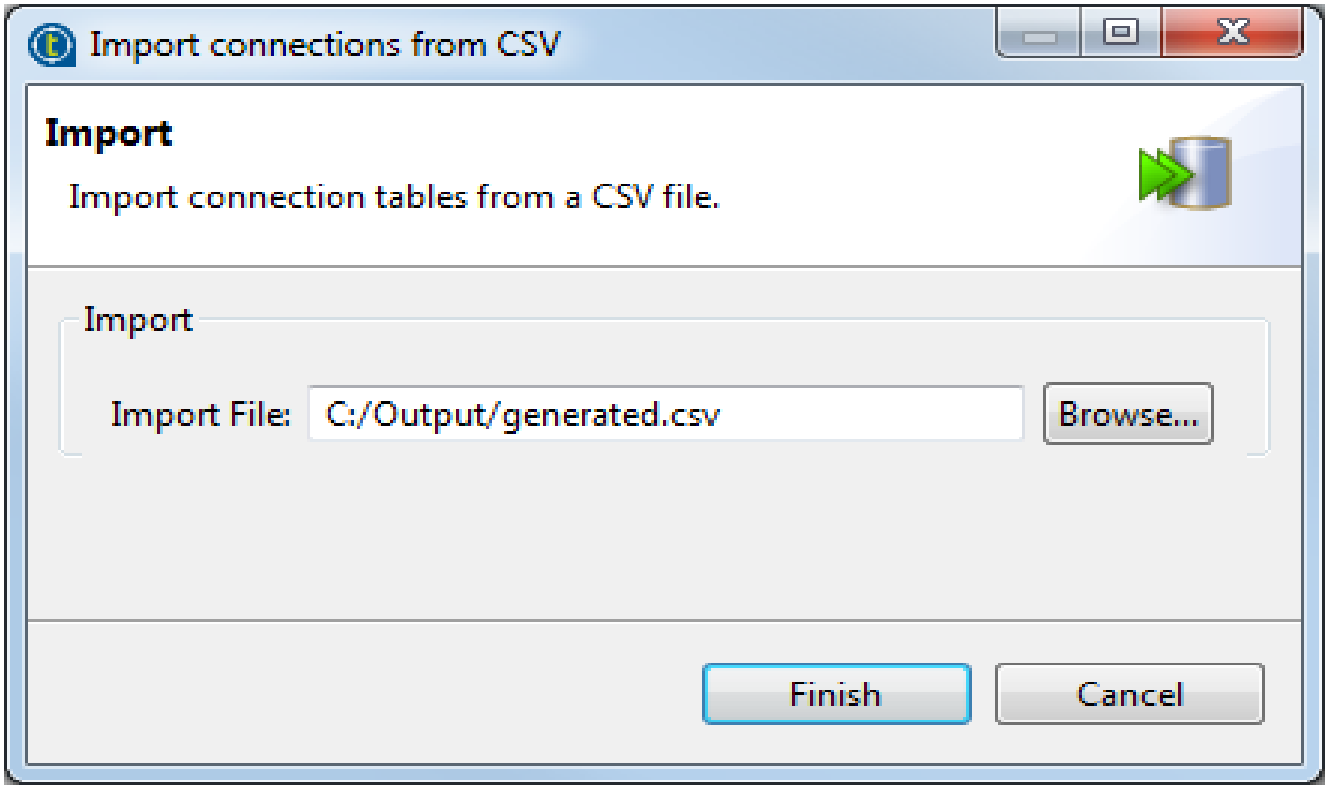
- To import database connection metadata from a defined CSV file, do the following:

1. In the Repository tree view, expand the Metadata node and right-click Db connections.
2. In the contextual menu, select Import connections from CSV.



Importing tables from DB (Contd..)

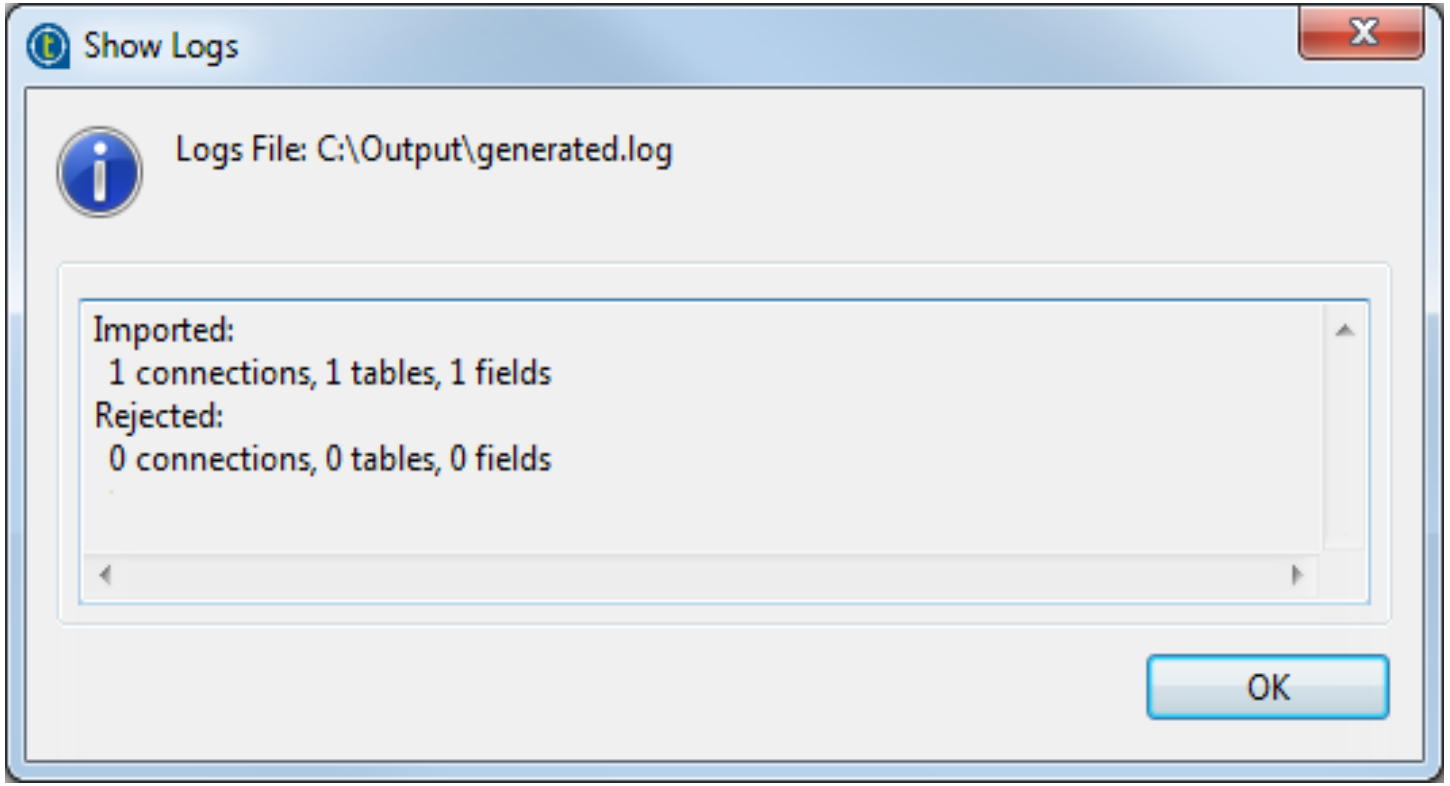
3. Click Browse... and go to the CSV file that holds the metadata of the database connection.



Importing tables from DB (Contd..)

4. Click Finish to close the dialog box.

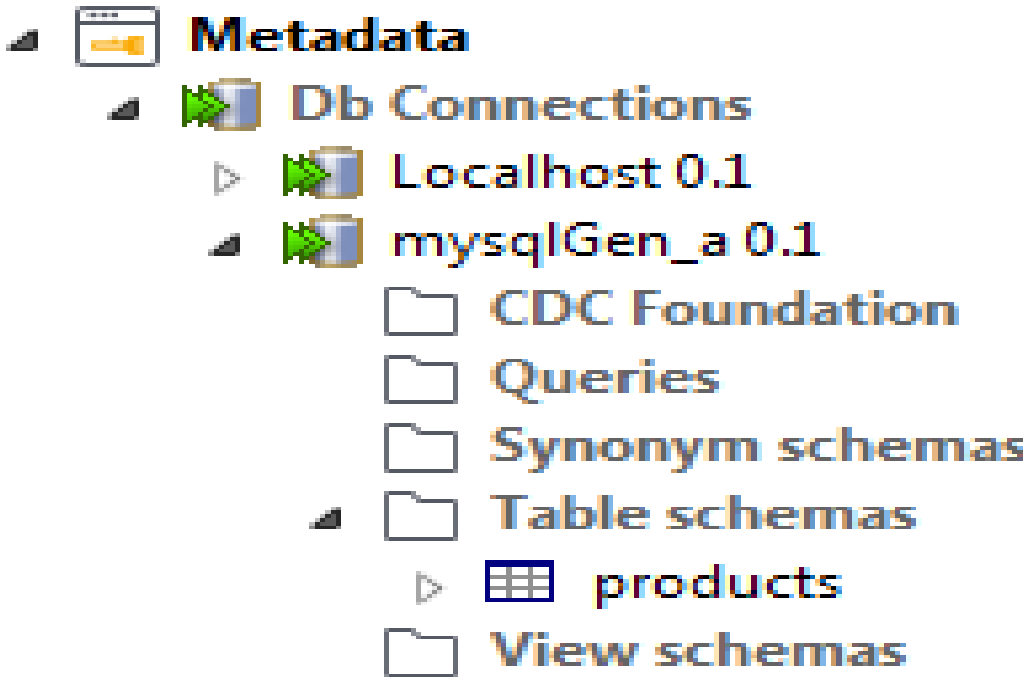
✓The [Show Logs] dialog bow displays to list imported and rejected metadata, if any.



Importing tables from DB (Contd..)

5. Click OK to close the dialog box.

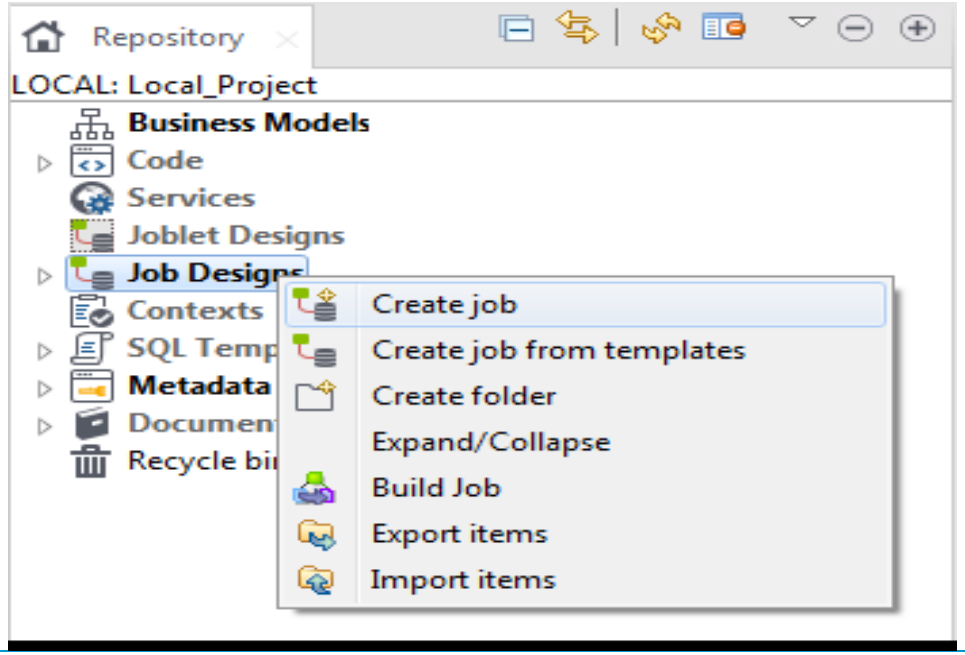
✓The imported metadata displays under the DB connections node in the Repository tree view.



Creating a Sample Job

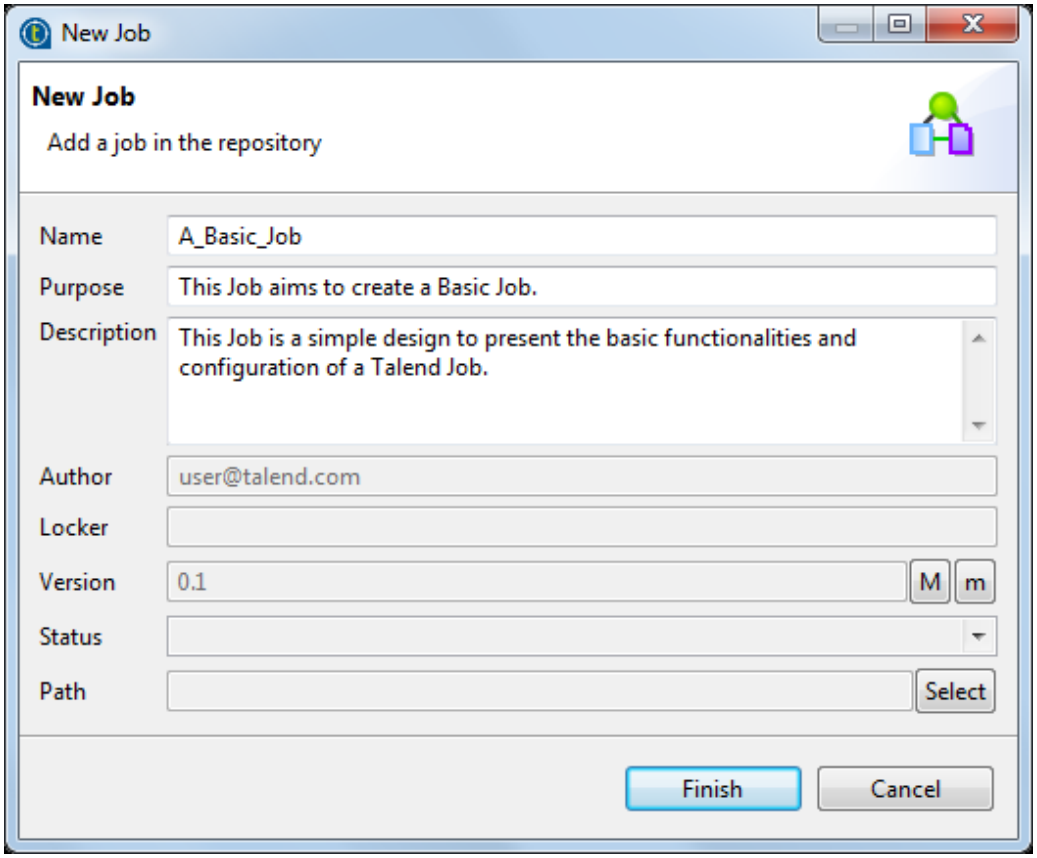
- Talend Studio enables you to create a Job by dropping different technical components from the Palette onto the design workspace and then connecting these components together.

1. In the Repository tree view of the Integration perspective, right-click the Job Designs node and select Create job from the contextual menu



Creating a Sample Job (Contd..)

- ✓The [New Job] wizard opens to help you define the main properties of the new Job.



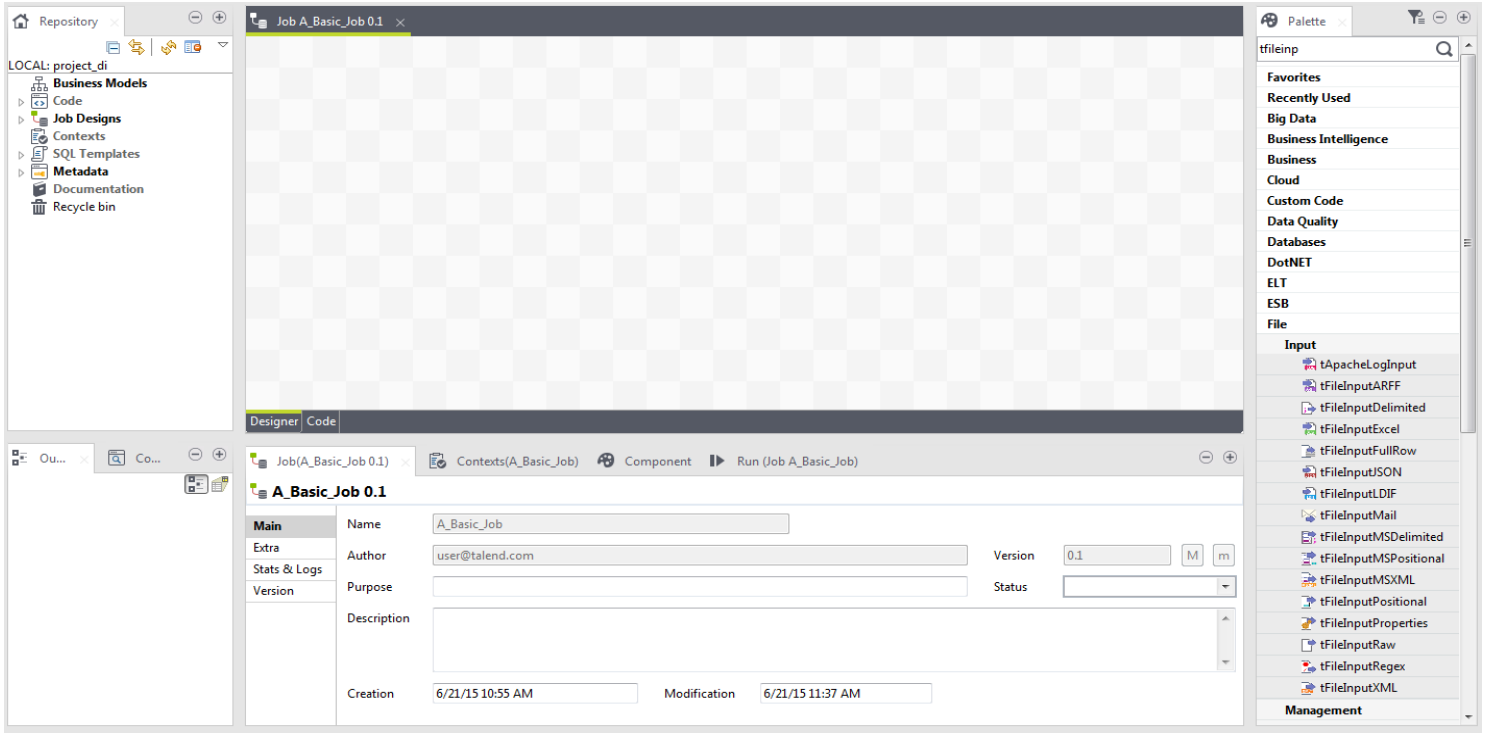
Creating a Sample Job (Contd..)

2. Fill the Job properties as shown in the previous screenshot. The fields correspond to the following properties:

Field	Description
Name	the name of the new Job. Note that a message comes up if you enter prohibited characters.
Purpose	Job purpose or any useful information regarding the Job use.
Description	Job description containing any information that helps you describe what the Job does and how it does it.
Version	a read-only field. You can manually increment the version using the M and m buttons. For more information, see the Getting Started Guide.
Locker	a read-only field that shows by default the login of the user who owns the lock on the current Job. This field is empty when you are creating a Job and has data only when you are editing the properties of an existing Job.
Author	a read-only field that shows by default the current user login.
Status	a list to select from the status of the Job you are creating.
Path	a list to select from the folder in which the Job will be created.

Creating a Sample Job (Contd..)

3. An empty design workspace opens up showing the name of the Job as a tab label.



✓ The Job you created is now listed under the Job Designs node in the Repository tree view.

✓ You can open one or more of the created Jobs by simply double-clicking the Job label in the Repository tree view.

Context Variables

Module Outline

- Context Parameters
- Defining Contexts- Steps
- How to define variables from the Component view?
- Running the Job Locally

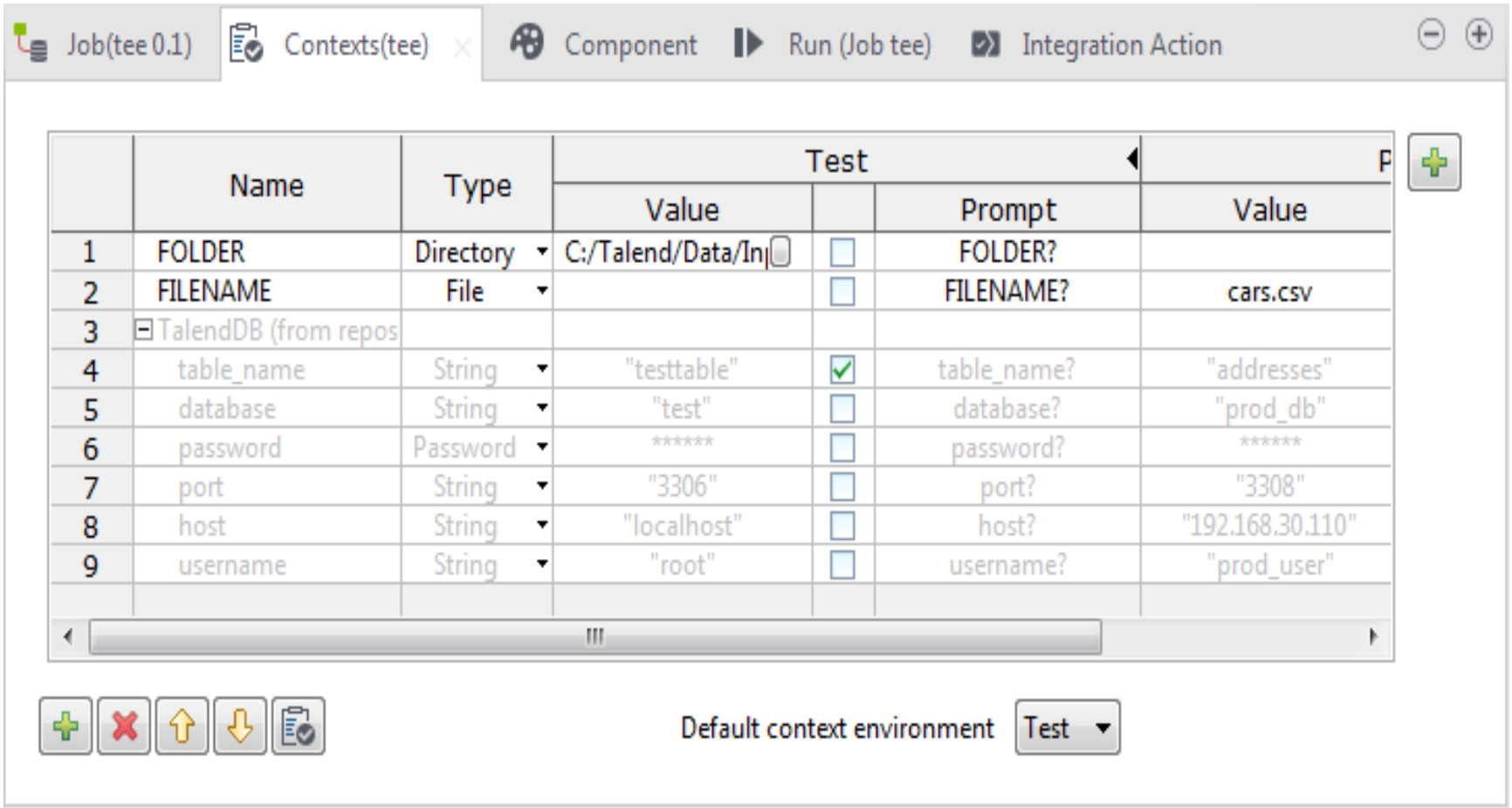
Context Parameters

- You can define context variables for a particular Job in two ways:
  - Using the Contexts view of the Job.
  - Using the F5 key from the Component view of a component.

How to define context variables in the Contexts view?

- The Contexts view is positioned among the configuration tabs below design workspace.
  - ✓ If you cannot find the Contexts view on the tab system of Talend Studio, go to Window > Show view > Talend, and select Contexts.
  - ✓ The Contexts tab view shows all of the variables that have been defined for each component in the current Job and context variables imported into the current Job.

Context Parameters (Contd..)

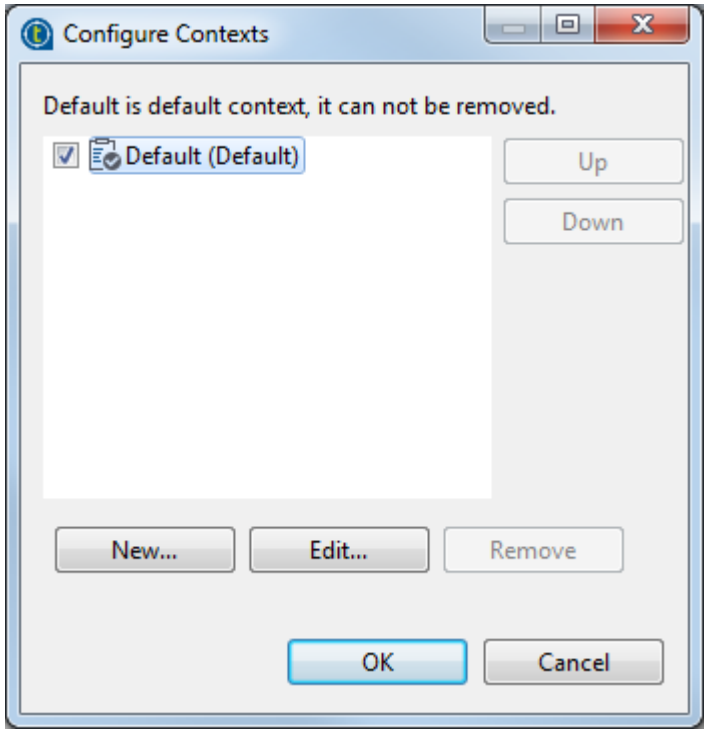


Context Parameters (Contd..)

- From this view, you can manage your built-in variables:
  - Create and manage built-in contexts.
  - Create, edit and delete built-in variables.
  - Reorganize the context variables.
  - Add built-in context variables to the Repository.
  - Import variables from a Repository context source for use in the current Job.
  - Edit Repository-stored context variables and update the changes to the Repository.
  - Remove imported Repository variables from the current Job.

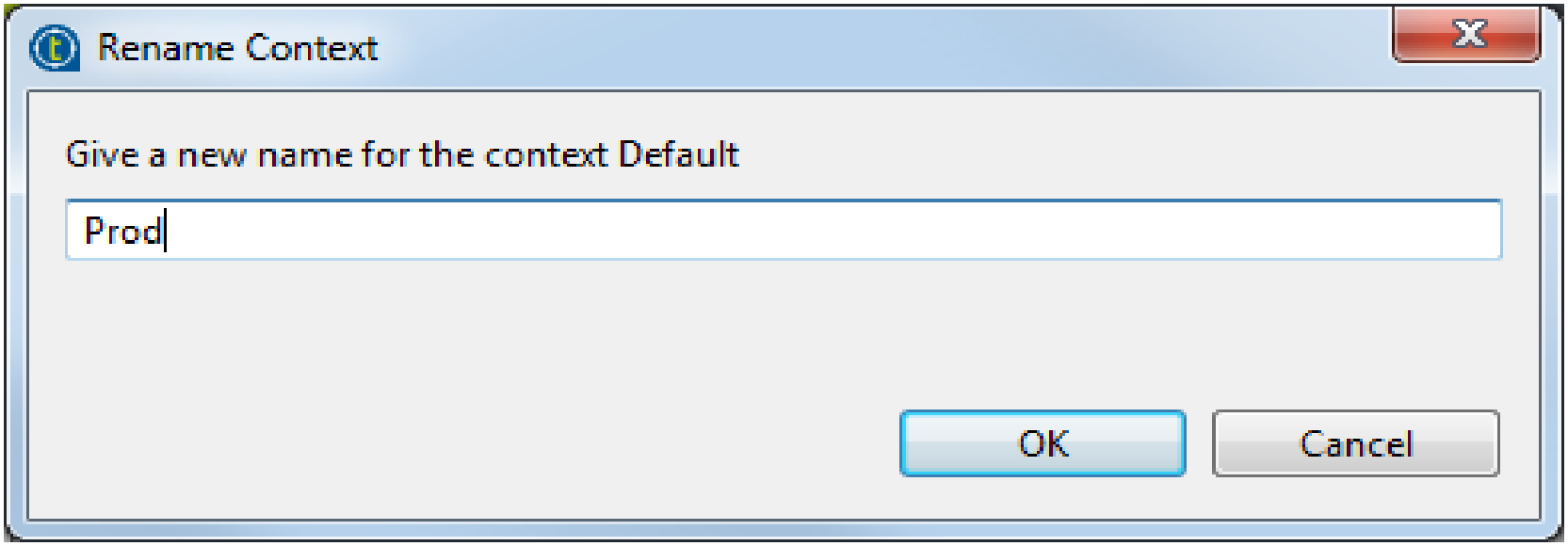
Defining Contexts

1. Open the Job in the design workspace.
2. Select the Contexts tab view and click the [+] button at the upper right corner of the view.
3. The [Configure Contexts] dialog box pops up.
4. A context named Default has been created and set as the default one by the system.



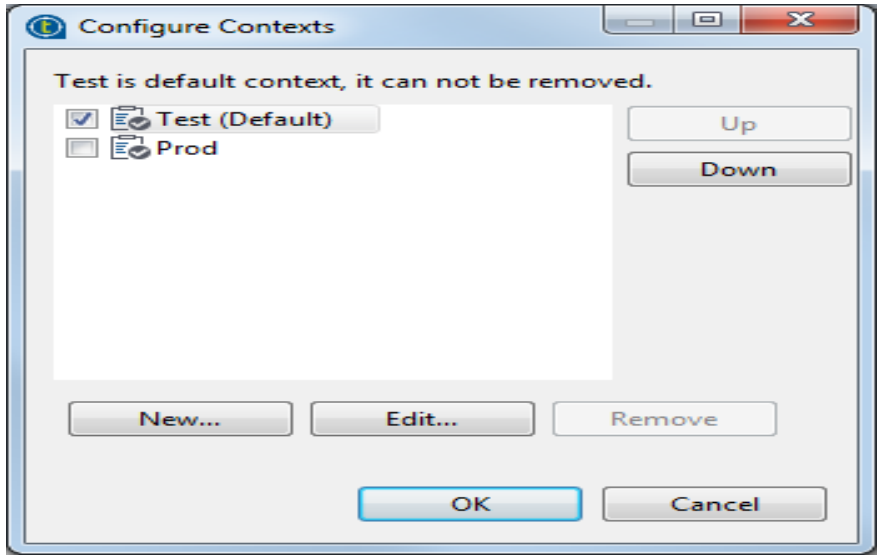
Defining Contexts (Contd..)

4. Select the context Default, click the Edit... button and enter Prod in the [Rename Context] dialog box that opens to rename the context Default to Prod.
5. Then click OK to close the dialog box.



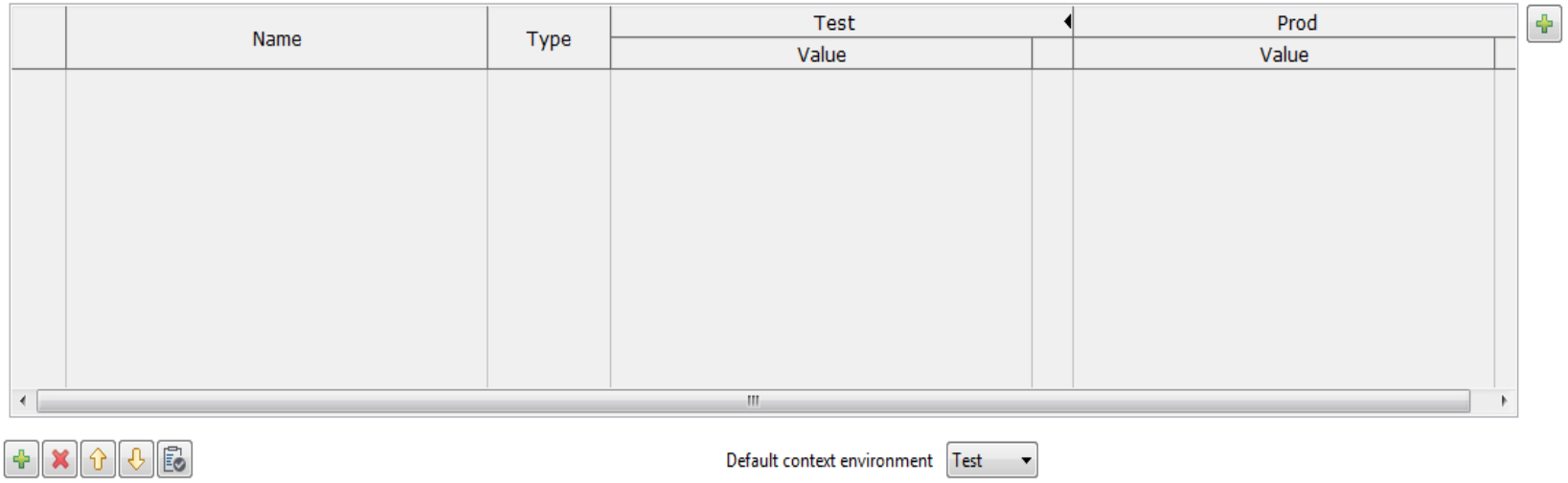
Defining Contexts (Contd..)

6. Click the New... button and enter *Test* in the [New Context] dialog box. Then click OK to close the dialog box.
7. Select the check box preceding the context you want to set as the default context. You can also set the default context by selecting the context name from the Default context environment list in the Contexts tab view.
8. If needed, move a context up or down by selecting it and clicking the Up or Down button. In this example, set Test as the default context and move it up.



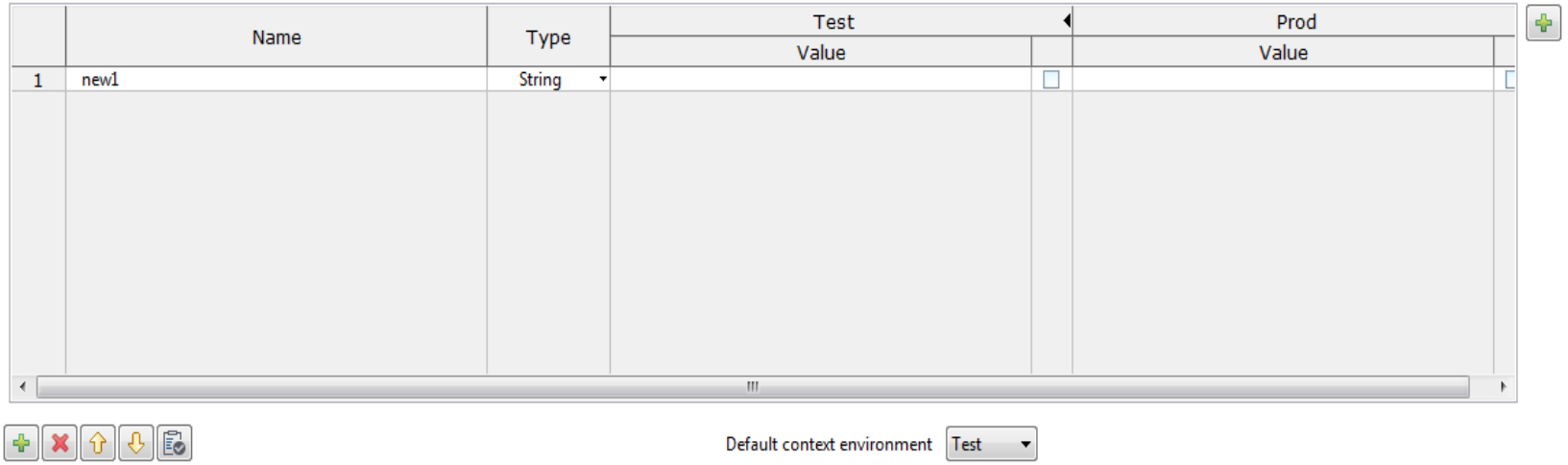
Defining Contexts (Contd..)

9. Click OK to validate your context definition and close the [Configure Contexts] dialog box.
10. The newly created contexts are shown in the context variables table of the Contexts tab view.



Defining Variables

1. Click the [+] button at the bottom of the Contexts tab view to add a parameter line in the table.



Defining Variables (Contd..)

2. Click in the Name field and enter the name of the variable you are creating, host in this example.
3. From the Type list, select the type of the variable corresponding to the component field where it will used, String for the variable host in this example.
4. For different variable types, the Value field appear slightly different when you click in it and functions differently:

Defining Variables (Contd..)

Type	Value field	Default value
String (default type)	Editable text field	null
Password	Editable text field; text entered appears encrypted.	
List of Value	Editable text field, with a button to open the [Configure Values] dialog box for list creation and configuration.	(Empty)
File	Editable text field, with a button to open the [Open] dialog box for file selection.	
Directory	Editable text field, with a button to open the [Browse for Folder] dialog box for folder selection.	
Date	Editable text field, with a button to open the [Select Date & Time] dialog box.	
Character, Double, Integer, Long, Short,Object, BigDecimal	Editable text field	
Boolean	Drop-down list box with two options: true and false	

Defining Variables (Contd..)

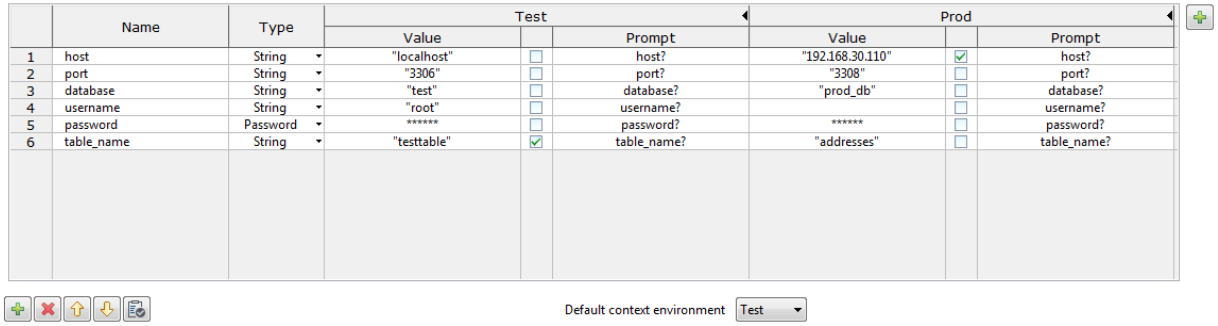
5. Click in Value field and enter the variable value under each context.

4. If needed, select the check box next the variable of interest and enter the prompt message in the corresponding Prompt field. This allows you to see a prompt for the variable value and to edit it at the execution time.

5. You can show/hide a Prompt column of the table by clicking the black right/left pointing triangle next to the relevant context name.

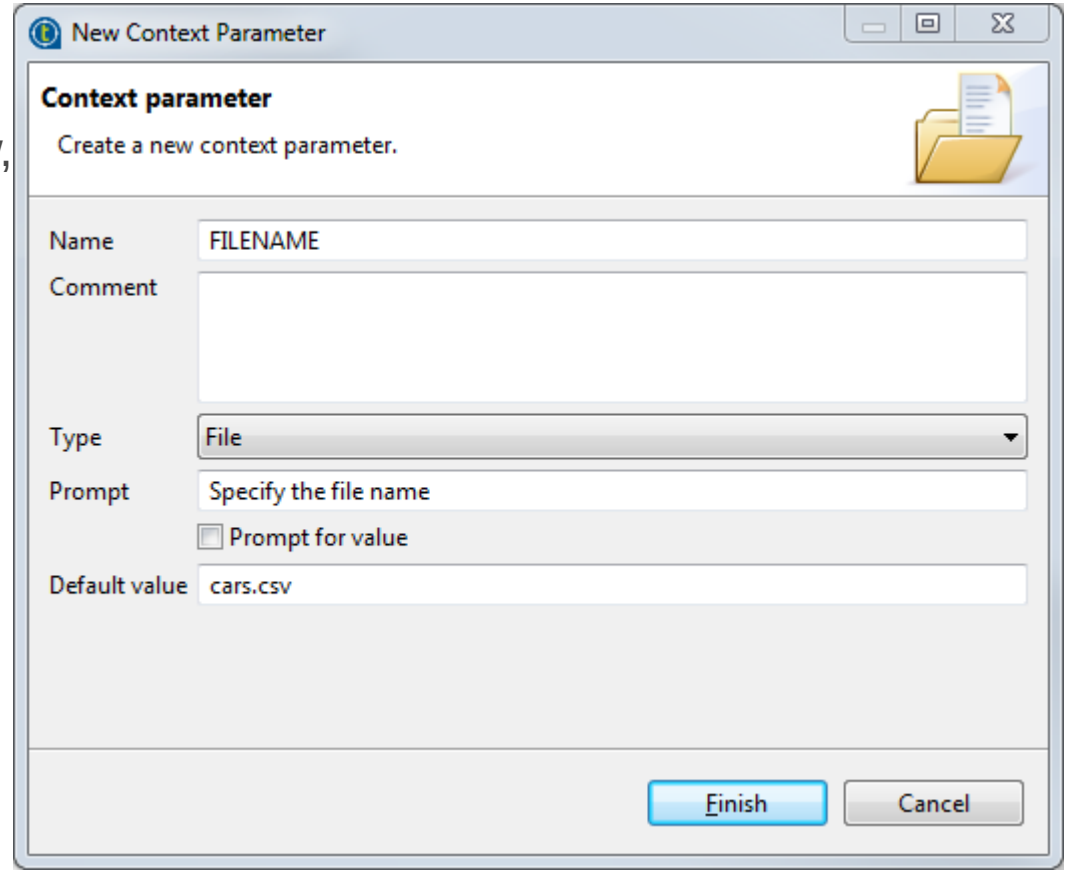
Defining Variables (Contd..)

7. Repeat the steps above to define all the variables in this example.
   - port, type String,
   - database, type String,
   - username, type String,
   - password, type Password,
   - table name, type String.
8. All the variables created and their values under different contexts are displayed in the table and are ready for use in your Job. You can further edit the variables in this view if needed.



How to define variables from the Component view?

1. The quickest way to create a single context variable is to use the F5 key from the Component view
2. On the relevant Component view, place your cursor in the field you want to parameterize
3. Press F5 to display the [New Context Parameter] dialog box



How to define variables from the Component view? (Contd..)

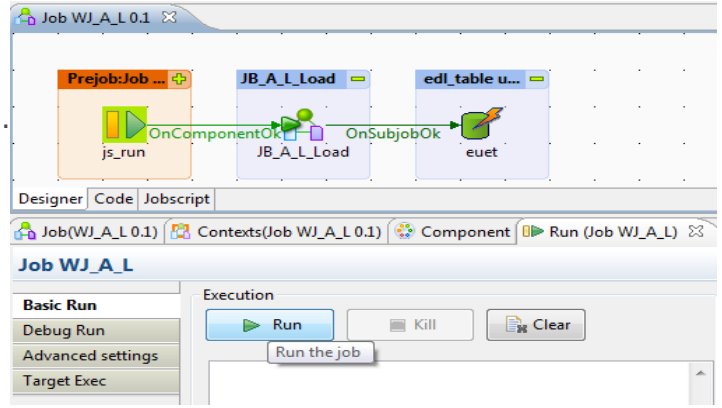
4. Give a Name to this new variable, fill in the Comment field if needed, and choose the Type.
5. Enter a Prompt to be displayed to confirm the use of this variable in the current Job execution (generally used for test purpose only), select the Prompt for value check box to display the prompt message and an editable value field at the execution time.
6. If you filled in a value already in the corresponding properties field, this value is displayed in the Default value field. Else, type in the default value you want to use for one context.
7. Click Finish to validate.
8. Go to the Contexts view tab. Notice that the context variables tab lists the newly created variables.
9. The newly created variables are listed in the Contexts view.

How to define variables from the Component view? (Contd..)

10. The variable created this way is automatically stored in all existing contexts, but you can subsequently change the value independently in each context. For more information on how to create or edit a context, see Defining contexts.

Running the Job Locally

- How a Talend Spark Job works:
  - Using the Spark-specific components, a Talend Spark Job makes use of the Spark framework to process RDDs (Resilient Distributed Datasets) on top of a given Spark cluster.
  - Depending on which framework you select for the Spark Job you are creating, this Talend Spark Job implements the Spark Streaming framework or the Spark framework when generating its code.
  - A Talend Spark Job can be run locally. The Studio builds the Spark environment in itself at runtime to run the Job locally within the Studio. With this mode, each processor of the local machine is used as a Spark worker to perform the computations. This mode requires minimum parameters to be set in this configuration view.
    - Note: this local machine is the machine in which the Job is actually run.



Processing Data

- You have used the tMap component to process date. Some other
- components that can help you to process your data are:

Name: tSortRow
Purpose:
- Sort input data based on one or more columns
- Specify sort type and order

Name: tUniqRow
Purpose: Filters out duplicate entries

Name: tAggregateRow
Purpose:
• Aggregates input data based on one or more columns
• Sample operations: min, max, avg, sum, first, last, ….

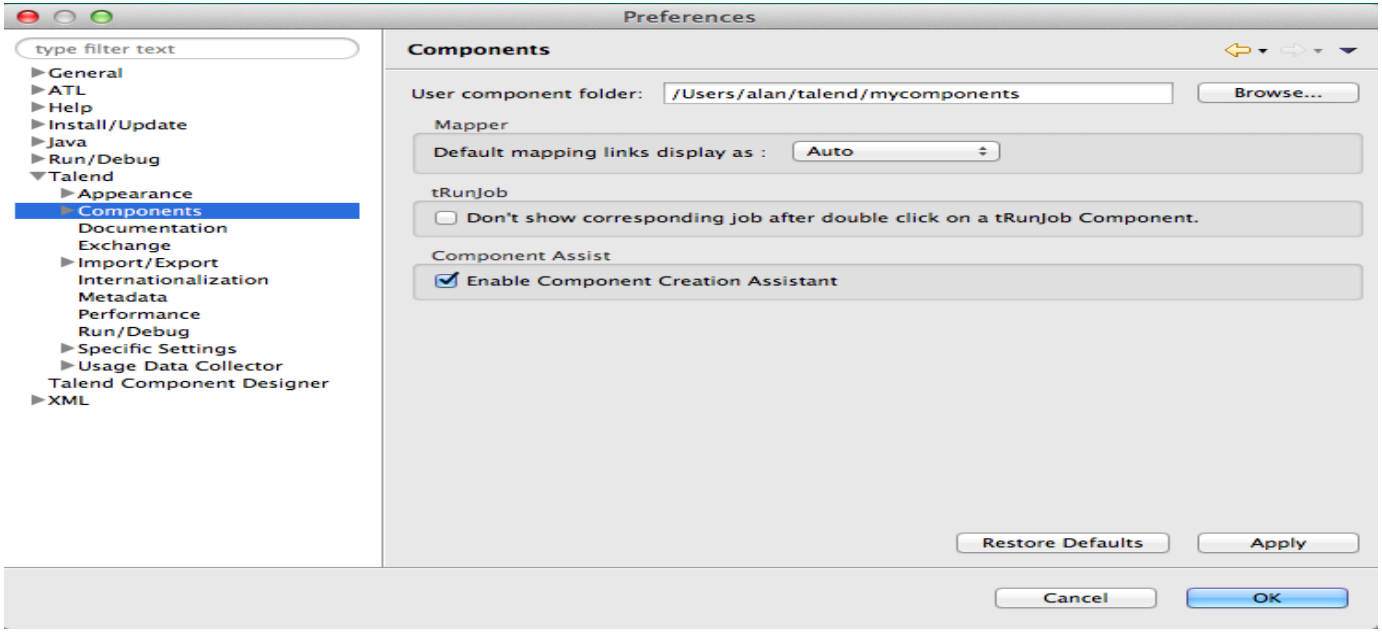
TDI – Processing and Transformation Components

Module Outline

- Transformation Components
  - tMap component
  - Selecting Specific Columsn with tFilterColumns
  - Mapping data using a filter and a simple explicit join
  - Filtering Data with tFilterRow Component
  - tReplace component
  - tSampleRow component
  - tFileCopy Component
  - tFileInputFullRow Component
  - Sorting Data using TsortRow Component
  - Merging Data using tUnite and tunique Component
- Transformation Components (Cont..)
  - tAggregateRow component
  - tRowGenerator component
  - Joining data using tJoin Component
  - tConvertType component
  - tReplicate component
  - Tsleep component

Important Components of Talend DI

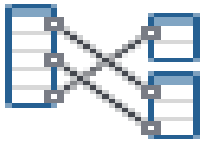
- A component is a preconfigured connector used to perform a specific data integration operation
- A component can minimize the amount of hand-coding required to work on data from multiple, heterogeneous sources.



Components of Talend DI

Components of Talend DI		
Big Data components	Databases - other components	Misc. group components
Business components	DotNET components	Orchestration components
Business Intelligence components	ELT components	Processing components
Cloud components	ESB components	System components
Custom Code components	File components	Talend MDM components
Data Quality components	Internet components	Technical components
Databases - traditional components	Logs & Errors components	XML components
Databases - appliance/ data warehouse components		

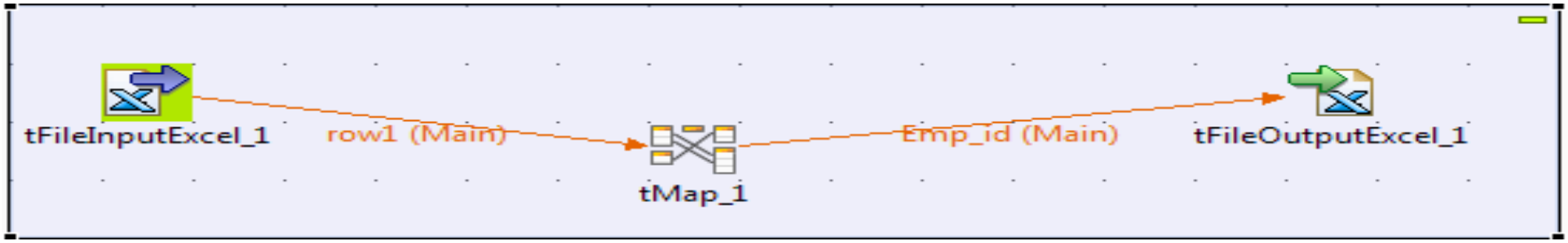
Transformation Components - tMap



- **Function**
  - **tMap** is an advanced component, which integrates itself as plugin to *Talend Studio*.
- **Purpose**
  - **tMap** transforms and routes data from single or multiple sources to single or multiple destinations.
- **Usage**
  - Possible uses are from a simple reorganization of fields to the most complex Jobs of data multiplexing or de-multiplexing transformation, concatenation, inversion, filtering and more...

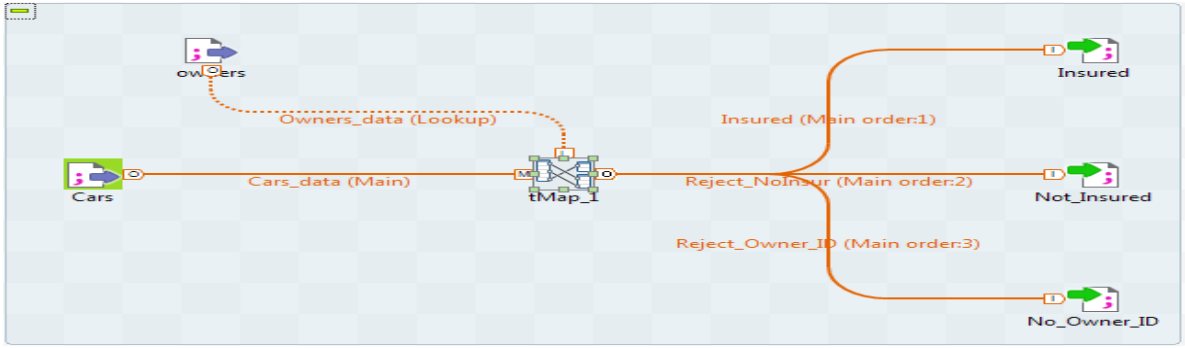
Transformation Components (Contd..)

- tMap: In this transformation **tMap** transforms and routes data from single or multiple sources to single or multiple destinations.
- In the below mentioned example I have mapped the Emp_id column of input excel with the output excel, plus I have also added function in tMap and added one more column in the output excel, which shows the insert timestamp of the record. tMap component is also mainly used to perform all your lookup tasks as well, you can also define variable and expressions in this component.



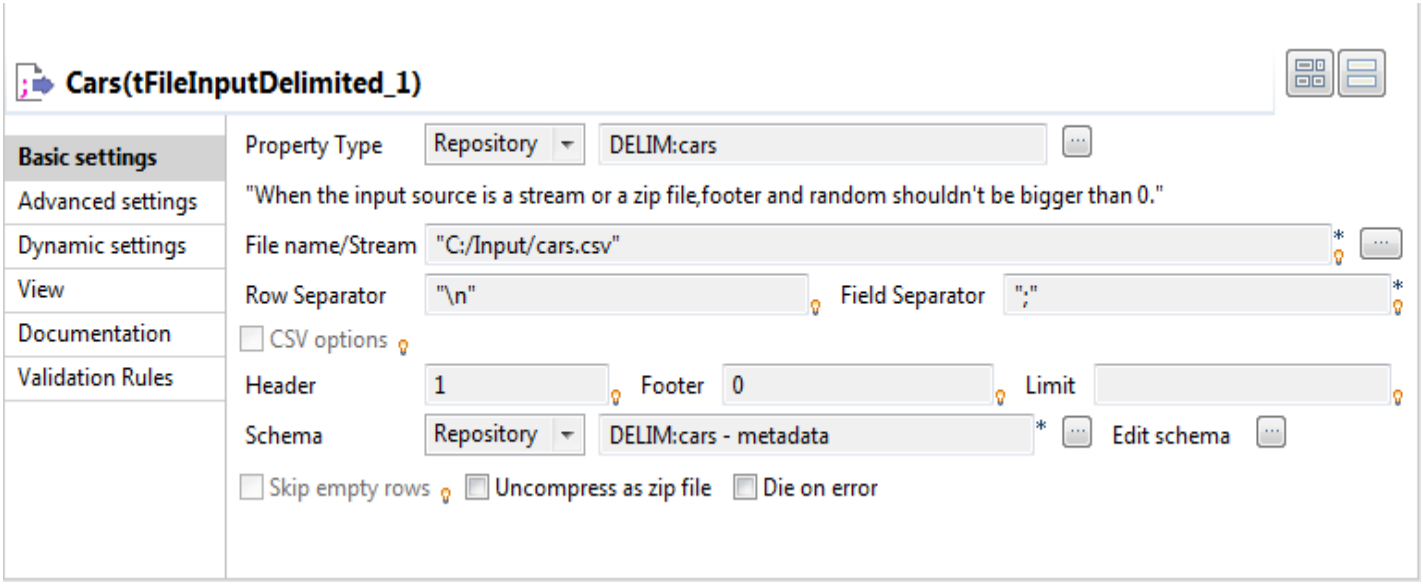
Mapping data using a filter and a simple explicit join

- **Problem Scenario:** The Job read data from a csv file with its schema stored in the Repository, looking up at a reference file, the schema of which is also stored in the Repository, then extracting data from these two files based on a defined filter to an output file and reject files.
- Step 1: Drop two **tFileInputDelimited** components, **tMap** and three **tFileOutputDelimited** components onto the design workspace.
- Connect **tMap** to the three output components using



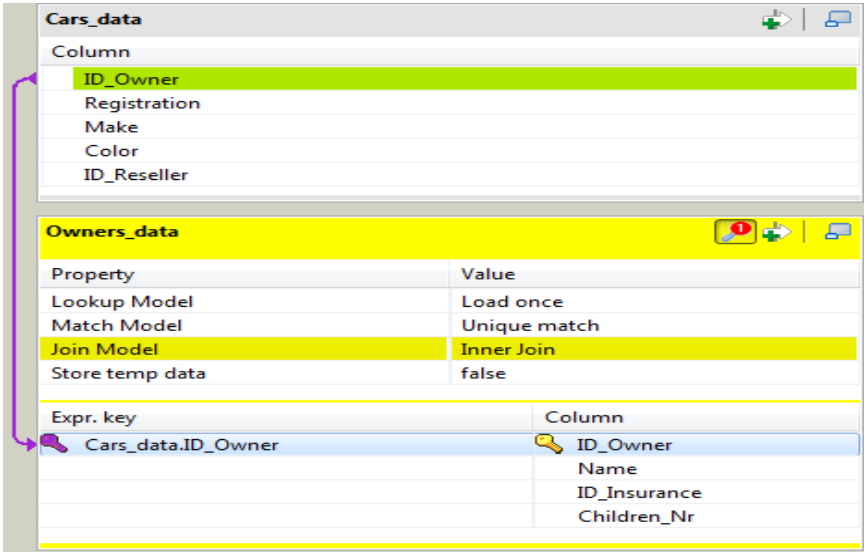
Mapping data using a filter and a simple explicit join (Cont…)

- **Step 3:** Double-click the **tFileInputDelimited** component labelled *Cars and do following configuration*



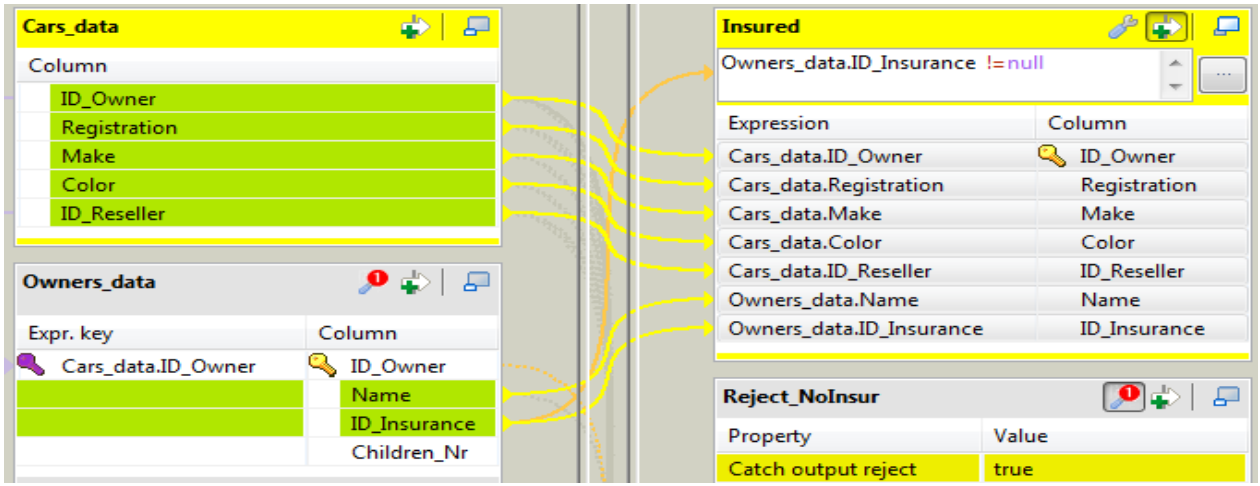
Mapping data using a filter and a simple explicit join (Cont…)

- Step 4: Double-click the **tMap** component to open the **Map Editor**.
- Step 5: Create a join between the two tables on the ***ID_Owner*** column by simply dropping the *ID_Owner* column from the *Cars_data* table onto the *ID_Owner* column in the *Owners_data* table.



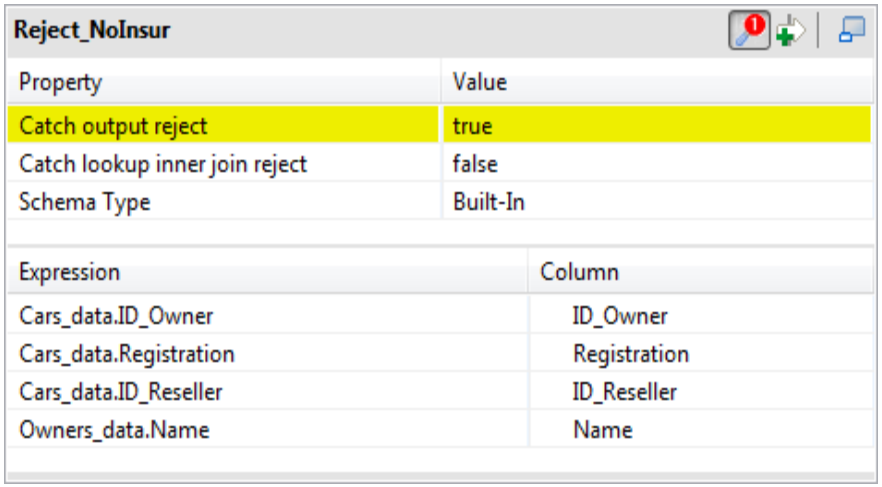
Mapping data using a filter and a simple explicit join (Cont…)

- Drag all the columns of the *Cars_data* table to the *Insured* table.
- Drag the *ID_Owner*, *Registration*, and *ID_Reseller* columns of the *Cars_data* table and the *Name* column of the *Owners_data* table to the *Reject_NoInsur* table.
- Drag all the columns of the *Cars_data* table to the *Reject_OwnerID* table.



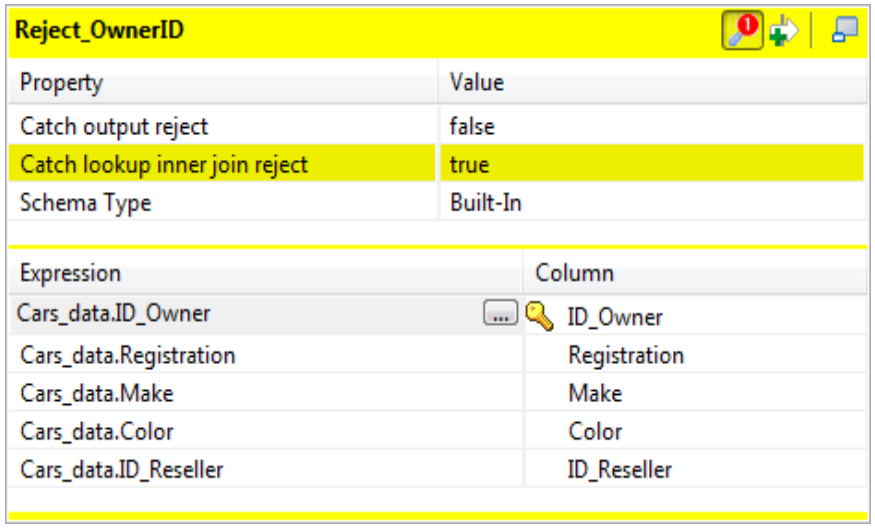
Mapping data using a filter and a simple explicit join (Cont…)

- Click the **tMap settings** button at the top of the *Reject_NoInsur* table and set **Catch output reject** to **true**



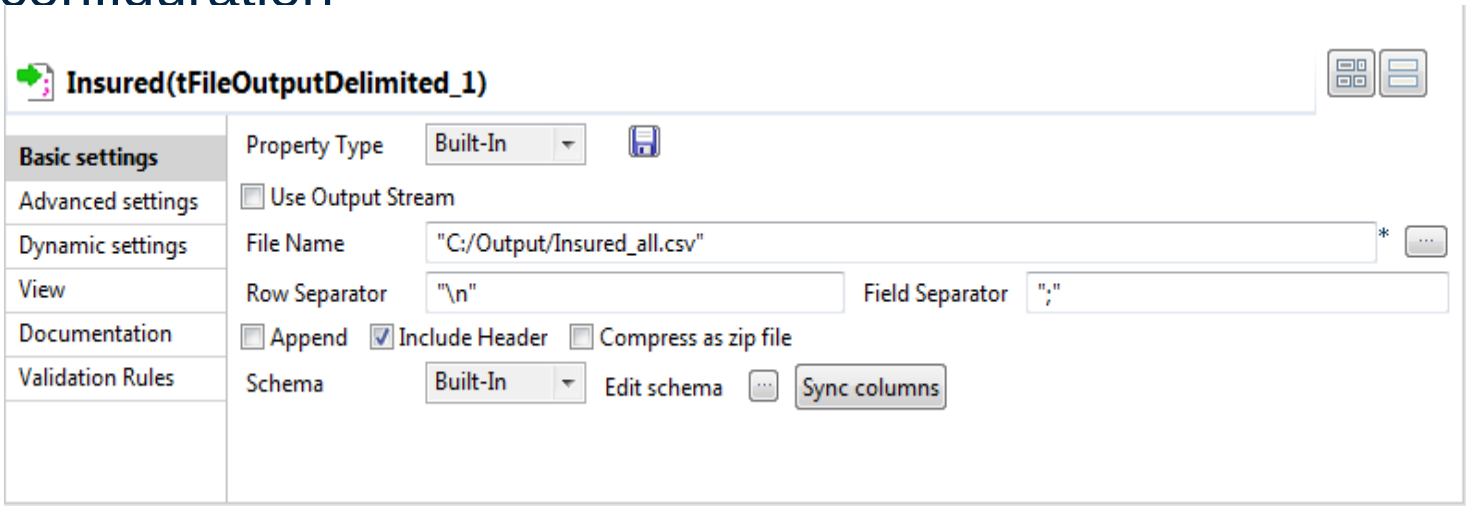
Mapping data using a filter and a simple explicit join (Cont…)

- Click the **tMap settings** button at the top of the *Reject_OwnerID* table and set **Catch lookup inner join reject** to **true**

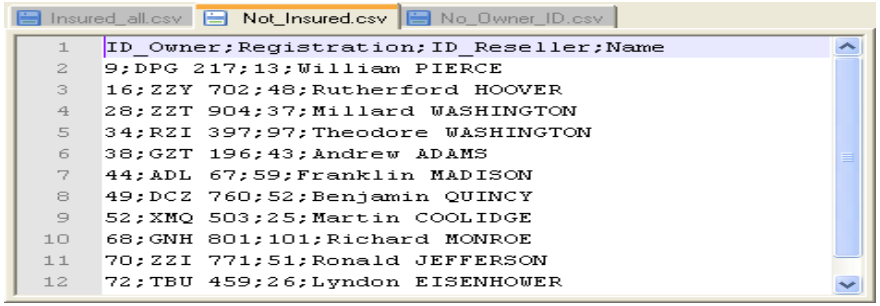


Mapping data using a filter and a simple explicit join (Cont…)

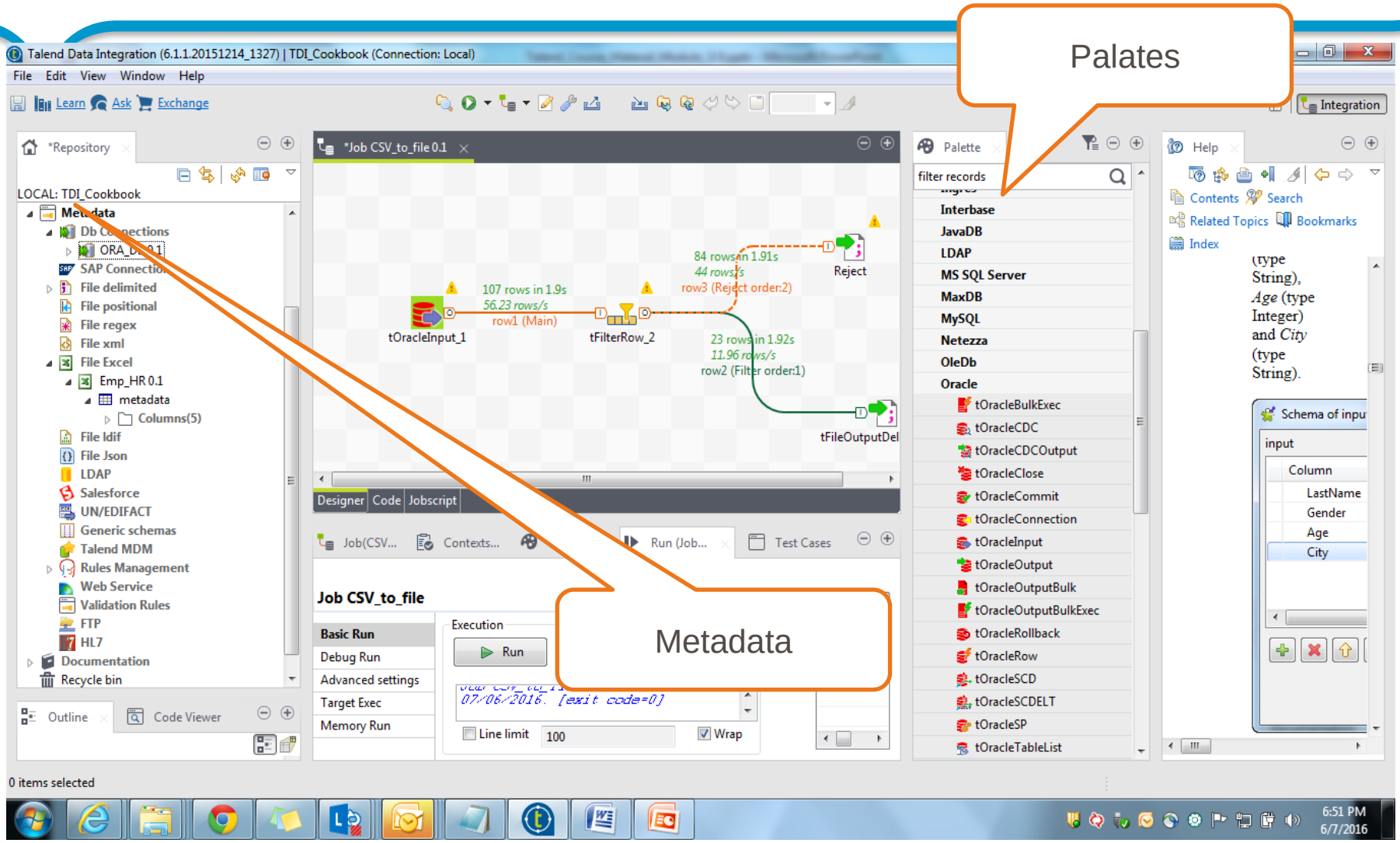
- Double-click each of the output components and do following configuration



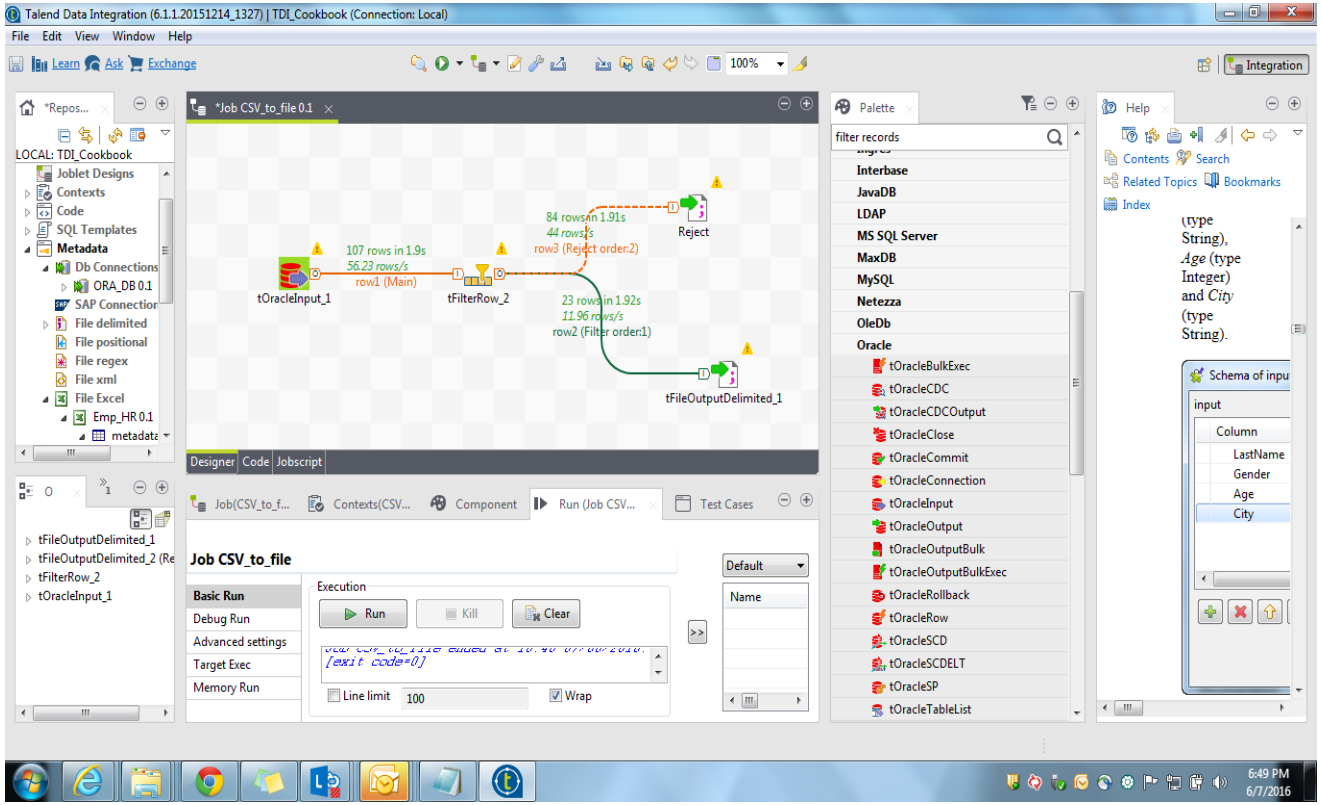
The output files are created, which contain the relevant data as defined.



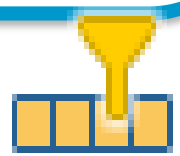
Repository



Job Design view



Filtering Data with tFilterRow – Component

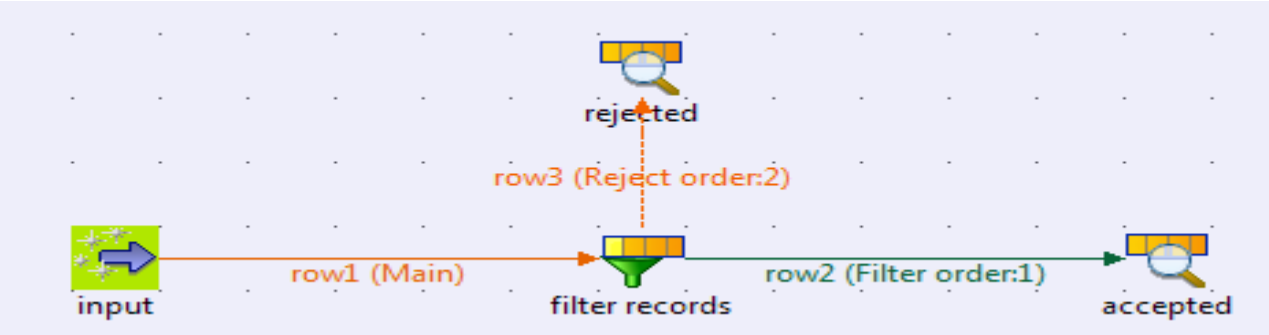


- **tFilterRow** filters input rows by setting one or more conditions on the selected columns
- **tFilterRow Properties**
  - A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component.
    - The schema is either **Built-In** or stored remotely in the **Repository**.
  - *Logical operator used to combine conditions*
    - Using AND or OR can be used
  - *Conditions -*

Setting Properties

- **tFilterRow Properties**
  - *Conditions –*
    - ***Input column:*** *Select the column of the schema the function is to be operated on*
    - ***Function:*** *Select the function on the list*
    - ***Operator:*** *Select the operator to bind the input column with the value*
    - ***Value:*** *Type in the filtered value, between quotes if needed*

Steps to Design job along with Demo



Dropping and linking components

1. Drop **tFixedFlowInput**, **tFilterRow** and **tLogRow** from the **Palette** onto the design workspace.

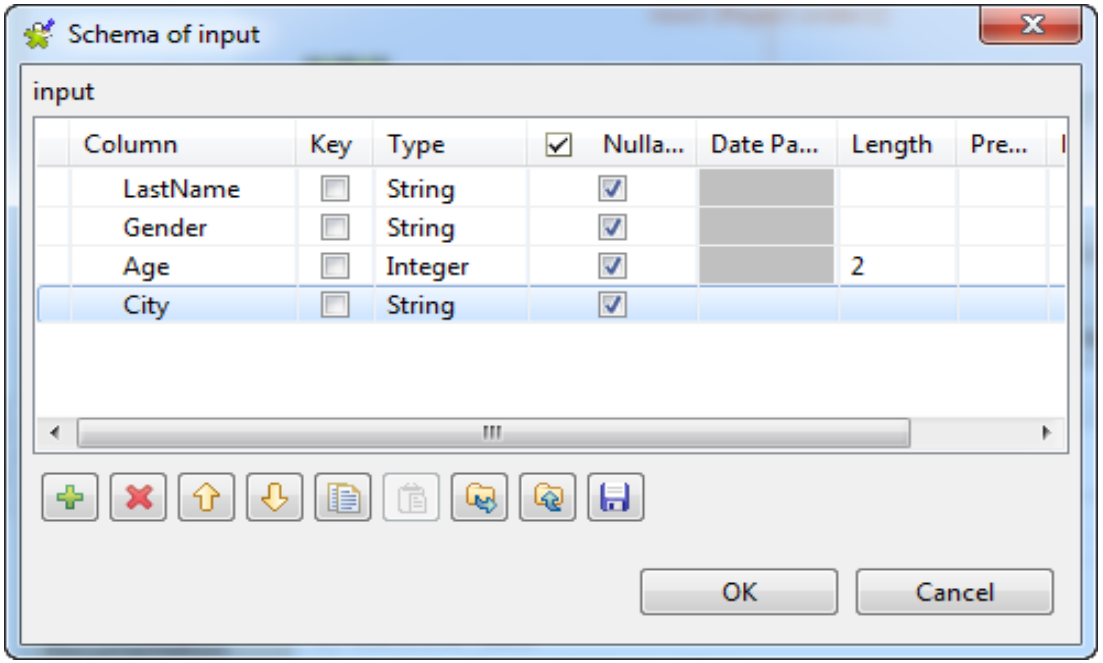
2. Connect the **tFixedFlowInput** to the **tFilterRow**, using a **Row** > **Main** link. Then, connect the **tFilterRow** to the **tLogRow**, using a **Row** > **Filter** link.

3. Drop **tLogRow** from the **Palette** onto the design workspace and rename it as *reject*. Then, connect the **tFilterRow** to the *reject*, using a **Row** > **Reject** link.

4. Label the components to better identify their roles in the Job.

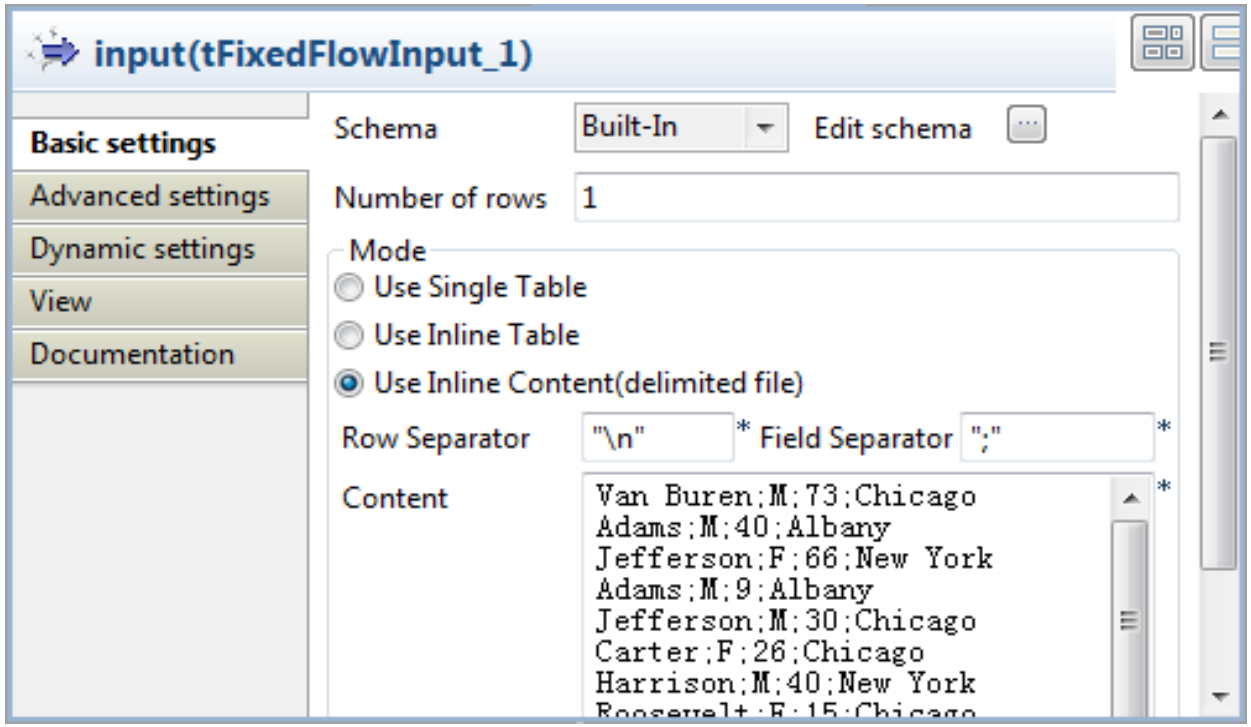
Step - 1

1. Double-click **tFixedFlowInput** to display its **Basic settings** view and define its properties



Step- 2

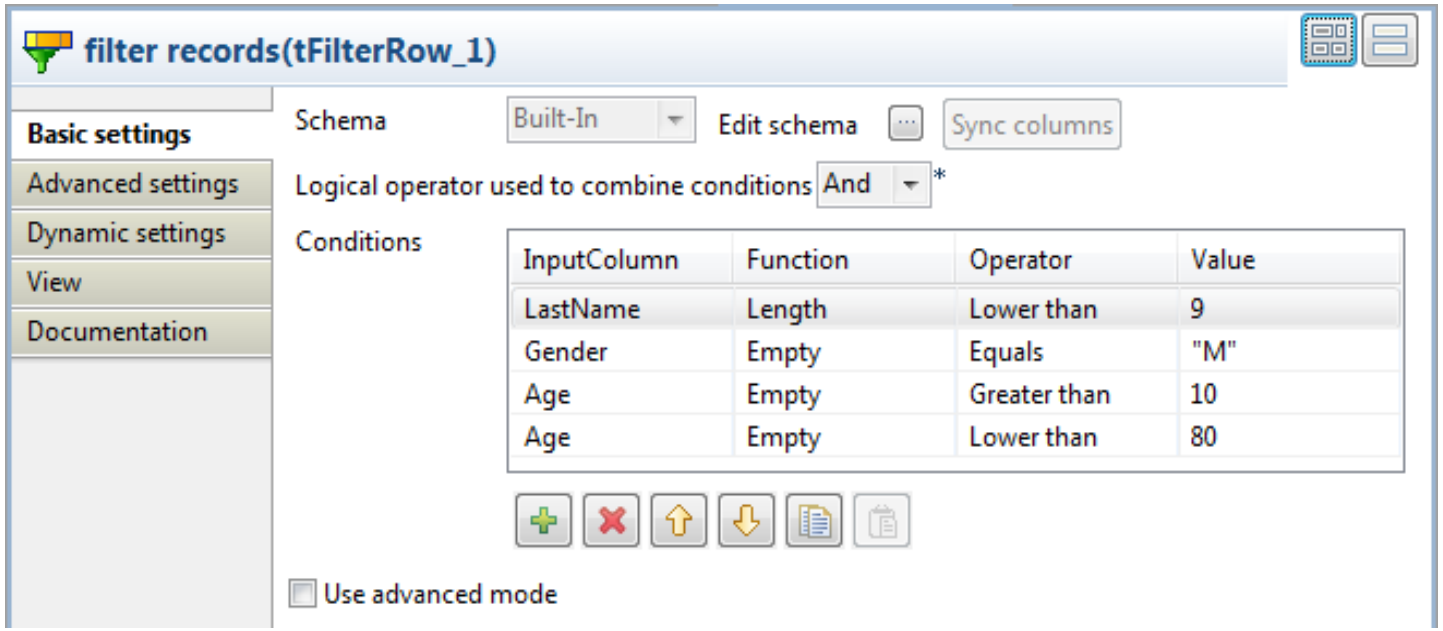
• Select the **Use Inline Content(delimited file)** option in the **Mode** area and type in the input data in the **Content** field.



Step- 3

•Double-click **tFilterRow** to display its **Basic settings** view and define its properties.

• In the **Conditions** table, add four conditions and fill in the filtering parameters.



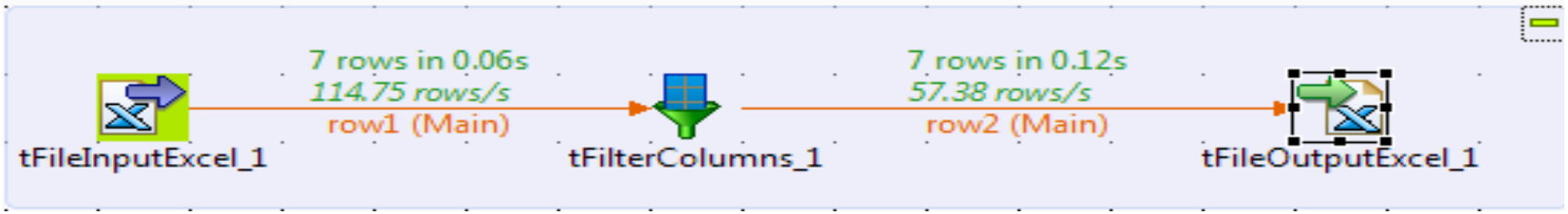
Step - 4

- Save your Job and press **F6** to execute it.

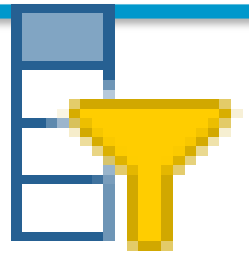
```
[statistics] connected
.--------+------+---+--------.
|          accepted           |
|=--------+------+---+-------=|
|LastName|Gender|Age|City     |
|=--------+------+---+-------=|
|Adams   |M     |40 |Albany   |
|Harrison|M     |40 |New York |
|Arthur  |M     |20 |Albany   |
|Pierce  |M     |18 |New York |
|McKinley|M     |70 |Boston   |
|Monroe  |M     |60 |Chicago  |
'--------+------+---+--------'
.---------+------+---+--------+-------------------------------------------------------------------------.
|                                        rejected                                                         |
|=--------+------+---+--------+-------------------------------------------------------------------------=|
|LastName |Gender|Age|City    |errorMessage                                                             |
|=--------+------+---+--------+-------------------------------------------------------------------------=|
|Van Buren|M     |73 |Chicago | LastName.length() < 9 failed                                            |
|Jefferson|F     |66 |New York| LastName.length() < 9 failed|Gender.compareTo("M") == 0 failed          |
|Adams    |M     |9  |Albany  |Age.compareTo(10) > 0 failed                                             |
|Jefferson|M     |30 |Chicago | LastName.length() < 9 failed                                            |
|Carter   |F     |26 |Chicago |Gender.compareTo("M") == 0 failed                                        |
|Roosevelt|F     |15 |Chicago | LastName.length() < 9 failed|Gender.compareTo("M") == 0 failed          |
|Monroe   |M     |8  |Boston  |Age.compareTo(10) > 0 failed                                             |
|Quincy   |F     |83 |Albany  |Gender.compareTo("M") == 0 failed|Age.compareTo(80) < 0 failed           |
|Coolidge |M     |4  |Chicago |Age.compareTo(10) > 0 failed                                             |
'---------+------+---+--------+-------------------------------------------------------------------------'
[statistics] disconnected
```

Transformation Components (Contd..)

- tFilterColumn: This transformation is used to remove any unwanted column from you input.
- For example in the below snapshot; In the input excel file I have 4 columns, after using the tFilterColumn I have removed the one of the columns and in the output excel there are only 3 columns



Selecting Specific Columsn with tFilterColumns – Component



- **tFilterColumns-** Makes specified changes to the schema defined, based on column name mapping

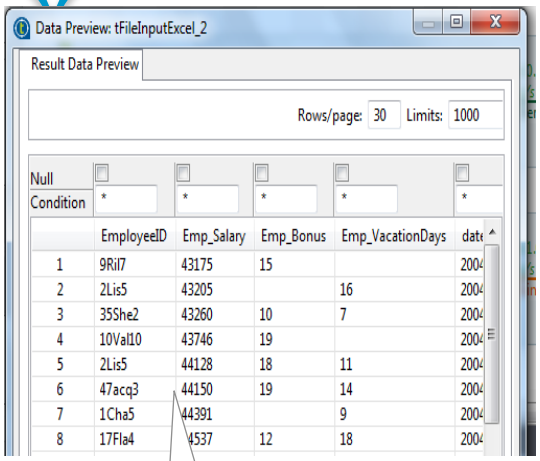
- **tFilterColumns Properties**
  - A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. The schema is either **Built-In** or stored remotely in the **Repository**.

tFilterColumns Component

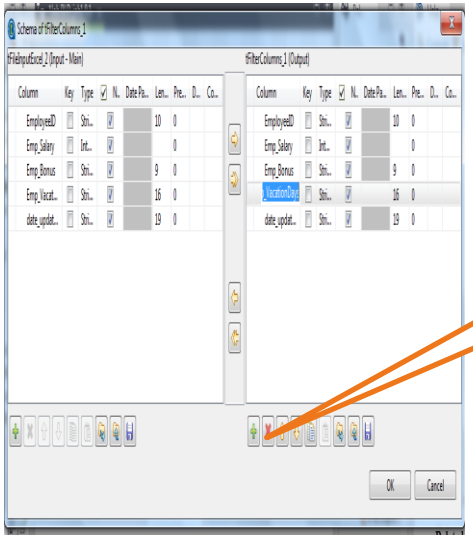
- ***Click on Schema* and *Edit***
- *Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:*
  - ***View schema: choose** this option to view the schema only.*
  - ***Change to built-in property:** choose this option to change the schema to Built-in for local changes.*
  - ***Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion.*
  - *If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the Repository Content.*

Demo Job

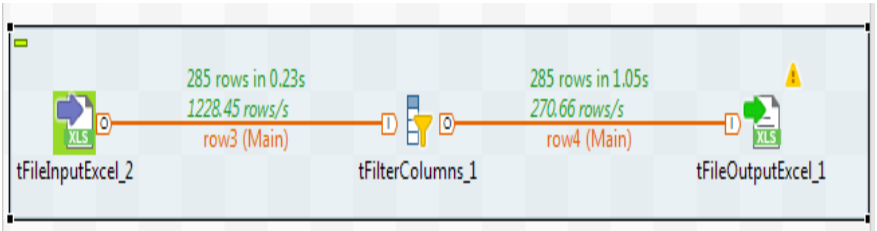
Click on Schema and Edit Columns



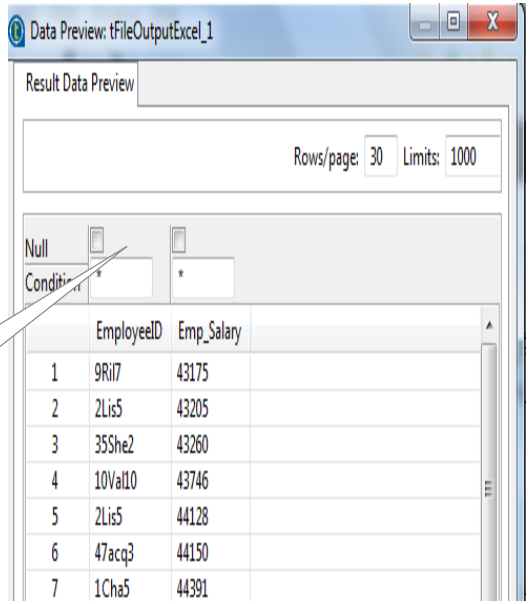
Input Columns



Select target column and delete



Output Columns



Transformation Components- tReplace

Function

Carries out a Search & Replace operation in the input columns defined.

Purpose

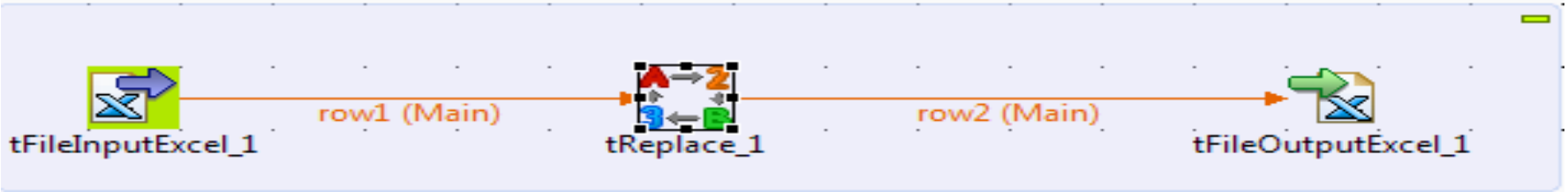
Helps to cleanse all files before further processing.

Usage

This component is not startable as it requires an input flow. And it requires an output component.

Transformation Components- tReplace (cont..)

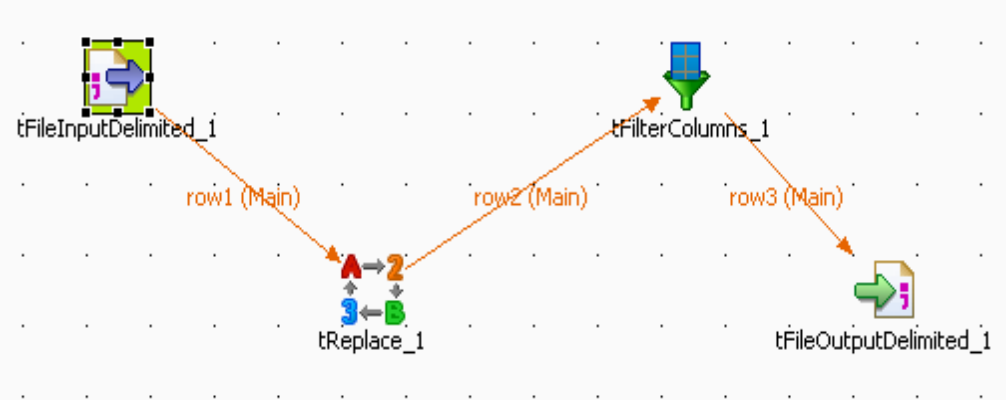
- tReplace: In this transformation tReplace we can search and replace records in the given input.
- In the below mentioned snapshot I am passing one excel as input . By using this transformation we replace a specific records with any other value. Like here I am searching for the record ‘Andheri’ in the column Emp_Address and if it’s found then it is replaced with the value ‘Kandivali’



Search/Replace

InputColumn	Search	Replace with	Whole w...	Ca
Emp_Address	"Andheri"	"Kandivali"	☑	☐

Transformation Components- tReplace (cont..)



Input Data

Street	FirstName	Name	Amount
streat	John	Kennedy	98.30$
streat	Richad	Nikson	78.23$
streat	Richard	Nikson	78.2$
streat	toto	Nikson	78.23$
streat	Richard	Nikson	78.23$
street	Georges *t	bush	99.99$

1. Setting Component properties in tFileInputDelimited

Property Type: Built-In

File Name: 'D:/Input/replace.csv' *

Row Separator: "\n" Field Separator: ";"

Header: 0 Footer: 0 Limit:

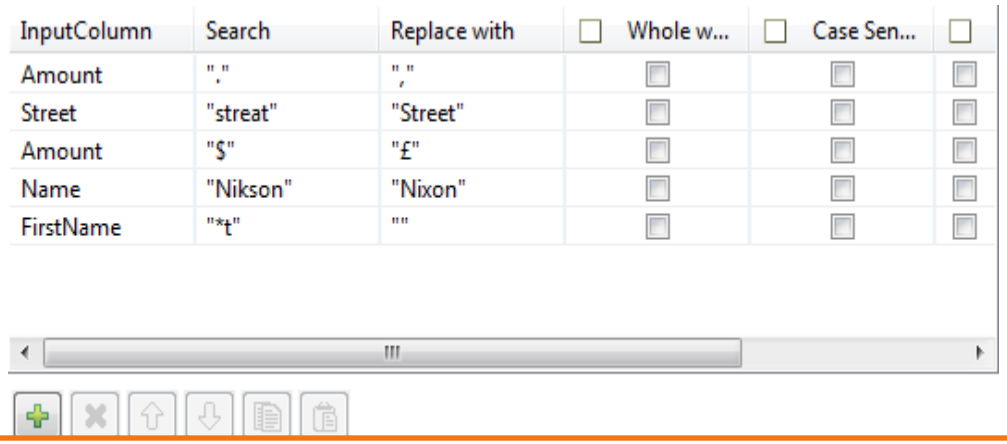
Schema: Built-In Edit schema Skip empty rows

Extract lines at random

Encoding Type: ISO-8859-15

Configuring tReplace and tFilter Columns Components

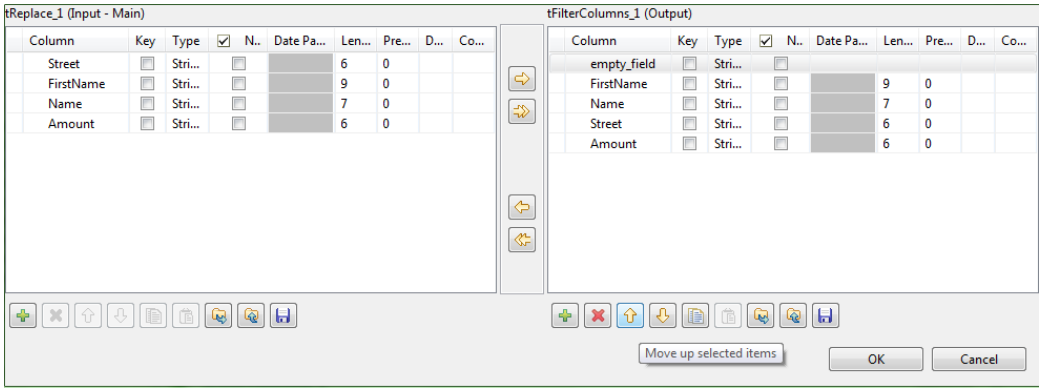
2. Double Click on **tReplace** component to set the search & replace parameters.



Input Data

Street	FirstName	Name	Amount
streat	John	Kennedy	98.30$
streat	Richad	Nikson	78.23$
streat	Richard	Nikson	78.2$
streat	toto	Nikson	78.23$
streat	Richard	Nikson	78.23$
street	Georges *t	bush	99.99$

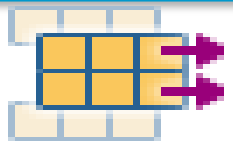
3. Select the next component in the Job, **tFilterColumn to** take required columns



Output Data

	John	Kennedy	Street	98,30£
	Richad	Nixon	Street	78,23£
	Richard	Nixon	Street	78,2£
	toto	Nixon	Street	78,23£
	Richard	Nixon	Street	78,23£
	Georges	bush	street	99,99£

tSampleRow components



Function

tSampleRow filters rows according to line numbers.

Purpose

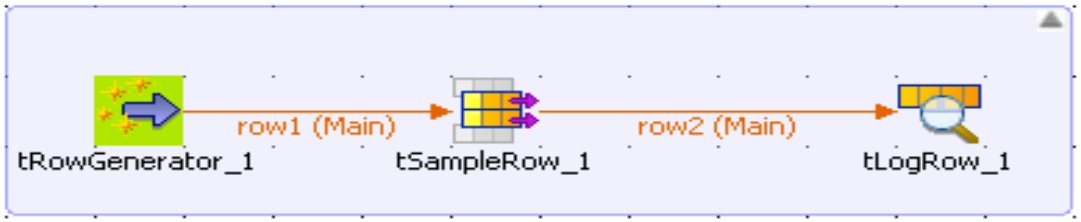
tSampleRow helps to select rows according to a list of single lines and/or a list of groups of lines.

Usage

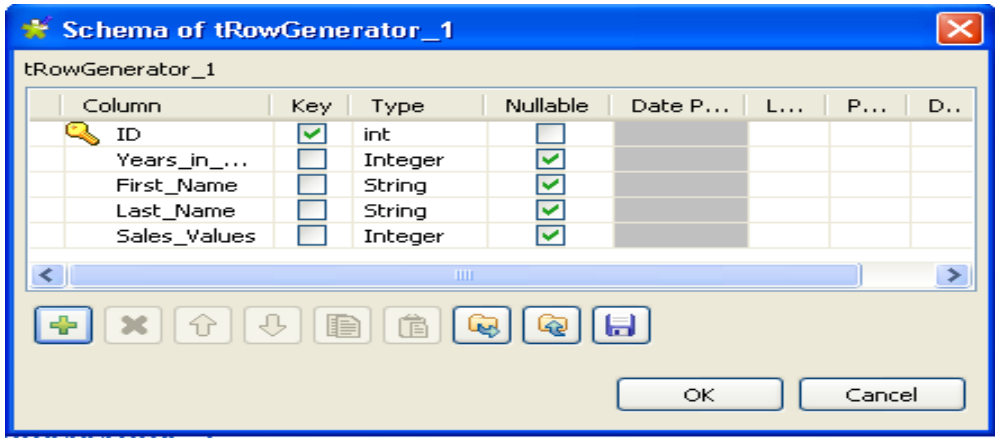
This component handles flows of data therefore it requires input and output components.

tSampleRow components (cont…)

1. Drop the following components from the **Palette** onto the design workspace: **tRowGenerator**, **tSampleRow**, and **tLogRow**.

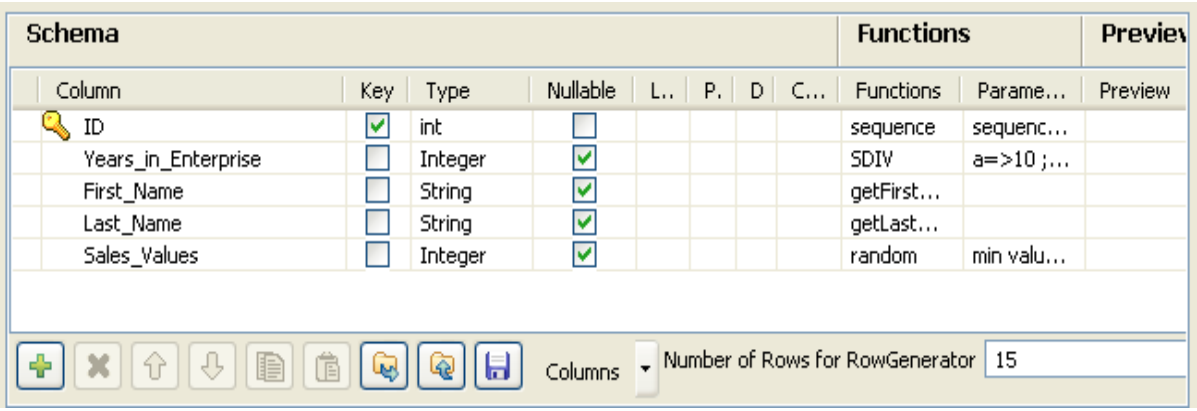


2. select **tRowgenerator**, and click the **Component** tab to define the basic settings for **tRowGenerator**.

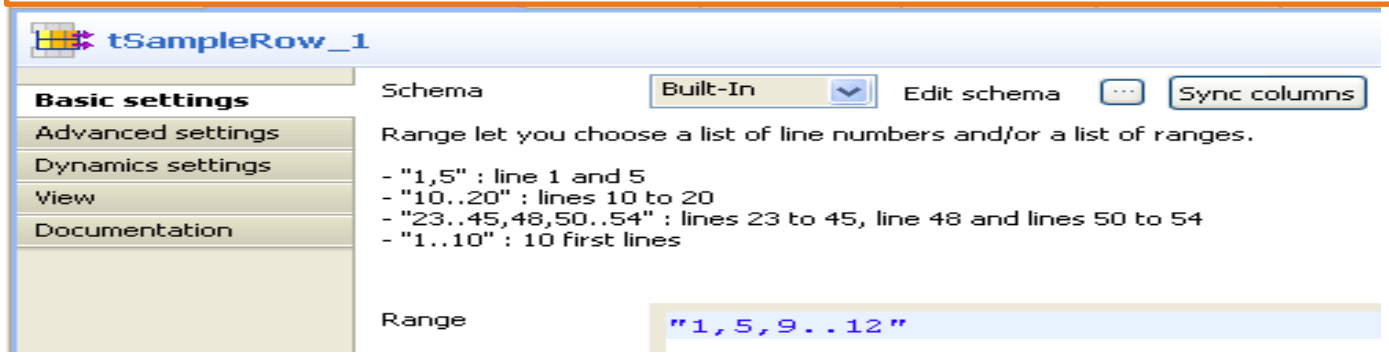


tSampleRow components (cont…)

3. In the **RowGenerator Editor**, specify the number of rows to be generated in the **Number of Rows for RowGenerator** field



4. select **tSampleRow** and click the **Component** tab to define the basic settings for **tSampleRow**.



tSampleRow components (cont…)

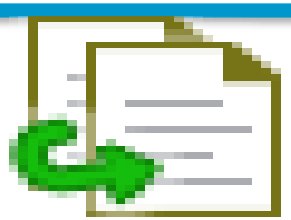
5. Press **Ctrl+S** to save your Job.

6. Press **F6**, or click **Run** on the **Run** tab to execute the Job.

```
Starting job sample_Row at 11:16 20/08/2008.
.--+-------------------+---------+---------+----------.
|                        tLogRow_1                      |
|=-+-------------------+---------+---------+----------=|
|ID|Years_in_Enterprise|First_Name|Last_Name|Sale_Values|
|=-+-------------------+---------+---------+----------=|
|1 |1                  |Martin    |Taft     |16        |
|5 |1                  |Franklin  |Adams    |45        |
|9 |1                  |William   |Madison  |15        |
|10|1                  |George    |Kennedy  |47        |
|11|1                  |George    |Van Buren|16        |
|12|1                  |Theodore  |Grant    |68        |
'--+-------------------+---------+---------+----------'

Job sample_Row ended at 11:16 20/08/2008. [exit code=0]
```

tFileCopy Component



Function

This component copies a source file or folder into a target directory.

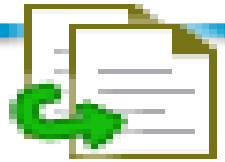
Purpose

This component helps to streamline processes by automating recurrent and tedious copy tasks.

Usage

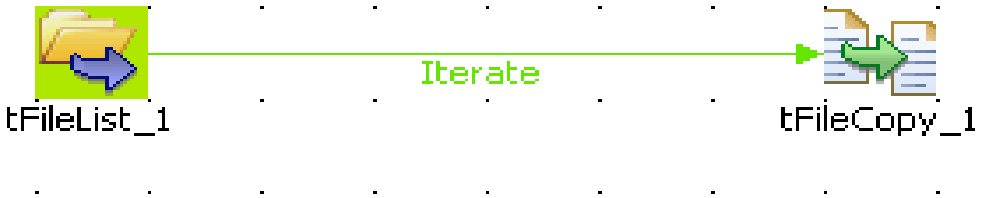
This component can be used as a standalone component.

tFileCopy Component (cont…)



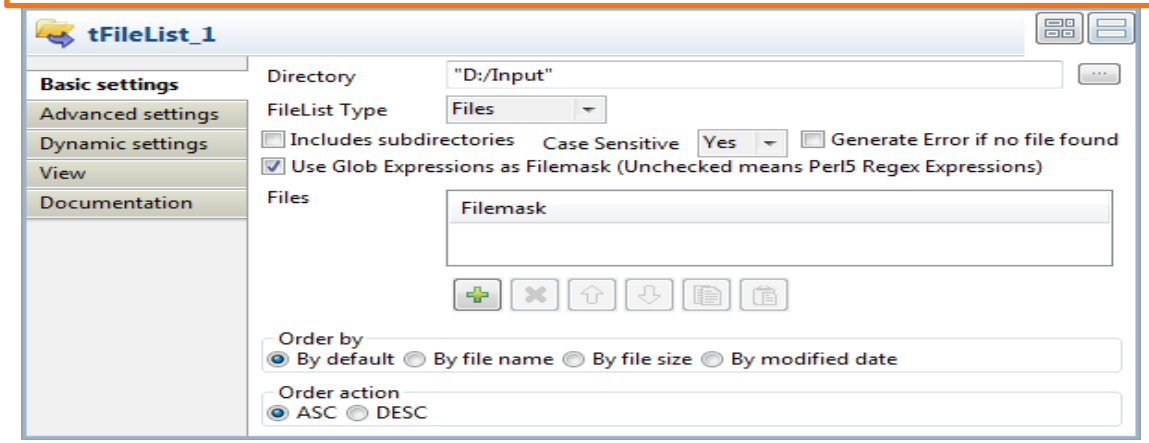
1. Add a **tFileList** component and a **tFileCopy** component by typing their names in the design workspace or dropping them from the **Palette**.

2. Connect **tFileList** to **tFileCopy** using a **Row** > **Iterate** link.

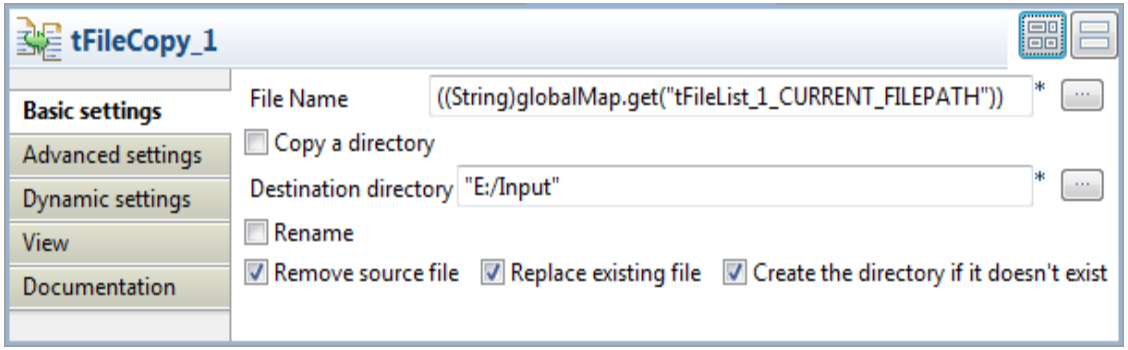


tFileCopy Component (cont…)

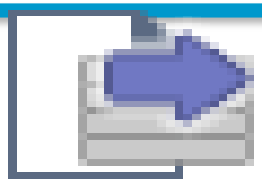
3. Double-click **tFileList** to open its **Basic settings** view.



4. Double-click **tFileCopy** to open and configure its **Basic settings.**



tFileInputFullRow Component



Function

tFileInputFullRow reads a given file row by row.

Purpose

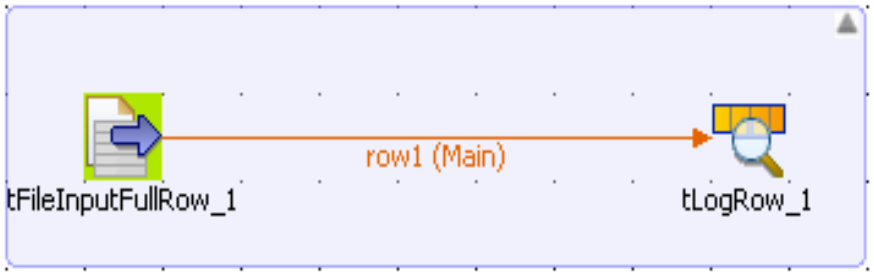
tFileInputFullRow opens a file and reads it row by row and sends complete rows as defined in the Schema to the next Job component, via a Row link.

Usage

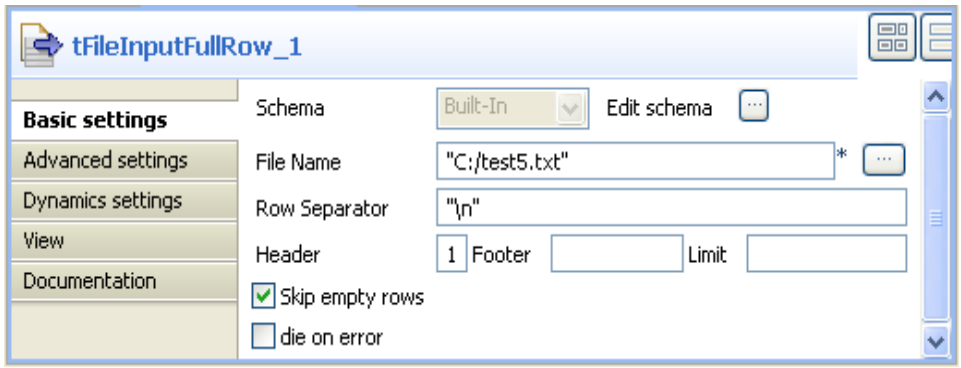
Use this component to read full rows in delimited files that can get very large. You can also create a rejection flow using a **Row > Reject** link to filter the data which does not correspond to the type defined.

tFileInputFullRow Component (cont…)

1. Drop a **tFileInputFullRow** and a **tLogRow** from the **Palette** onto the design workspace.

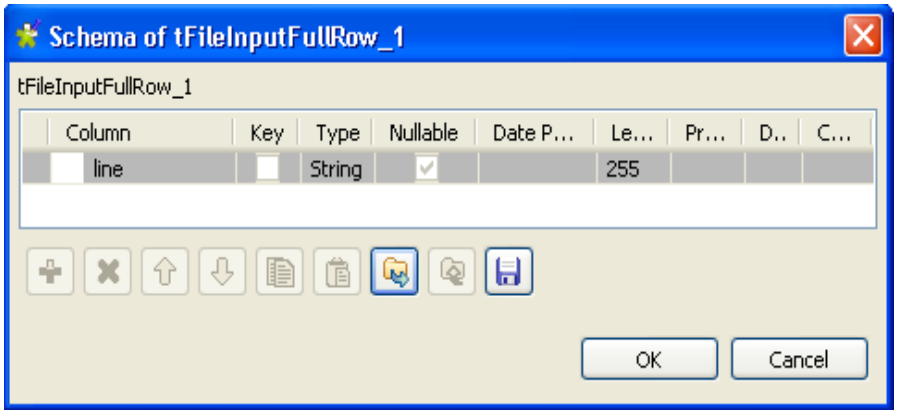


2. In the design workspace, select **tFileInputFullRow** and set components



tFileInputFullRow Component (cont…)

3. The **Basic settings** view, set **Schema** to **Built-In**. it is read only



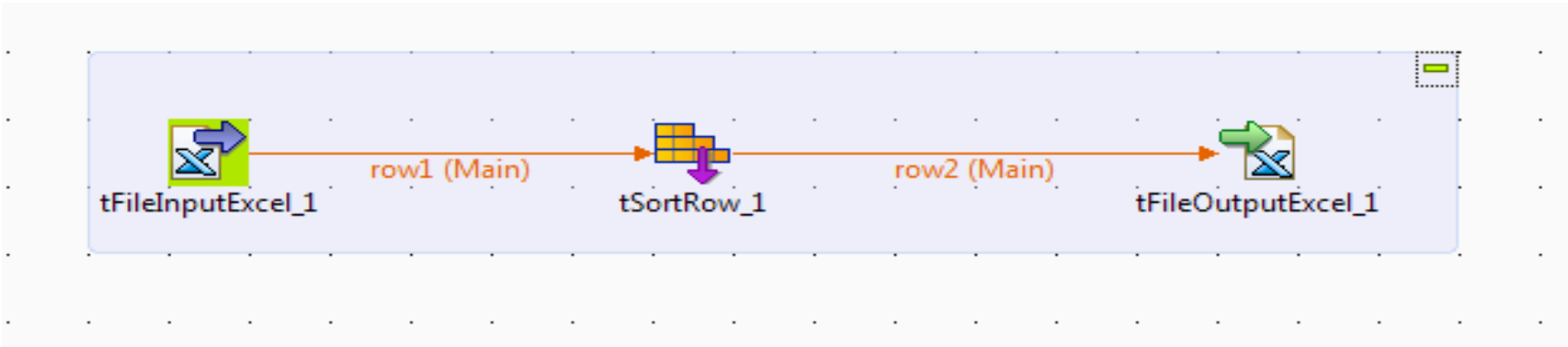
4. tFileInputFullRow reads the three rows one by one ignoring field separators, and the complete rows are displayed on the **Run** console.

```
Starting job Input_Full_Row at 11:13 26/08/2008.
.--------------------.
|     tLogRow_1      |
|=-------------------=|
|line                |
|=-------------------=|
|Janet,Anderson;I9988|
|Martin,Chairman;9889|
|Lily,Massy;9988     |
'--------------------'

Job Input_Full_Row ended at 11:14 26/08/2008. [exit
```

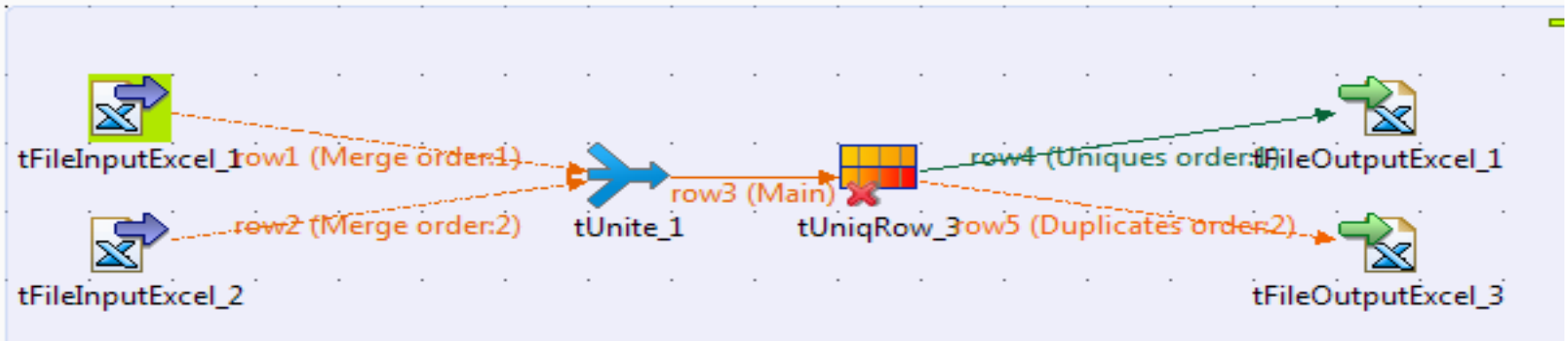
Sorting Data using- TsortRow Component

- tSortRow: This transformation component is used for sorting the input based on one or several columns by sort type.
- In the below snapshot I am passing an excel as an input and applying the sorting transformation to sort the data in the excel, and the sorted output is transferred to other excel.



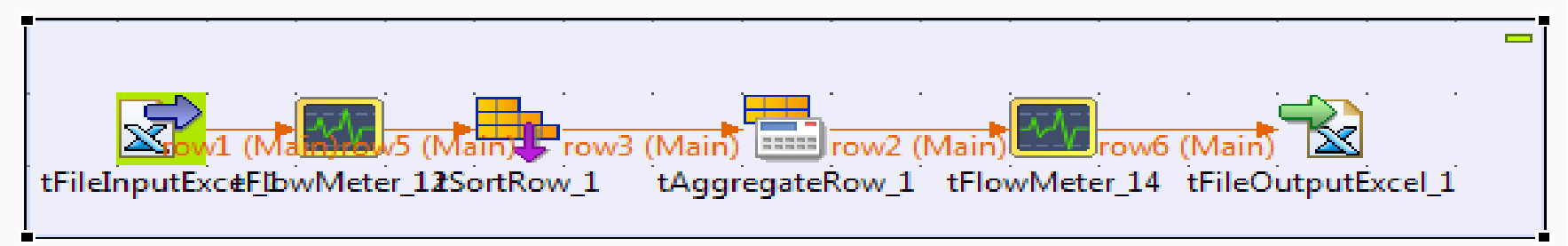
Merging Data using tUnite and tunique Component

- tUnite & tUniqRow: The **tUnite** transformation is used for merging the data from various sources, based on a common schema. The **tUniqRow** transformation is used to remove duplicate records.
- In the below snapshot I have merged the records from two different excels and have applied the tUniqrow transformation on the output and have split the output in two excels one containing the duplicate rows and other one containing the unique



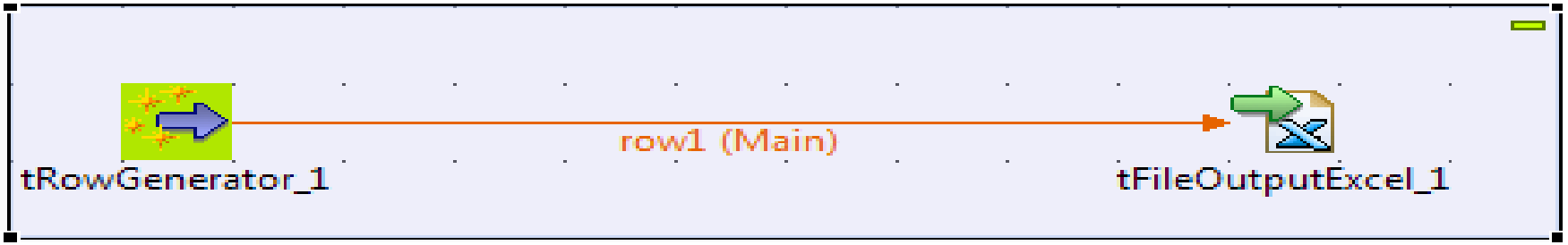
Aggregating data using tAgrregateRow and tFlowMeter

- tAggregateRow & tFlowMeter: The transformation **tAggregateRow** is used to perform aggregate operations such as Count, Min, Max, Avg, Sum Etc.
- The transformation **tFlowMeter** is used to count the number of rows processed in the defined flow.
  - ✓ In the below example passing an Excel as a input having few employee related records, first it will sort the records using **tSortRow** transformation based on the Emp_ID, further aggregate operations such as count, min and max are performed on the sorted input and output is finally stored in the excel.
  - ✓ Here the tFlowMeter between the input file and tSortRow will give the number of records passed to the tSortRow transformation.



Transformation Components – tRowGenerator

- tFilterRow: In this transformation **tRowGenerator** generates as many rows and fields as are required using random values taken from a list.
- There is an option to define functions to generate the records.

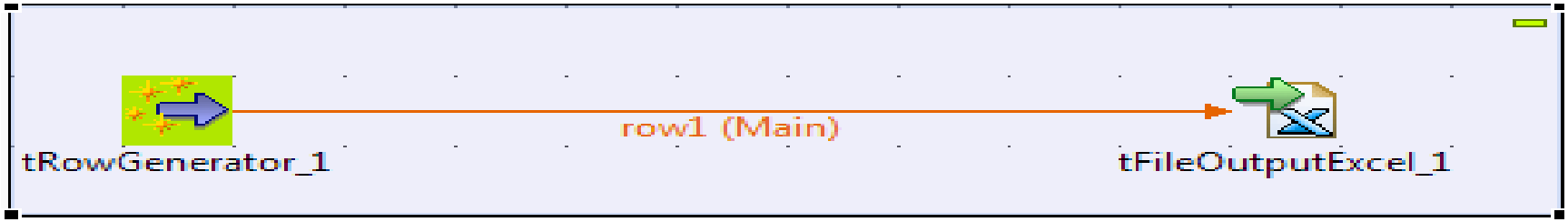


Schema | **Functions**

Column	Key	Type	N..	Length	Precision	Default	Functions	Environ...
Number	☐	Integer	☑	3			Mathematical.NUM	=>"1" ;
Text	☐	String	☑	20			DemoRoutines.helloE...	
Date	☐	String	☑	10			DataOperation.DTX	i=>1 ;
Flag	☐	Boolean	☑	5			ngHandling.IS_ALPHA	input=>...

Transformation Components – tRowGenerator (Cont…)

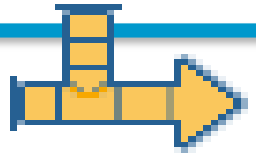
- tRowGenerator: In this transformation **tRowGenerator** generates as many rows and fields as are required using random values taken from a list.
- There is an option to define functions to generate the records.



Schema | **Functions**

Column	Key	Type	N..	Length	Precision	Default	Functions	Environ...
Number	☐	Integer	☑	3			Mathematical.NUM	=>"1" ;
Text	☐	String	☑	20			DemoRoutines.helloE...	
Date	☐	String	☑	10			DataOperation.DTX	i=>1 ;
Flag	☐	Boolean	☑	5			ngHandling.IS_ALPHA	input=>...

Joining data **using tJoin** Component



Function

tJoin joins two tables by doing an exact match on several columns. It compares columns from the main flow with reference columns from the lookup flow and outputs the main flow data and/or the rejected data.

Purpose

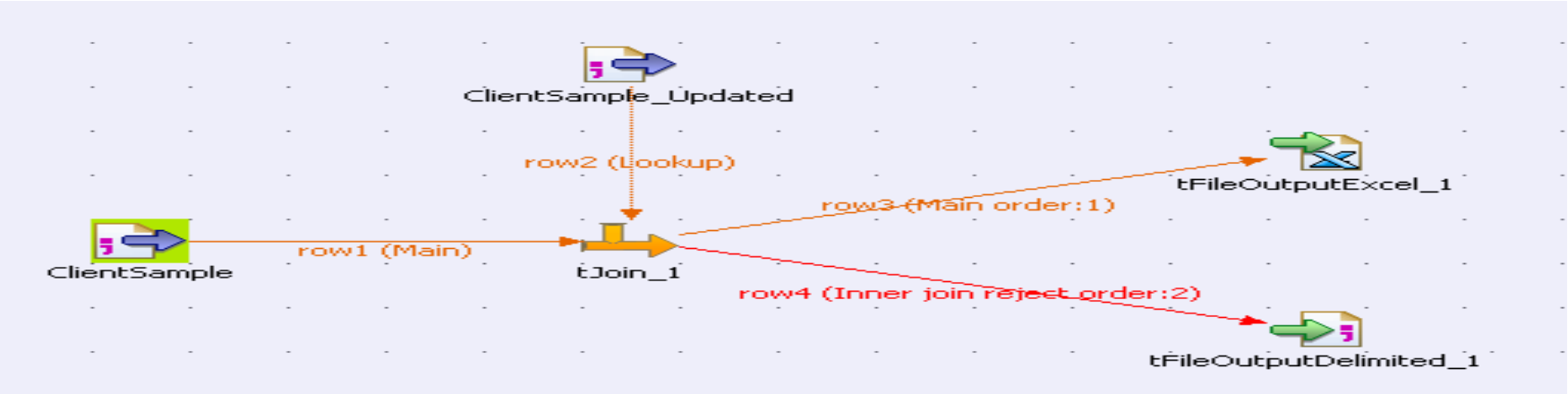
This component helps you ensure the data quality of any source data against a reference data source.

Usage

This component is not startable and it requires two input components and one or more output components.

Joining data **using tJoin** Component (cont…)

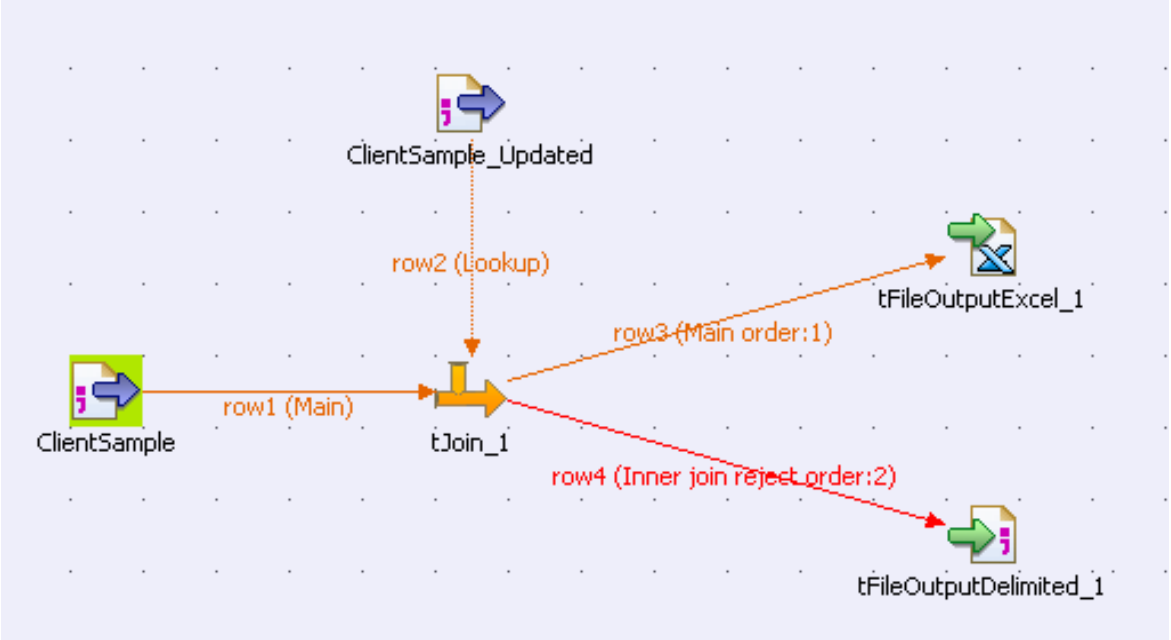
- tJoin: It joins two tables by doing an exact match on several columns.
- It compares columns from the main flow with reference columns from the lookup flow and outputs the main flow data and/or the rejected data.
- This component helps you ensure the data quality of any source data against a reference data source.



Joining data **using tJoin** Component (cont…)

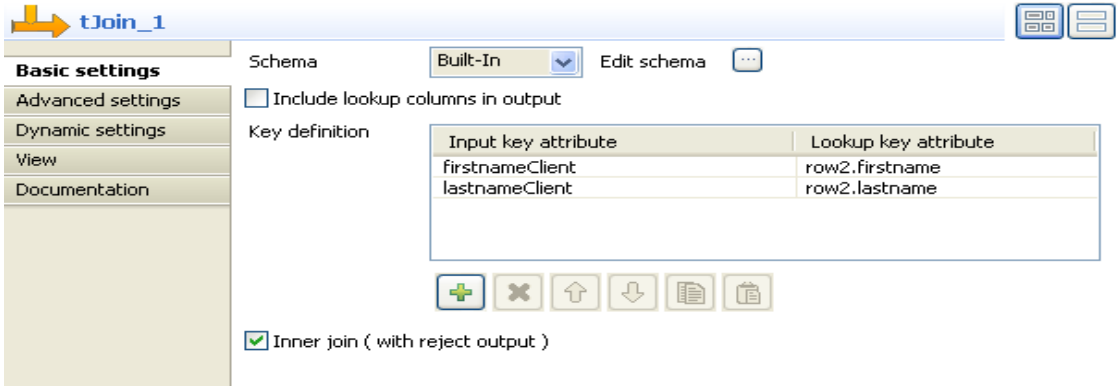
1. The tFileInputDelimited to fetch a file called ClientSample and ClientSample_Upda

2. Drop the following components from the **Palette** onto the design workspace: **tJoin**, **tFileOutputExcel**, and **tFileOutputDelimited**.

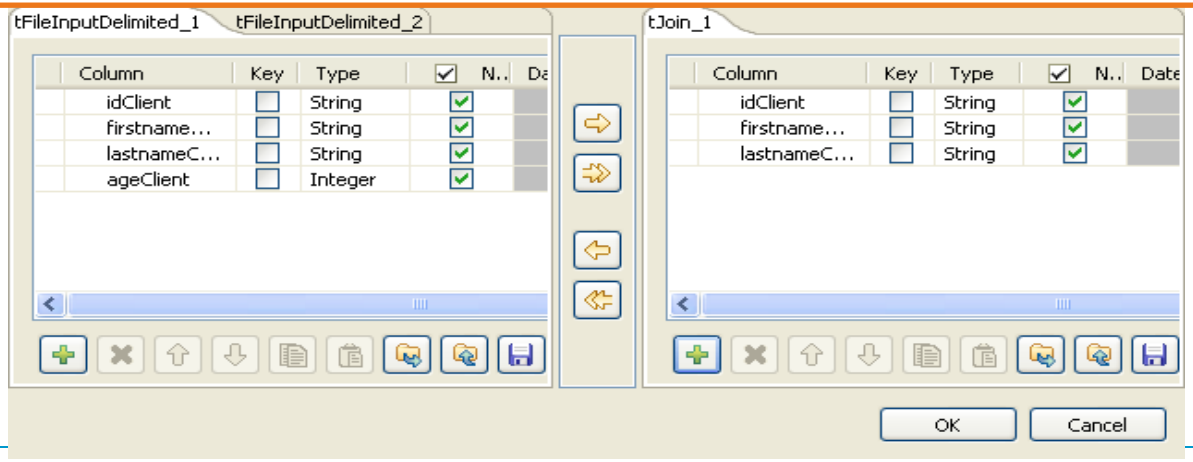


Joining data **using tJoin** Component (cont…)

3. Double click tJoin to display its Basic settings view and define its properties.

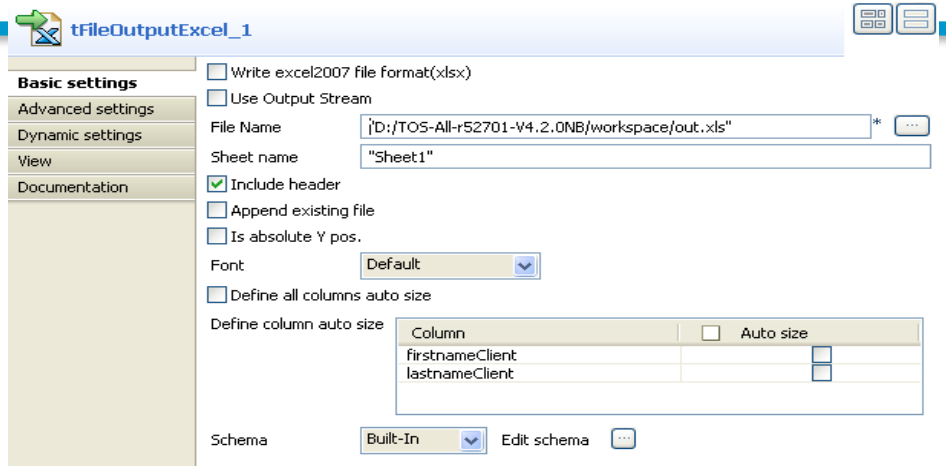


4. Click the **Edit schema** button to open a dialog box that displays the data structure of the input files,

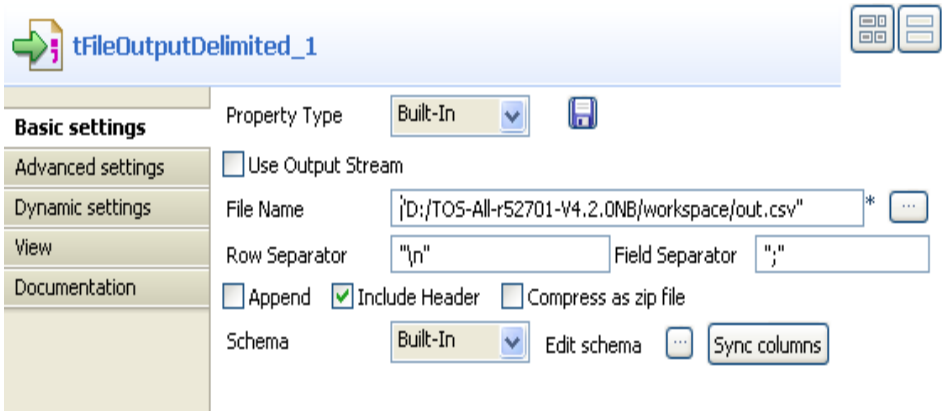


Joining data **using tJoin** Component (cont…)

5. Double click **tFileOutputExcel** to display its **Basic settings** view and define its properties.

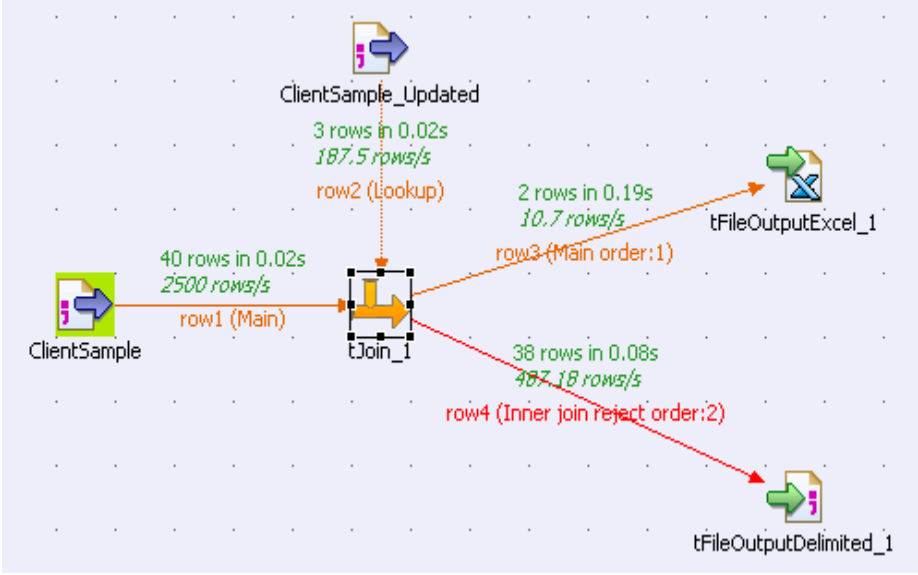


6. Double click **tFileOutputDelimited** to display its **Basic settings** view and define its properties.



Joining data **using tJoin** Component (cont…)

7. Press **Ctrl+S** to save your Job.
8. Press **F6**, or click **Run** on the **Run** tab to execute the Job.



9. The output of the exact match on the *firstnameClient* and *lastnameClient* columns is written to the defined Excel file

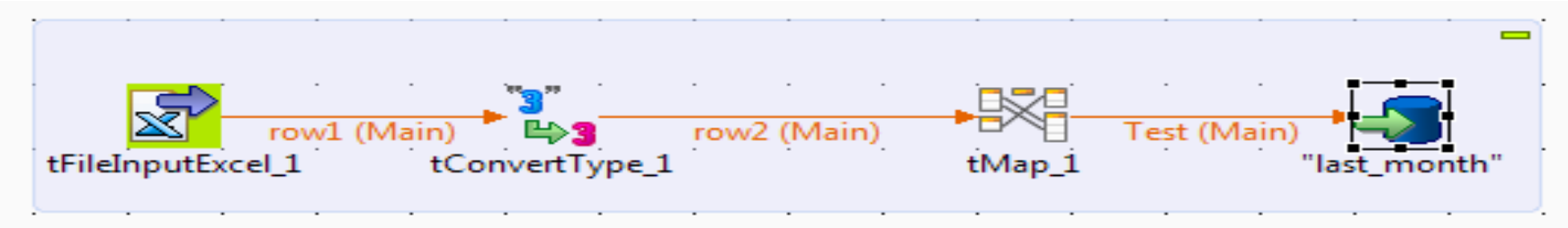
	A	B	C
1	idClient	firstnameClient	lastnameClient
2	28	Herbert	Eisenhower
3	36	Chester	Grant

10. The rejected data is written to the defined delimited file

```
1  idClient;firstnameClient;lastnameClient
2  1;Dwight;Madison
3  2;Franklin;Jackson
4  3;Ronald;Buchanan
5  4;Bill;Cleveland
6  5;William;Harrison
7  6;William;Fillmore
8  7;Harry;Adams
9  8;Harry;McKinley
10 9;Herbert;Reagan
11 10;Lyndon;Jefferson
12 11;Bill;Jackson
13 12;John;Hayes
14 13;Ulysses;Reagan
```

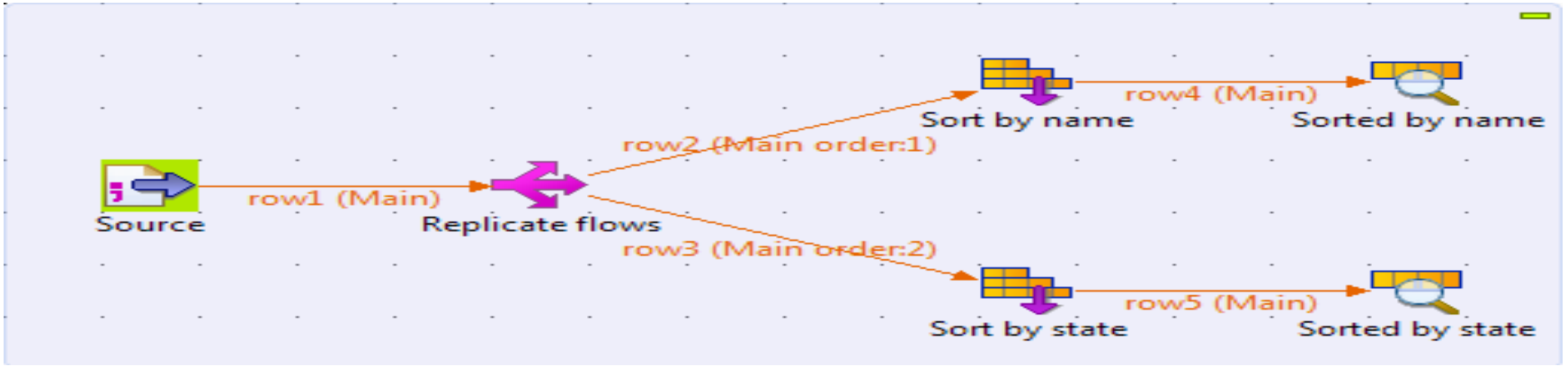
Transformation Components - tConvertType

- tConvertType: In this transformation **tConvertType** component allows specific conversions at run time from one data type to another data type without writing any parsing statement.
- In the below mentioned example; the Input is flat file so all the columns data type is string but in the final target table all the columns are not string , there are integer datatype column in this parsing is required, so we can use tConvertType which will handle all the parsing requirements.



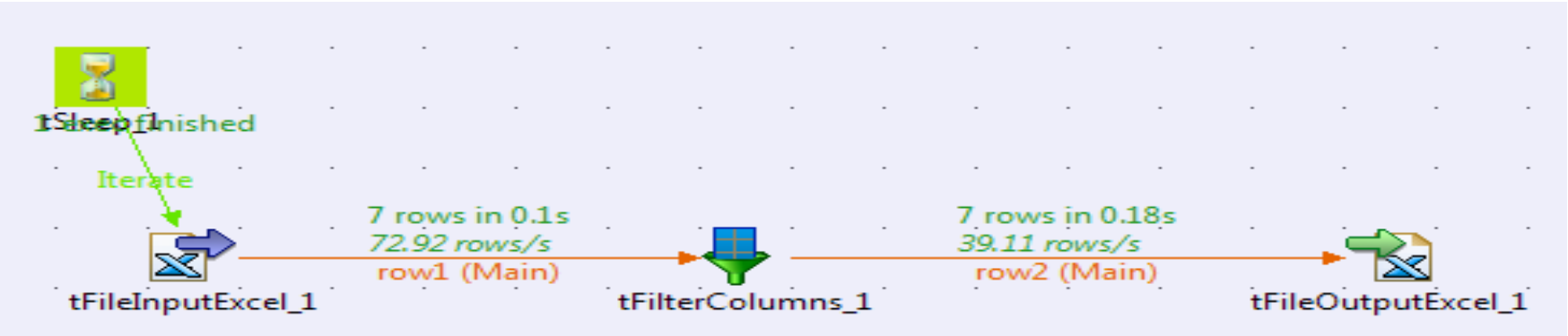
Transformation Components - tReplicate

- tReplicate: Duplicate the incoming schema into two identical output flows. Allows you to perform different operations on the same schema. A schema is a row description.
- It defines the number of fields (columns) to be processed and passed on to the next component. The schema is either Built-In or stored remotely in the Repository.



Transformation Components - tsleep

- tsleep: By using this transformation we need to implement time off in job execution. The amount of time you want to off in the job execution is mentioned in terms of seconds.

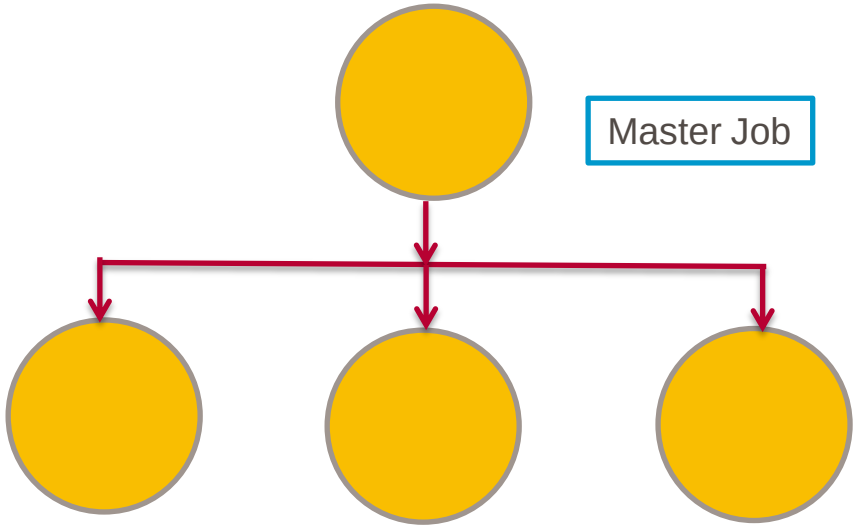


Working With Master Jobs

Module Outline

- Create a Master Job
- Override context variables
- Export a Job and its dependencies

Job Orchestration - Master Jobs



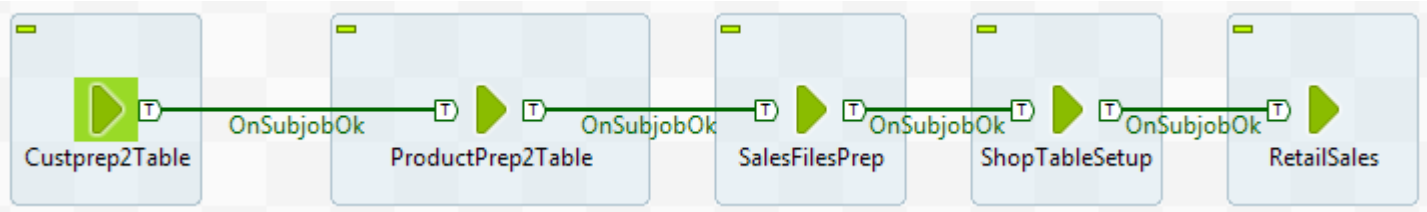
Parent job

Child Jobs

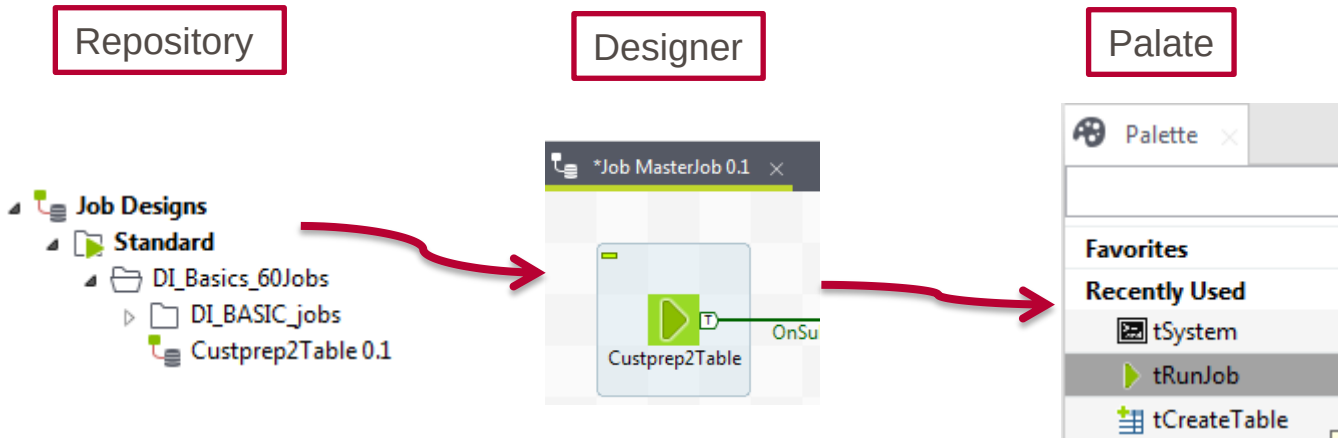
Master Job – Calls and orchestrates child jobs

Job Orchestration - Master Jobs- When, Why and How

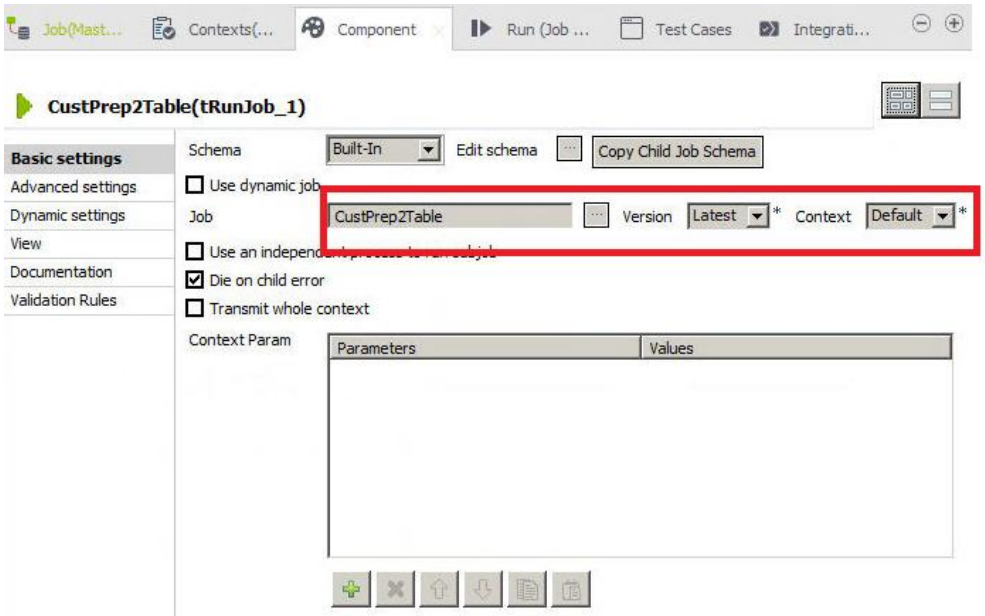
- When?
  - Complex Jobs that need to run in sequence
- Why?
  - Easy to use orchestration layer within Talend Studio
- How?
  - Largely through tRunJob component



Job Orchestration - Master Jobs- tRunJob



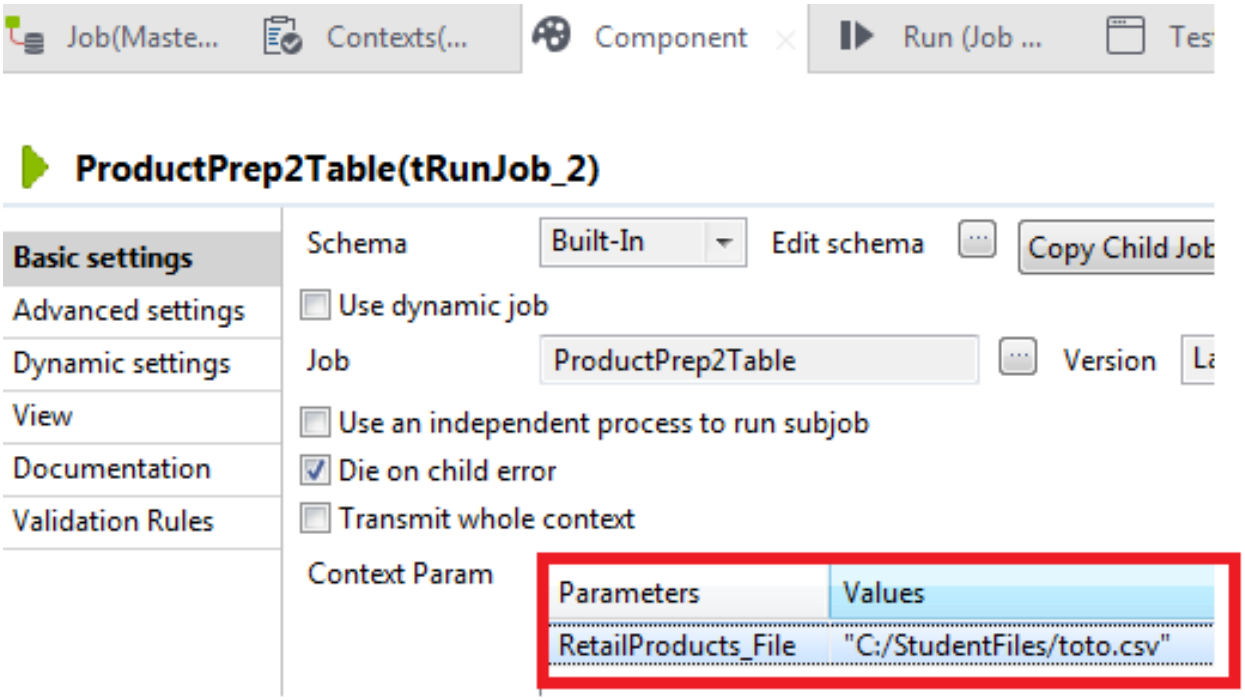
Job Orchestration - Master Jobs- tRunJob



- You can specify:
  - Job Name
  - Job Version
  - Context

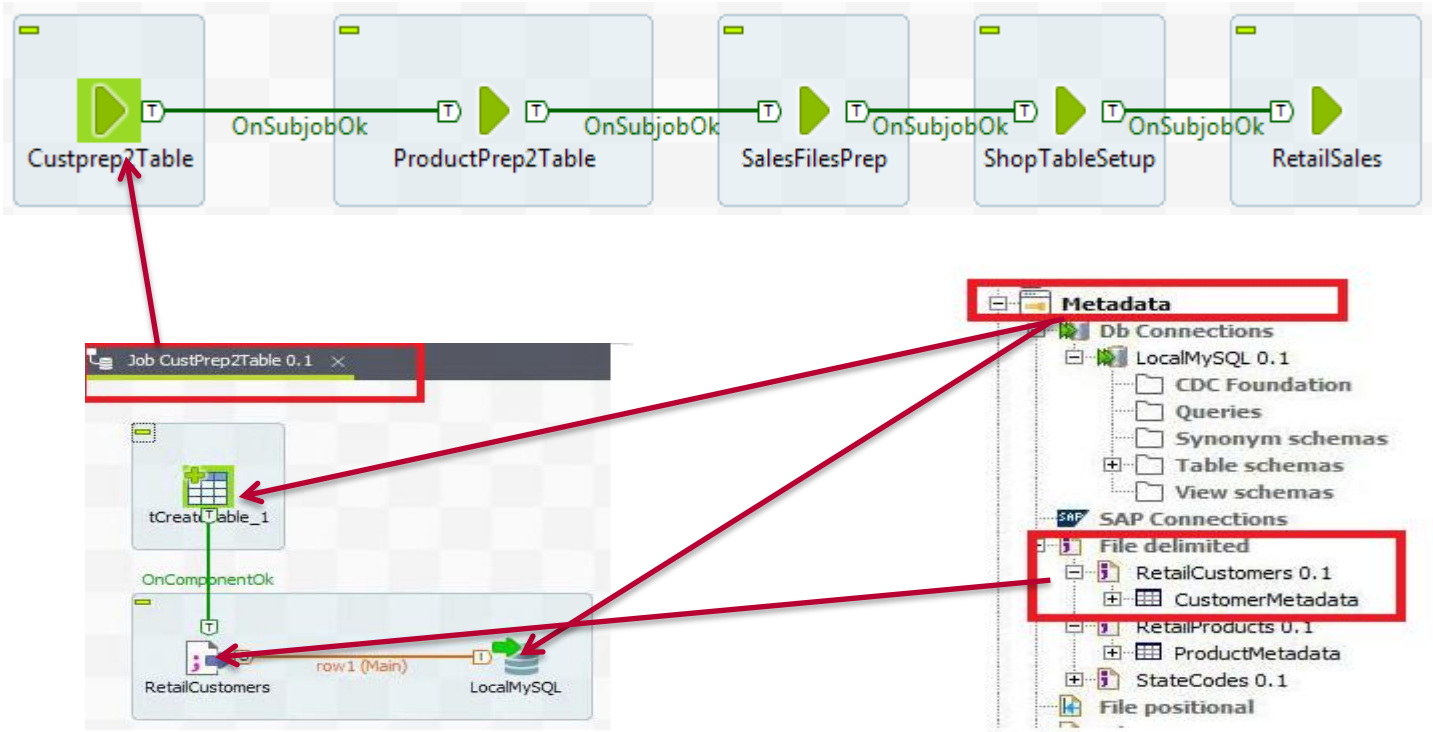
Job Orchestration - Master Jobs- tRunJob

- The value of any context variable can be modified:



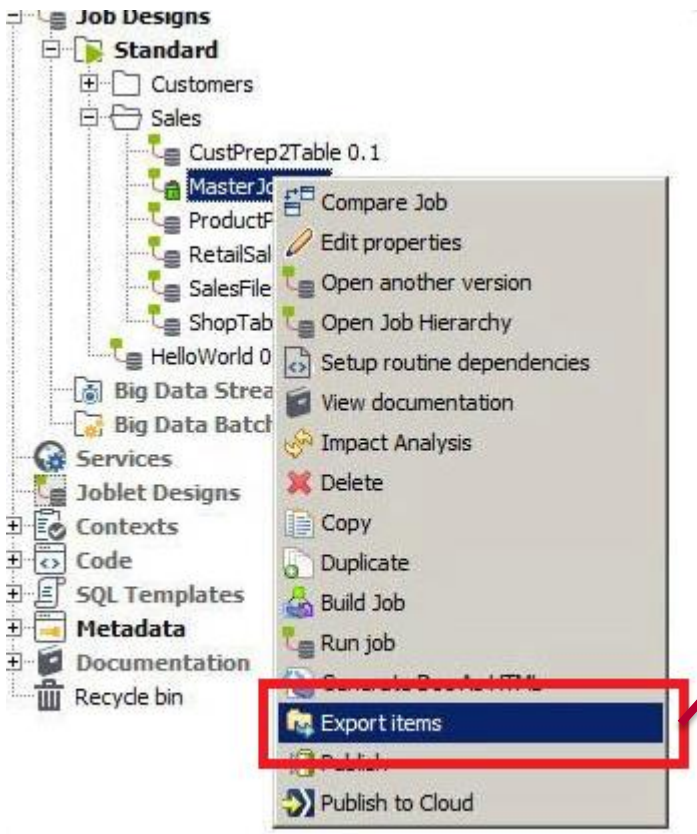
Exporting Jobs

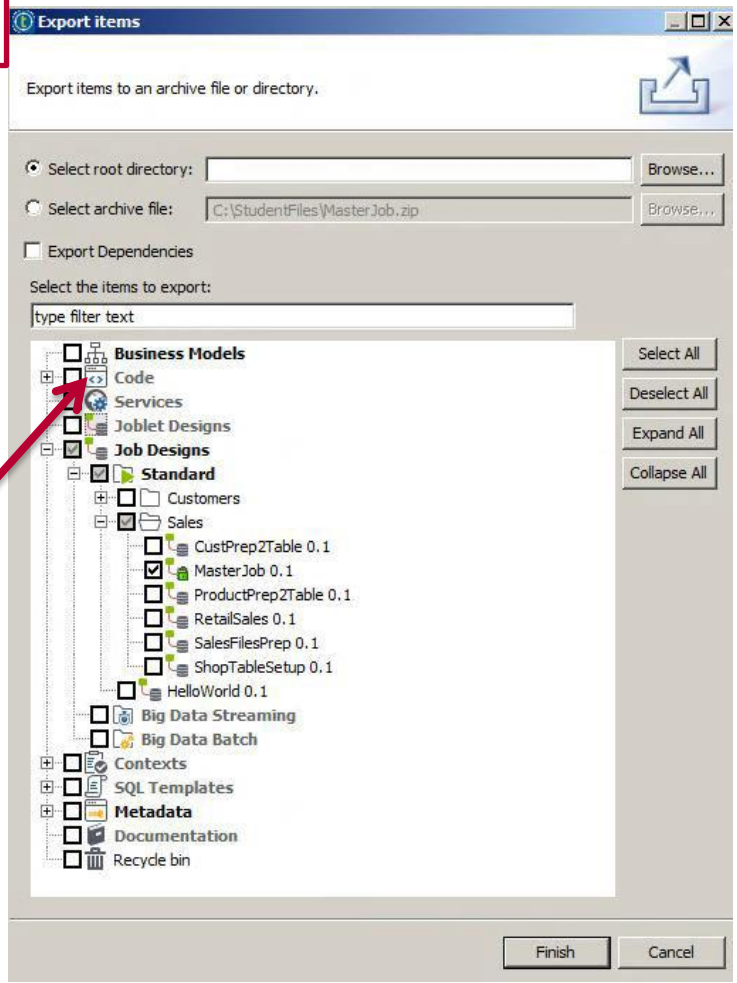
Imagine that you would like to share the Master Job with colleagues.
It needs to be packaged with all dependencies in order to make it work in another Talend Studio instance:



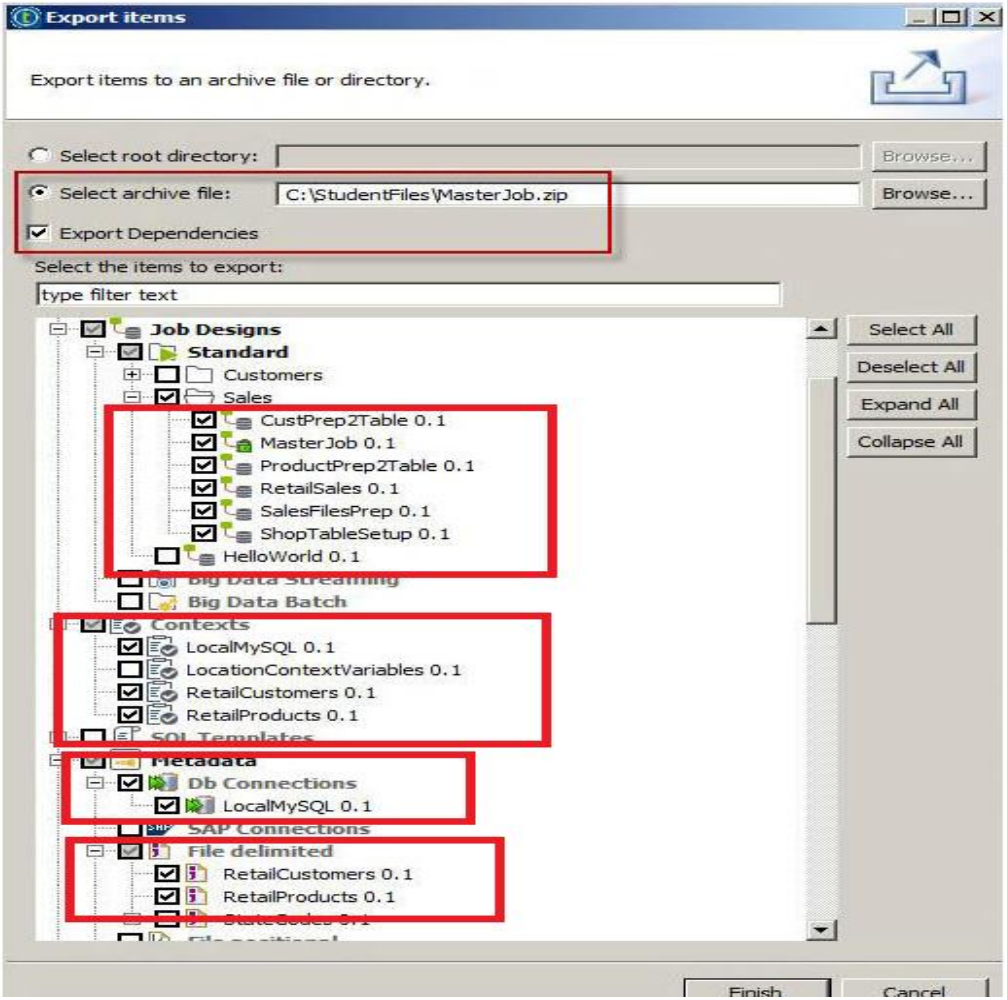
Exporting Jobs

Jobs can be exported from the Repository:





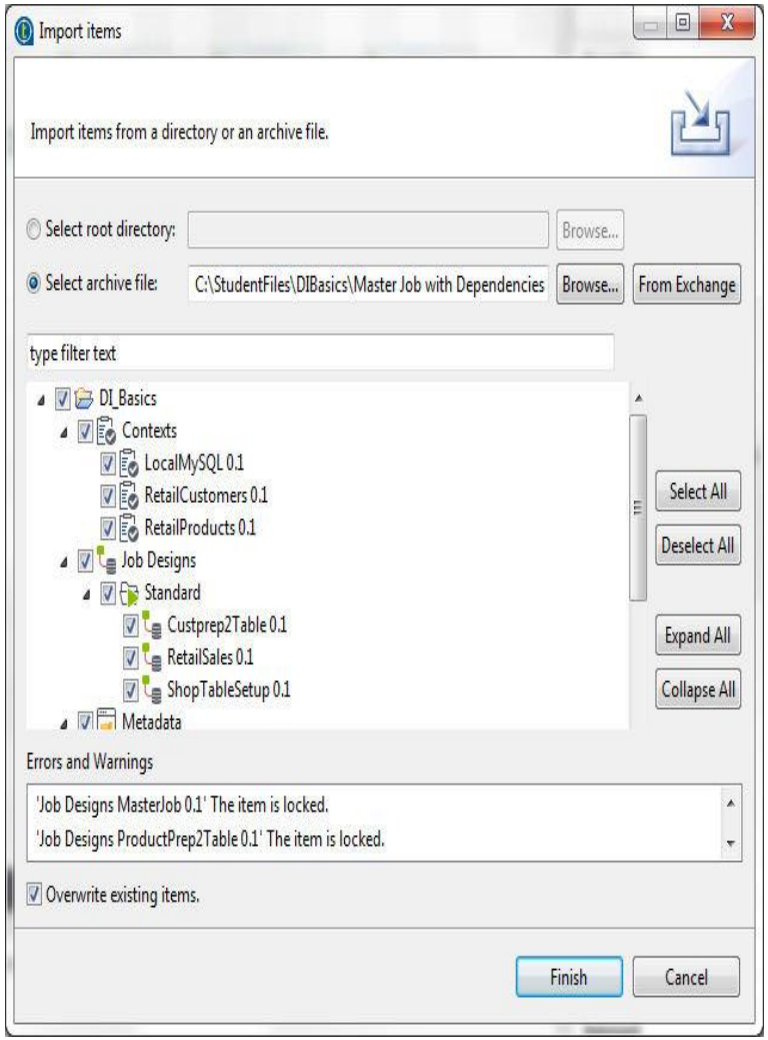
Exporting Jobs



- Jobs can be exported:
  - As a .zip file
  - with all dependencies

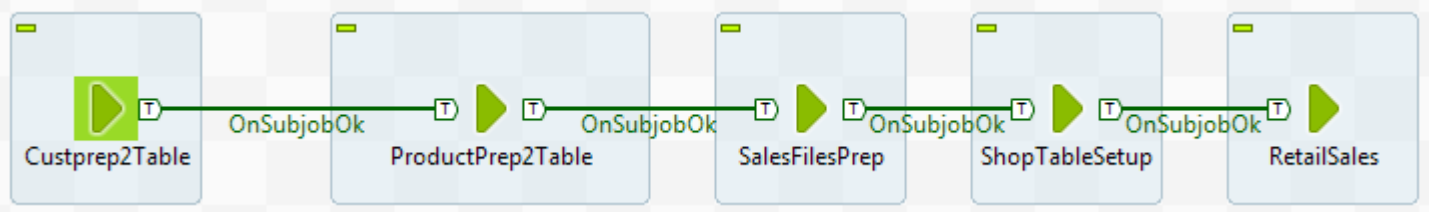
Importing Jobs

- **Jobs can be imported from the Repository**:
- Select/unselect items prior to import
- Select Overwrite as needed to clear Errors and Warnings
- if some items already exist in the Repository
- Finish is not actionable until Errors are resolved



Lab Overview

- You will create a Master Job to control the execution of other DI Jobs.



- Then, you will export the Job and all its dependencies.

Building Standalone jobs

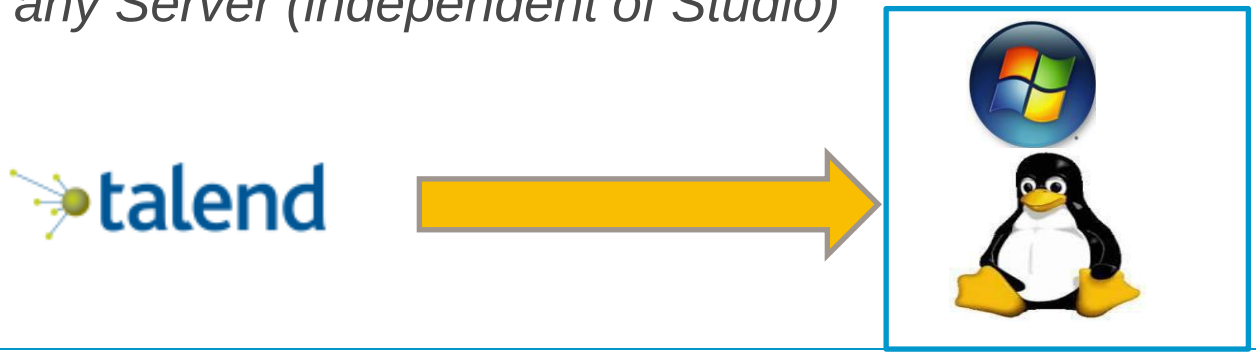
Module Outline

- Build and export a Job
- Run an exported Job independently from the Talend Studio
- Create a new version of an existing Job

Building a Job

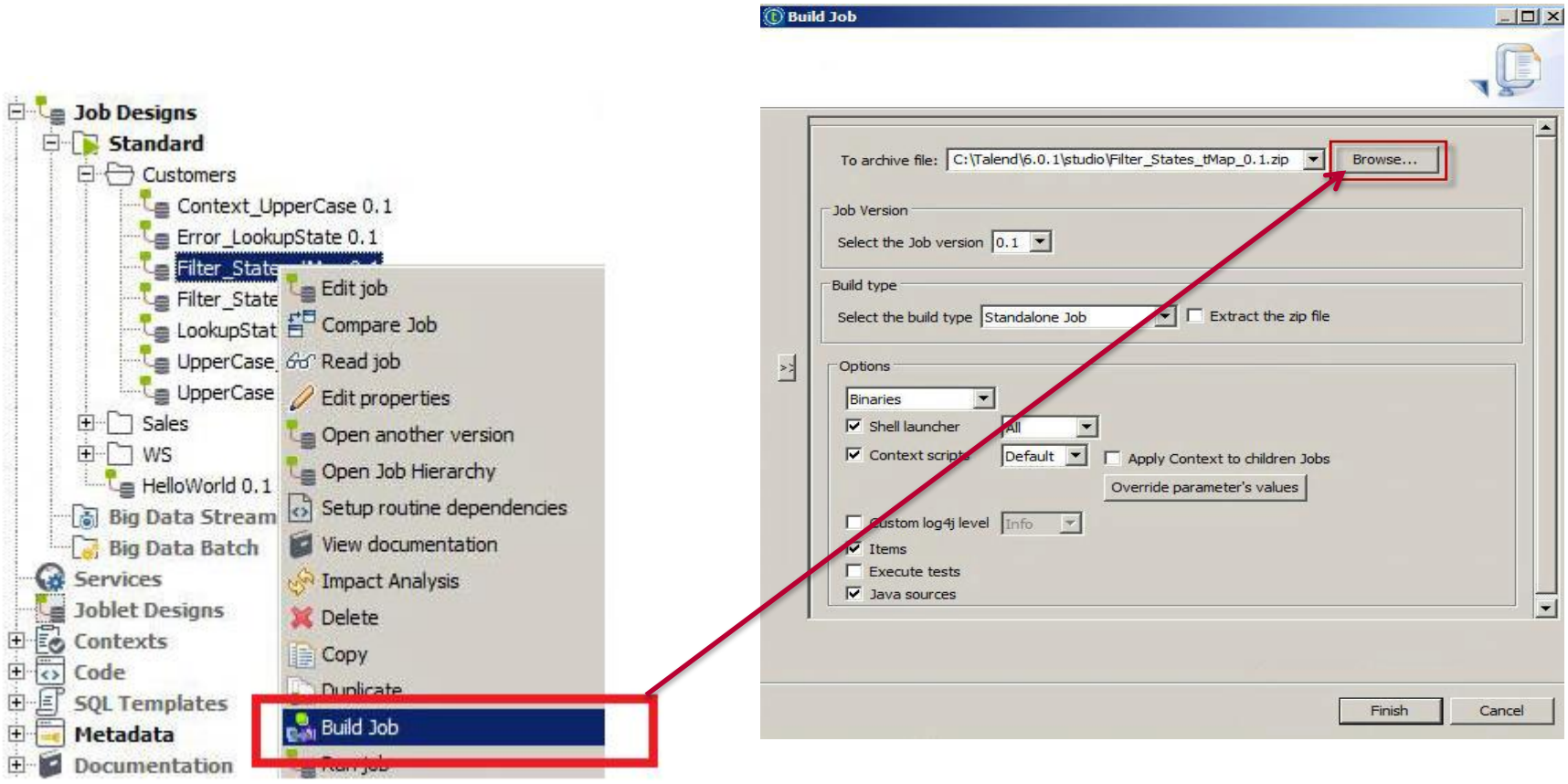
- In the previous lesson you exported a Job
  - Export a Job and its dependencies
  - Only usable within the Studio

- Build Job
  - Build and deploy a Job
  - Run on *any Server (independent of Studio)*



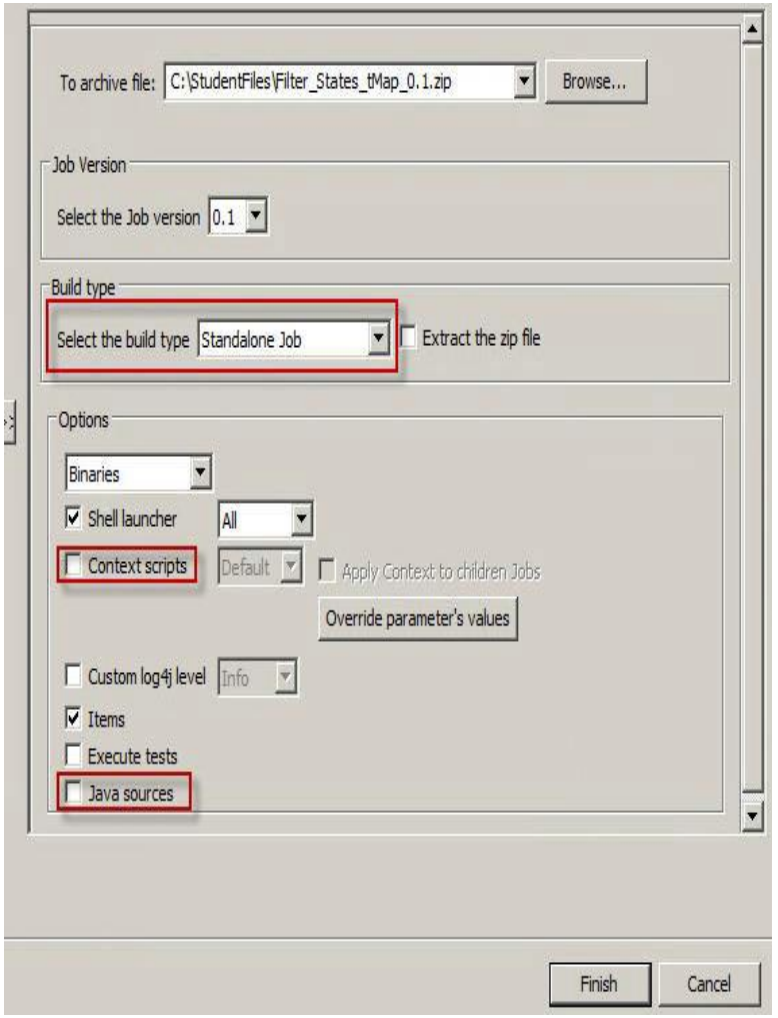
Building a Job

- Select your Job in the Repository and start the Job build:



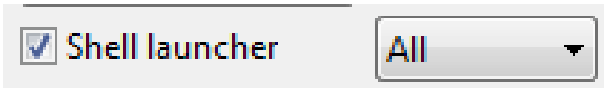
Building a Job

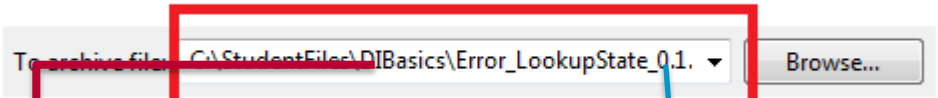
- Select the Build type:
- Selecting Standalone Job you will get the following in the .zip archive file:
  - JAR files
    - Java Archive Files
    - Package of files needed for deployment
  - *.bat files - Windows batch files
  - *.sh files – Unix shell script files



Extract and Run

- Extract
  - Go to Archive folder
  - Extract .zip
    - Windows (.bat file)
    - Unix (.sh file)
    - All – Exports both (.bat and .sh)

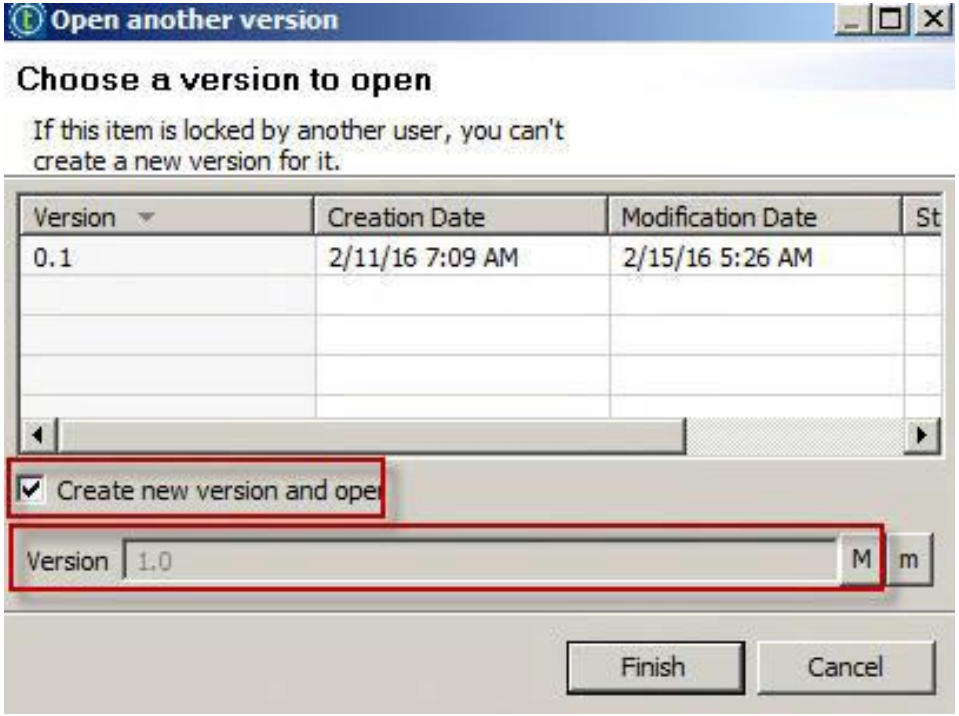




- Run
  - Invoke executable
  - Error_LookupState_0.1.bat
  - Error_LookupState_0.1.sh

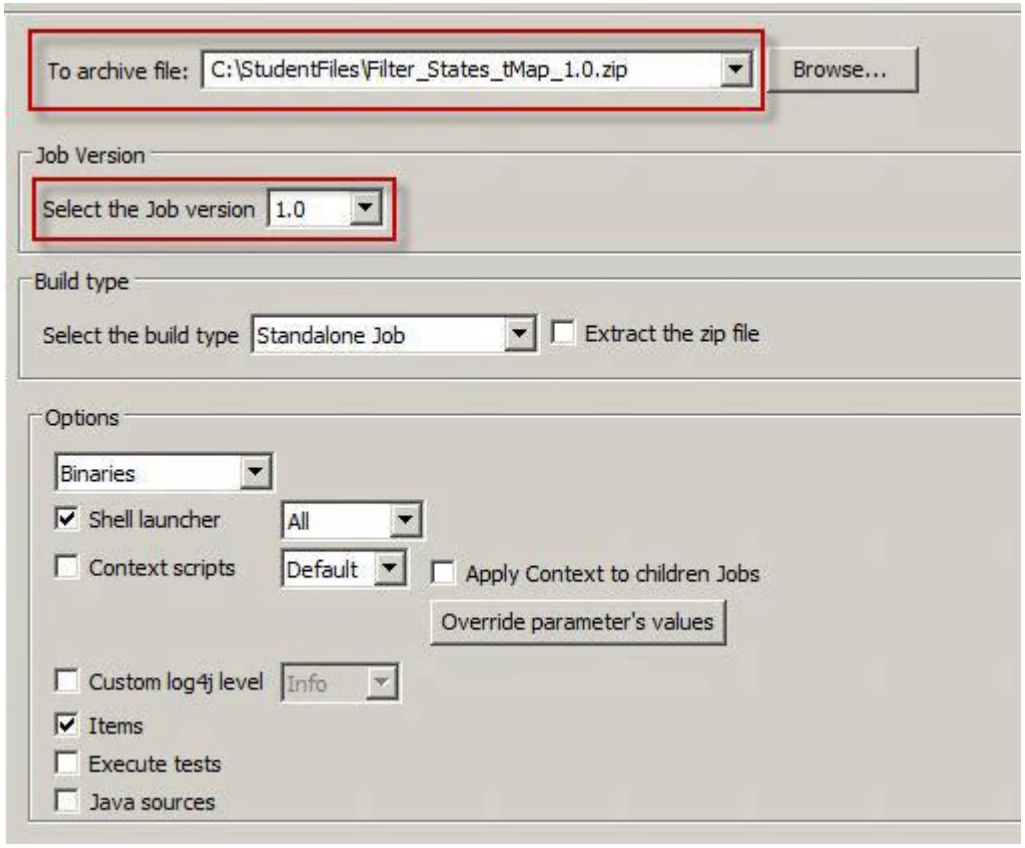
Job Versioning

- Studio offers basic version control (Major.minor)
  - Create Job
  - Open another version



Job Versioning

- Studio offers basic version control (Major.minor)
  - Build another version



Lab Overview - Stand Alone Jobs

- Build Jobs that can be run outside of the Talend Studio
- Use basic version control

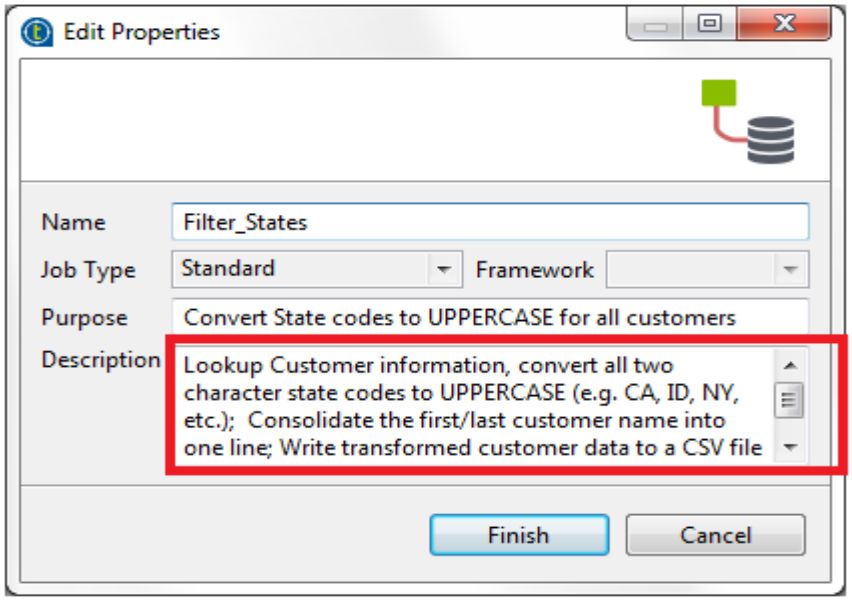
Talend Best Practices

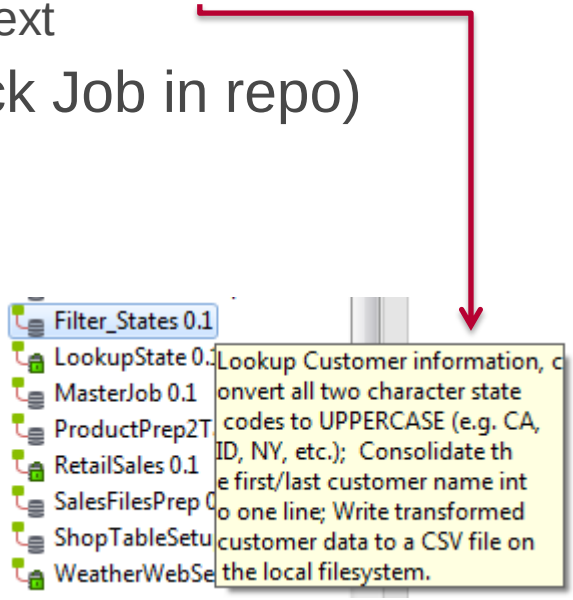
Module Outline

- Name a Job and fill out its properties using best practices
- Provide additional information for Job components
- Create a Business Model
  - Add and connect shapes
  - Associate items (Jobs, Documents) to Shapes
- Generate and view HTML documentation for your Job

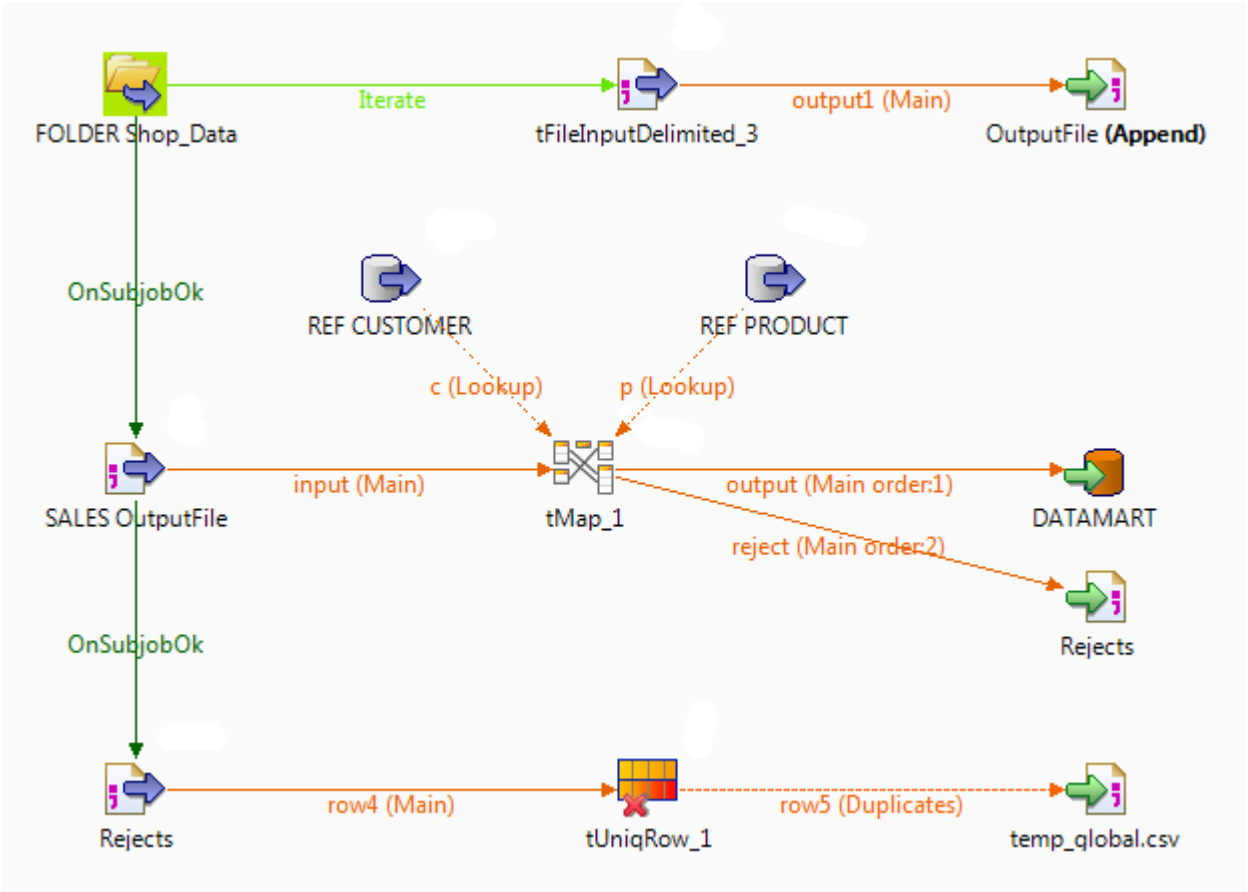
Best Practices - Jobs

- Create a New Job
  - Name – Having a naming convention is recommended
  - Purpose – Brief comment
  - Description – More detailed. Shows in repo hover-text
- Modify anytime with Edit Properties (Rt-Click Job in repo)



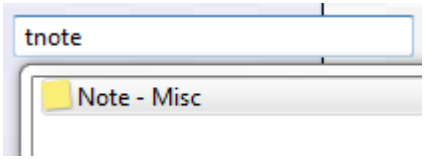


Best Practices - Job Layout

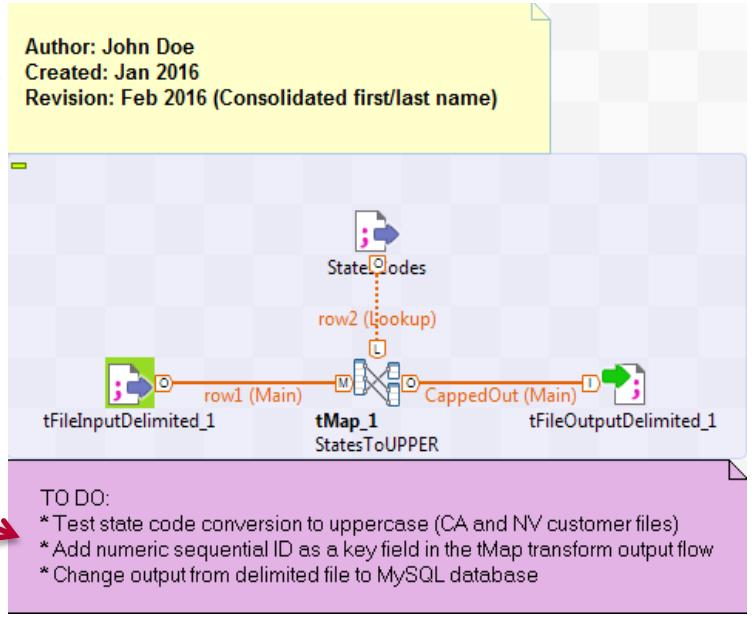


Best Practices – Jobs – Writing Notes

- Place Note in Design Workspace
- Possible Conventions



- Author: John Doe
- Created: Jan 2016
- Revision: Feb 2016 (Consolidated first and last name)



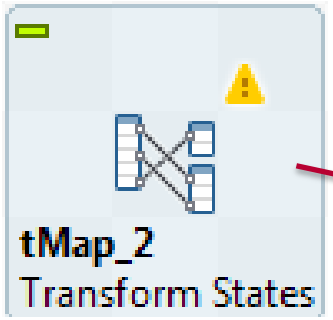
- To Do: Build out a list…
- Basic format options
  - Font, size, bold, *italics*
  - Opacity
  - Colors (text, border, fill)
  - Vertical/Horizontal alignment

Best Practices – Jobs naming convences

Field	Description	Default
Label format	Text label shown in design workspace	__UNIQUE_NAME__

Double-click Component > View > Label format

__UNIQUE_NAME__
Transform States



At a glance:
- Icon (develop familiarity)
- Component Name
- Primary function

ELT

ELT

- The ELT family groups together the most popular database connectors and processing components.
- This mode supports all of the most popular databases including Teradata, Oracle, Vertica, Netezza, Sybase, etc
- In ELT data is migrated in bulk according to the data set and the transformation process occurs after the data has been loaded into the targeted DBMS in its raw format.
- Less stress is placed on the network and larger throughput is gained.
- For example: As SQL is less powerful than Java, the scope of available data transformations is limited. ELT requires users that have high proficiency in SQL tuning and DBMS tuning.

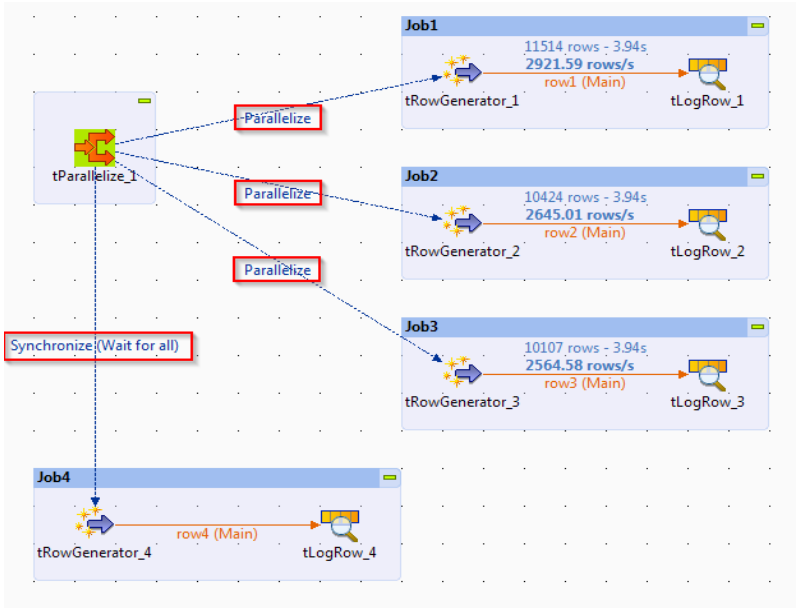
ELT (Contd..)

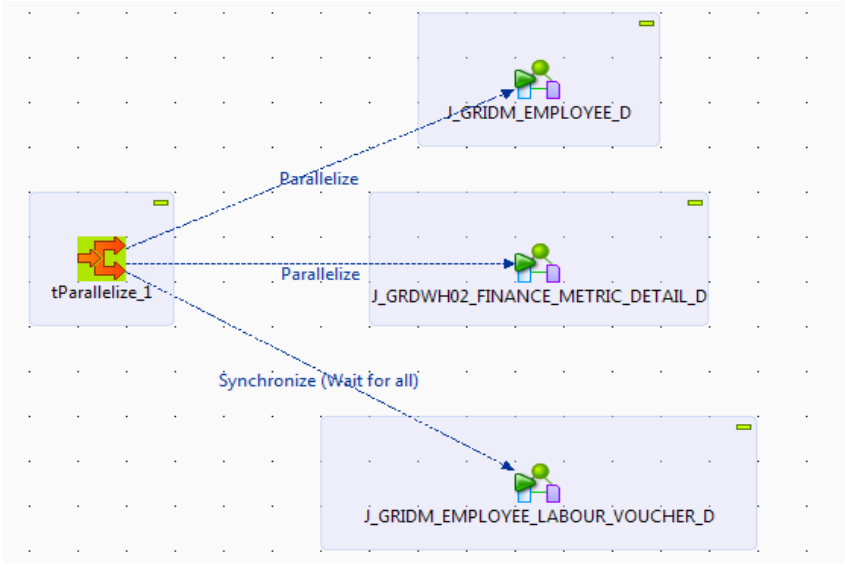
Types of ELT Components

tAccessConnection - Elt	tELTHiveInput	tELTMysqlOutput	tELTSybaseMap	tHiveConnection - Elt	tSAPHanaConnection - Elt
tAS400Connection - Elt	tELTHiveMap	tELTNetezzaInput	tELTSybaseOutput	tIngresConnection - Elt	tSQLiteConnection - Elt
tCombinedSQLAggregate	tELTHiveOutput	tELTNetezzaMap	tELTTeradataInput	tInterbaseConnection - Elt	tSQLTemplate
tCombinedSQLFilter	tELTJDBCInput	tELTNetezzaOutput	tELTTeradataMap	tJDBCConnection - Elt	tSQLTemplateAggregate
tCombinedSQLInput	tELTJDBCMap	tELTOracleInput	tELTTeradataOutput	tMSSqlConnection - Elt	tSQLTemplateCommit
tCombinedSQLOutput	tELTJDBCOutput	tELTOracleMap	tELTVerticaInput	tMysqlConnection - Elt	tSQLTemplateFilterColumns
tDB2Connection - Elt	tELTMSSqlInput	tELTOracleOutput	tELTVerticaMap	tNetezzaConnection - Elt	tSQLTemplateFilterRows
tELTGreenplumInput	tELTMSSqlMap	tELTPostgresqlInput	tELTVerticaOutput	tOracleConnection - Elt	tSQLTemplateMerge
tELTGreenplumMap	tELTMSSqlOutput	tELTPostgresqlMap	tEXAConnection - Elt	tParAccelConnection - Elt	tSQLTemplateRollback
tELTGreenplumOutput	tELTMysqlInput	tELTPostgresqlOutput	tFirebirdConnection - Elt	tPostgresPlusConnection - Elt	tSybaseConnection - Elt
tVectorWiseConnection - Elt	tELTMysqlMap	tELTSybaseInput	tGreenplumConnection - Elt	tPostgresqlConnection - Elt	tTeradataConnection - Elt

Parent and Child Job (tParallelize & tRun)

- TParallelize: The tParallelize component is an Orchestration component. The tParallelize (available in Enterprise Edition) component is used for multiple executions of the jobs at the same time. This can be achieved by multi-threading the jobs.





tParallelize & tRun (Contd..)

- tRun: Component allows you to embed one Talend Job within another so that it may be executed as a Talend SubJob.
- It executes the Job called in the component's properties, in the frame of the context defined.
- tRun helps mastering complex Job systems which need to execute one Job after another. By this means can be utilized to have a parent and child relationship among the Jobs.
- tRunJob Component from the Component Palette (Orchestration) of you can drag an existing Job from Job Designs in the Repository Browser.

ERROR - Handling

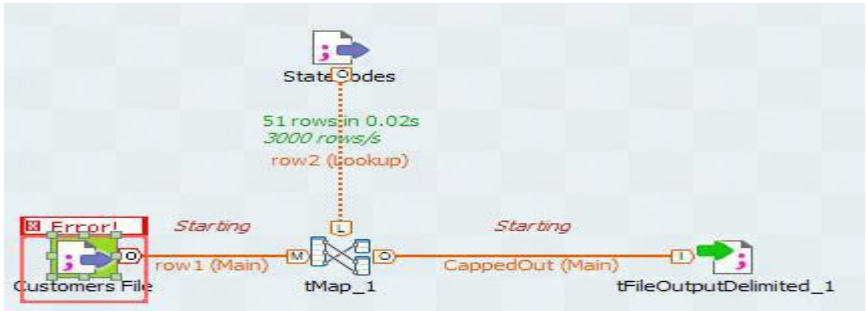
Module outline

- Handle Errors
  - Kill a Job when an error occurs
  - Implement specific Job execution path on the component error
    - Raise a Warning under specific conditions
    - Configure the log level in the execution console

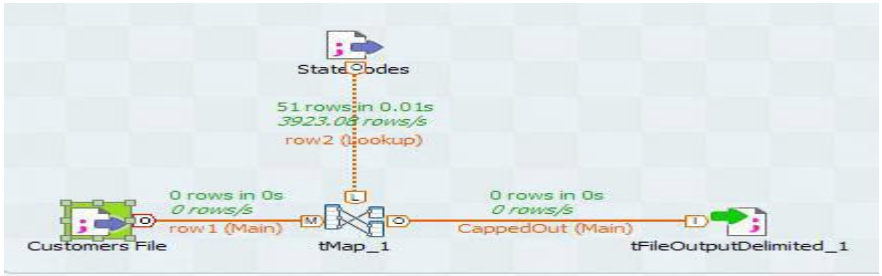
Overview

- Suppose a component is in error
  - Ex: The input file does not exist
  - Job execution options:

1. The Job stops when error is encountered



2. The Job completes execution despite the error

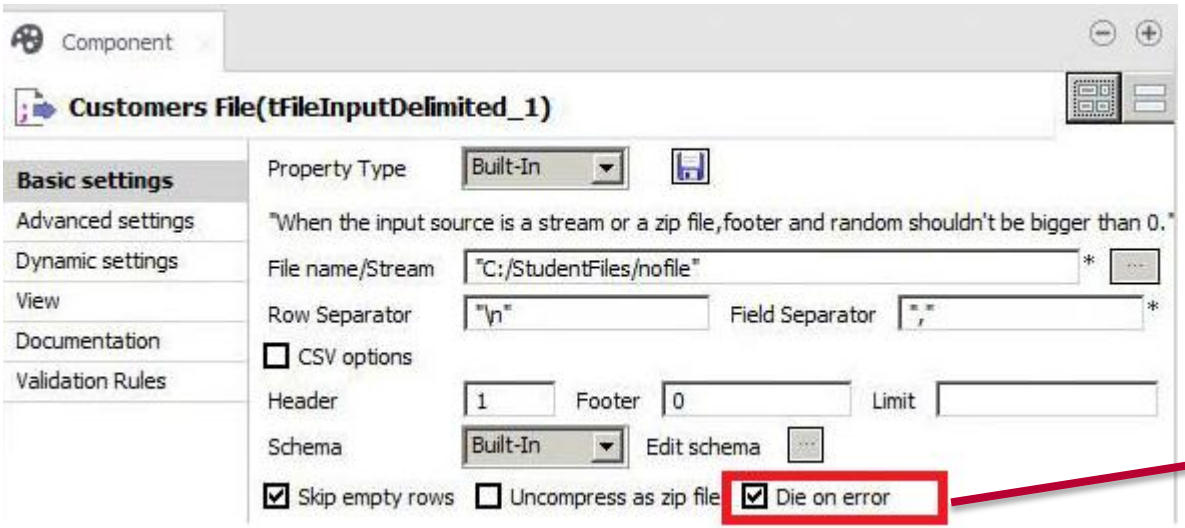


Kill a Job on Component Error

- Kill a Job on Component Error
- **Die On Error option** - available for certain components

Check **Die On Error option**

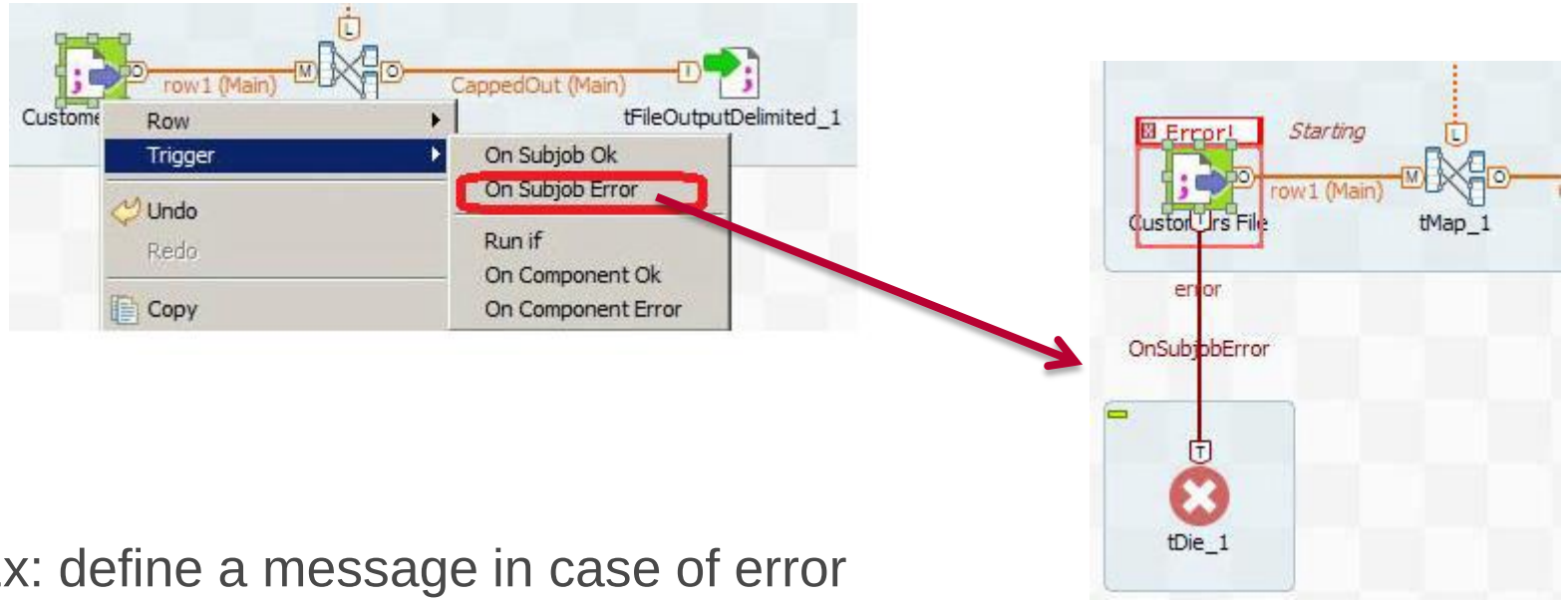
Job stops execution when error is encountered



Some components always die on error, so the option is not available

Triggers

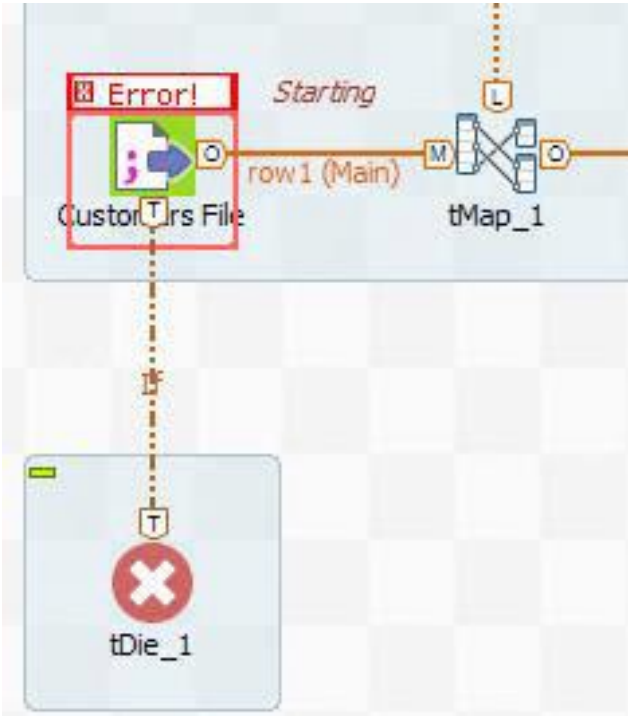
- Triggers allow implementation of different execution paths:
  - the status of components/subjobs
  - specific conditions



- Ex: define a message in case of error
  - Use tDie component to set the Die Message

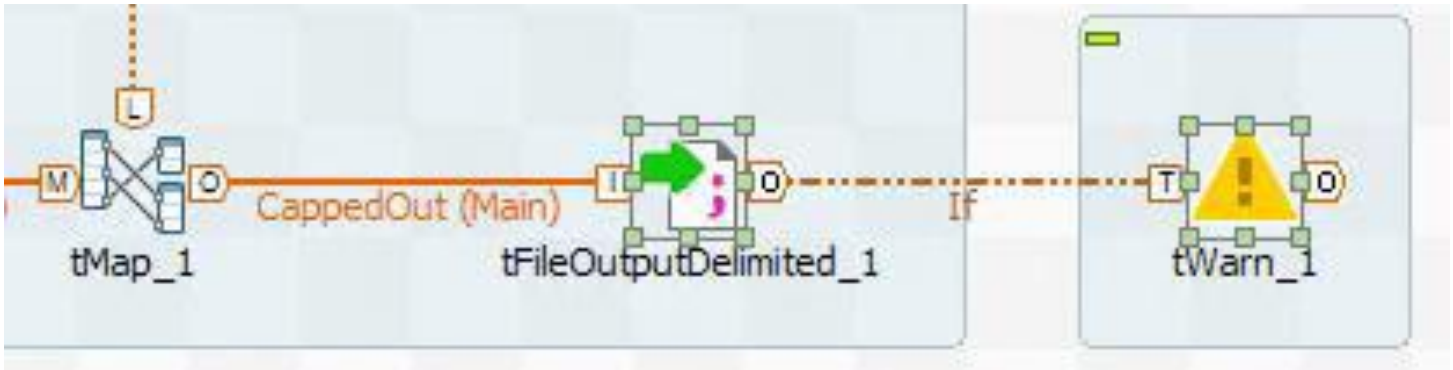
Kill a Job Under Specific Conditions

- Requirement:
  - Kill a Job under specific conditions (no technical error)
    - Ex: The input file has less then a certain number of lines
- Solution:
  - Use the tDie component to kill the Job
  - Use Run If trigger to specify the condition
    - The Job execution stops if the condition is met



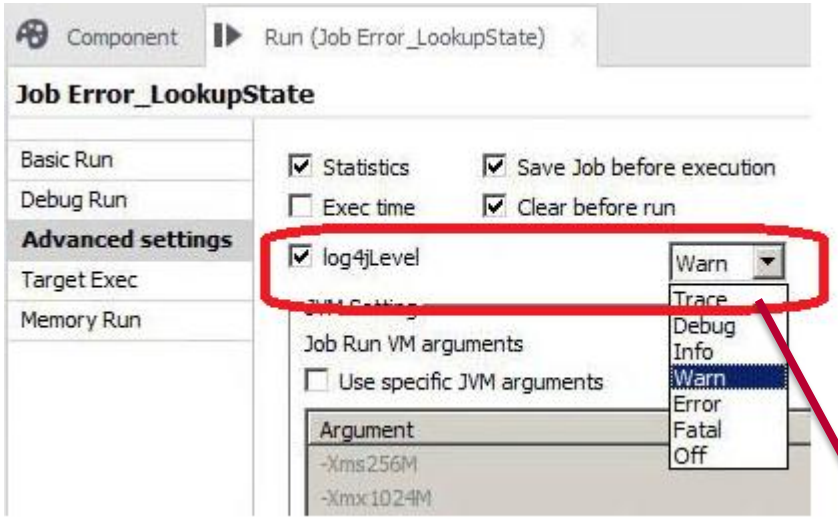
Raise a Warning

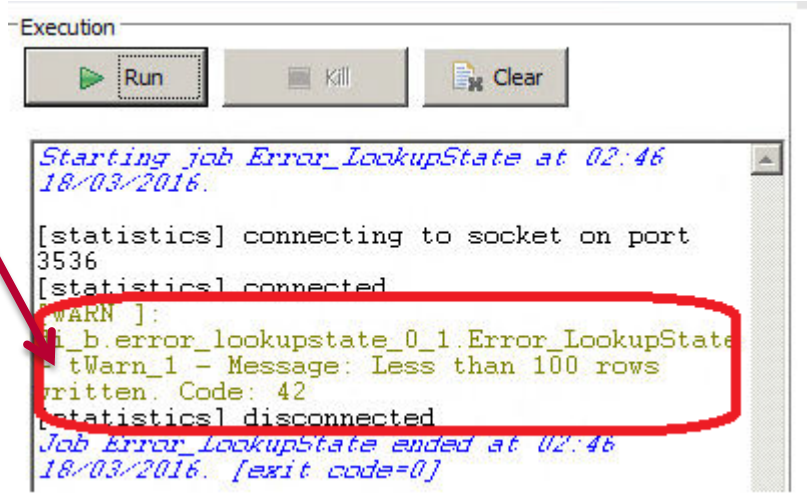
- Need:
  - Raise a Warning when a business rule is not met
    - Ex: The output file has not the expected number of lines
- Solution:
  - Use the tWarn component to raise a warning
  - Use Run If trigger to specify the condition
    - A warning is raised if the condition is met



Configuration

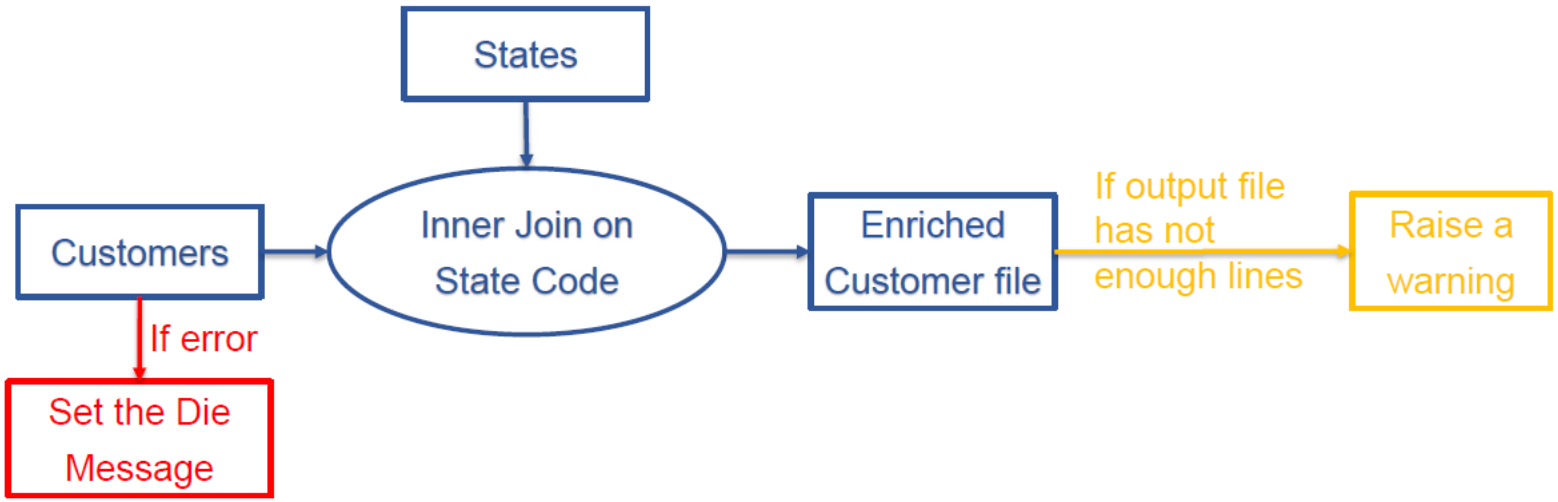
- log4jLevel
  - Available under the Advanced Settings tab of the Run View
  - Specifies the types of log messages to be displayed





Lab Exercises

- Create a technical error: input file does not exist
  - Kill the Job
  - Set the Die message
- Create a business error: output file does not have enough lines
  - Raise a warning



How to catch information about your jobs' executions

Module Outline

- tMsgBox and Tdoe Components
- tWarn and TLogCatcher Components
- tLogRow Component
- Sample Job – Steps

tMsgBox and Tdoe Components

- tMsgBox - **tMsgBox** is a graphical break in the job execution progress.
  - This component can be used as intermediate step in a data flow or as a start or an end object in the Job flowchart.
  - It can be connected to the next/previous component using either a **Row** or **Iterate** link.

- tDie **-** This component throws an error and kills the job.
  - Triggers the **tLogCatcher** component for exhaustive log before killing the Job.
  - Both **tDie** and **tWarn** components are closely related to the **tLogCatcher** component
  - This component cannot be used as a start component and it is generally used with a **tLogCatcher** for the log purpose.

tWarn and TLogCatcher Components

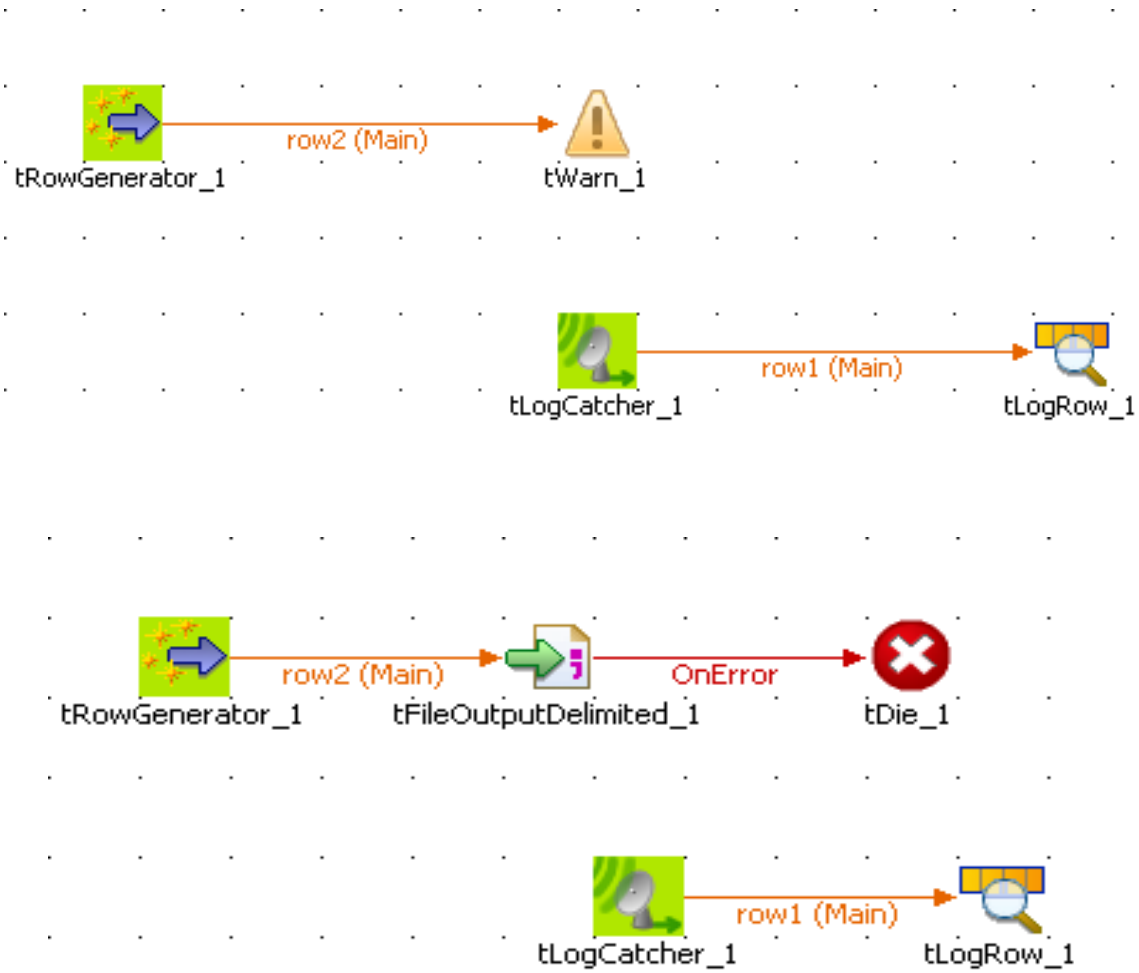
- **tWarn -** This component provides a priority-rated message to the next component. It does not stop your Job in case of error.
  - Triggers a warning often caught by the **tLogCatcher** component for exhaustive log.
  - Both **tDie** and **tWarn** components are closely related to the **tLogCatcher** component
  - Cannot be used as a start component. If an output component is connected to it, an input component should be preceding it.

- **tLogCatcher -** Fetches set fields and messages from Java Exception, **tDie** and/or **tWarn** and passes them on to the next component.
  - Operates as a log function triggered by one of the three: Java exception, tDie or tWarn, to collect and transfer log data.
  - This component is the start component of a secondary Job which automatically triggers at the end of the main Job

tLogRow Component

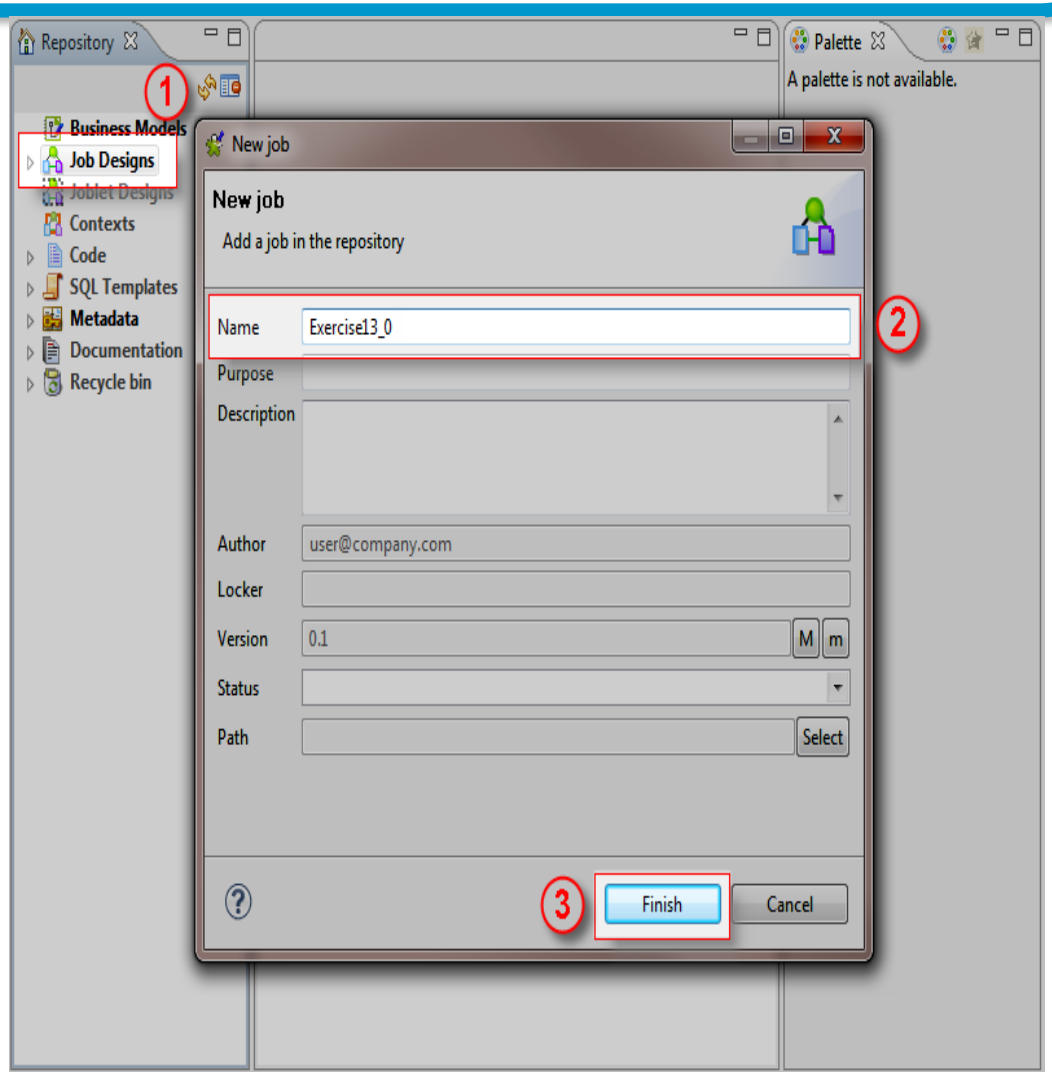
- **tLogRow –** Displays data or results in the **Run** console.
- **tLogRow** is used to monitor data processed.
- This component can be used as intermediate step in a data flow or as a n end object in the Job flowchart.

Sample Jobs



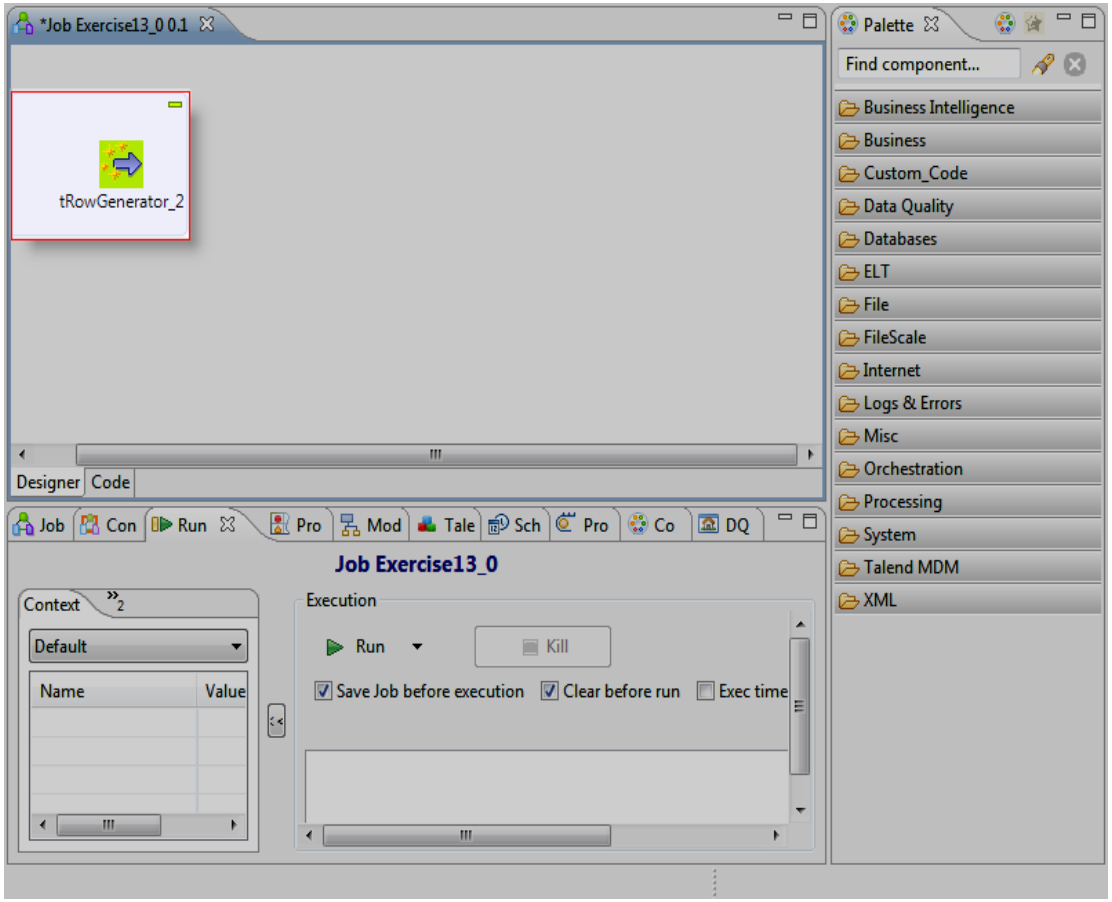
Catching information about ‘Job' executions

- In the Repository:
- Right-click on Job Designs.
- In the menu, click Create Job to open the New Job wizard.



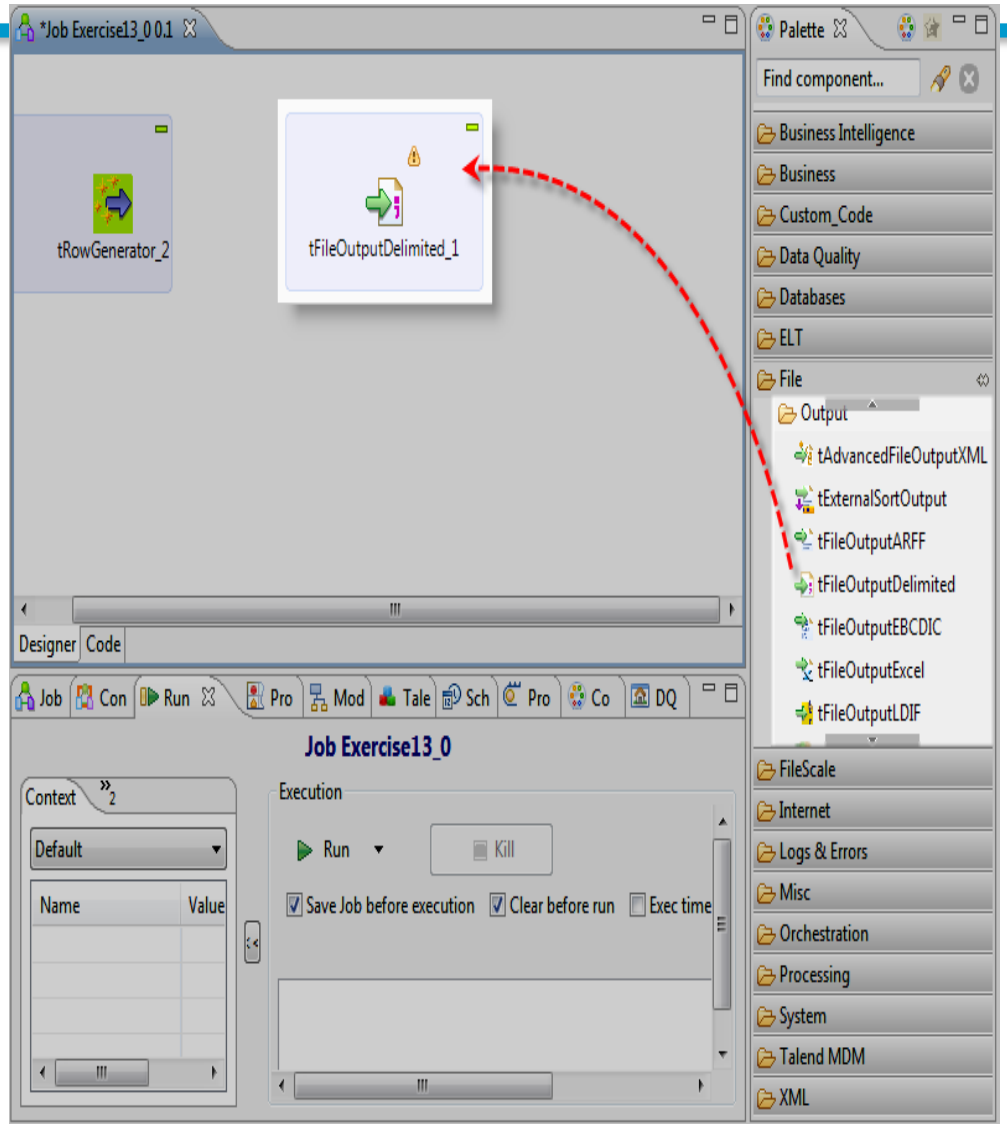
Catching information about ‘Job' executions (Cont.…)

- Open Exercise1_0 Job and copy the tRowGenerator component.
- Paste it in the Exercise13_0 Job.



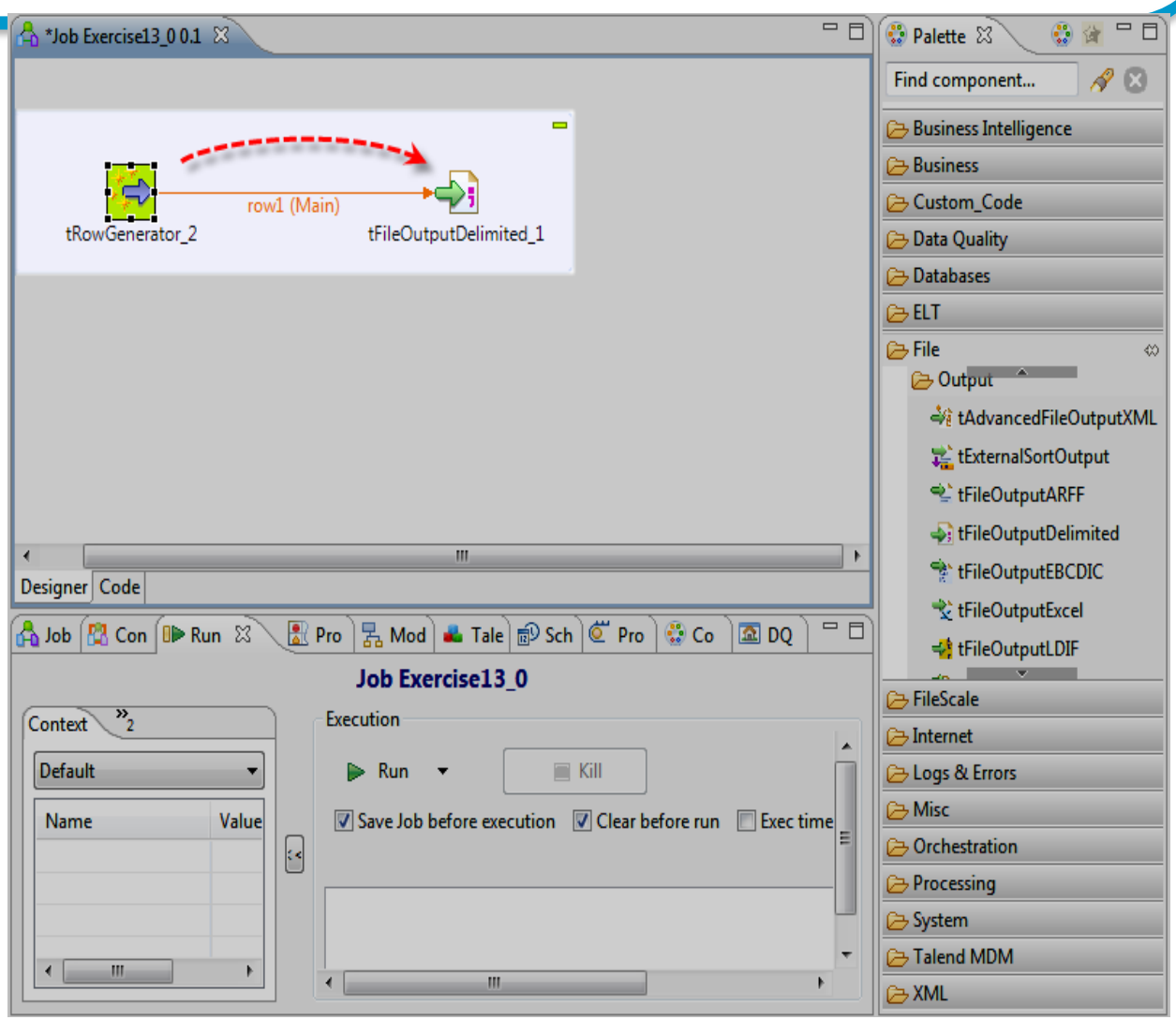
Catching information about ‘Job' executions (Cont.…)

- In the Palette:
- Click
  the File > Output folde
  r.
- Click
  the tFileOutputDelimit
  ed and drop it on the
  Job Designer to place
  it to the right of
  thetRowGenerator.



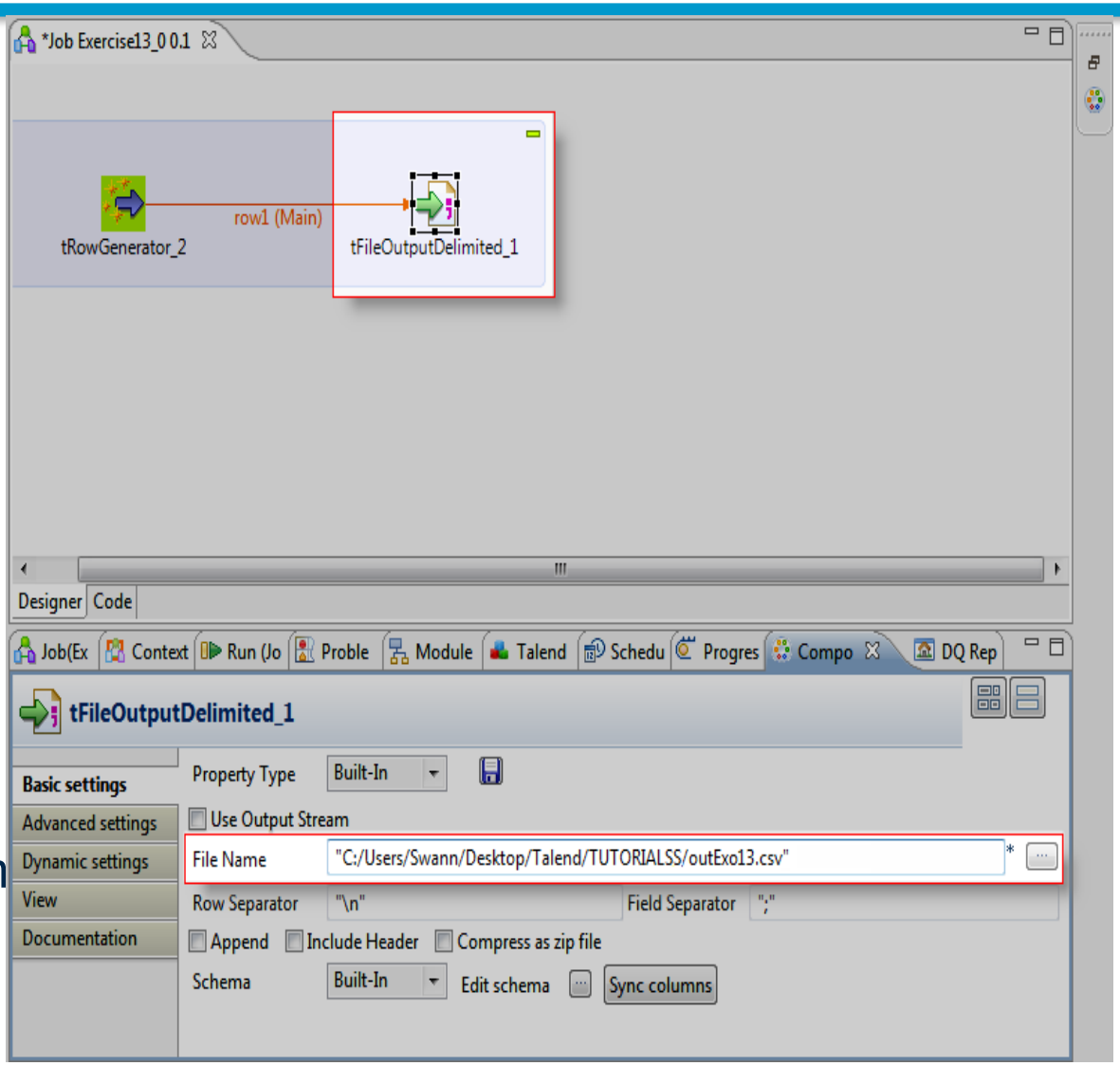
Catching information about ‘Job' executions (Cont.…)

- In the Job Designer:
- Right-click the tRowGenerator, hold and drag to the tFileOutputDelimited to create the row link.



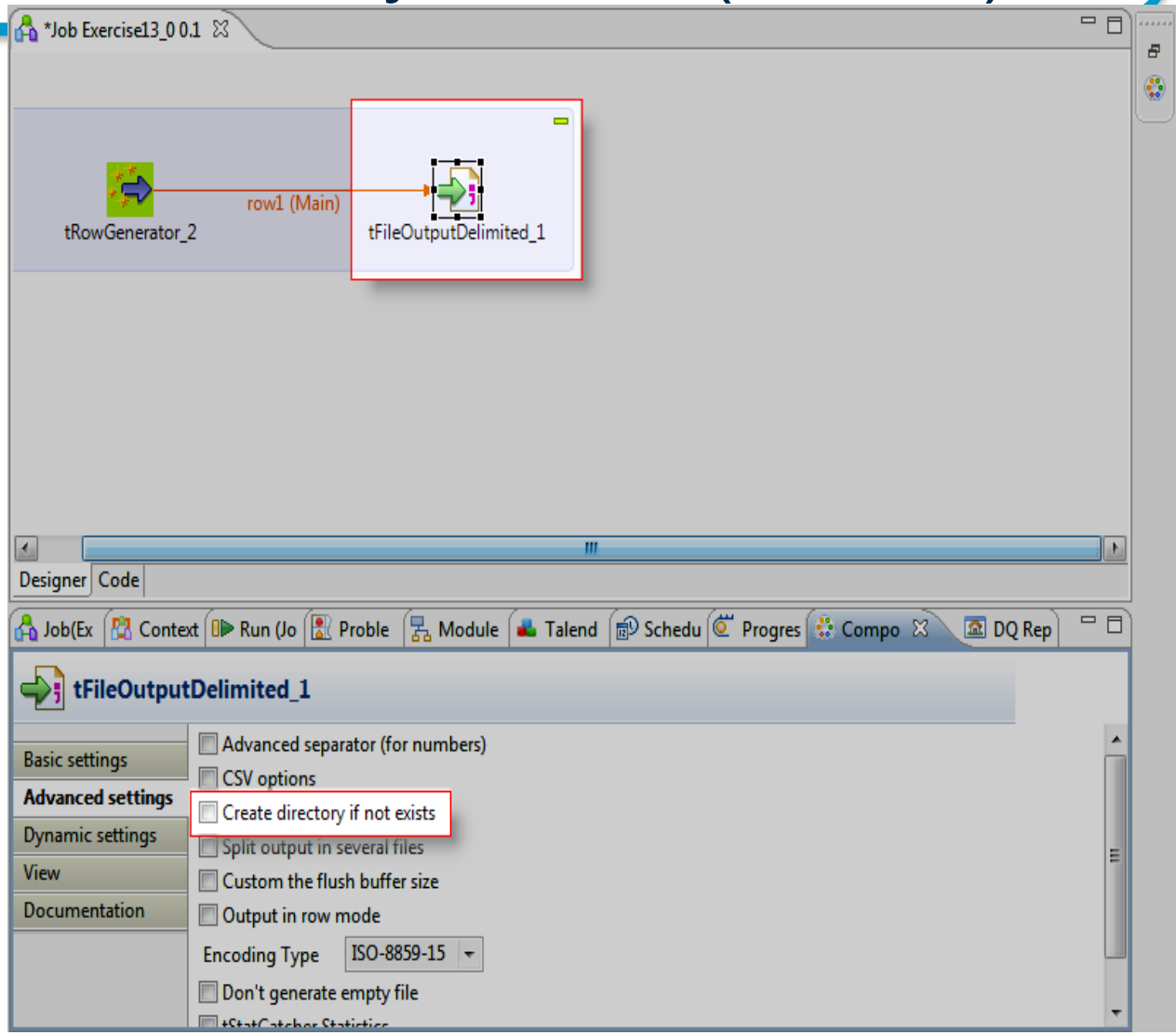
Catching information about ‘Job' executions - Step 1: Include an error in your Job

- In the Job Designer:
- Double-click the tFileOutputDelimited component to display its Component view.
- Click [...] next to the File Name field to specify the path and name of the file your are creating. The path shall not exist.



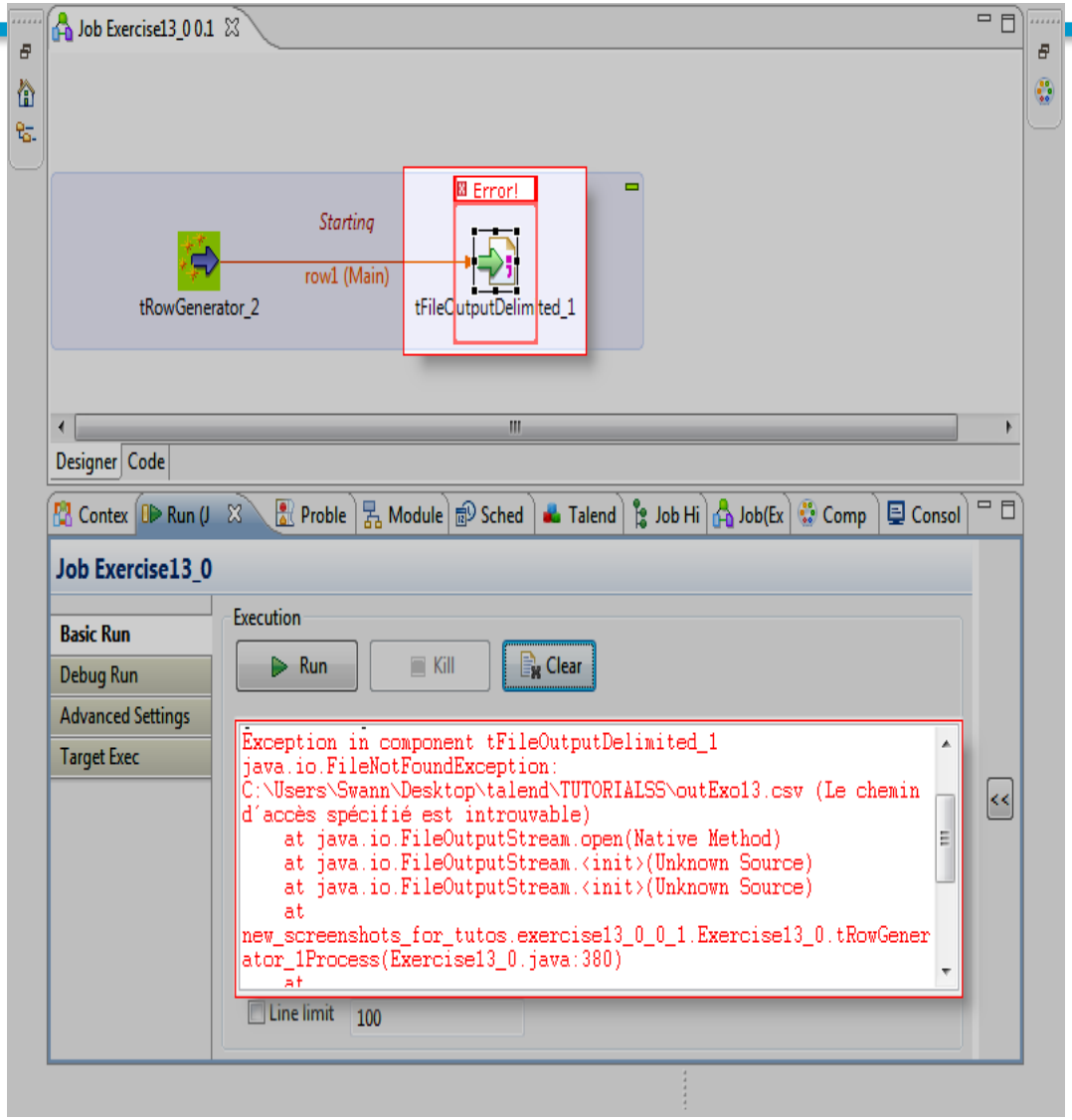
Catching information about ‘Job' executions - Step 1: Include an error in your Job (Cont…)

- In the Component view:
- Click the Advanced settings tab.
- Uncheck the Create directory if not exists box to obtain an error when executing the Job.



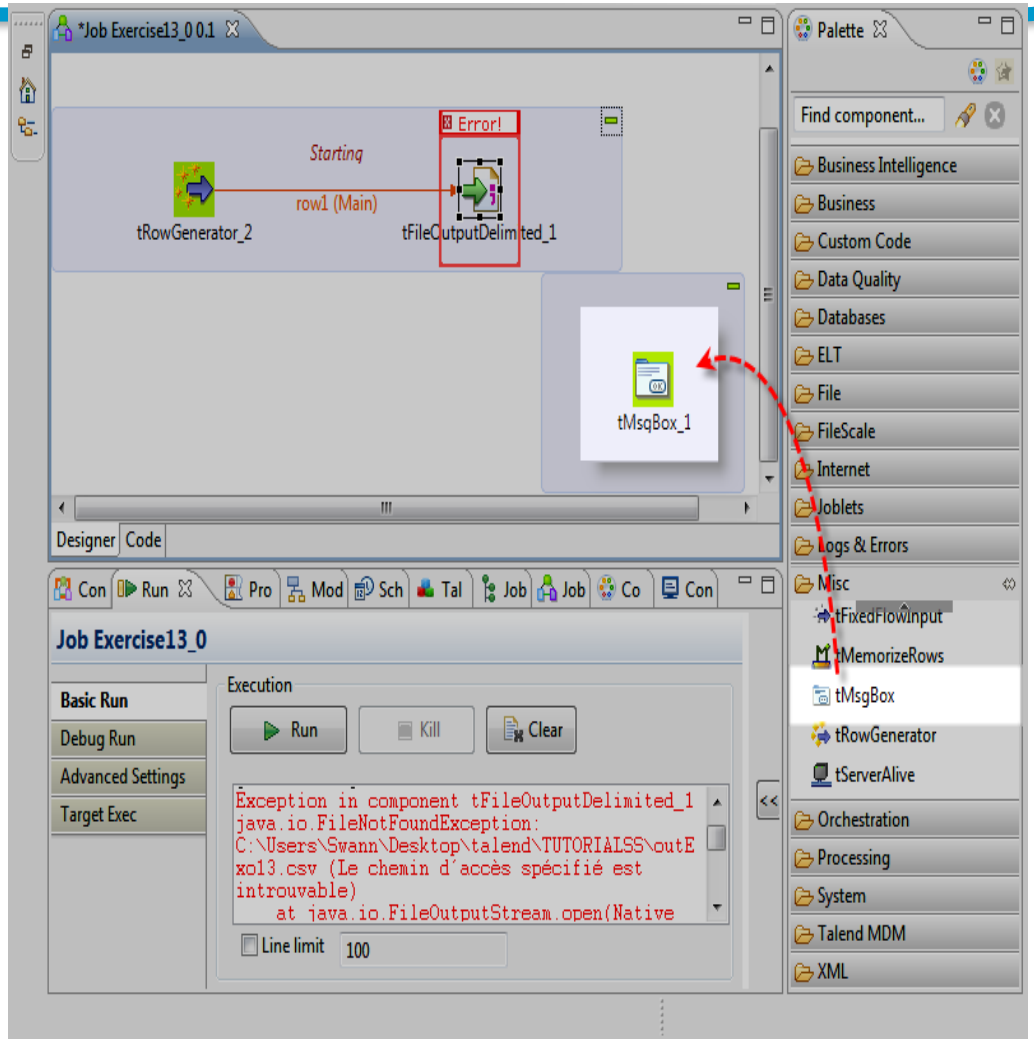
Catching information about ‘Job' executions - Step 1: Include an error in your Job (Cont…)

- Press F6 to run the Job and see the errors.



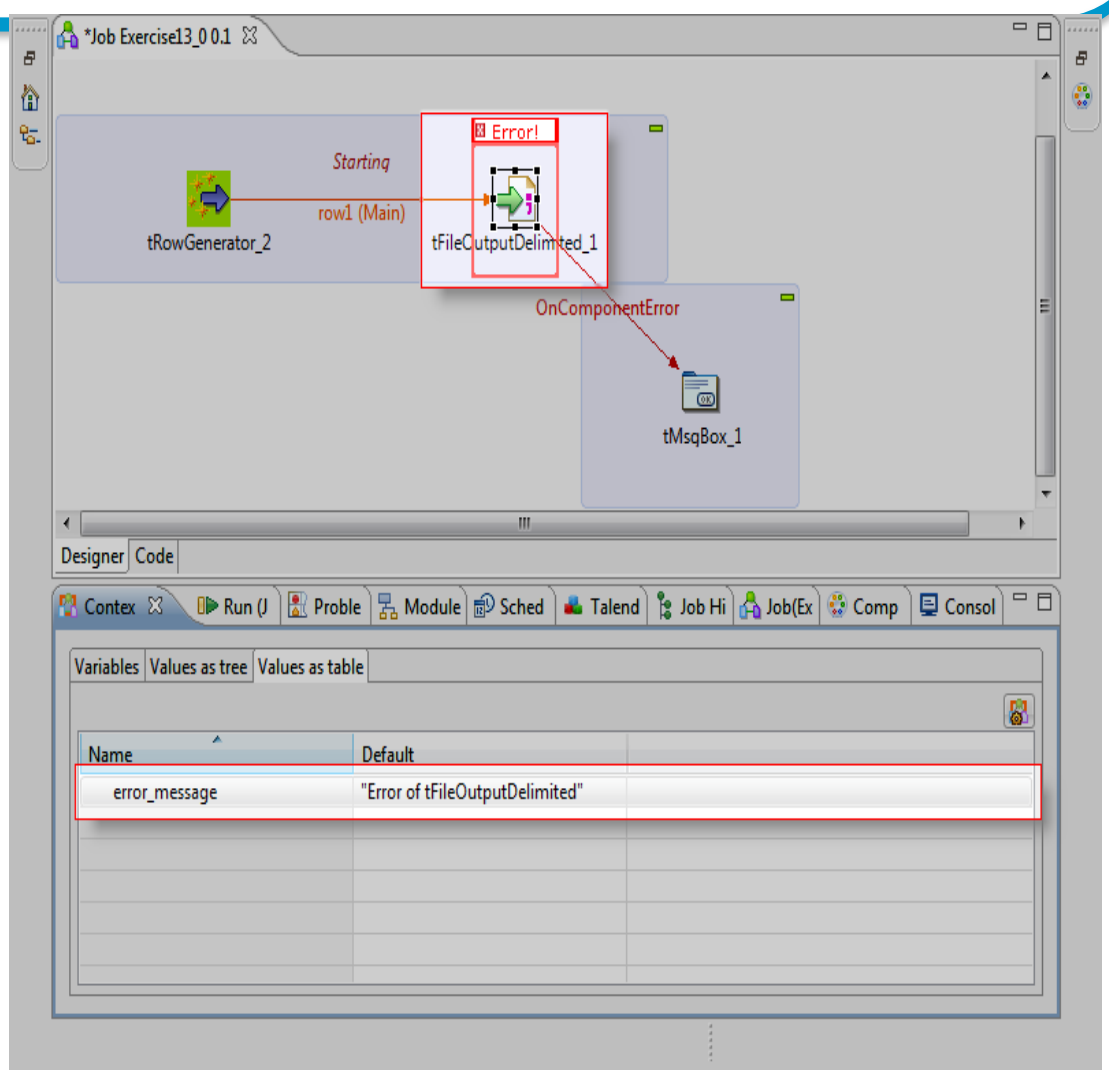
Catching information about ‘Job' executions – Add tMsgBox component

- In the Palette:
- Click the Misc folder.
- Click the tMsgBox component and drop it on the Job Designer below the tFileOutputDelimited.



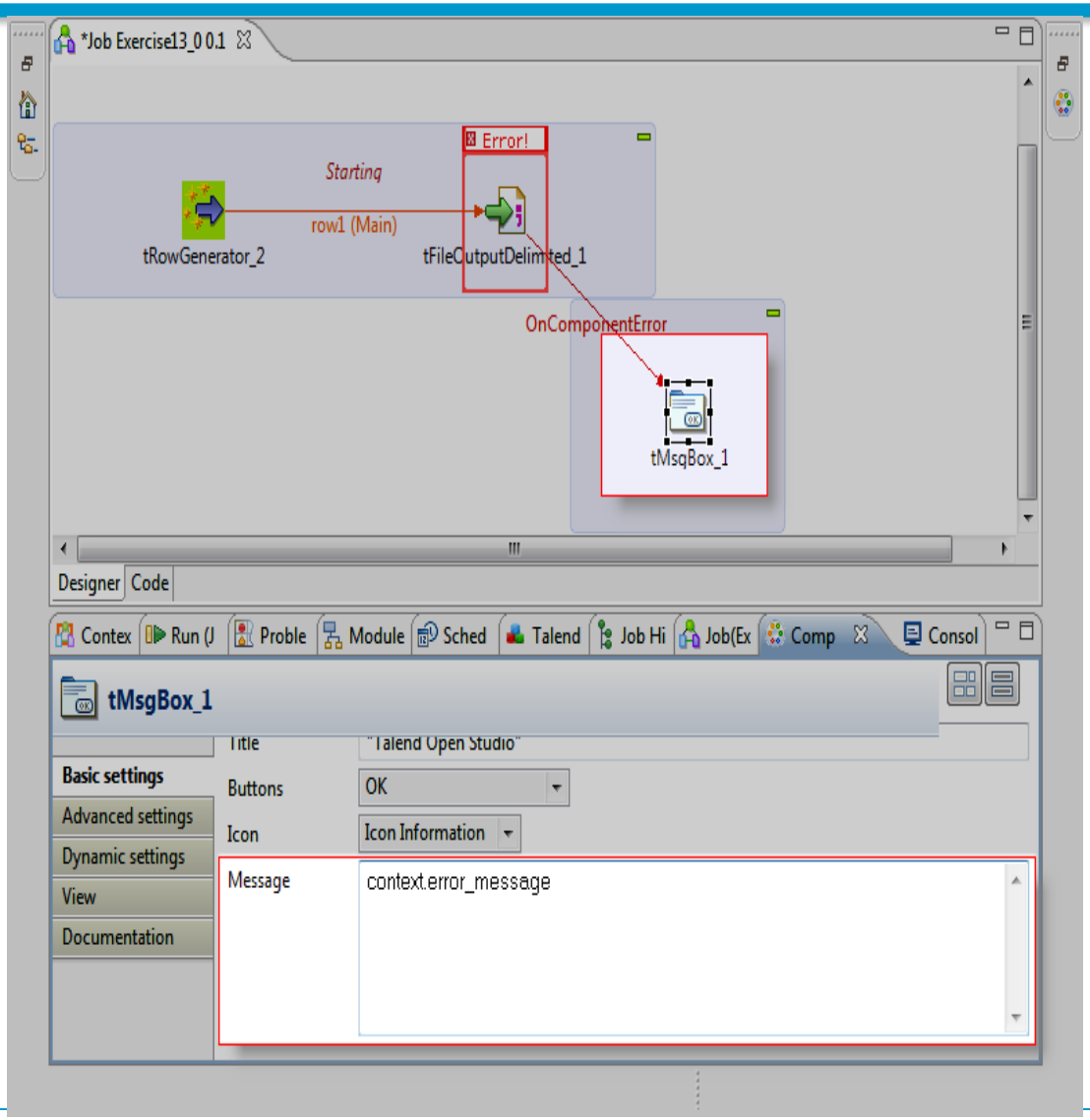
Catching information about ‘Job' executions - Step 2: Use the OnError Component link

- In the Job Designer:
- Define the context variable that will be used to display the error message.
- Click the Context view.
- Click [+] to add a line in the Variables tab and name this new variable: error_message.
- Select this new variable and click the Values as table tab.
- In the Default field, type in "Error of tFileOutputDelimited".



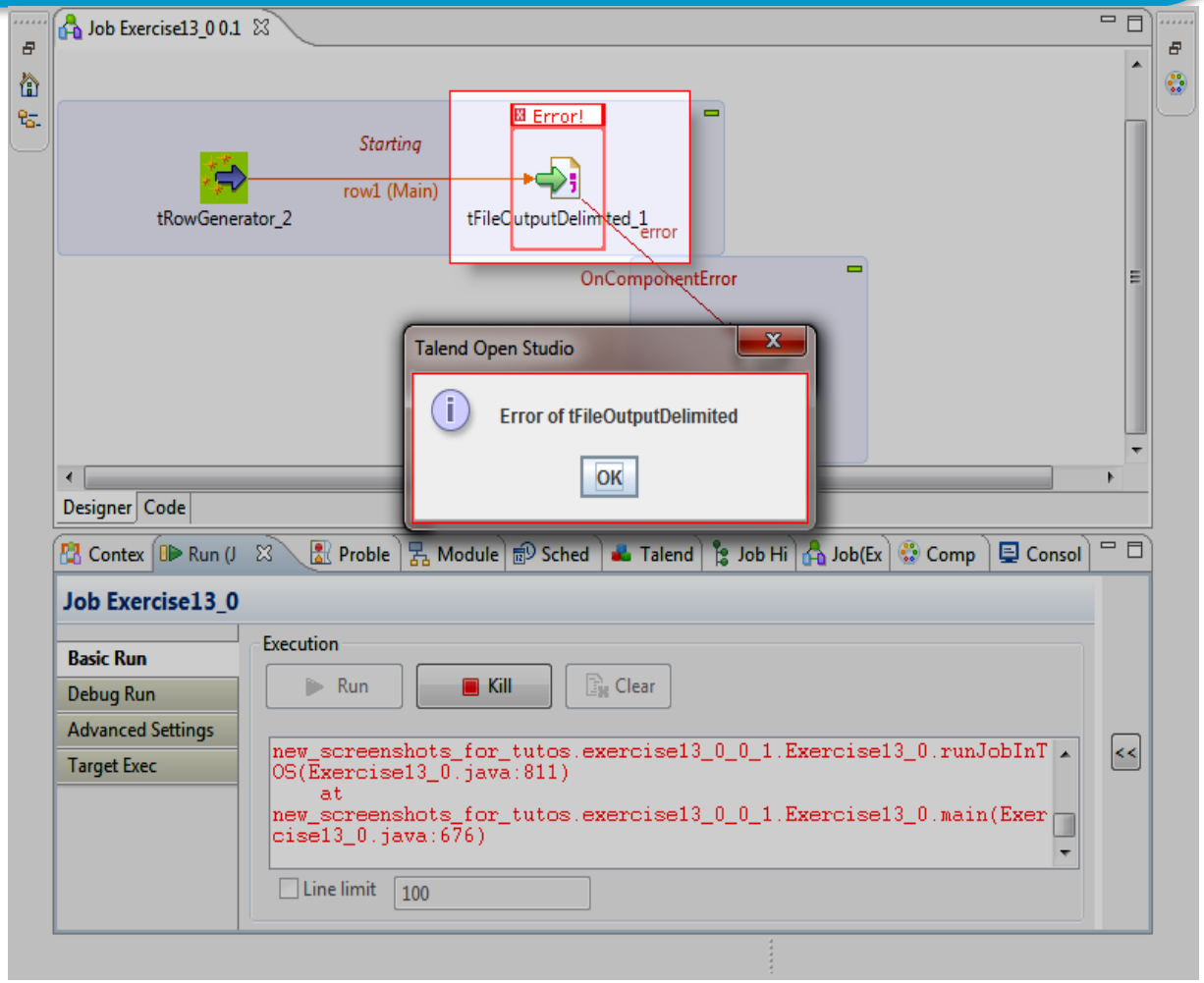
Catching information about ‘Job' executions - Step 2: Use the OnError Component link

- In the Job Designer:
- Double-click the tMsgBox to display the Component view.
- In the Message field, press Ctrl+Space to display the autocompletion list. In this list, click the new context.error_messag e variable.



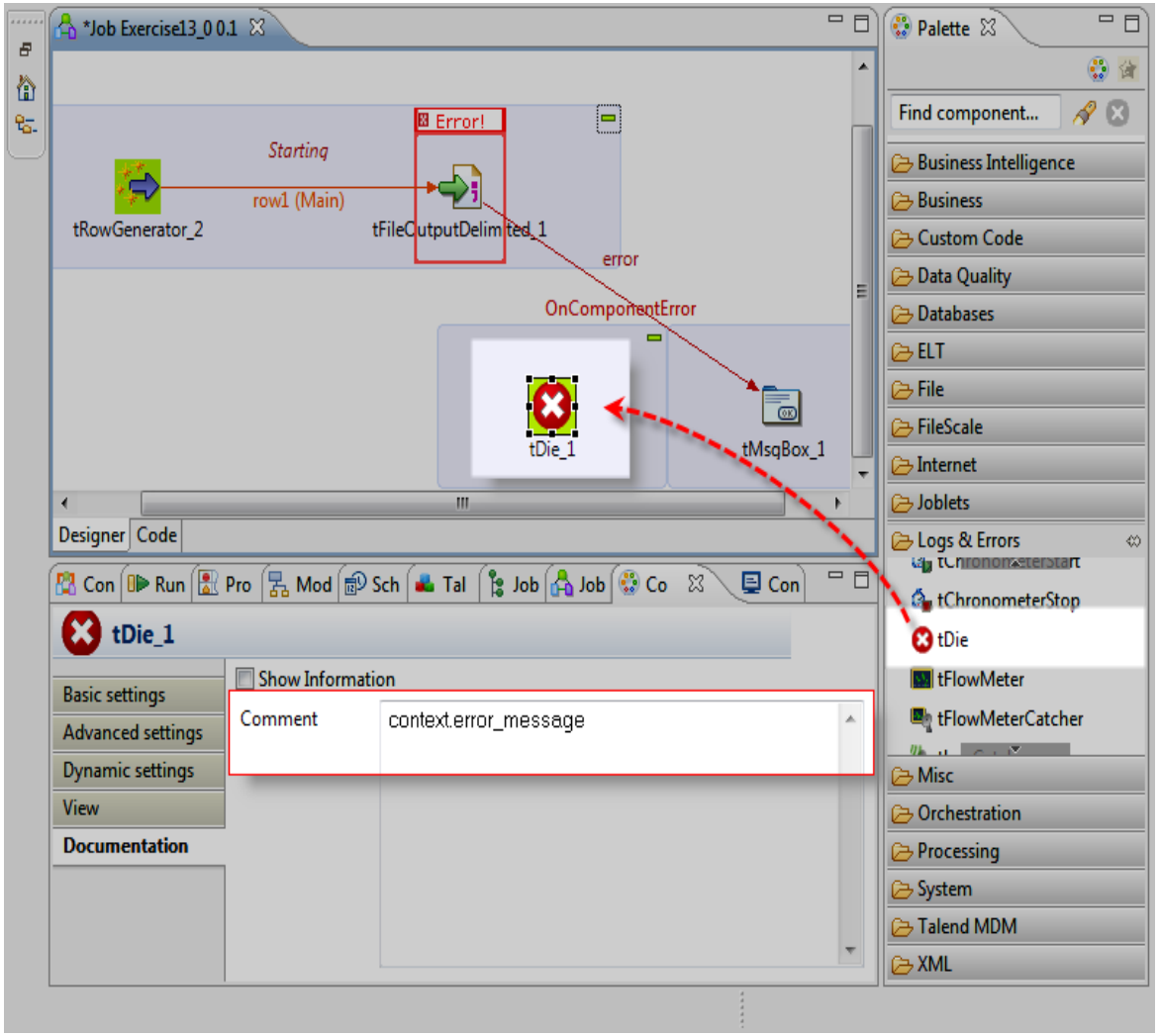
Catching information about ‘Job' executions - Step 2: Use the OnErrorComponent link

- Press F6 to run the Job. A message box will appear and display the error message.



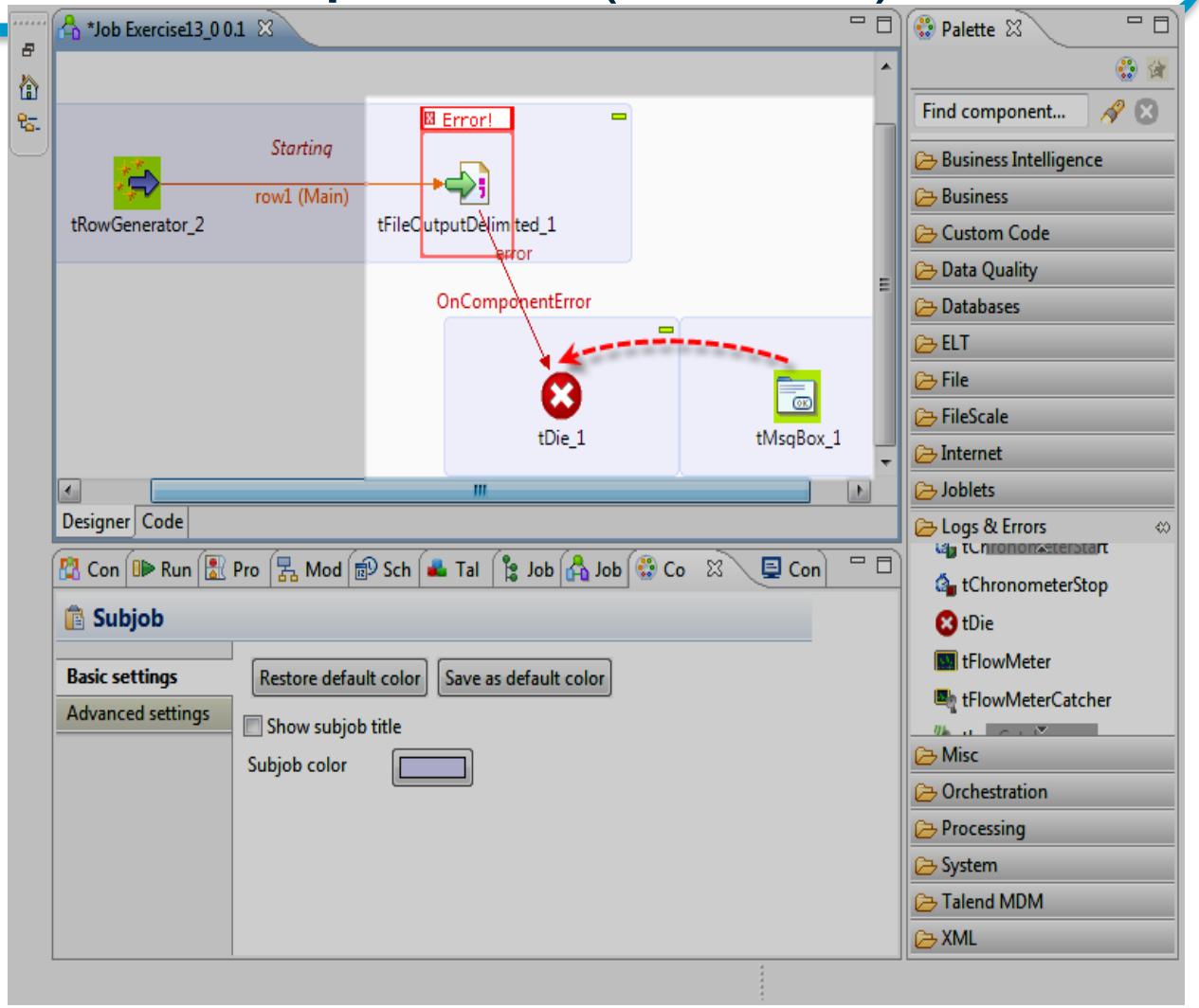
Catching information about ‘Job' executions - Step 3: Add a tDie component

- In the Palette:
- Click the Logs & Errors folder.
- Click the tDie component and drop it on the Job Designer instead of the tMsgBox.
- Double-click the tDie to display its Component view.
- In the Comment field, press Ctrl+Space to display the autocompletion list. In this list, click the context.error_message variable.



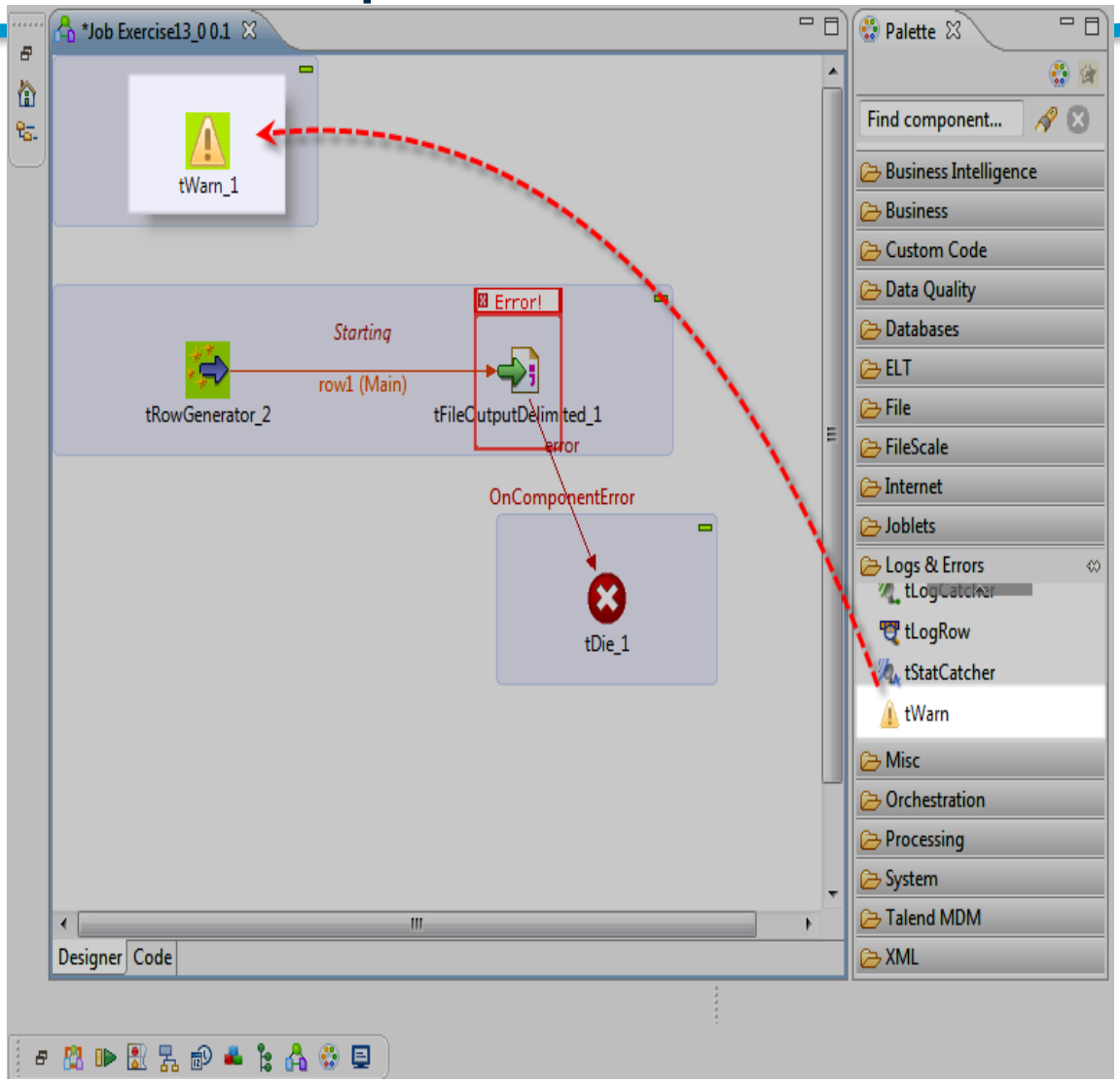
Catching information about ‘Job' executions - Step 3: Add a tDie component (Cont…)

- Select the OnComponentError link at the tMsgBox and move it to the tDie. Then, delete the tMsgBox component.



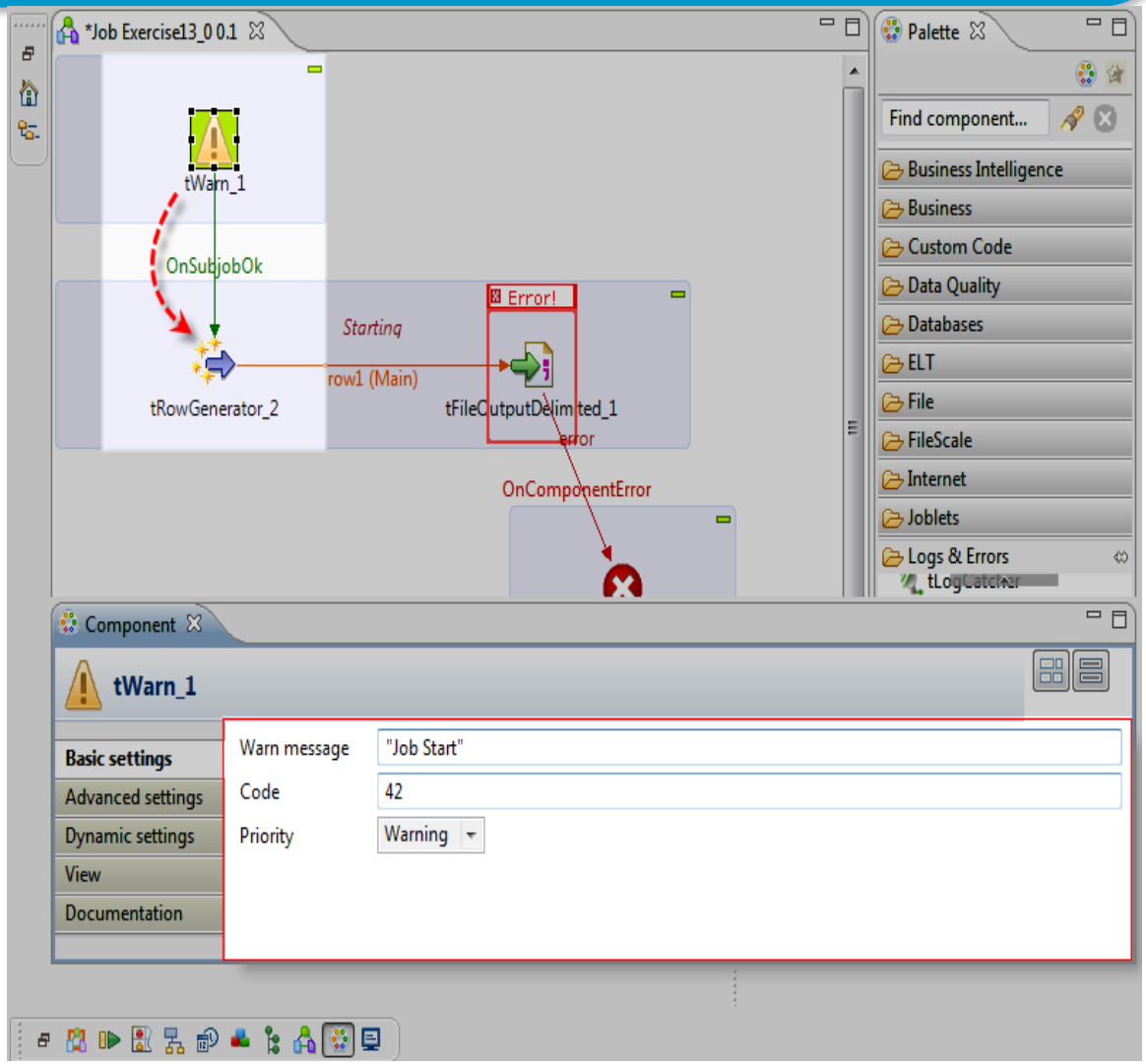
Catching information about ‘Job' executions - Step 4: Add a tWarn component

- In the Palette:
- Click the tWarn component and drop it on the Job Designer above the tRowGenerator.



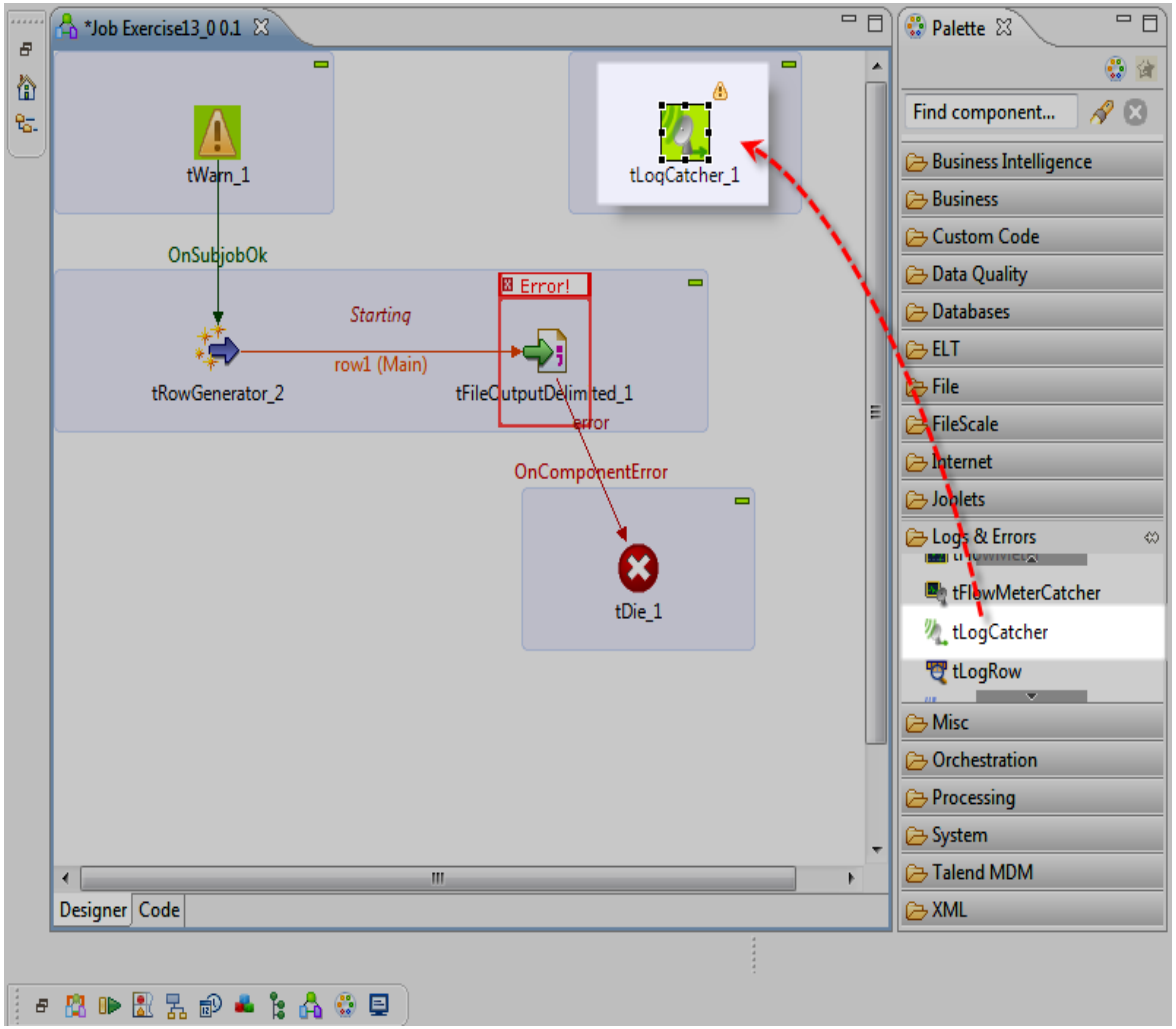
Catching information about ‘Job' executions - Step 4: Add a tWarn component (Cont…)

- In the Job Designer:
- Double-click the tWarn to display its Component view.
- In the Warn message field, type in "Job Start".
- Right-click the tWarn, select Trigger > OnSubjobOk in the menu and then click the tRowGenerator to create a link.



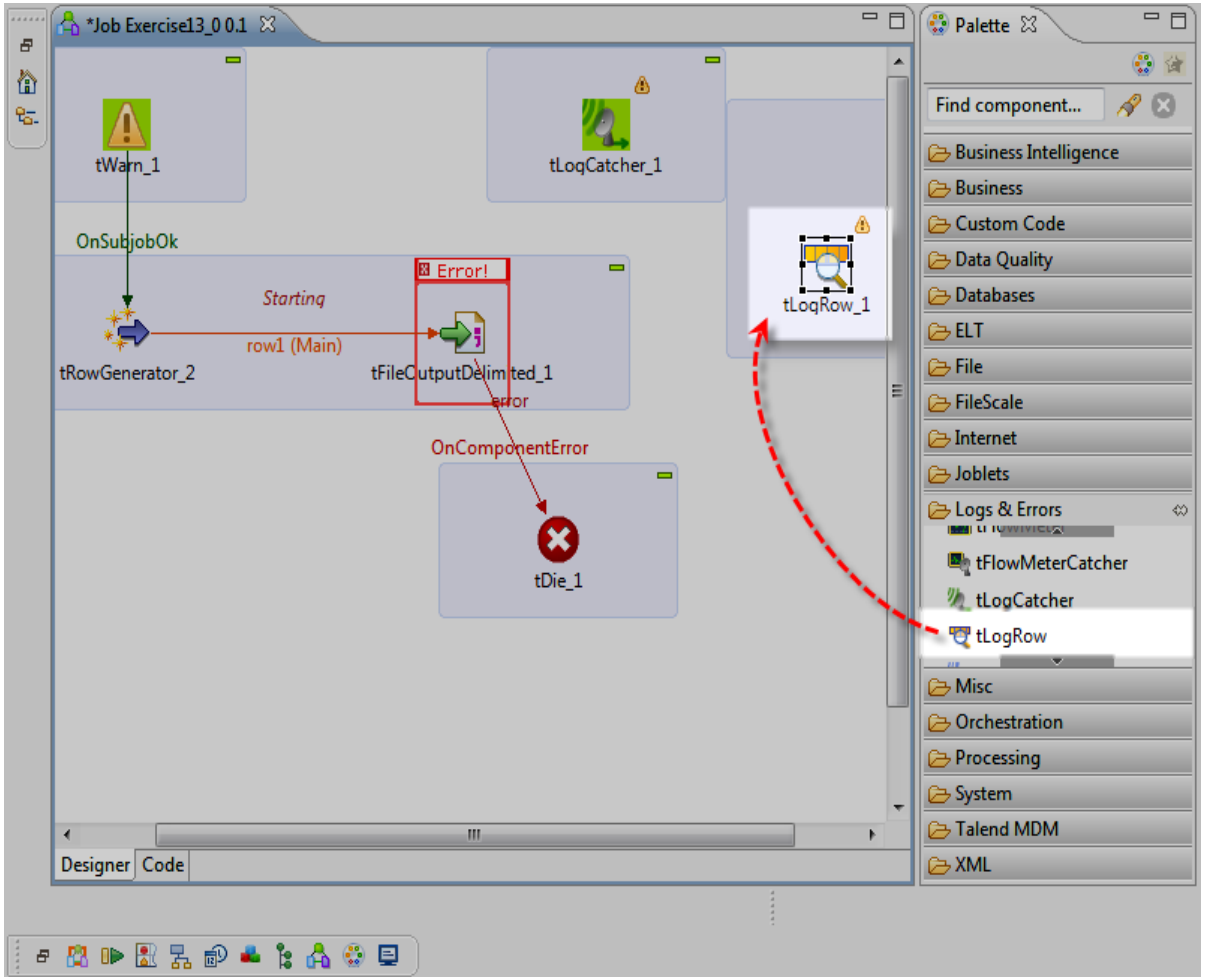
Catching information about ‘Job' executions - Step 5: Catch the message with a tLogCatcher

- In the Palette:
- Click the tLogCatcher and drop it on the Job Designer to the top right of the Job Designer.



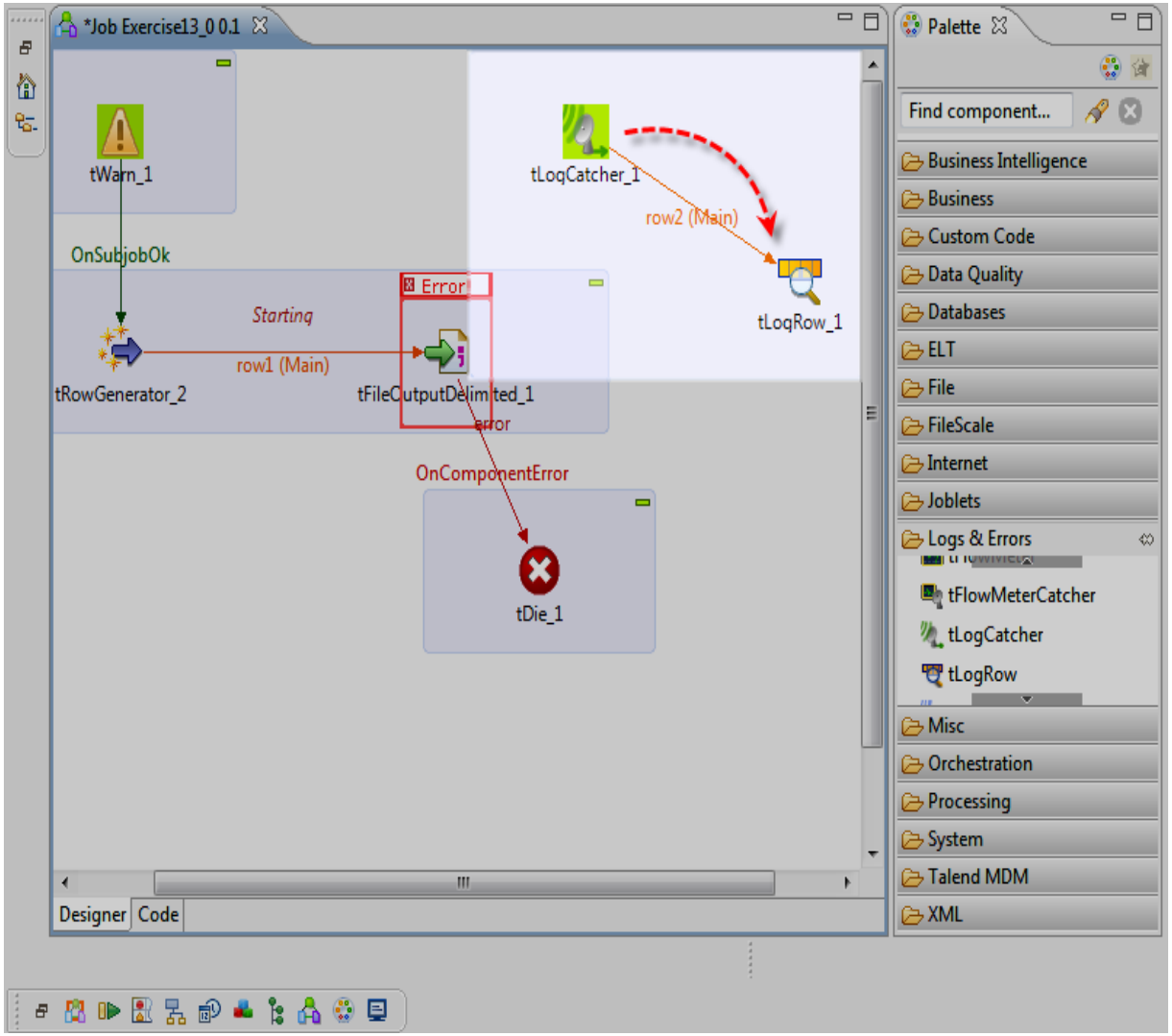
Catching information about ‘Job' executions - Step 5: Catch the message with a tLogCatcher (Cont…)

- In the Palette:
- Click the tLogRow component and drop it on the Job Designer below the tLogCatcher.



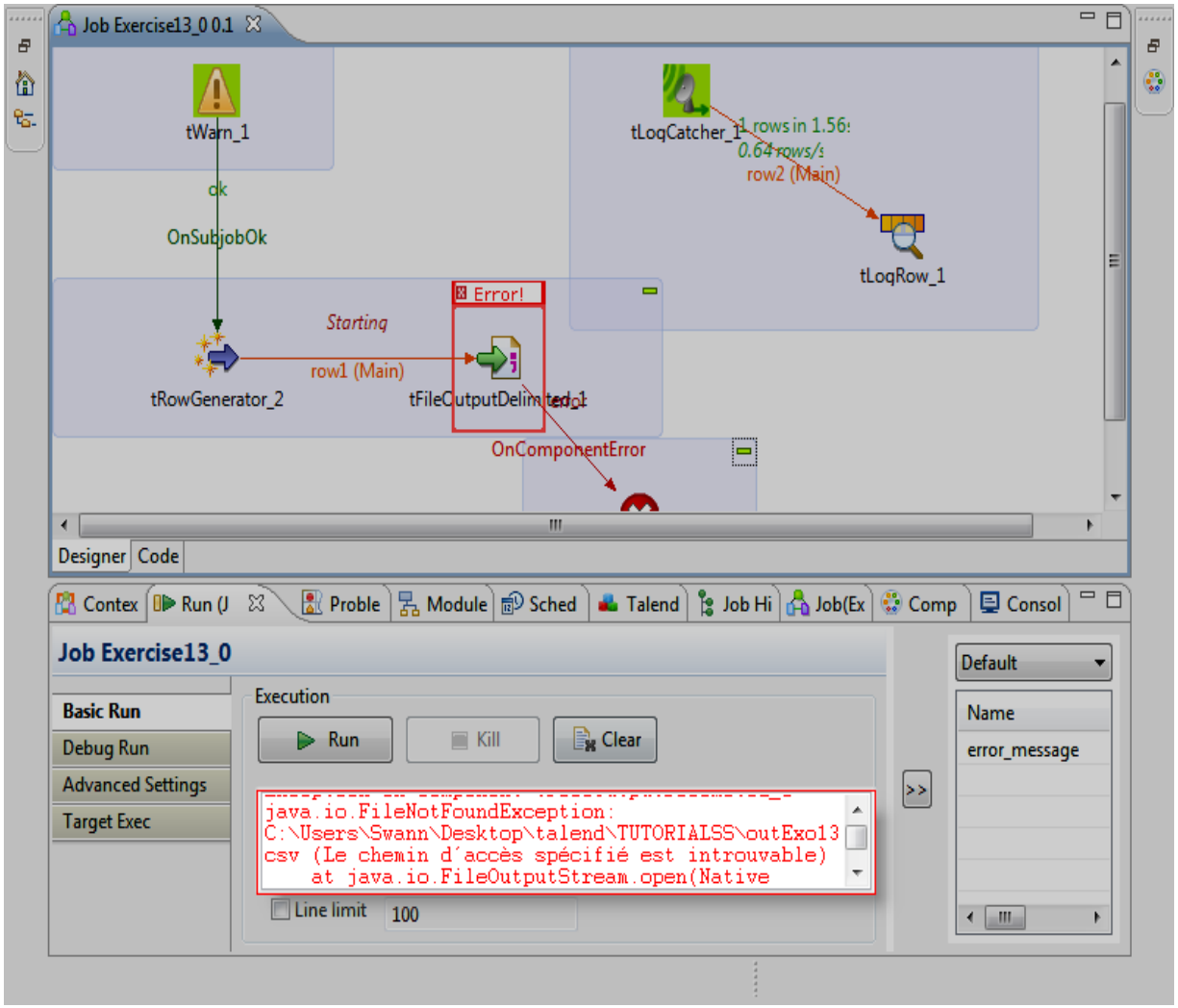
Catching information about ‘Job' executions - Step 5: Catch the message with a tLogCatcher (Cont…)

- In the Job Designer:
- Right-click the tLogCatcher, hold and drag to the tLogRow to create a row link.



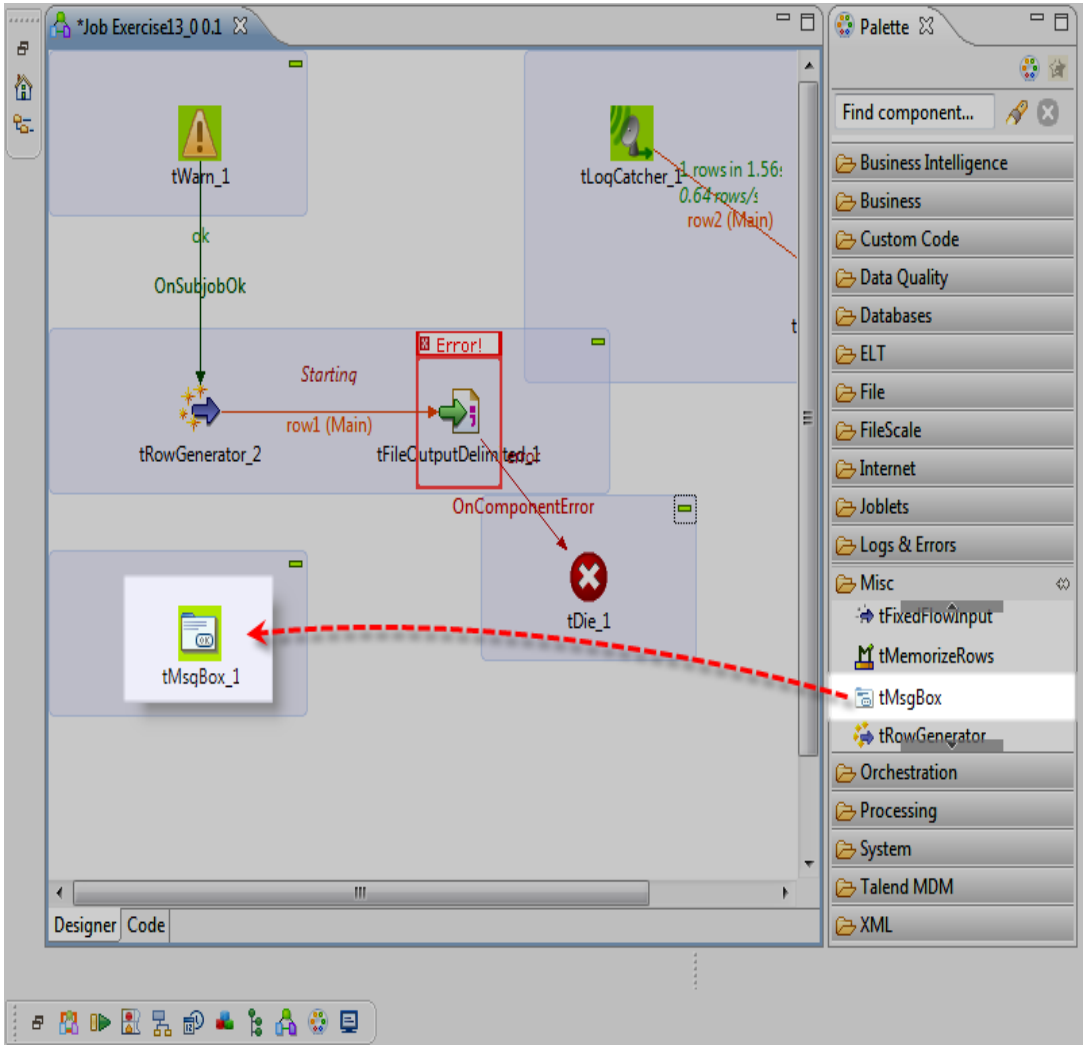
Catching information about ‘Job' executions - Step 5: Catch the message with a tLogCatcher (Cont…)

- Press F6 to run the Job and look at the log messages in the Console, including the one which is linked to the file creation error.



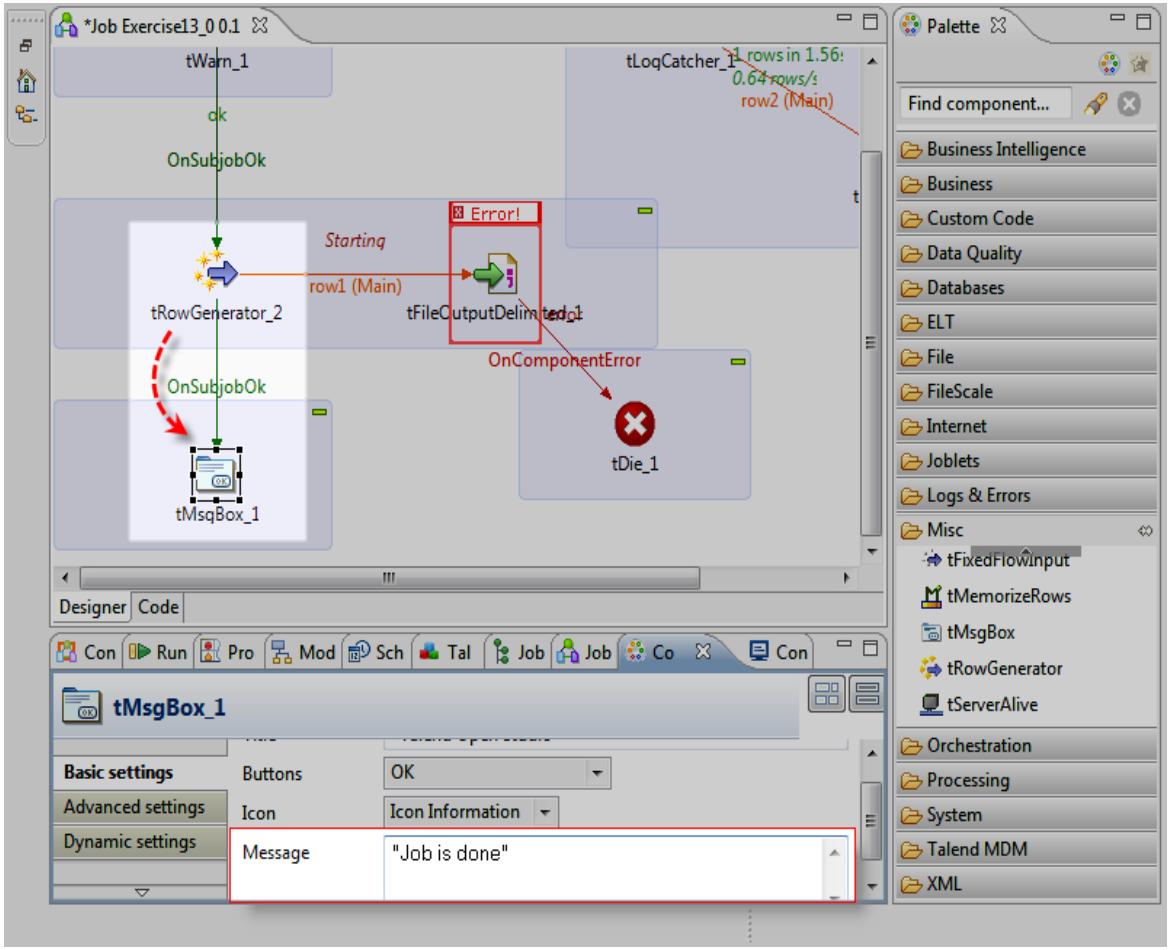
Catching information about ‘Job' executions - Step 5: Catch the message with a tLogCatcher (Cont…)

- In the Palette:
- Click the Misc folder.
- Click the tMsgBox component and drop it on the Job Designer below the tRowGenerator



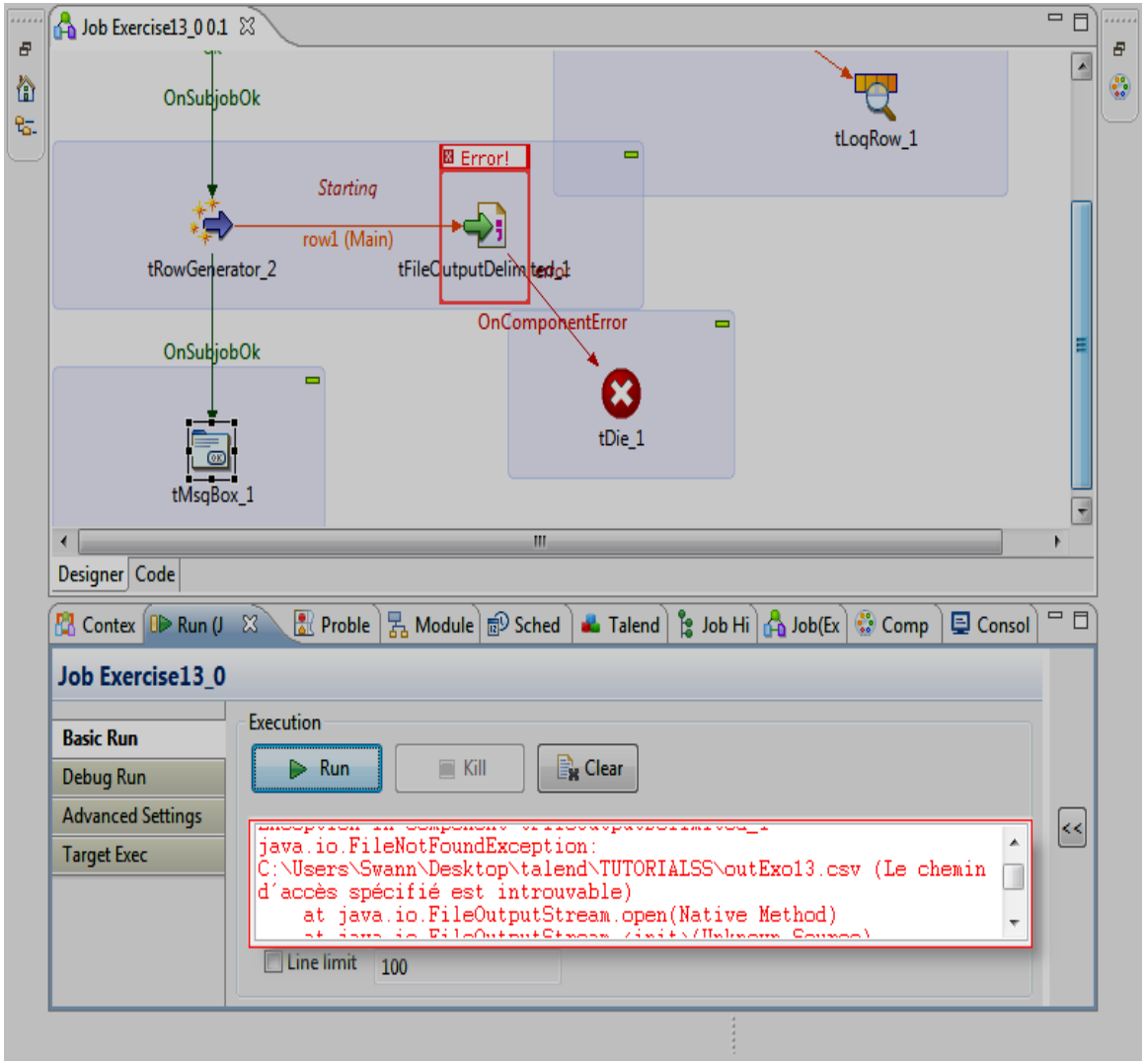
Catching information about ‘Job' executions - Step 5: Catch the message with a tLogCatcher (Cont…)

- In the Job Designer:
- Right-click the tRowGenerator component and select Trigger > On SubJob Ok in the menu, then click the tMsgBox.
- Double-click the tMsgBox to display its Component view.
- In the Message field, type in the message: "Job is done!".



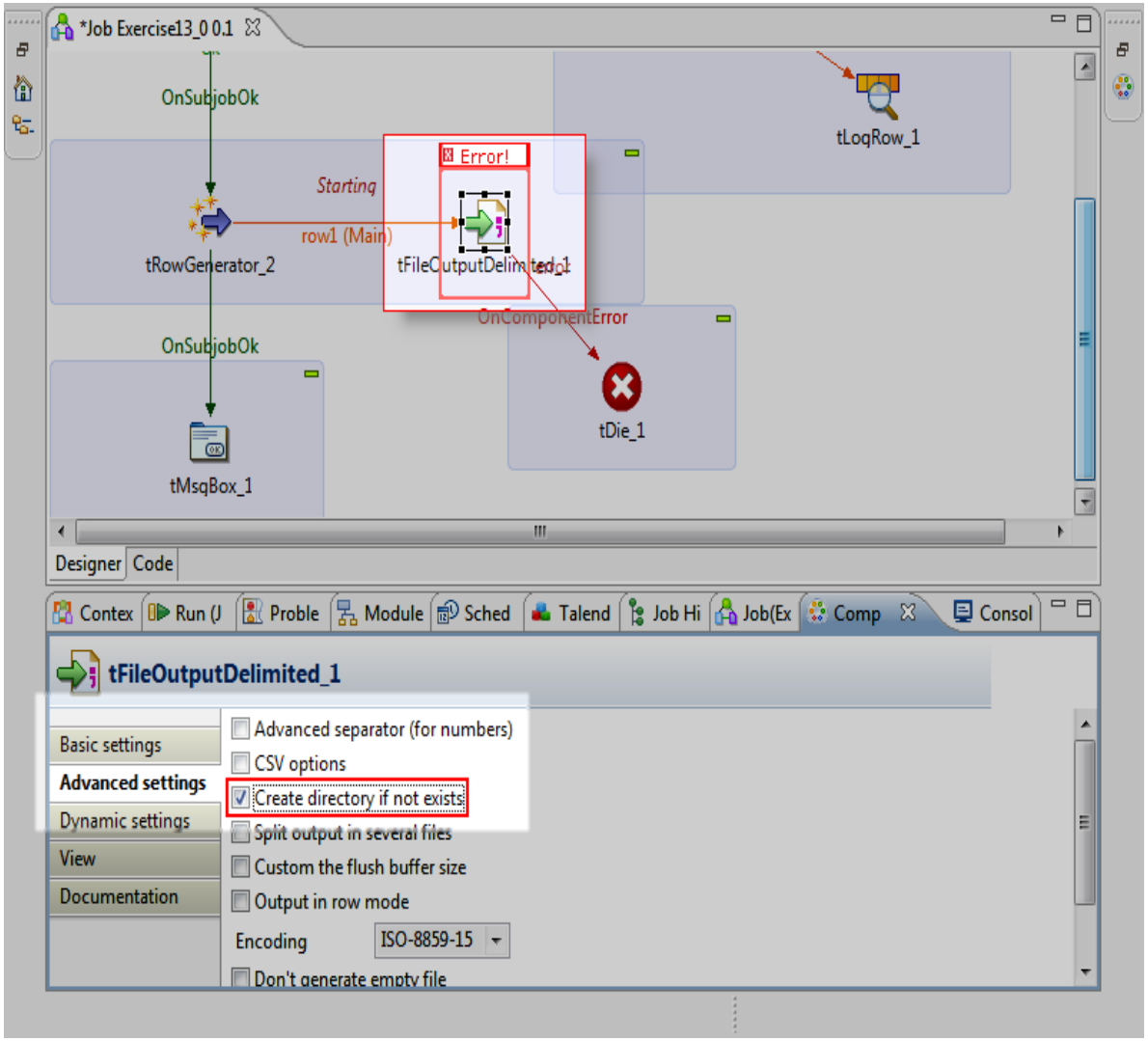
Catching information about ‘Job' executions - Step 5: Catch the message with a tLogCatcher (Cont…)

- Press F6 to run the job and observe that the tDie component kills the job so that the tMsgBox is not executed.



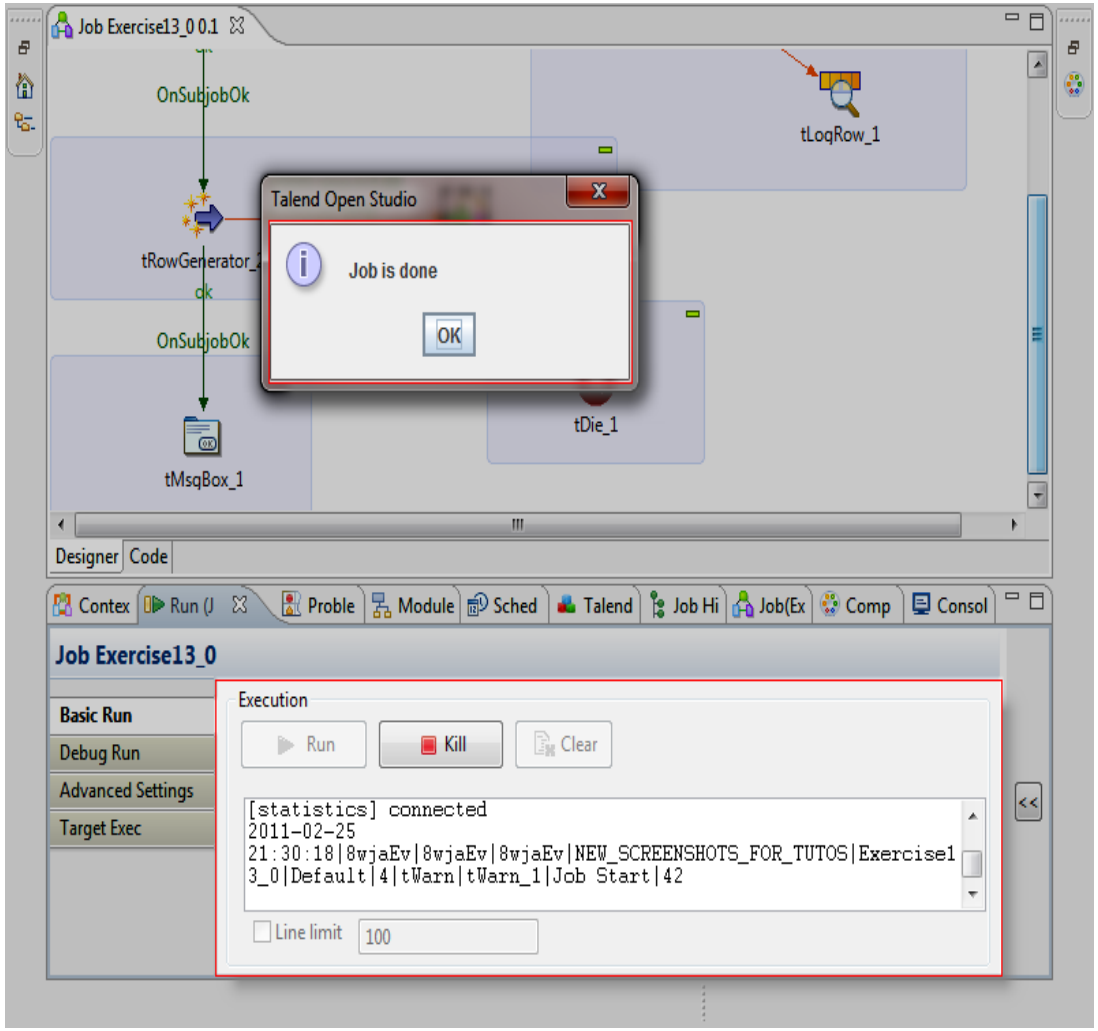
Catching information about ‘Job' executions - Step 6: Correct the error and run the Job

- Double-click the tFileOutputDelimited component to display the Component view.
- Click the Advanced settings tab.
- Check the Create directory if not exists box.



Catching information about ‘Job' executions - Step 6: Correct the error and run the Job

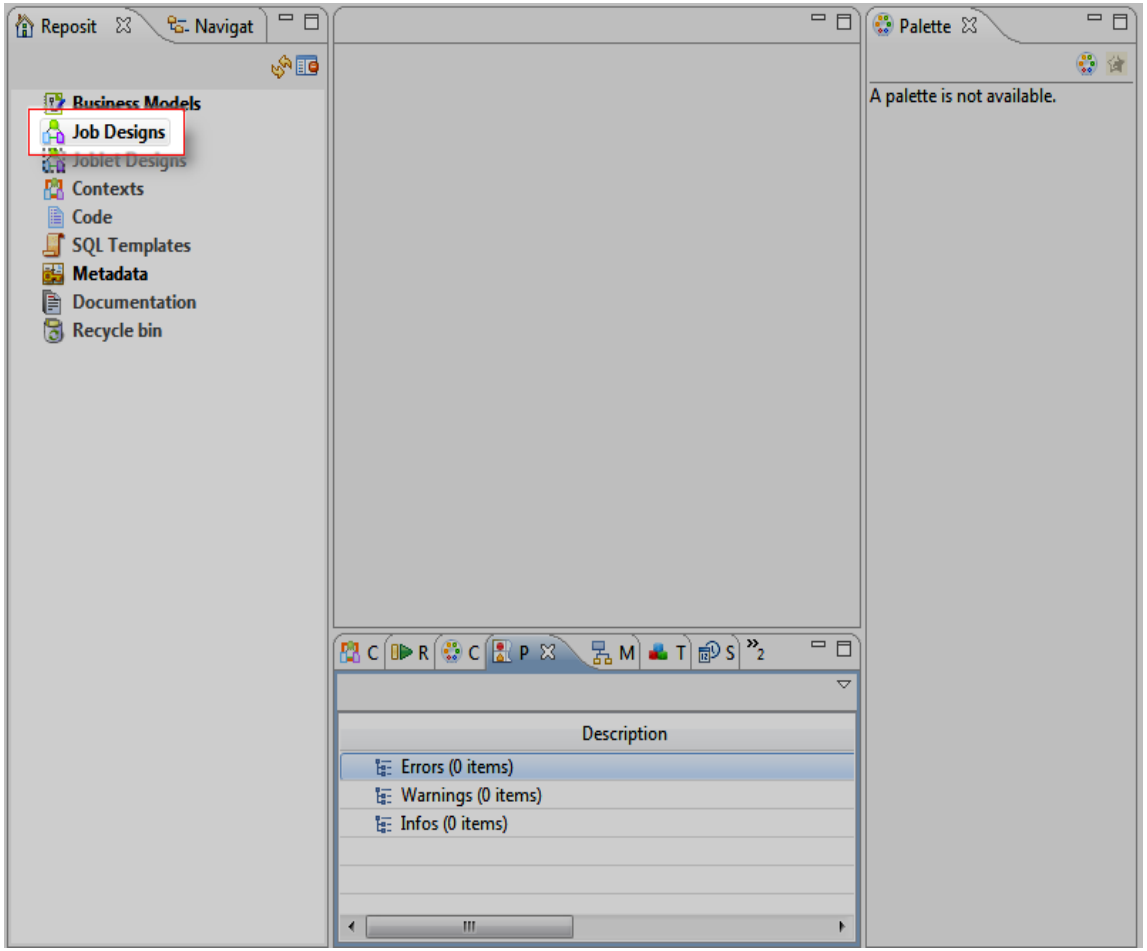
- Press F6 to run the Job again, analyze the Log messages in the Console and observe the execution of the tMsgbox.



How to use a Multi Schema component

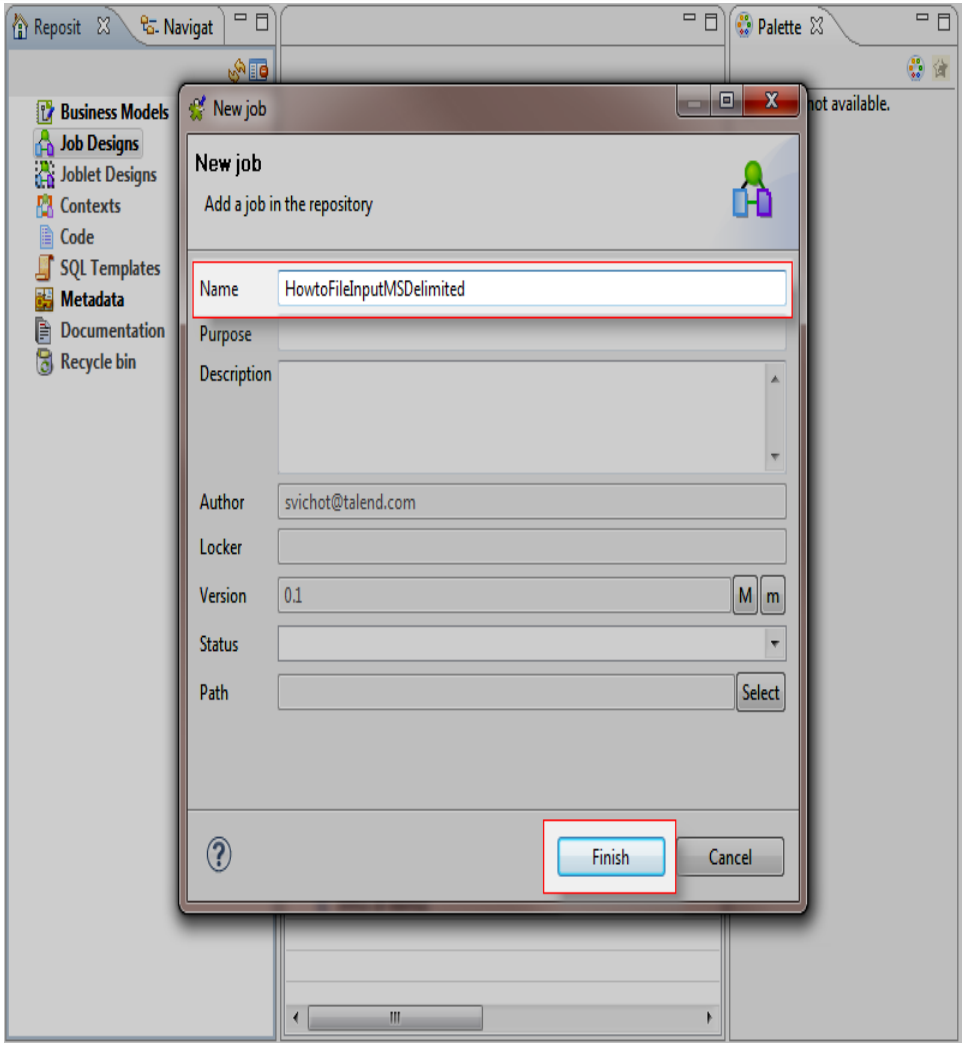
How to use a Multi Schema component

- In the Repository on the left :
- Right-click Job Designs.
- In the menu, click Create Job to open the New Job wizard



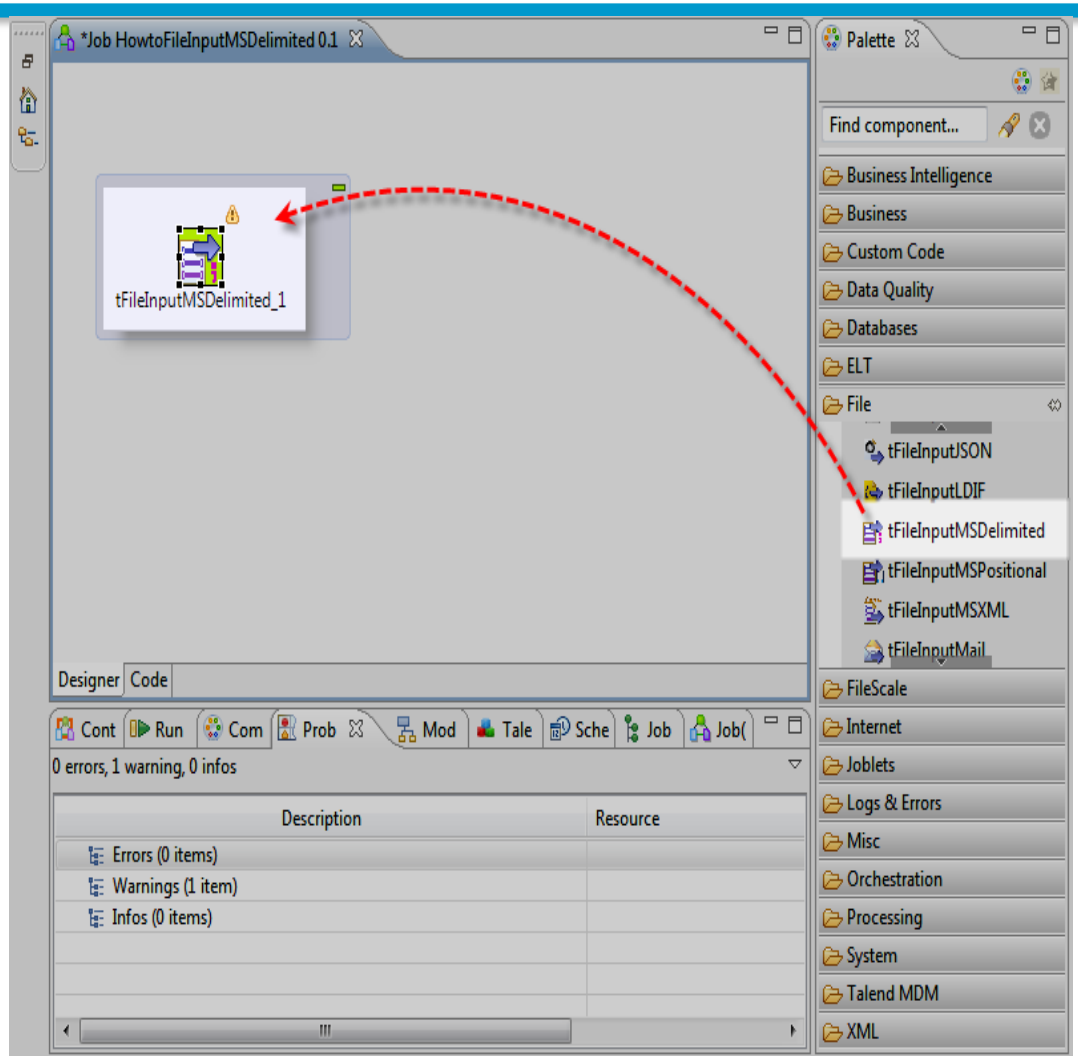
How to use a Multi Schema component (Cont…)

- In the New Job wizard:
- In the Name field, enter the name of your Job: HowtotFileInputMSDelimited.
- Click Finish to close the wizard and create your Job.
- The Job Designer opens an empty Job



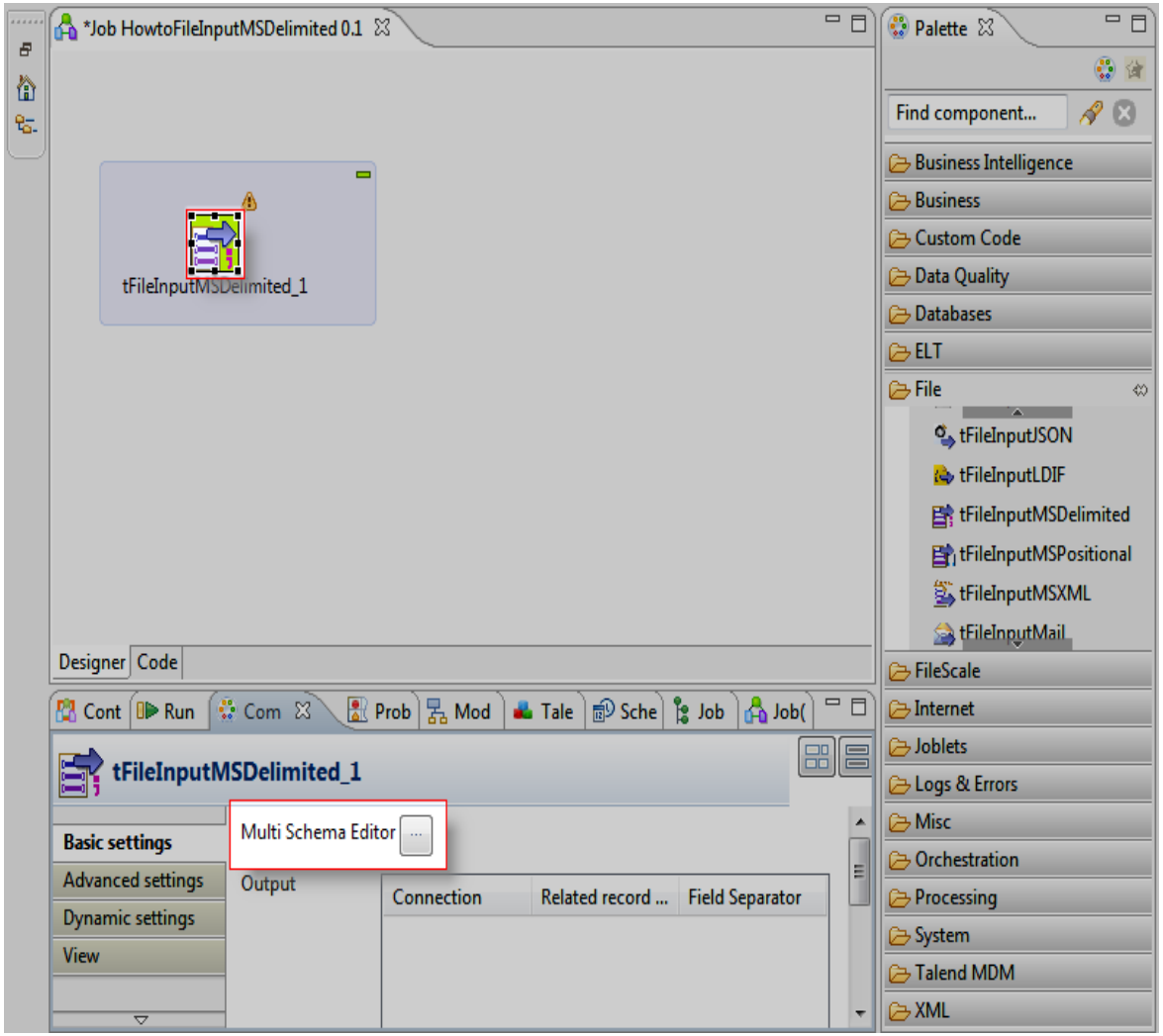
How to use a Multi Schema component (Cont…)

- In the Palette on the right:
- To add the input component, click the File family and then the Input sub-family.
- Click tFileInputMSDelimited and drop it on the Job Designer.



How to use a Multi Schema component (Cont…)

- Double-click the component to open the schema editor, or click the component and then Multi Schema Editor.

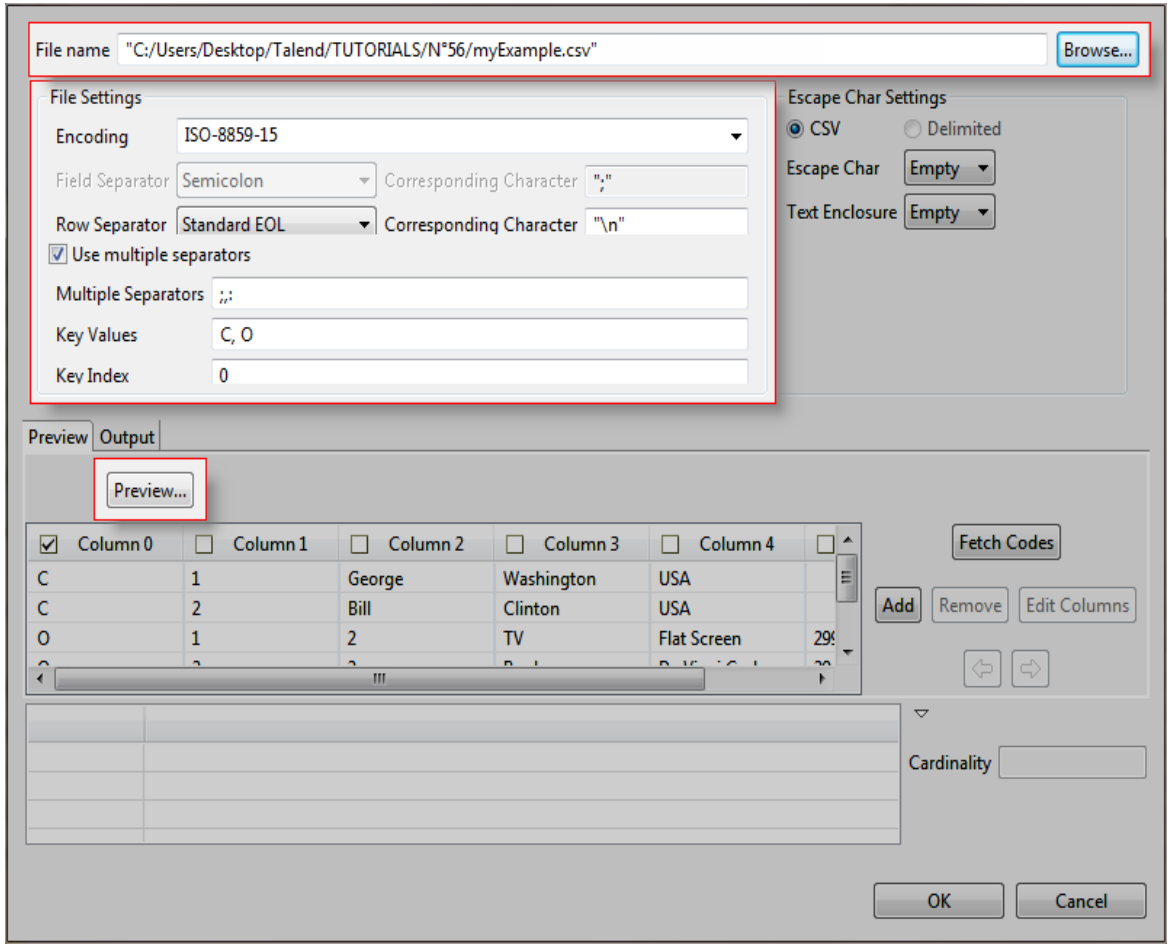


How to use a Multi Schema component (Cont…)

In the schema editor, set the file properties:

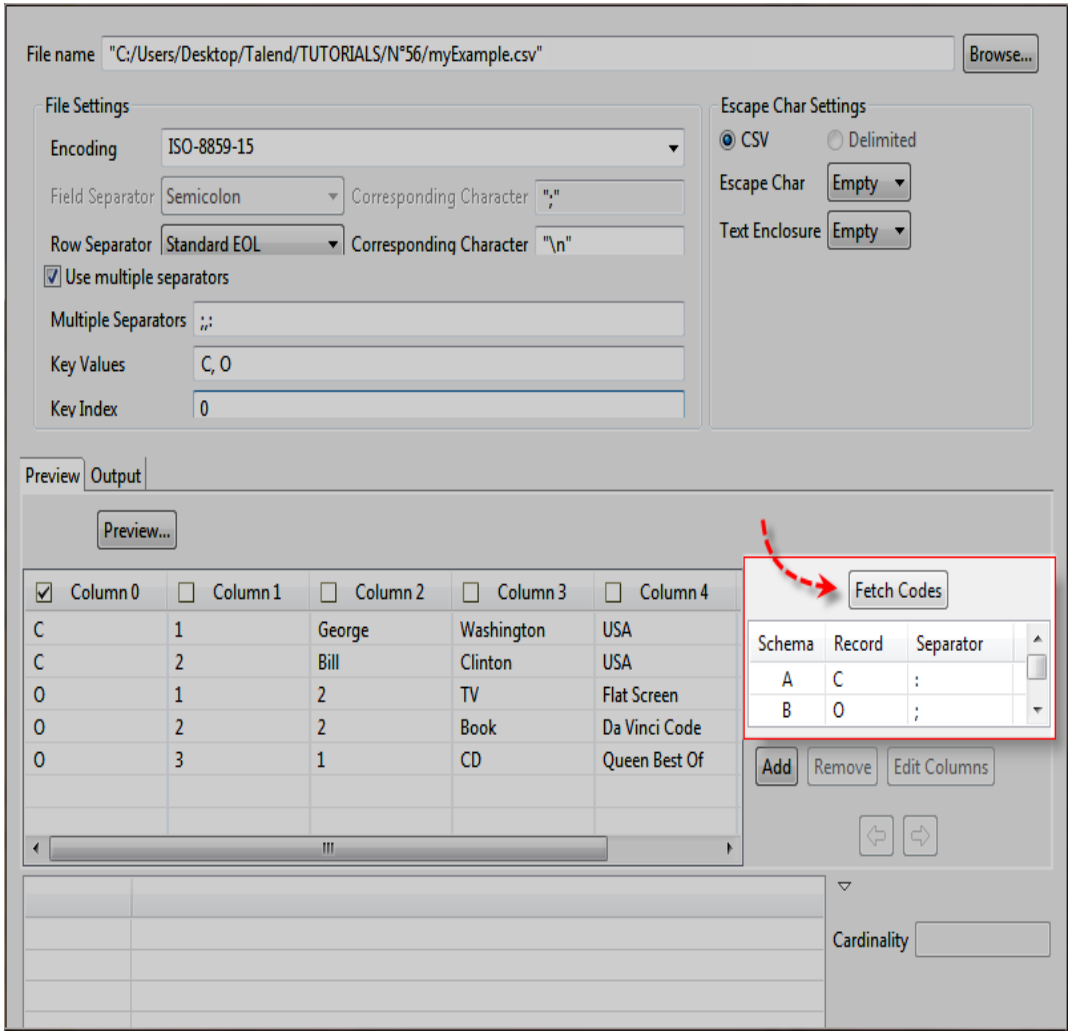
- File name: path and name of your file.
- Encoding: select your file's encoding type.
- Use multiple separators: check this box if the separators are different from one schema to another.
- Multiple Separators: fill in the separator characters used in your schemas. (In this field, use a comma as separator)
- Key Values: enter all of the values which enable identification of your file schemas. (In this field, use a comma as separator)
- Key Index: enter the position of the key value columns.

Once you have provided all of the file information, you can preview it by clicking on Preview.



How to use a Multi Schema component (Cont…)

- If the data preview matches your file schemas, click Fetch Codes to the right.
- Every schema has an associated code.



How to use a Multi Schema component (Cont…)

- Select a schema to display its properties:

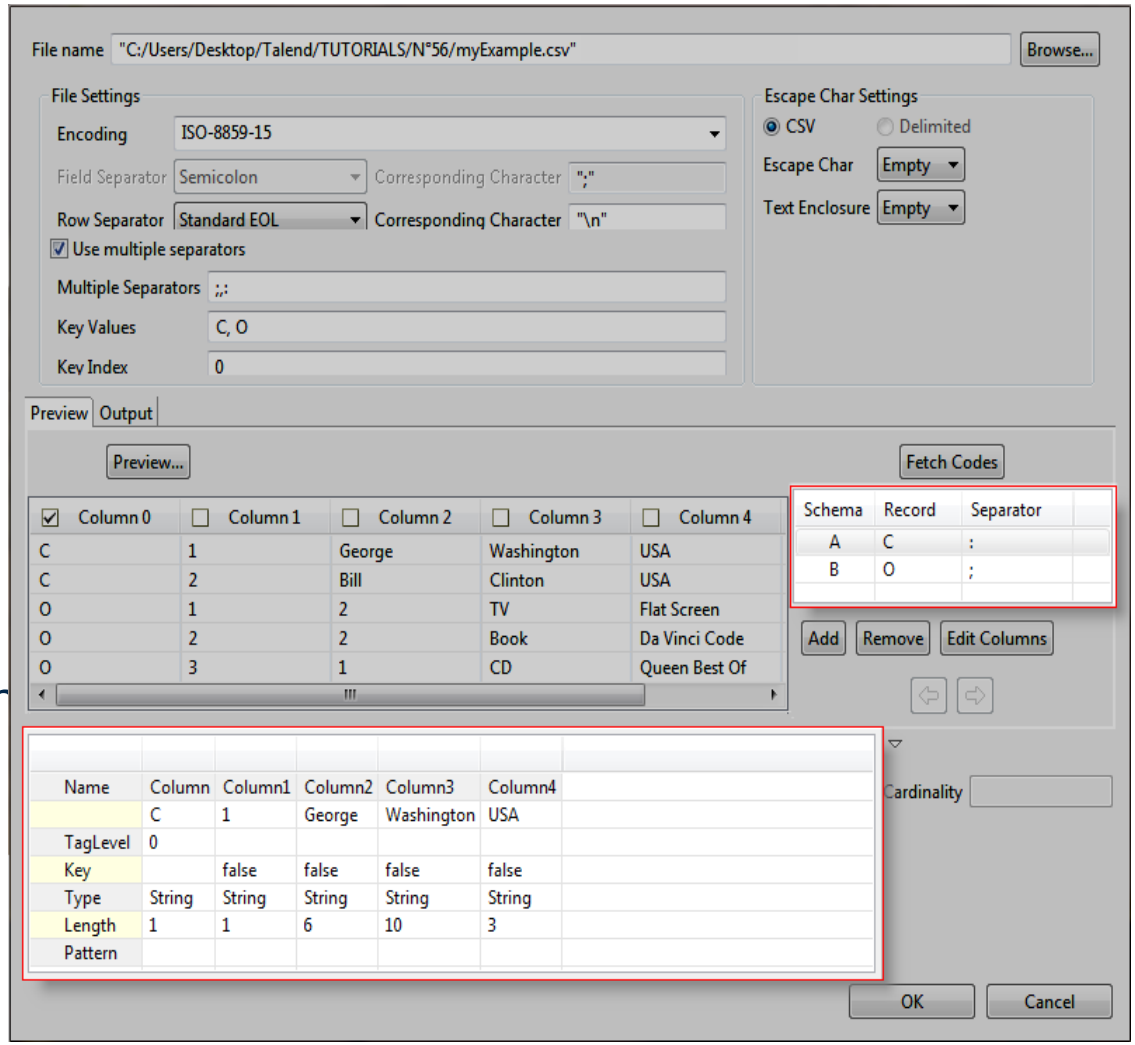
- Name: the column name.

- TagLevel: indicates the schema level if there is no parent-child relationship between the schemas.

- Type: data type of the corresponding column.

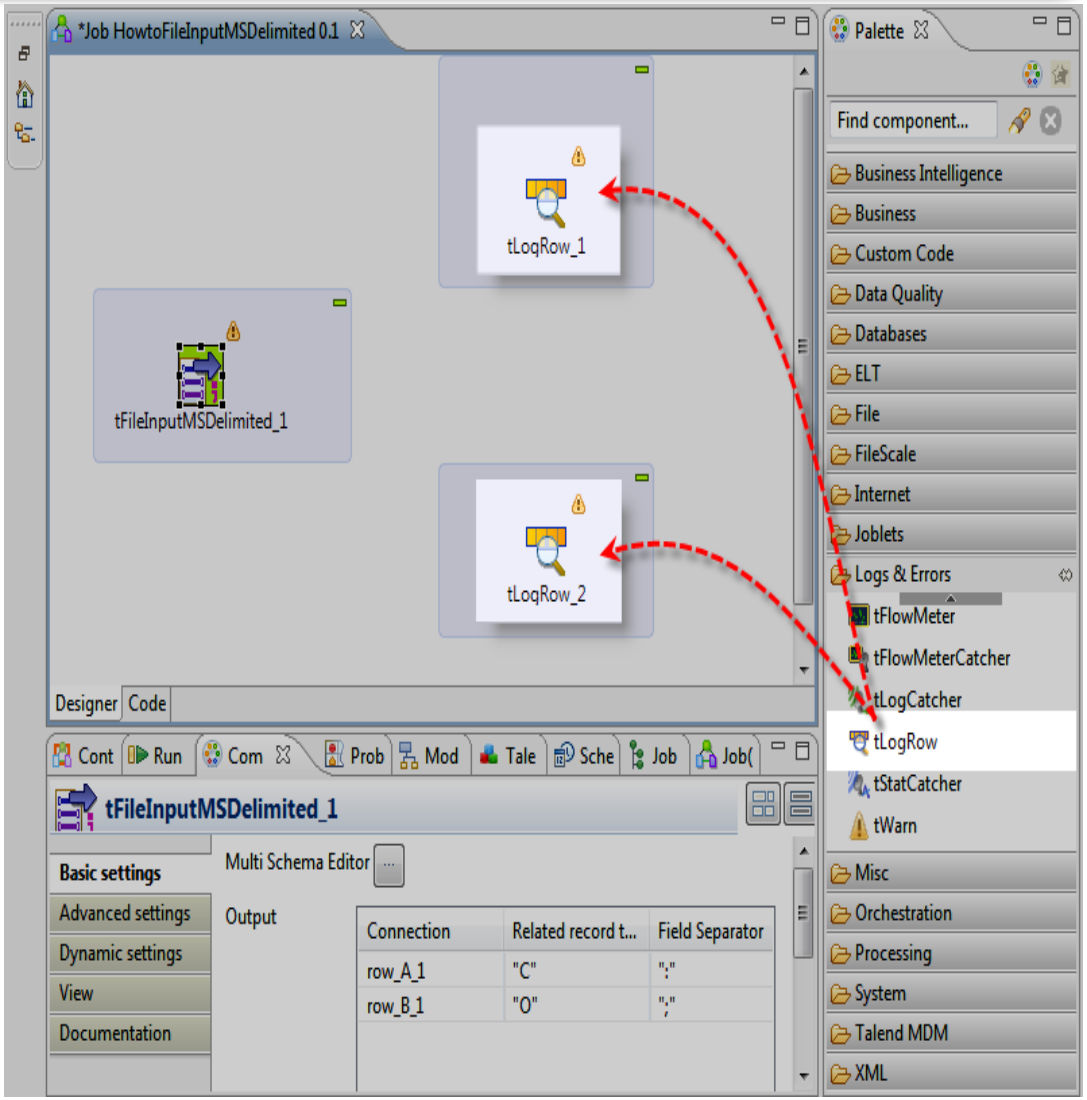
- Length: length of the column

- Pattern: data pattern of the column, if required.



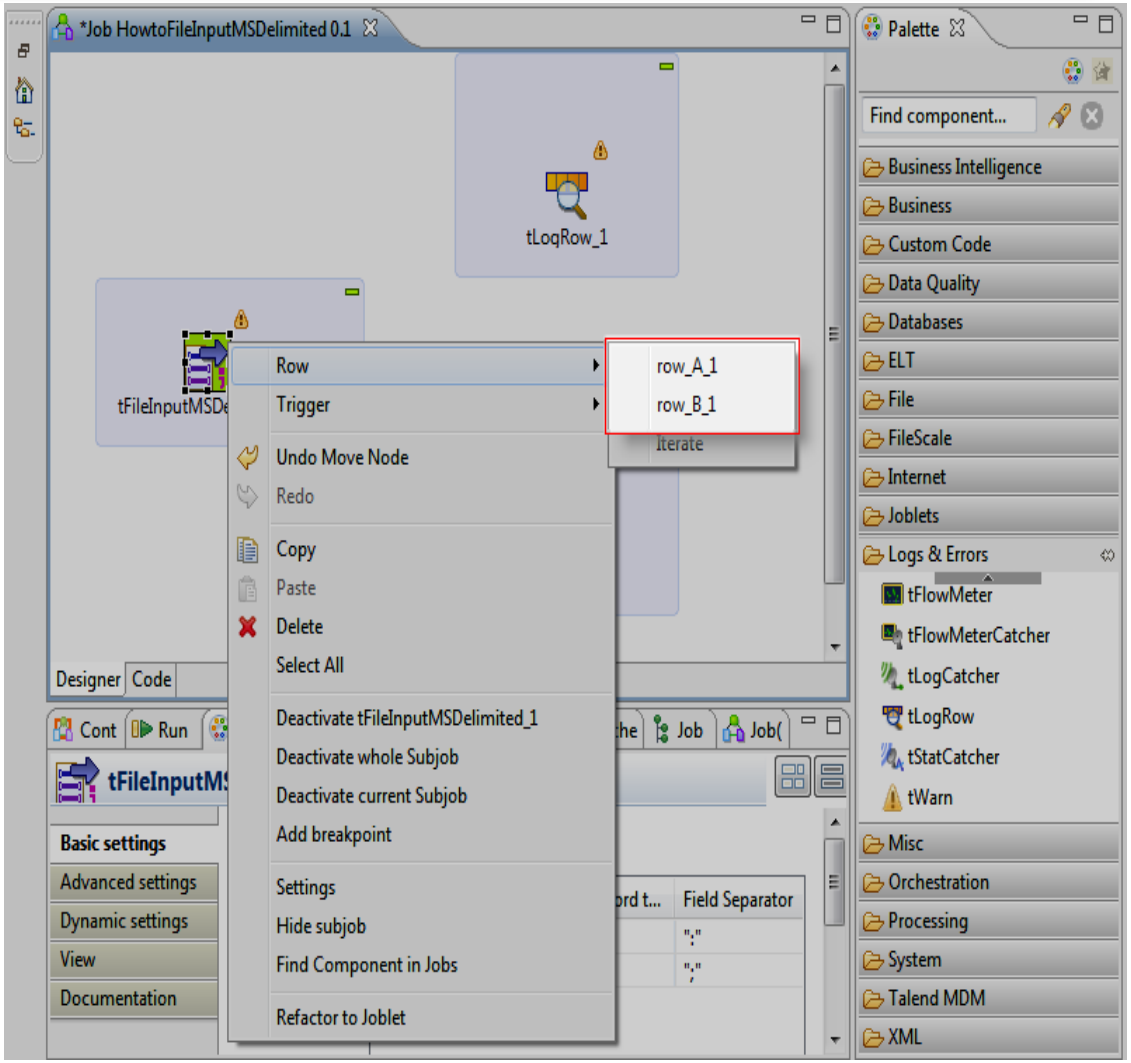
How to use a Multi Schema component (Cont…)

- To add the output components, click Logs & Errors in the Palette.
- Click tLogRow and drop it on the Job Designer.
- Repeat this operation as many times as required until you have a tLogRow component for every schema in your file (in this example two tLogRow components are required).
- You can also use tFileOutputDelimited components for the output data flow instead of tLogRow components.



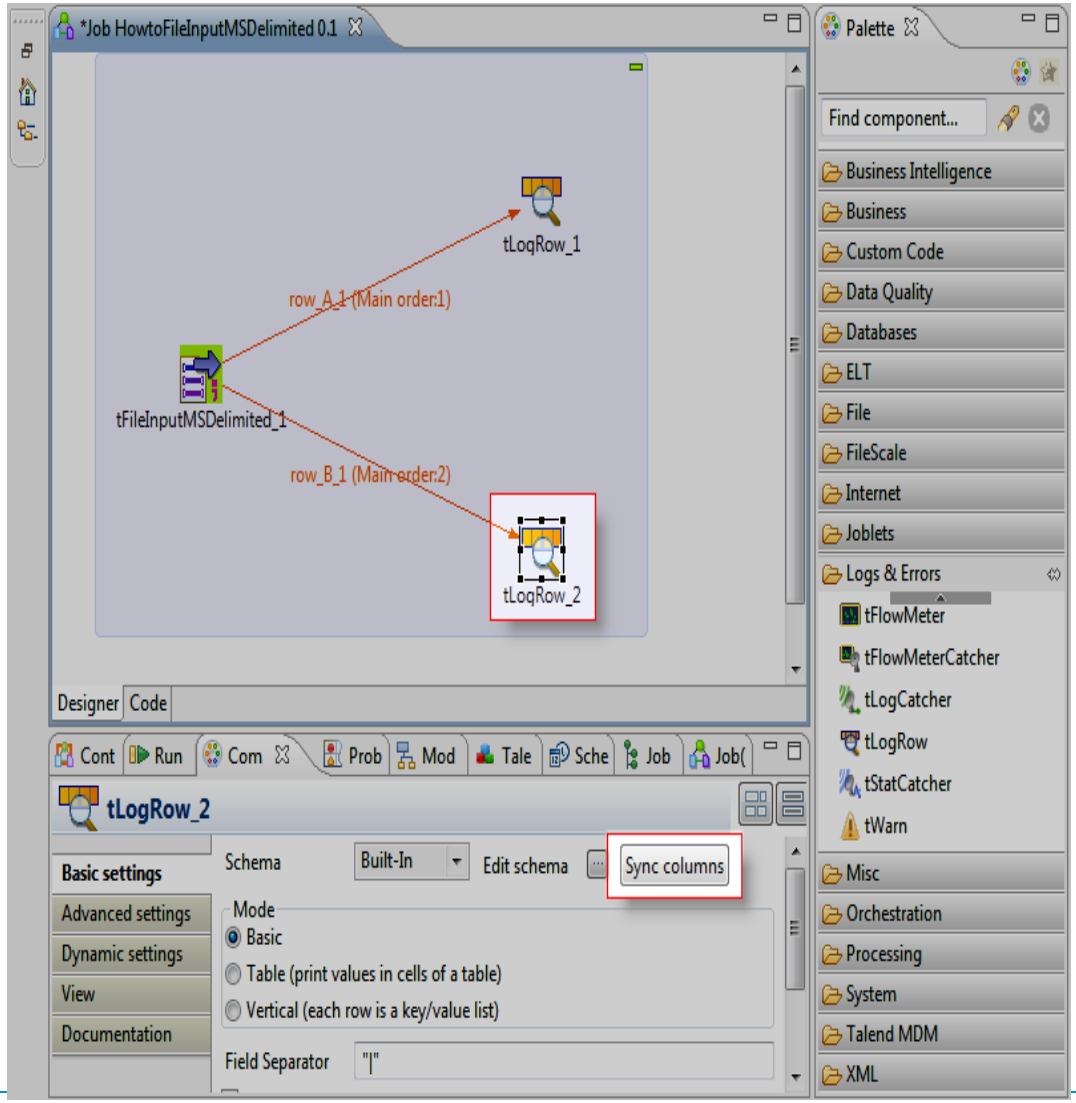
How to use a Multi Schema component (Cont…)

- In the Job Designer:
- To link the components together, right-click tFileInputMSDelimited_1, select one of the rows and click the target tLogRow component.



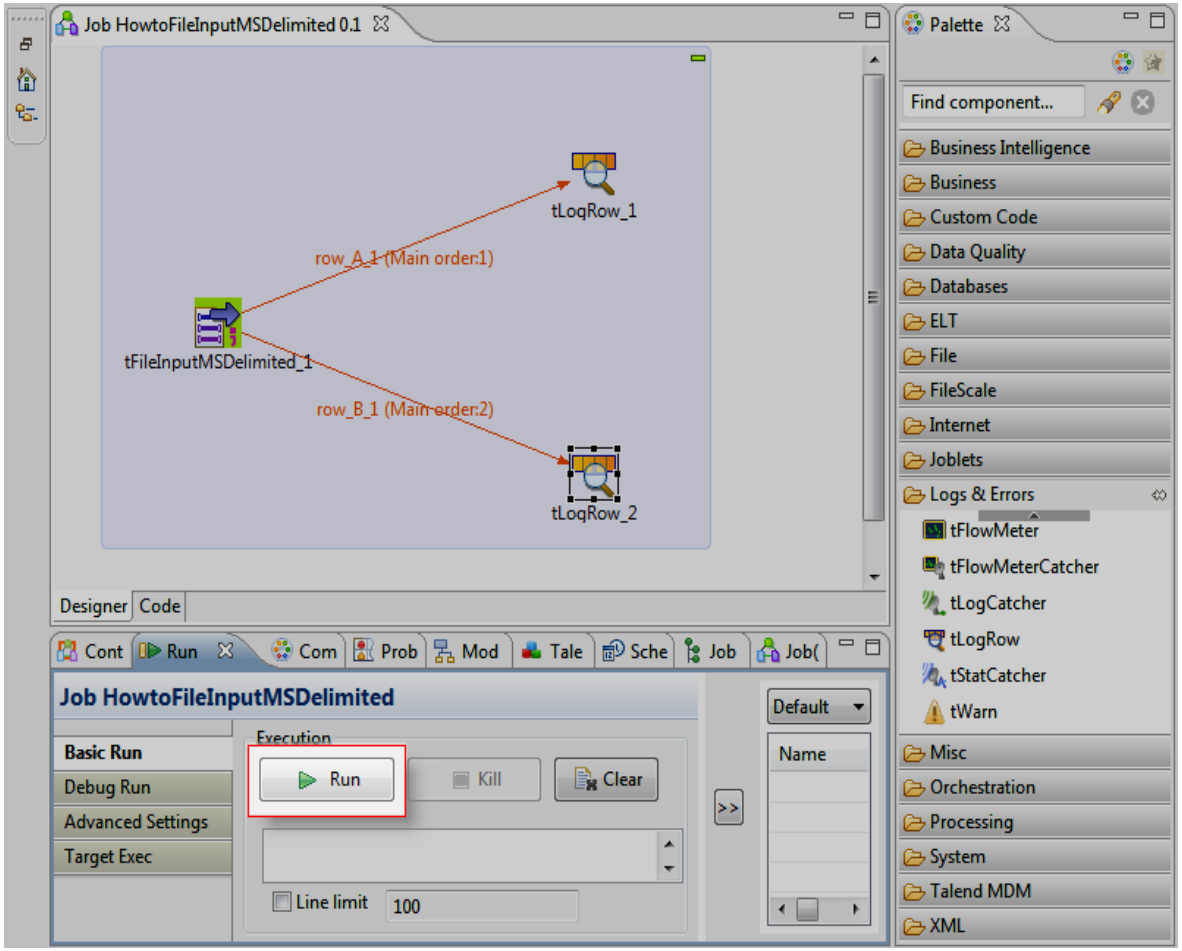
How to use a Multi Schema component (Cont…)

- Click each tLogRow component and synchronize the input and output schemas by clicking the Sync columns button.



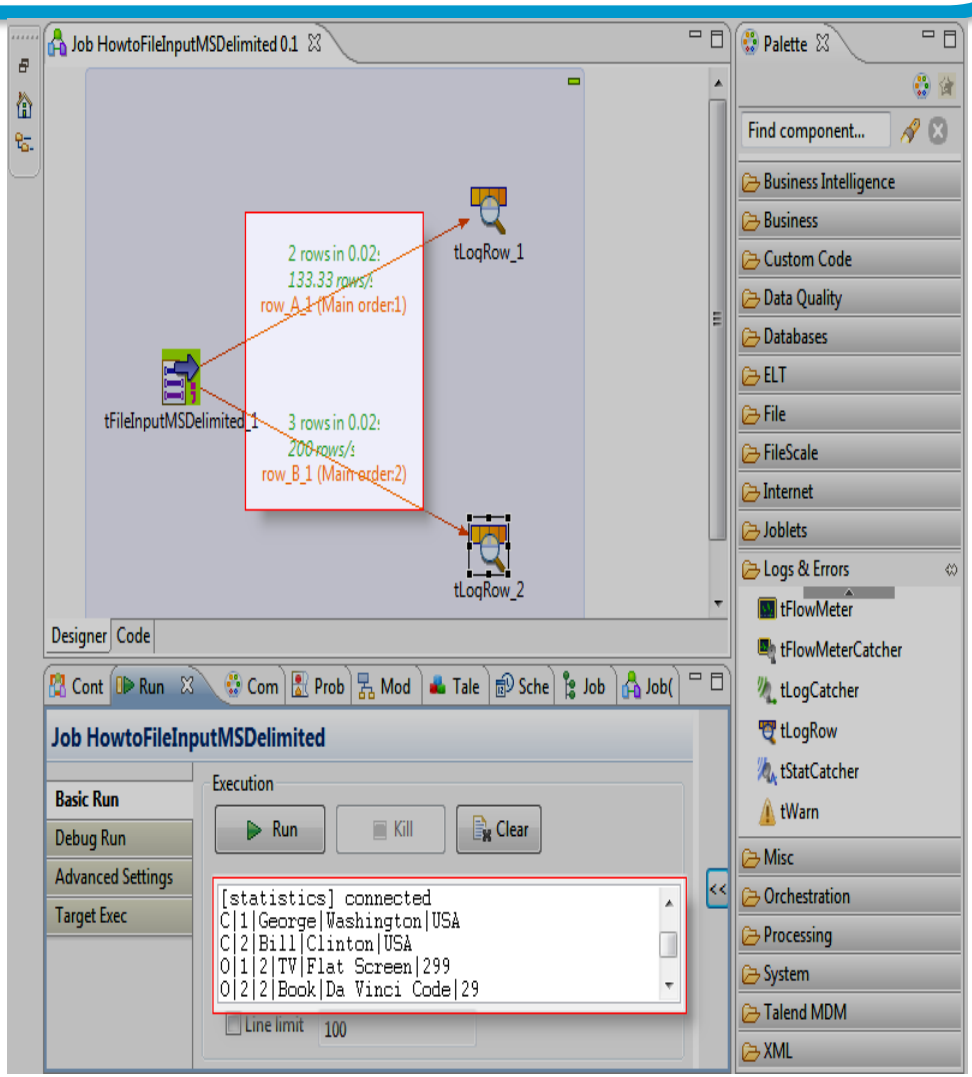
How to use a Multi Schema component (Cont…)

- In the Job Designer:
- Press Ctrl+S to save the Job.
- Press F6 to run it (or click Run in the Run view).
- The Run view appears at the bottom of Talend Open Studio and the console follows the execution of the Job.



How to use a Multi Schema component (Cont…)

- The file data is correctly split into two flows, one for each schema.
- You now know how to use this component to read multi-structured delimited files and how to separate the fields in these files using a specified separator.



Talend Big Data DI

and

Hadoop Components

Module Outline

- Introduction to Big Data and Hadoop
- Hadoop Components and how they are using Talend Studio
- Over of Hadoop and Talend Studio
- Writing and reading data into Hadoop using tHDFSInput and tHDFSOutput
- Reading and writing data from Hbase using tHBaseInput and tHBaseOutput
- Loading Data from Hadoop tHiveLoad
- Loading Data from Hadoop using tPigLoad
- Storing Pig Results into hadoop using tPigStorageResult
- Importing and exporting data from RDBMS into Hadoop using tSqoopImport

Talend Big Data DI

BIG DATA- What is it?

- Big data refers to the extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions. Eg: NASA Climate Simulation (32 Petabytes)
- Specifically, Big Data relates to data creation, storage, retrieval and analysis that is remarkable in terms of **volume**, **velocity** and **variety**.
- Big data refers to the extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.

Talend Big Data DI (Contd..)

- It includes:
  - Very large data set volumes
  - Very Long/ unacceptable processing times
  - Very large data velocity (inputs / outputs)
  - Very large varieties of data
  - High level of complexity

Obstacles: Big Data Integration Challenges

- **Technology**. A successful big data initiative requires acquiring, integrating and managing several big data technologies. It can become an expensive custom-coding nightmare to integrate large sets of diverse structured and unstructured datasets that also becomes difficult to maintain and manage.
  - ✓ Conventional data management tools fail when trying to integrate, search and analyze big datasets, which (for now) range from terabytes to multiple petabytes of information.

- **People**. As with any new technology, staff needs to be trained in big data technologies to learn proper skills and best practices.
  - ✓ A recent Talend survey, “How Big is Big Data Adoption”, found that the two biggest big data implementation challenges are finding in-house expertise, and allocation of sufficient budget, time and resources.

- **Quality Processes**: The survey also found that many big data projects do not have explicit project management structure, data governance, and lack the necessary big data quality procedures when processing unstructured sets of data.

Solution: Talend Big Data

Big Data Integration

Landing big data (large volumes of log files, data from operational systems, social media, sensors, or other sources) into a big data platform such as Apache Hadoop, Google Cloud Platform, Netezza, Teradata or Vertica is a cinch with the breadth of big data components provided by Talend. A full set of Talend data integration components (application, database, service and even a master data hub) is available, so that data movement can be orchestrated from any source or into almost any target. NoSQL connectivity to MongoDB and Cassandra is simplified through pre-built graphical connector components.

Big Data Quality

Talend provides data quality functions that take advantage of the massively parallel environment of Hadoop, allowing you to understand the completeness, accuracy and integrity of data as well as to remove duplicates. Hadoop data profiling allows you do collect information and statistics about big data to assess data quality, repurposing and metadata. Additional functions include standardization, parsing, enrichment, matching, survivorship and monitoring of ongoing data quality.

Solution: Talend Big Data (Contd..)

- **Big Data Manipulation**

  Talend supports Apache Pig and HBase so you can perform basic transformations and analysis on massive amounts of data in little time. With these scripting languages you can compare, filter, evaluate and group data within an HDFS cluster. Google Big Query supports allows you to interactively analyze very large datasets. Talend speeds development and collaboration by providing a set of components that allow these scripts to be defined in a graphical environment and as part of a data flow.

- **Big Data Project Governance and Administration**

  Governance of a big data project is very similar to any integration project; however, big data projects sometimes lack necessary management functions. Talend presents a simple, intuitive environment to implement and deploy a big data program with the ability to schedule, monitor and deploy any big data job. Also included is a common repository so developers can collaborate and share project metadata and artifacts.

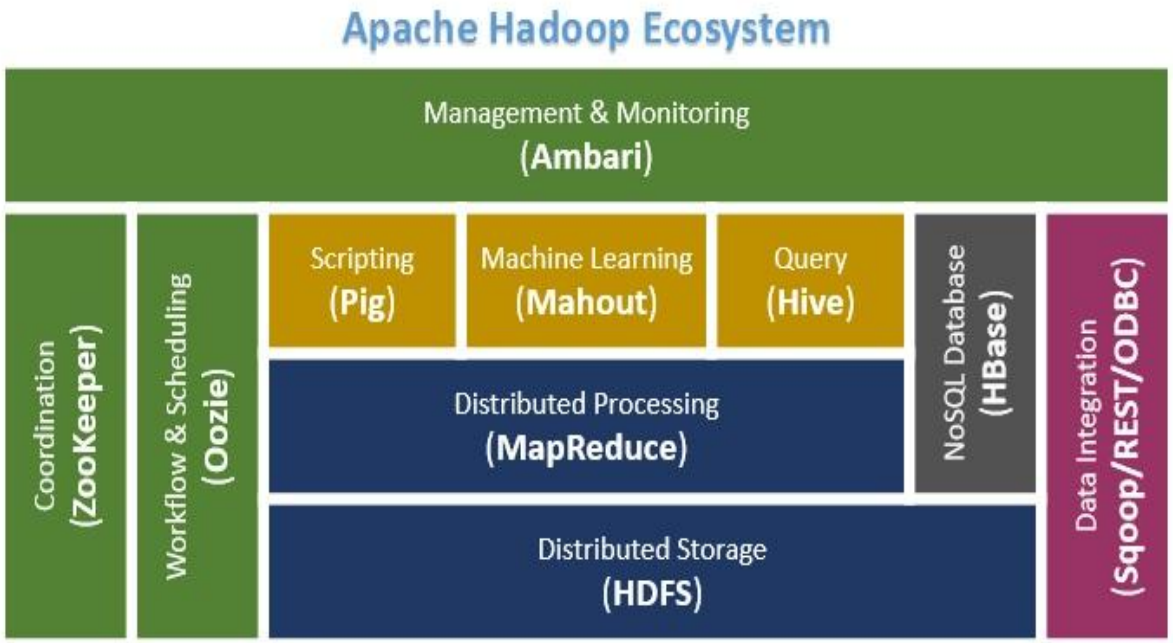
Talend Big Data DI (Contd..)

- Big data integration is a key operational challenge for today's enterprise IT departments. IT groups may find their skill sets, workload, and budgets over-stretched by the need to manage terabytes or petabytes of data in a way that delivers genuine value to business users.
- Talend, the leading provider of open source data management solutions, helps organizations large and small meet the big data challenge by making big data integration easy, fast, and affordable.

Hadoop Components

CORE HADOOP COMPONENTS:

- The Hadoop Ecosystem comprises of 4 core components –



1) Hadoop Common-

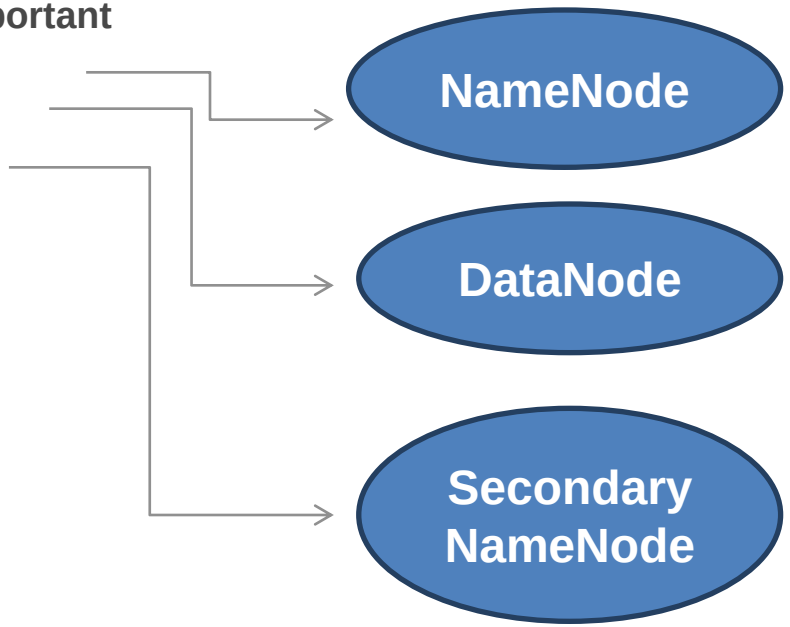
Apache Foundation has pre-defined set of utilities and libraries that can be used by other modules within the Hadoop ecosystem. For example, if HBase and Hive want to access HDFS they need to make of Java archives (JAR files) that are stored in Hadoop Common.

Hadoop Components (Contd..)

2) Hadoop Distributed File System (HDFS) -

- The default big data storage layer for Apache Hadoop is HDFS
- Users can dump huge datasets into HDFS and the data will sit there until the user wants to leverage it for analysis.
- HDFS component creates several replicas of the data block to be distributed across different clusters for reliable and quick data access. **HDFS comprises of 3 important components:**

HDFS operates on a Master-Slave architecture model where the NameNode acts as the master node for keeping a track of the storage cluster and the DataNode acts as a slave node summing up to the various systems within a Hadoop cluster.



Hadoop Components (Contd..)

3) MapReduce- Distributed Data Processing Framework of Apache Hadoop

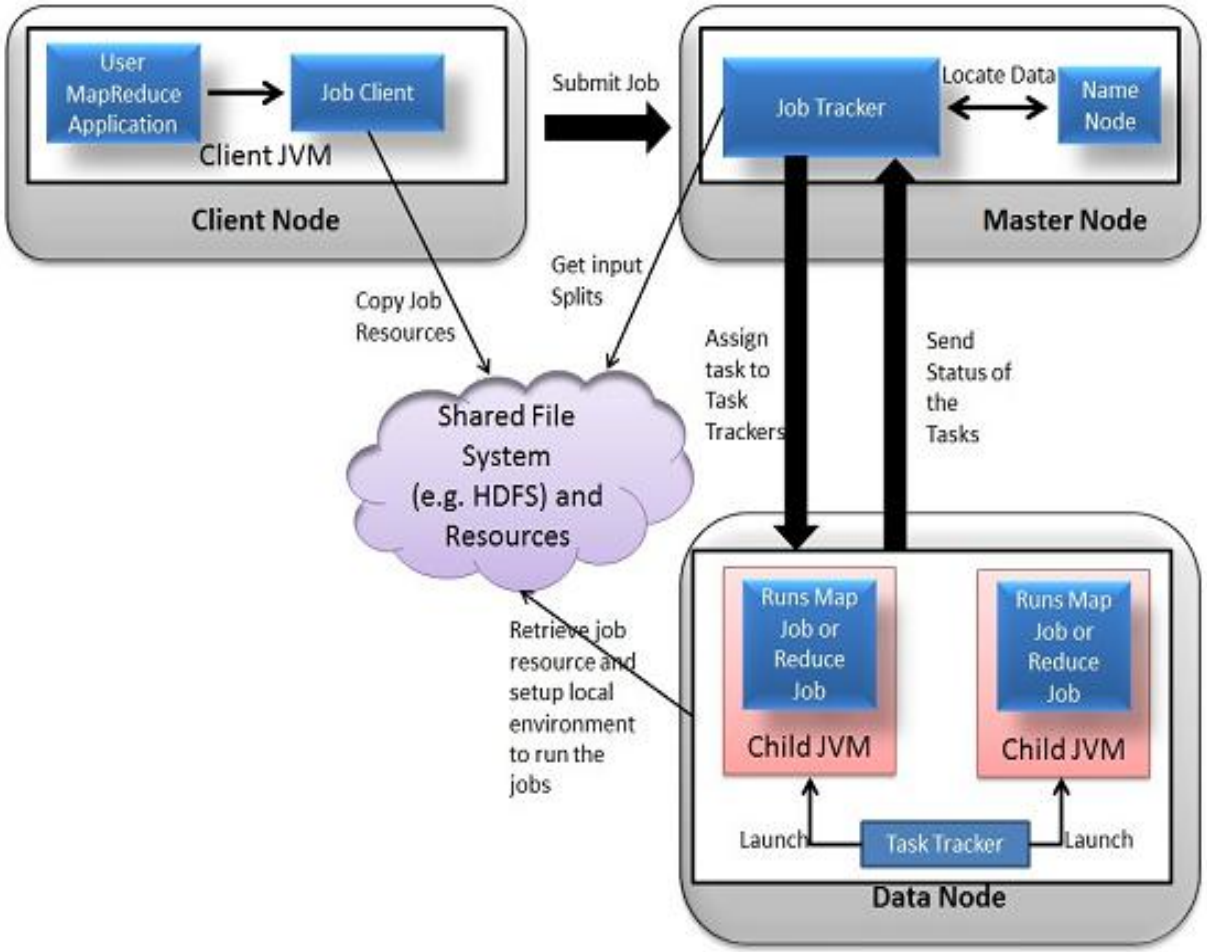
- MapReduce breaks down a big data processing job into smaller tasks.
- MapReduce is responsible for the analyzing large datasets in parallel before reducing it to find the results.
- In the Hadoop ecosystem, Hadoop MapReduce is a framework based on YARN architecture. YARN based Hadoop architecture, supports parallel processing of huge data sets and MapReduce provides the framework for easily writing applications on thousands of nodes, considering fault and failure management.

Hadoop Components (Contd..)

MapReduce framework forms the compute node while the HDFS file system forms the data node. Typically in the Hadoop ecosystem architecture both data node and compute node are considered to be the same. The delegation tasks of the MapReduce component are tackled by two daemons:

Hadoop Components (Contd..)

Job Tracker and **Task Tracker** as shown in the image below –



Hadoop Components (Contd..)

4) YARN:

YARN forms an integral part of Hadoop 2.0.YARN is great enabler for dynamic resource utilization on Hadoop framework as users can run various Hadoop applications without having to bother about increasing workloads.

Key Benefits of Hadoop 2.0 YARN Component-

- It offers improved cluster utilization
- Highly scalable
- Beyond Java
- Novel programming models and services
- Agility

Hadoop Components (Contd..)

Hadoop 2 - YARN Architecture

ResourceManager (RM)

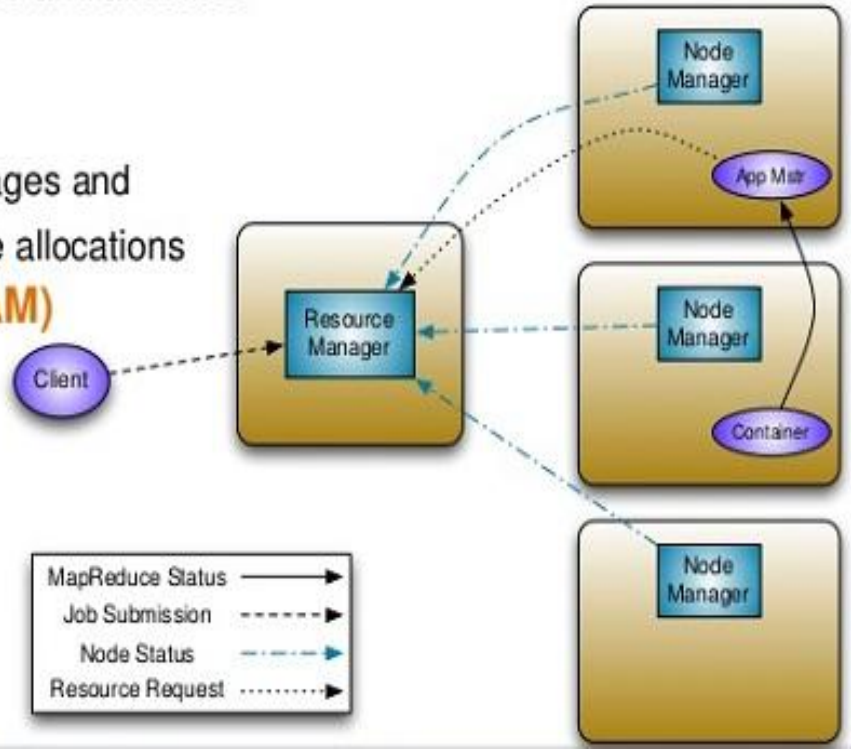
Central agent - Manages and allocates cluster resources

NodeManager (NM)

Per-Node agent - Manages and enforces node resource allocations

ApplicationMaster (AM)

Per-Application – Manages application lifecycle and task scheduling



Hadoop Components (Contd..)

➢ **DATA ACCESS COMPONENTS OF HADOOP ECOSYSTEM- Pig and Hive**

PIG

- structure is open to considerable parallelization
- high level data flow language Pig Latin

- data warehouse built on top of Hadoophigh level data flow language Pig Latin
- provides a simple language known as HiveQL similar to SQL for querying, data summarization and analysis.

HIVE

Hadoop Components (Contd..)

- **DATA INTEGRATION COMPONENTS OF HADOOP ECOSYSTEM- Sqoop and Flume**

Sqoop

- used for importing data from external sources into related Hadoop components like HDFS, HBase or Hive

- used for collecting data from its origin and sending it back to the resting location (HDFS).

Flume

Hadoop Components (Contd..)

➢ **DATA STORAGE COMPONENT OF HADOOP Ecosystem –HBase**

HBase

- Is a column-oriented database that uses HDFS for underlying storage of data.
- random reads and also batch computations using MapReduce.

Hadoop Components (Contd..)

- **MONITORING, MANAGEMENT AND ORCHESTRATION COMPONENTS OF HADOOP ECOSYSTEM- Oozie and Zookeeper**

Oozie

- is a workflow scheduler where the workflows are expressed as Directed Acyclic Graphs.
- makes use of a database to store all the running workflow instances, their states ad variables along with the workflow definitions to manage Hadoop jobs

Zookeeper

- provides simple, fast, reliable and ordered operational services for a Hadoop cluster.

Hadoop Components (Contd..)

AMBARI

- provides easy to use web user interface for Hadoop management.
- provides step-by-step wizard for installing Hadoop ecosystem services.
- facilitates the metrics collection, alert framework, which can monitor the health status of the Hadoop cluster.

MAHAUT

- provides implementation of various machine learning algorithms.
- helps with considering user behavior in providing suggestions, categorizing and classifying the items, supporting in implementation group mining or item set mining.

Hadoop Components (Contd..)

Big Data / Hive

tHiveClose — Close a selected Hive connection.

tHiveConnection — Creates a connection to a Hive database

tHiveCreateTable — create hive table

tHiveInput — Reads a Hive table and extracts fields based on an HiveQL query

tHiveLoad — load data to hive table or directory

tHiveRow — Executes an SQL query at each of Talend flow's iterations

Databases / HBase

tHBaseClose — Close a selected HBase connection.

tHBaseConfiguration — Define an HBase configuration.

tHBaseConnection — Reads an HBase table and extracts fields based on an SQL query

tHBaseInput — Read rows from an HBase table.

tHBaseOutput — Insert or update rows in an HBase table.

Hadoop Components (Contd..)

Databases / Greenplum

 tGreenplumBulkExec	Loads data efficiently from a file
 tGreenplumClose	Close a selected Greenplum connection.
 tGreenplumCommit	Commits all transactions not already commited in a selected Greenplum connection
 tGreenplumConnection	Creates a connection to a Greenplum DB
 tGreenplumGPLoad	Bulk load data from a file or named-pipe using greenplum's gpload utility
 tGreenplumInput	Reads a Greenplum table and extracts fields based on an SQL query
 tGreenplumOutput	Inserts or updates lines in a Greenplum table
 tGreenplumOutputBulk	Writes a temporary file to feed the Greenplum DB
 tGreenplumOutputBulkExec	Efficiently loads data from a flow
 tGreenplumRollback	Rollbacks all transactions not already commited in a selected Greenplum connection
 tGreenplumRow	Executes an SQL query at each of Talend flow's iterations
 tGreenplumSCD	Slowly Changing Dimension algorithm for Greenplum

Hadoop and Talend Studio

- When IT specialists talk about 'big data', they are usually referring to data sets that are so large and complex that they can no longer be processed with conventional data management tools.
  - These huge volumes of data are produced for a variety of reasons. Streams of data can be generated automatically (reports, logs, camera footage, and so on).
  - Or they could be the result of detailed analyses of customer behavior (consumption data), scientific investigations (the Large Hadron Collider is an apt example) or the consolidation of various data sources.

Hadoop and Talend Studio

- The Hadoop open source platform has emerged as the preferred framework for analyzing big data.
- This distributed file system splits the information into several data blocks and distributes these blocks across multiple systems in the network (the Hadoop cluster).
- By distributing computing power, Hadoop also ensures a high degree of availability and redundancy

Hadoop and Talend Studio

- However, to reap the benefits of Hadoop, data analysts need a way to load data into Hadoop and subsequently extract it from this open source system. This is where **Talend** studio comes in.

- Built on top of **Talend**'s data integration solutions, **Talend** studio enable users to handle big data easily by leveraging Hadoop and its databases or technologies such as HBase, HCatalog, HDFS, Hive, Oozie and Pig.

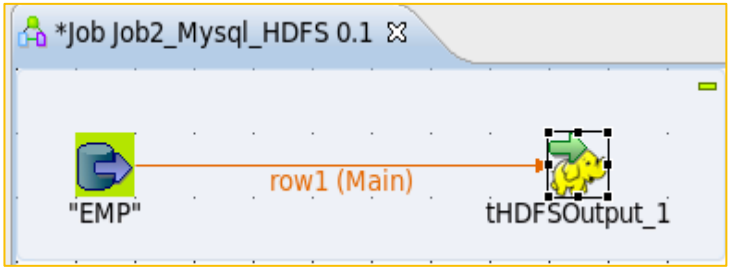
Hadoop and Talend Studio

- **Talend** studio is an easy-to-use graphical development environment that allows for interaction with big data sources and targets without the need to learn and write complicated code.

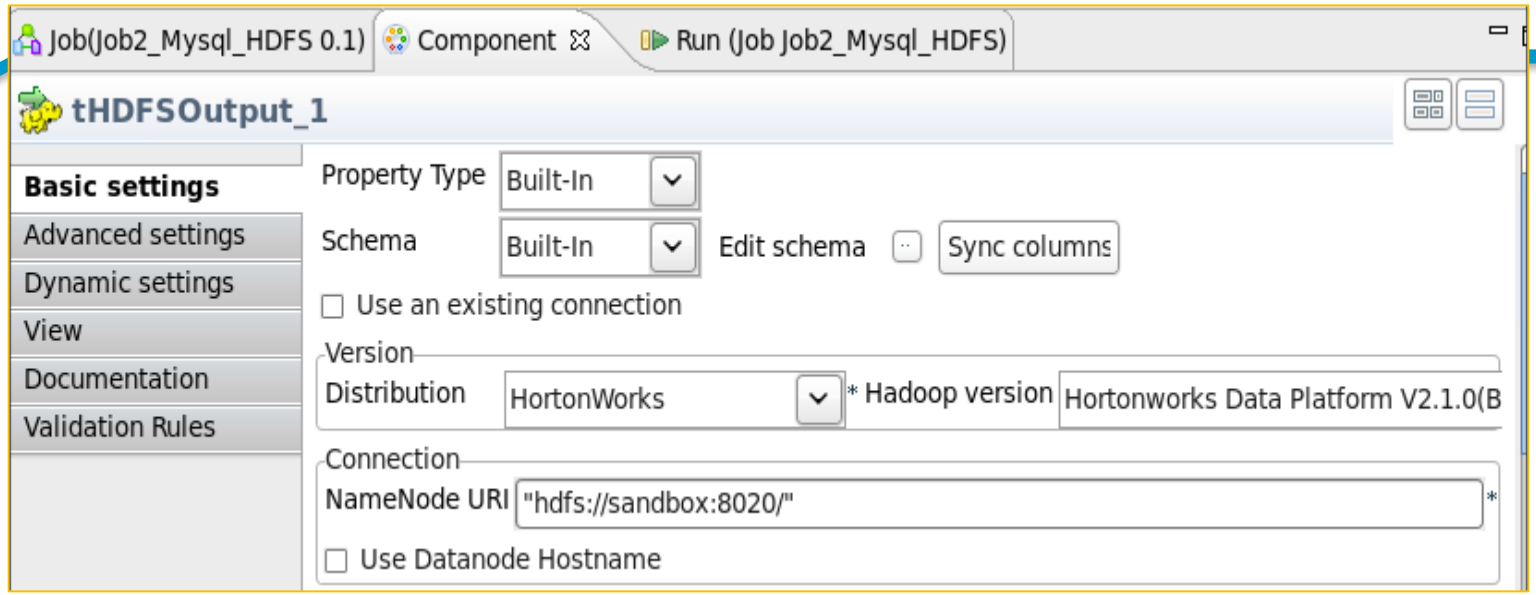
- Once a big data connection is configured, the underlying code is automatically generated and can be deployed as a service, executable or stand-alone Job that runs natively on your big data cluster - HDFS, Pig, HCatalog, HBase, Sqoop or Hive.

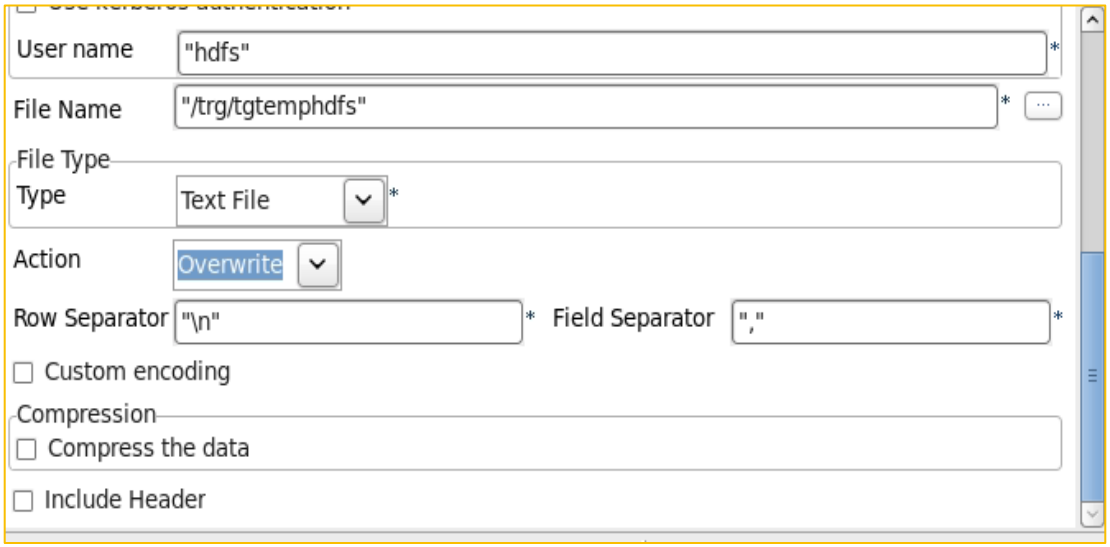
tHDFSOutput

- **Function:**
  - **tHDFSOutput** writes data flows it receives into a given Hadoop distributed file system (HDFS).
- **Purpose :**
  - **tHDFSOutput** transfers data flows into a given HDFS file system



tHDFSOutput - Properties





tHDFSInput

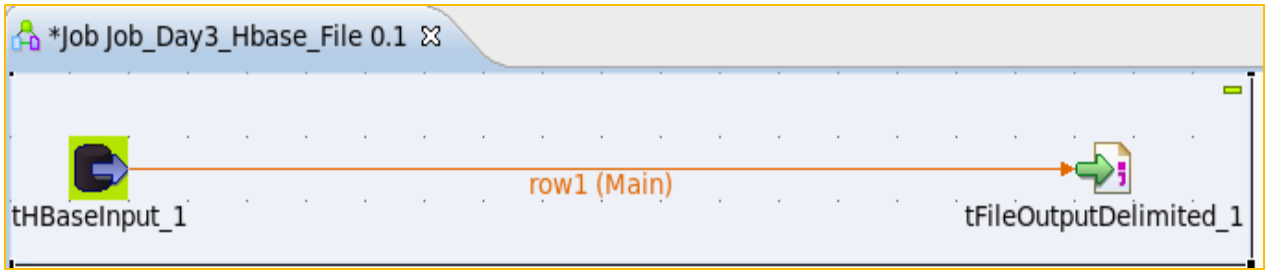
- **Function :**
  - **tHDFSInput** reads a file located on a given Hadoop distributed file system (HDFS) and puts the data of interest from this file into a Talend schema.
  - Then it passes the data to the component that follows.
- **Purpose**
  - **tHDFSInput** extracts the data in a HDFS file for other components to process it.

tHDFSProperties

- **Basic settings** *Property type*
  - Either **Built-in** or **Repository**
    - Built-in: No property data stored centrally.
    - Repository: Select the repository file in which the properties are stored.
  - *Hadoop version:* Select the version of the Hadoop distribution
  - *NameNode URI :*Type in the URI of the Hadoop NameNode
  - *User name:*In the User name field, enter the login user name for your distribution
  - *Schema* and *Edit schema* A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. The schema is either **Built-In** or stored remotely in the **Repository**

tHBaseInput

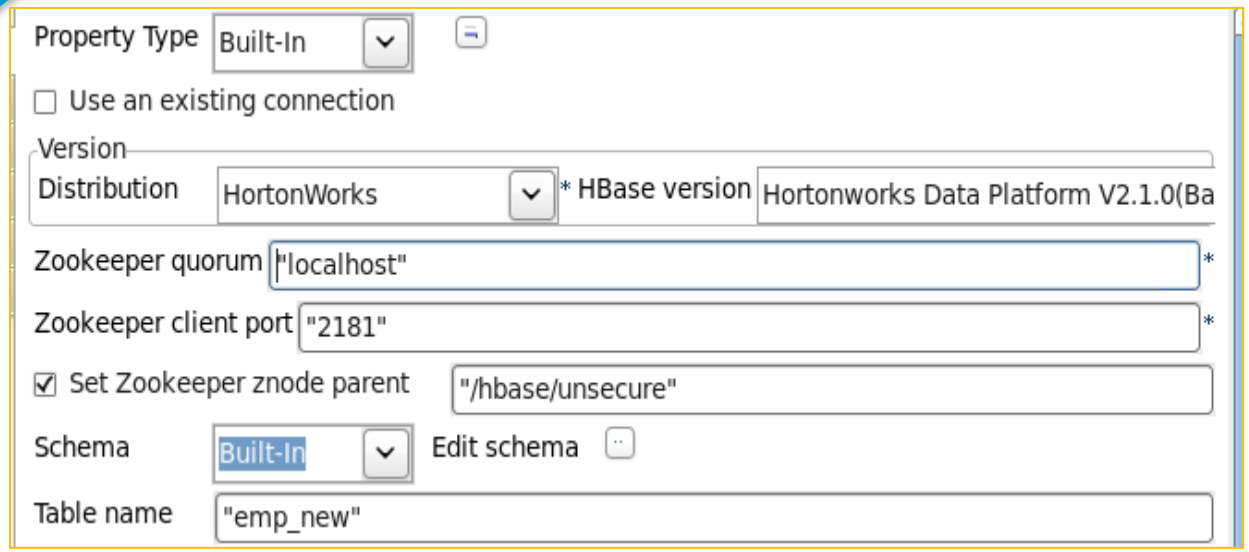
- **Function:**
  - **tHBaseInput** extracts columns corresponding to schema definition. Then it passes these columns to the next component via a Main row link.
- **Purpose:**
  - **tHBaseInput** reads data from a given HBase database and extracts columns of selection. Hbase is a distributed, column-oriented database that hosts very large, sparsely populated tables on clusters

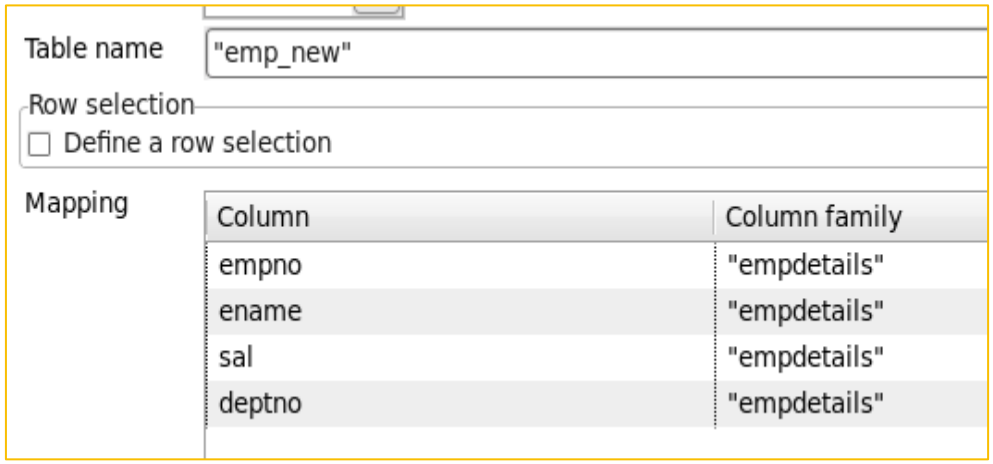


tHBaseInput properties

- **Basic settings** *Property type*
  - Either **Built-in** or **Repository**
    - Built-in: No property data stored centrally.
    - Repository: Select the repository file in which the properties are stored
  - *HBase version* Select the version of the Hadoop distribution

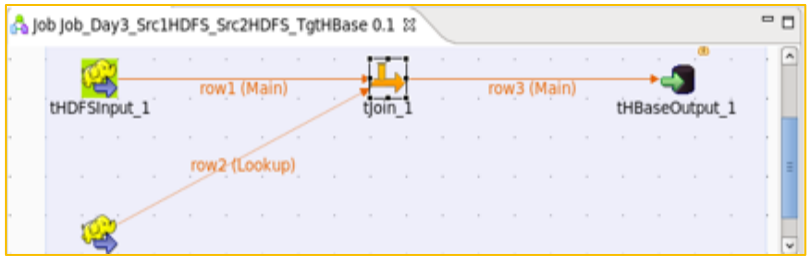
tHBaseInput properties





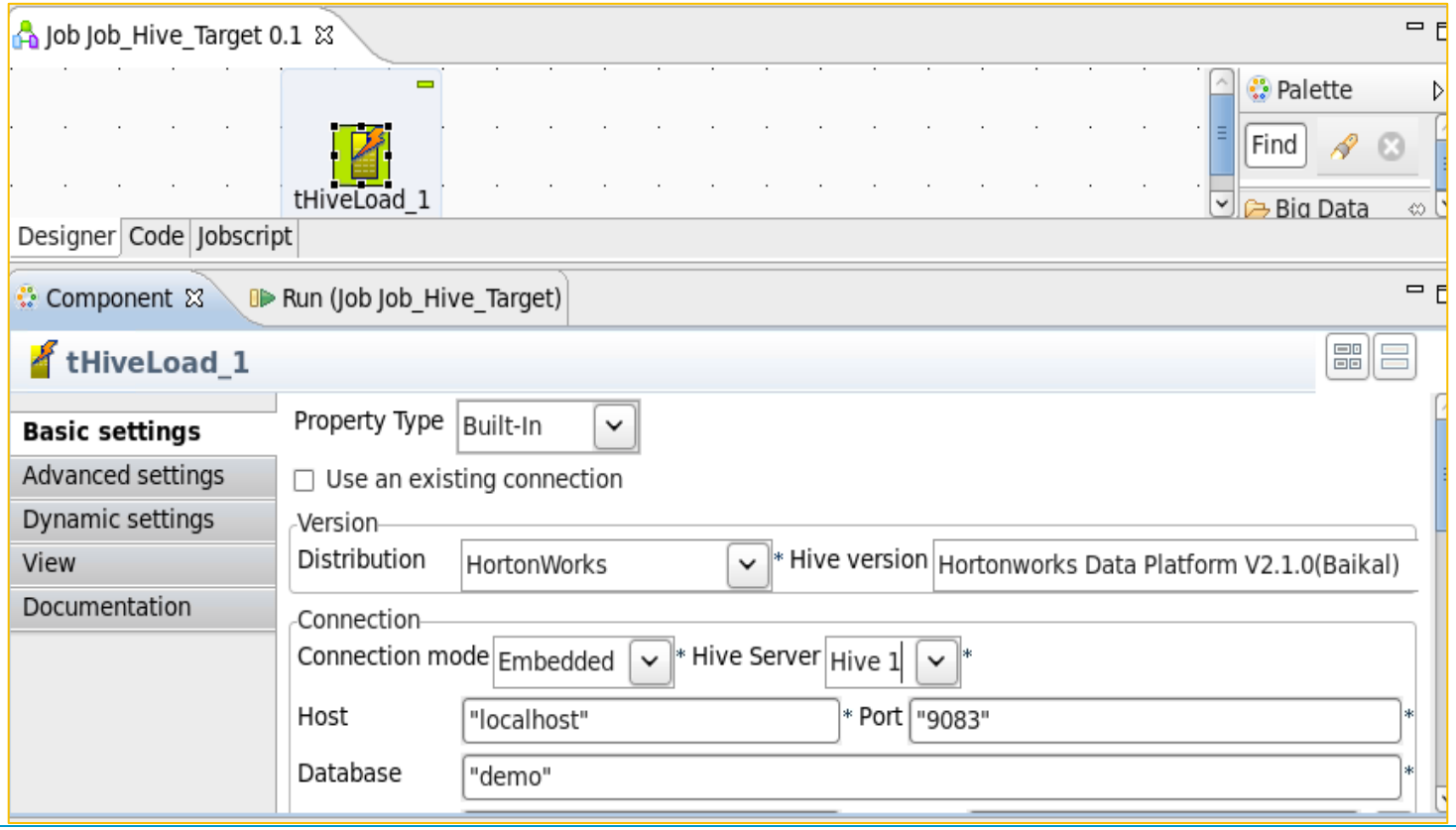
tHBaseOutput

- **Function :**
  - **tHBaseOutput** receives data from its preceding component, creates a table in a given Hbase database and writes the received data into this HBase table.
- **Purpose:**
  - **tHBaseOutput** writes columns of data into a given HBase database.

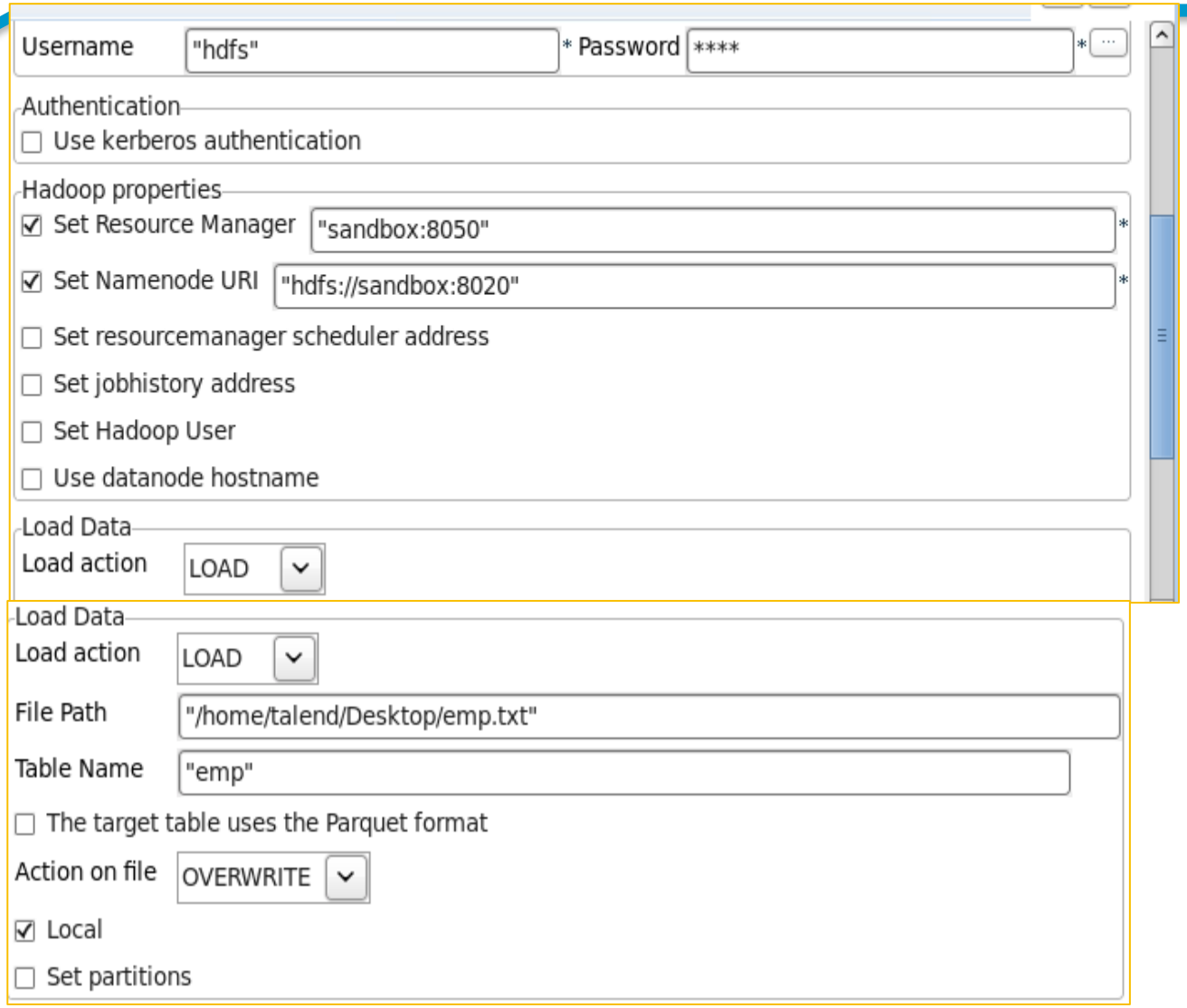


tHiveLoad

- **Function:**This component connects to a given Hive database and copies or moves data into an existing Hive table or a directory you specify.
- **Purpose:** This component is used to write data of different formats into a given Hive table or to export data from a Hive table to a directory.

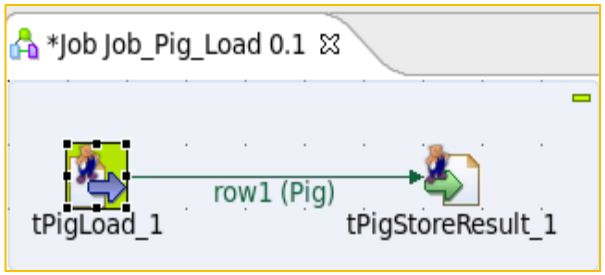


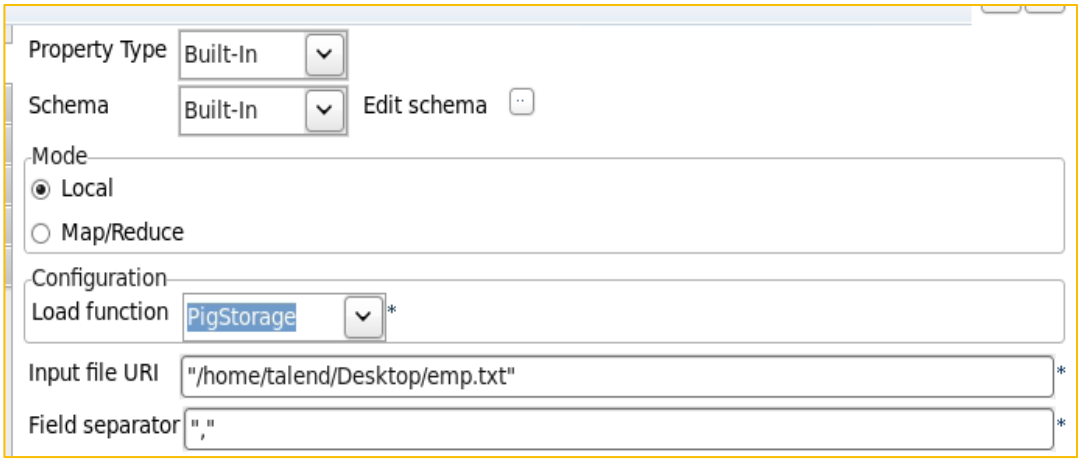
tHiveLoad - Property



tPigLoad

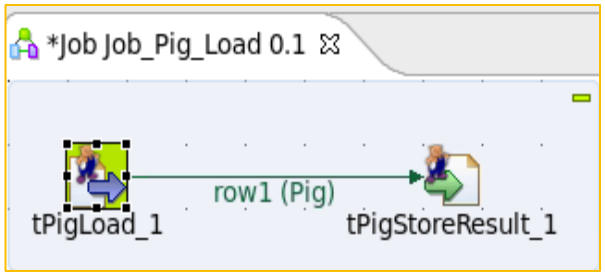
- Function: This component allows you to set up a connection to the data source for a current transaction.
- Purpose: The tPigLoad component loads original input data to an output stream in just one single transaction, once the data has been validated

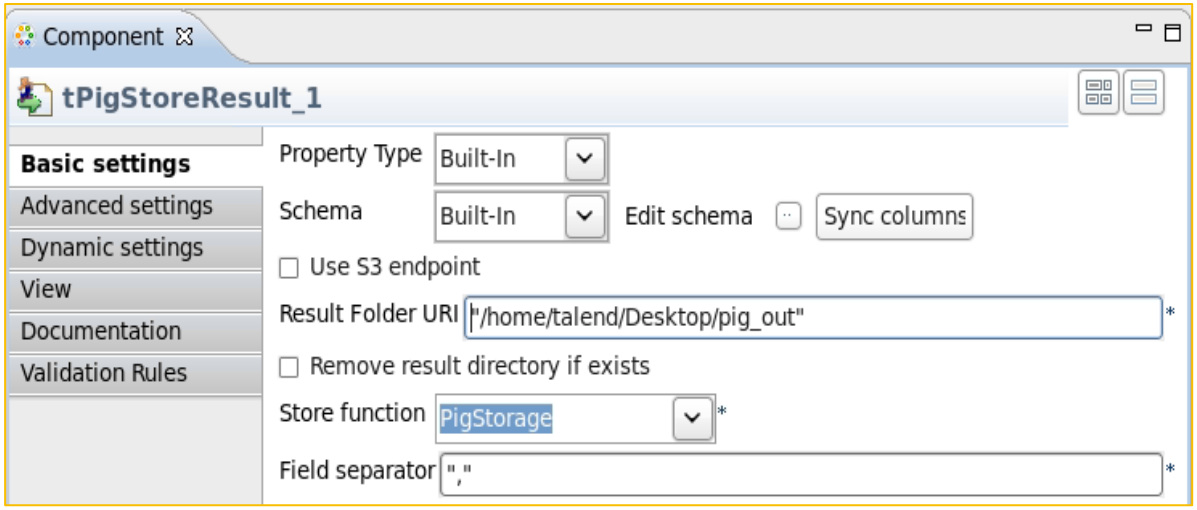




tPigStoreResult

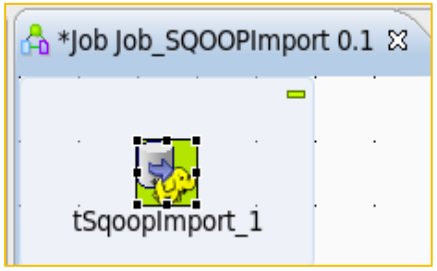
- Function: This component allows you to store the result of your Pig Job into a defined data storage space.
- Purpose: The tPigLoad component loads original input data to an output stream in just one single transaction, once the data has been validated



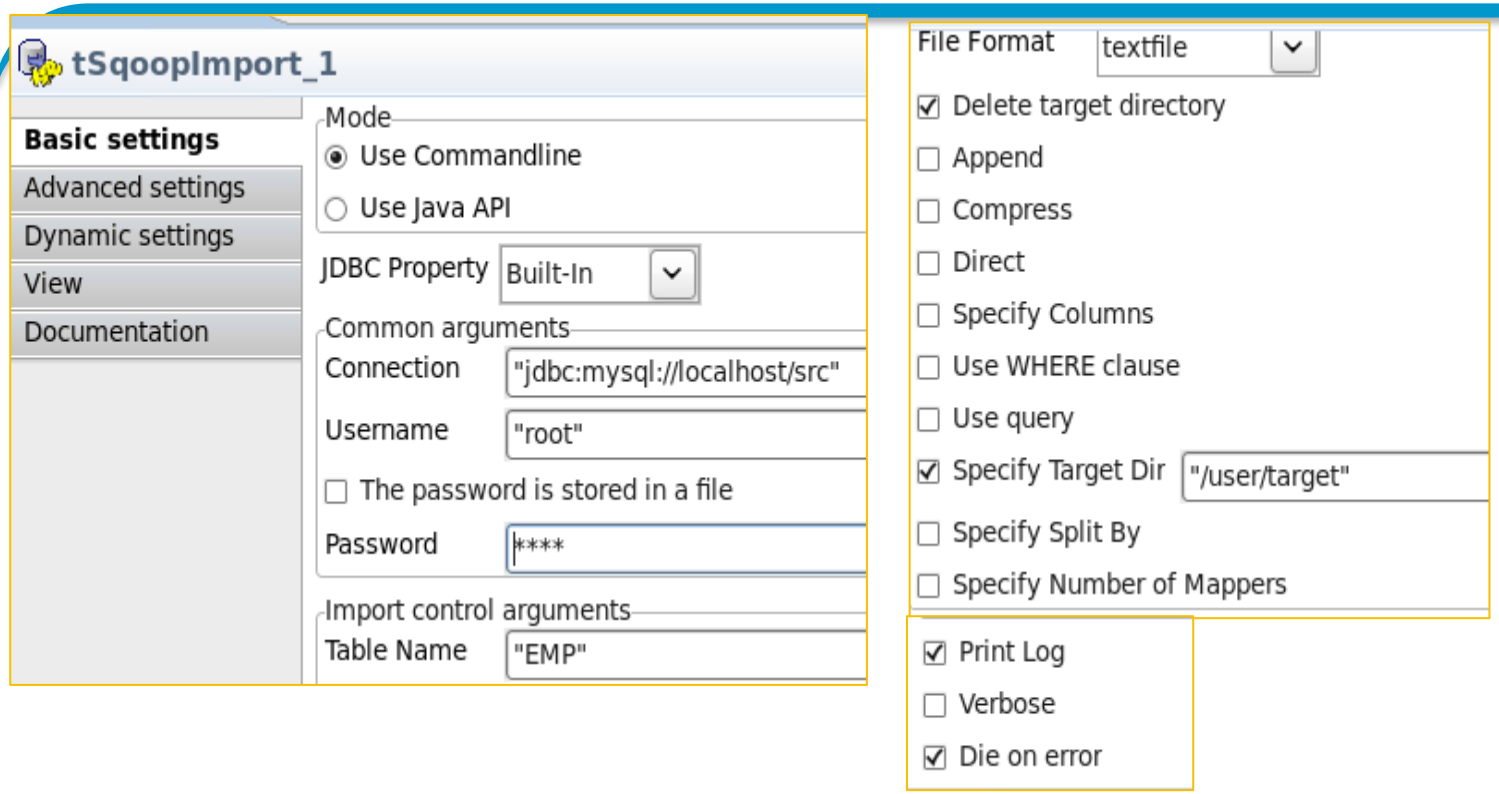


tSqoopImport

- Function:**tSqoopImport** calls Sqoop to transfer data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS).
- Purpose: **tSqoopImport** is used to define the arguments required by Sqoop for writing the data of your interest into HDFS.



tSqoopImport



Contact information

Insert contact picture

Name
Last Name
Title/Role
name.lastname@capgemini.com

Capgemini Office (Optional)
Address Line 1
Address Line 2
Address Line 3

Insert contact picture

Name
Last Name
Title/Role
name.lastname@capgemini.com

Capgemini Office (Optional)
Address Line 1
Address Line 2
Address Line 3

Insert contact picture

Name
Last Name
Title/Role
name.lastname@capgemini.com

Capgemini Office (Optional)
Address Line 1
Address Line 2
Address Line 3

Insert contact picture

Name
Last Name
Title/Role
name.lastname@capgemini.com

Capgemini Office (Optional)
Address Line 1
Address Line 2
Address Line 3

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.

About Capgemini

With 180,000 people in over 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.573 billion.

Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness.

A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

www.capgemini.com