



Talend Open Studio for Data Integration

Getting Started Guide

6.3.0

Adapted for v6.3.0. Supersedes previous releases.

Publication date: October 27, 2016

Copyright

This documentation is provided under the terms of the Creative Commons Public License (CCPL).

For more information about what you can and cannot do with this documentation in accordance with the CCPL, please read: <http://creativecommons.org/licenses/by-nc-sa/2.0/>

Notices

Talend is a trademark of Talend, Inc.

All brands, product names, company names, trademarks and service marks are the properties of their respective owners.

License Agreement

The software described in this documentation is licensed under the Apache License, Version 2.0 (the "License"); you may not use this software except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0.html>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product includes software developed at AOP Alliance (Java/J2EE AOP standards), ASM, Amazon, AntLR, Apache ActiveMQ, Apache Ant, Apache Axiom, Apache Axis, Apache Axis 2, Apache Batik, Apache CXF, Apache Chemistry, Apache Common Http Client, Apache Common Http Core, Apache Commons, Apache Commons Bcel, Apache Commons JXPath, Apache Commons Lang, Apache Derby Database Engine and Embedded JDBC Driver, Apache Geronimo, Apache Hadoop, Apache Hive, Apache HttpClient, Apache HttpComponents Client, Apache JAMES, Apache Log4j, Apache Lucene Core, Apache Neethi, Apache POI, Apache ServiceMix, Apache Tomcat, Apache Velocity, Apache WSS4J, Apache WebServices Common Utilities, Apache Xml-RPC, Apache Zookeeper, Box Java SDK (V2), CSV Tools, DataStax Java Driver for Apache Cassandra, Ehcache, Ezmorph, Ganymed SSH-2 for Java, Google APIs Client Library for Java, Google Gson, Groovy, Guava: Google Core Libraries for Java, H2 Embedded Database and JDBC Driver, Hector: A high level Java client for Apache Cassandra, Hibernate Validator, HighScale Lib, HsqlDB, Ini4j, JClouds, JLine, JSON, JSR 305: Annotations for Software Defect Detection in Java, JUnit, Jackson Java JSON-processor, Java API for RESTful Services, Java Agent for Memory Measurements, Jaxb, Jaxen, Jettison, Jetty, Joda-Time, Json Simple, LightCouch, MetaStuff, Mondrian, OpenSAML, Paracel JDBC Driver, PostgreSQL JDBC Driver, Resty: A simple HTTP REST client for Java, Rocoto, SL4J: Simple Logging Facade for Java, SQLite JDBC Driver, Simple API for CSS, SshJ, StAX API, StAXON - JSON via StAX, The Castor Project, The Legion of the Bouncy Castle, W3C, Woden, Woodstox: High-performance XML processor, Xalan-J, Xerces2, XmlBeans, XmlSchema Core, Xmlsec - Apache Santuario, Zip4J, atinject, dropbox-sdk-java: Java library for the Dropbox Core API, google-guice. Licensed under their respective license.

Table of Contents

Chapter 1. Introduction to Talend Open Studio for Data Integration	1
Chapter 2. Prerequisites to using Talend products	3
2.1. Memory requirements	4
2.2. Software requirements	4
2.3. Installing Java	4
2.4. Setting up the Java environment variable on Windows	5
2.5. Setting up the Java environment variable on Linux	5
2.6. Installing 7-Zip (Windows)	5
Chapter 3. Downloading and installing Talend Open Studio for Data Integration	7
3.1. Downloading Talend Open Studio for Data Integration	8
3.2. Installing Talend Open Studio for Data Integration	8
3.2.1. Extracting via 7-Zip (Windows recommended)	8
3.2.2. Extracting via Windows default unzipping tool	8
3.2.3. Extracting via the Linux GUI unzipper	9
Chapter 4. Configuring and setting up your Talend product	11
4.1. Launching the Studio for the first time	12
4.2. Logging in to the Studio	12
4.3. Installing additional packages	12
Chapter 5. Performing data integration tasks	15
5.1. Reading movies information from a CSV file	16
5.1.1. Creating your first Job	16
5.1.2. Dropping and linking components	17
5.1.3. Preparing the movies metadata	18
5.1.4. Configuring and executing your Job	22
5.2. Filtering the movies information	24
5.2.1. Preparing directors file metadata	24
5.2.2. Duplicating the existing Job	28
5.2.3. Adding a mapping component	29
5.2.4. Adding a lookup component	31
5.2.5. Configuring mappings and executing the Job	33
5.3. Gathering rejected movies information and saving processing results to a database	36
5.3.1. Adding database output components to your Job	36
5.3.2. Configuring mappings for rejected data	39
5.3.3. Configuring MySQL database outputs	40
5.4. What's next?	41



Chapter 1. Introduction to Talend Open Studio for Data Integration

Talend Open Studio for Data Integration provides unified development and management tools to integrate and process all of your data with an easy to use, visual designer.

Talend Open Studio for Data Integration offers solutions to the problems companies face due to growing system complexities by addressing both ETL for analytics and ETL for operational integration needs and offering industrialization features.



Chapter 2. Prerequisites to using Talend products

This chapter provides basic software and hardware information required and recommended to get started with your *Talend* product:

- [Memory requirements.](#)
- [Software requirements.](#)

It also guides you to install and configure required and recommended third-party tools:

- [Installing Java.](#)
- [Setting up the Java environment variable on Windows](#) or [Setting up the Java environment variable on Linux.](#)
- [Installing 7-Zip \(Windows\).](#)

To successfully install the software, you need administrative access to your computer. To get administrative access, contact your Administrator.

2.1. Memory requirements

To make the most out of your *Talend* product, please consider the following memory and disk space usage:

Memory usage	3GB minimum, 4 GB recommended
Disk space	3GB

2.2. Software requirements

To make the most out of your *Talend* product, please consider the following system and software requirements:

Required software

- Operating System for Talend Studio:

Support type	Operating System	Version	Processor
Recommended	Microsoft Windows Professional	7	64-bit
Recommended	Linux Ubuntu	14.04	64-bit
Supported	Apple OS X	El Capitan/10.11	64-bit
		Yosemite/10.10	64-bit
		Mavericks/10.9	64-bit

- Java 8 JRE Oracle. See [Installing Java](#).
- A properly installed and configured MySQL database, with a database named *gettingstarted*.

Optional software

- 7-Zip. See [Installing 7-Zip \(Windows\)](#).

2.3. Installing Java

To use your *Talend* product, you need Oracle Java Runtime Environment installed on your computer.

- From the [Java SE Downloads](#) page, under **Java Platform, Standard Edition**, click the **JRE Download**.
- From the **Java SE Runtime Environment 8 Downloads** page, click the radio button to **Accept License Agreement**.
- Select the appropriate download for your Operating System.
- Follow the Oracle installation steps to install Java.

When Java is installed on your computer, you need to set up the `JAVA_HOME` environment variable. For more information, see:

- [Setting up the Java environment variable on Windows](#).
- [Setting up the Java environment variable on Linux](#).

2.4. Setting up the Java environment variable on Windows

Prior to installing your *Talend* product, you have to set the `JAVA_HOME` and `Path` environment variables:

1. Go to the **Start Menu** of your computer, right-click on **Computer** and select **Properties**.
2. In the **[Control Panel Home]** window, click **Advanced system settings**.
3. In the **[System Properties]** window, click **Environment Variables...**
4. Under **System Variables**, click **New...** to create a variable. Name the variable `JAVA_HOME`, enter the path to the Java 8 JRE, and click **OK**.

Example of default JRE path: `C:\Program Files\Java\jre1.8.0_77`.

5. Under **System Variables**, select the **Path** variable and click **Edit...** to add the previously defined `JAVA_HOME` variable at the end of the `Path` environment variable, separated with semi colon.

Example: `<PathVariable>;%JAVA_HOME%\bin`.

2.5. Setting up the Java environment variable on Linux

Prior to installing your *Talend* product, you have to set the `JAVA_HOME` and `Path` environment variables:

1. Find the JRE installation home directory.

Example: `/usr/lib/jvm/jre1.8.0_65`

2. Export it in the `JAVA_HOME` environment variable.

Example:

```
export JAVA_HOME=/usr/lib/jvm/jre1.8.0_65
export PATH=$JAVA_HOME/bin:$PATH
```

3. Add these lines at the end of the user profiles in the `~/.profile` file or, as a superuser, at the end of the global profiles in the `/etc/profile` file.
4. Log on again.

2.6. Installing 7-Zip (Windows)

Talend recommends to install 7-Zip and to use it to extract the installation files: <http://www.7-zip.org/download.html>.

1. Download the 7-Zip installer corresponding to your Operating System.
2. Navigate to your local folder, locate and double-click the 7z exe file to install it.

The download will start automatically.



Chapter 3. Downloading and installing Talend Open Studio for Data Integration

Talend Open Studio for Data Integration is easy to install. After downloading it from *Talend's* Website, a simple unzipping will install it on your computer.

This chapter provides basic information useful to download and install it.

3.1. Downloading Talend Open Studio for Data Integration

Talend Open Studio for Data Integration is a free open source product that you can download directly from *Talend's* Website:

1. Go to *Talend Open Studio for Data Integration* [Download page](#).
2. Click **DOWNLOAD FREE TOOL**.

The download will start automatically.

3.2. Installing Talend Open Studio for Data Integration

Installation is done by unzipping the TOS_DI zip file previously downloaded.

This can be done either by using:

- 7Zip (Windows recommended): [Extracting via 7-Zip \(Windows recommended\)](#).
- Windows default unzipper: [Extracting via Windows default unzipping tool](#).
- Linux default unzipper (for a Linux based Operating System): [Extracting via the Linux GUI unzipper](#).

3.2.1. Extracting via 7-Zip (Windows recommended)

For Windows, *Talend* recommends you to install 7-Zip and use it to extract files. For more information, see [Installing 7-Zip \(Windows\)](#).

To install the studio, follow the steps below:

1. Navigate to your local folder, locate the **TOS** zip file and move it to another location with a path as short as possible and without any space character.

Example: *C:/Talend/*

2. Unzip it by right-clicking on the compressed file and selecting **7-Zip > Extract Here**.

3.2.2. Extracting via Windows default unzipping tool

If you do not want to use 7-Zip, you can use Windows default unzipping tool:

1. Unzip it by right-click the compressed file and select, **Extract All**.
2. Click on **Browse** and navigate to the *C: drive*.
3. Select **Make new folder** and name the folder *Talend*. Click **OK**.

4. Click on **Extract** to begin the installation.

3.2.3. Extracting via the Linux GUI unzipper

To install the studio, follow the steps below:

1. Navigate to your local folder, locate the **TOS** zip file and move it to another location with a path as short as possible and without any space character.

Example: *home/user/talend/*

2. Unzip it by right-clicking on the compressed file and selecting **Extract Here**.



Chapter 4. Configuring and setting up your Talend product

This chapter provides basic information required to configure and set up your *Talend* product, including:

- *Launching the Studio for the first time*
- *Logging in to the Studio*
- *Installing additional packages*

4.1. Launching the Studio for the first time

The Studio installation directory contains binaries for several platforms including Mac OS X and Linux/Unix.

To open the *Talend Studio* for the first time, do the following:

1. Double-click the executable file corresponding to your operating system, for example:
 - TOS_*-win-x86_64.exe, for Windows.
 - TOS_*-linux-gtk-x86_64, for Linux.
 - TOS_*-macosx-cocoa.app, for Mac.
2. In the **[User License Agreement]** dialog box that opens, read and accept the terms of the end user license agreement to proceed.

4.2. Logging in to the Studio

To log in to the *Talend Studio* for the first time, do the following:

1. In the *Talend Studio* login window, select **Create a new project**, specify the project name: *getting_started* and click **Finish** to create a new local project.
2. Depending on the product you are using, either of the following opens:
 - the Quick Tour. Play it to get more information on the User Interface of the Studio, and click **Stop** to end it.
 - the Welcome page. Follow the links to get more information about the Studio, and click **Start Now!** to close the page and continue opening the Studio.

Now you have successfully logged in to the *Talend Studio*. Next you need to install additional packages required for the *Talend Studio* to work properly.

4.3. Installing additional packages

Talend recommends that you install additional packages, including third-party libraries and database drivers, as soon as you log in to your *Talend Studio* to allow you to fully benefit from the functionalities of the Studio.

1. When the **[Additional Talend Packages]** wizard opens, install additional packages by selecting the **Required** and **Optional third-party libraries** check boxes and clicking **Finish**.

This wizard opens each time you launch the studio if any additional package is available for installation unless you select the **Do not show this again** check box. You can also display this wizard by selecting **Help > Install Additional Packages** from the menu bar.

For more information, see the section about installing additional packages in the *Talend Installation and Upgrade Guide*.

2. In the **[Download external modules]** window, click the **Accept all** button at the bottom of the wizard to accept all the licenses of the external modules used in the studio.

Depending on which libraries you selected, you may need to accept their license more than once.

Wait until all the libraries are installed before starting to use the studio.

3. If required, restart your *Talend Studio* for certain additional packages to take effect.



Chapter 5. Performing data integration tasks

This chapter takes the example of a company that provides movie rental and streaming video services, and shows how such a company could make use of Talend Open Studio for Data Integration.

You will work with data about movies and directors and data about your customers as you learn how to filter data in order to separate movie entries with valid director information from those without.

For more real-life examples, see the *Theory into practice* chapter of your *Talend Studio User Guide*.

5.1. Reading movies information from a CSV file

The examples provided in this chapter assume that:

- You have launched your **Talend Studio** and opened the **Integration** perspective.
- You have installed all the required third-part libraries and database drivers in your **Talend Studio**.
- You have properly installed and configured the MySQL database software, and created a database named *gettingstarted*.

In this scenario, you will learn:

- How to create a data integration Job. See [Creating your first Job](#) for details.
- How to add and link components in a data integration Job. See [Dropping and linking components](#) for details.
- How to create file metadata in the **Repository**. See [Preparing the movies metadata](#) for details.
- How to configure and execute a data integration Job. See [Configuring and executing your Job](#) for details.

If you want to replicate the example described in this document and use the exact input data, you can download the source files [here](#) and then save them in your local directory `C:\getting_started\input_data\`.

5.1.1. Creating your first Job

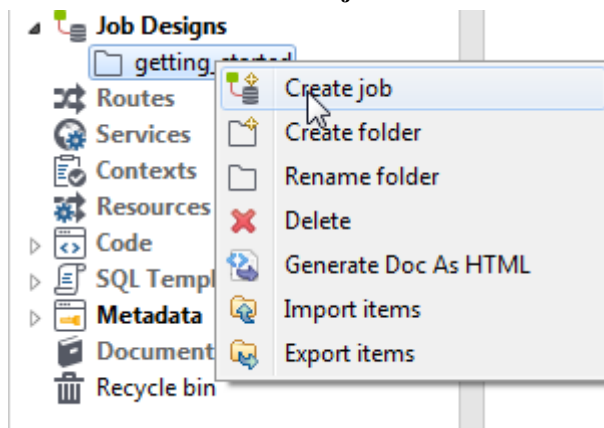
This procedure describes how to create a Job folder named *getting_started* and a Job named *movies* in the folder.

Follow the steps below to create a folder named *getting_started*:

1. In the **Repository** tree view, right-click the **Job Designs** node and select **Create folder** from the contextual menu.
2. In the **[New Folder]** wizard, name your Job folder *getting_started* and click **Finish** to create your folder.

Follow the steps below to create a Job named *movies* in the *getting_started* folder:

1. Right-click the *getting_started* folder and select **Create job** from the contextual menu.



2. In the **[New Job]** wizard, enter a name for the Job to be created and other useful information.

For example, enter *movies* in the **Name** field.

In this step of the wizard, **Name** is the only mandatory field. The information you provide in the **Description** field will appear as a tooltip when you move your mouse pointer over the Job in the **Repository** tree view.

3. Click **Finish** to create your Job.

An empty Job is opened in the Studio.

The Job is now created and automatically opened in the design workspace. Next you need to add components to the Job.

5.1.2. Dropping and linking components

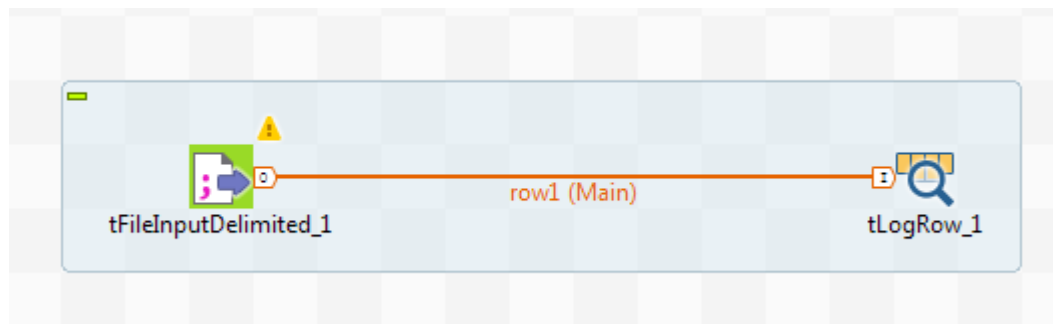
This example describes how to add and link components in the newly created Job, so that it will read a CSV file and display the data on the console.

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

You can find the **tFileInputDelimited** component in the **Input** group of the **File** family and the **tLogRow** component in the **Logs & Errors** family in the **Palette**.

2. Click the **tFileInputDelimited** component so that an **o** icon appears, drag and drop the **o** icon onto the **tLogRow** component.

The two components are now connected via a **Row > Main** connection.



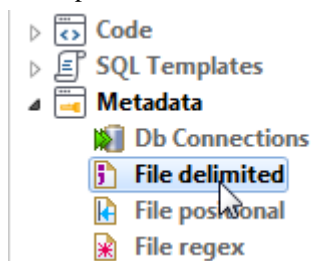
Now you have added the required components to the Job. In the next steps you will need to prepare the required metadata and configure the Job.

5.1.3. Preparing the movies metadata

This example describes how to set up the metadata of the source file *movies.csv* in the **Repository**. Repository metadata can be used across Jobs, allowing you to configure your Jobs quickly without having to define each parameter and schema manually.

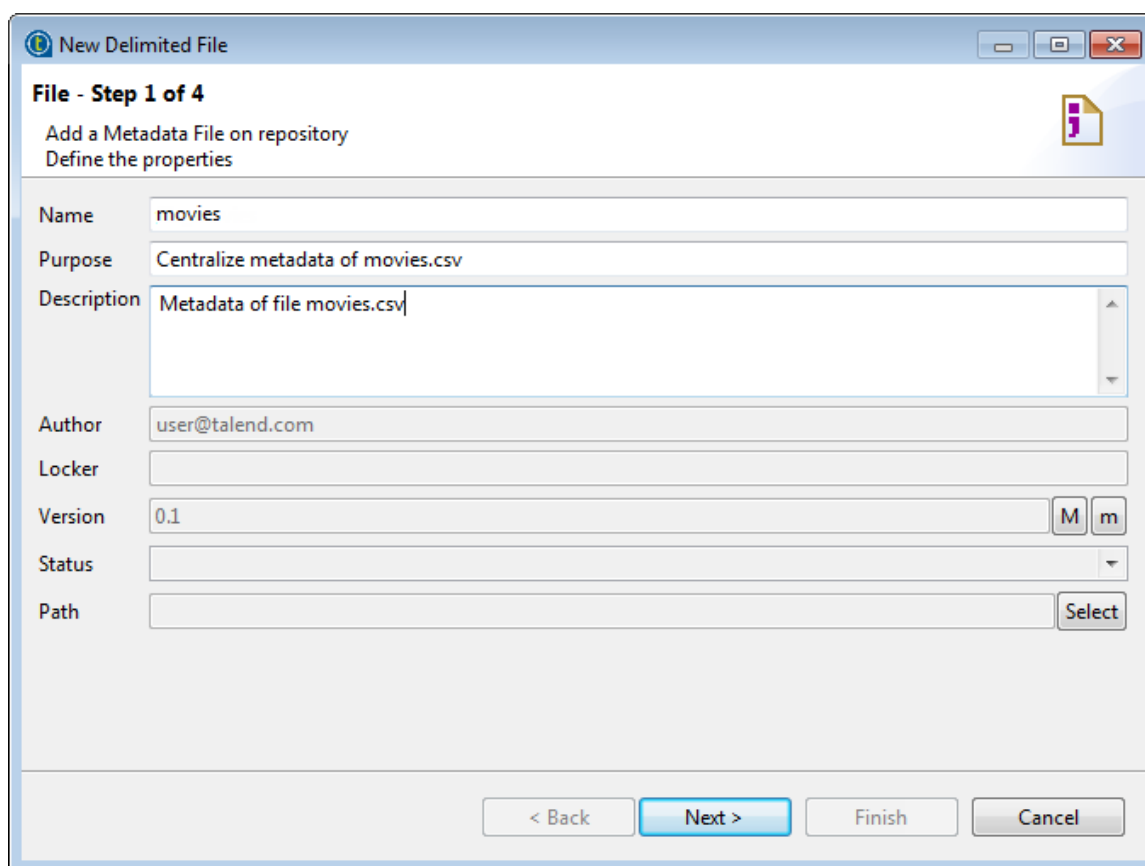
Prerequisites:

- You have the source file *movies.csv* ready in the directory *C:\getting_started\input_data*.
- In the **Repository** tree view, expand the **Metadata** node, right-click **File delimited**, and select **Create file delimited** from the contextual menu to open the [New Delimited File] wizard.



- In the [New Delimited File] wizard enter a name for the file metadata, *movies* in this example, and other useful information to better describe your file metadata, and then click **Next** to go to the next step and define the general properties of the file.

In this step of the wizard, **Name** is the only mandatory field. The information you provide in the **Description** field will appear as a tooltip when you move your mouse pointer over the file connection.



New Delimited File

File - Step 1 of 4

Add a Metadata File on repository
Define the properties

Name: movies

Purpose: Centralize metadata of movies.csv

Description: Metadata of file movies.csv

Author: user@talend.com

Locker:

Version: 0.1 M m

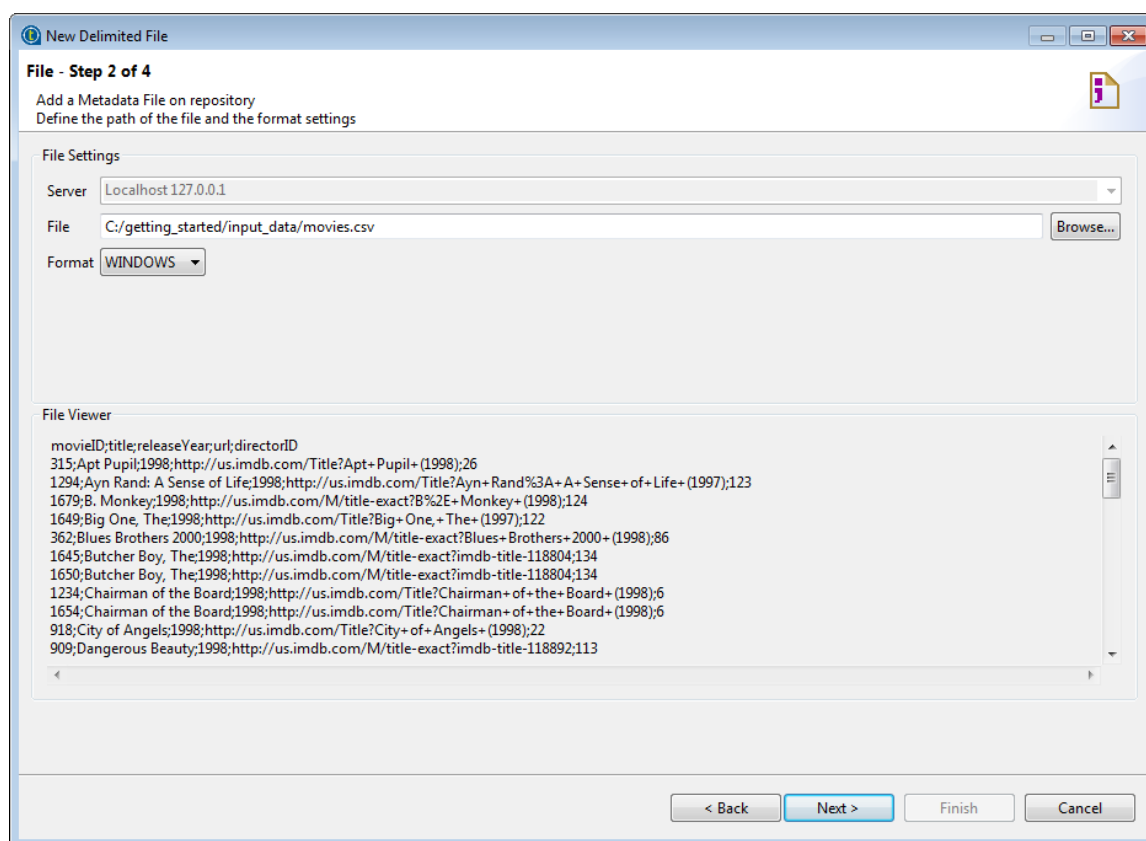
Status:

Path: Select

< Back Next > Finish Cancel

3. In the **File** field specify the path of the source file, or click **Browse** to browse to the file.

The **File Viewer** area displays an abstract of the file, allowing you to check the file consistency, the presence of header and more generally the file structure.



4. From the **Format** list, select your operating system, and click **Next** to parse the file.
5. On the **Preview** tab, select the **Set heading row as column names** check box to retrieve the file column names from the first row, and then click **Refresh Preview**.

The **Header** check box in the **Rows To Skip** area is automatically selected and the number of header rows to be skipped is incremented by 1. If the file contains more than one heading row, which need to be skipped in file parsing, specify the number in this field before clicking **Refresh Preview**.

New Delimited File

File - Step 3 of 4
Add a Metadata File on repository
Define the setting of the parse job

File Settings
 Encoding: US-ASCII
 Field Separator: Semicolon Corresponding Character: ";"
 Row Separator: Standard EOL Corresponding Character: "\n"

Escape Char Settings
☐ CSV ☒ Delimited
 Escape Char: Empty
 Text Enclosure: Empty
☐ Split row before field

Rows To Skip
 If any rows must be ignored, specify the following parameters
 Header: ☒ 1
 Footer: ☐
☐ Skip empty row

Limit Of Rows
 If the number of lines must be limited, specify this number
 Limit: ☐

Preview **Output**

☒ Set heading row as column names [Refresh Preview](#)

movieID	title	releaseYear	url	directorID
315	Apt Pupil	1998	http://us.imdb.com/Title?Apt+Pupil+(1998)	26
1294	Ayn Rand: A Sense of Life	1998	http://us.imdb.com/Title?Ayn+Rand%3A+A+Sense+of+Life+(1997)	123
1679	B. Monkey	1998	http://us.imdb.com/M/title-exact?B%2E+Monkey+(1998)	124
1649	Big One, The	1998	http://us.imdb.com/Title?Big+One,+The+(1997)	122

[Export as context](#) [Revert Context](#)

< Back **Next >** Finish Cancel

- Click **Next** to retrieve the file schema.

The **Description of the Schema** table displays the generated file schema.

- Name the schema *movies_schema* and check the file schema and edit it according to your actual needs. In this example, increase the length of the *title* and *url* columns.

New Delimited File

File - Step 4 of 4
Add a Schema on repository
Define the Schema

Name:

Comment:

Schema

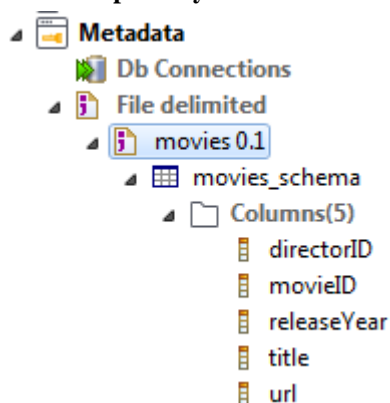
Click to update schema preview

Description of the Schema

Column	Key	Type	<input checked="" type="checkbox"/> N...	Date Pattern (Ctrl+Sp...	Length	Precision	Default	Comment
movieID	<input type="checkbox"/>	Integer	<input checked="" type="checkbox"/>		4	0		
title	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>		72	0		
releaseYear	<input type="checkbox"/>	Integer	<input checked="" type="checkbox"/>		4	0		
url	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>		120	0		
directorID	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>		3	0		

- Click **Finish** to validate the schema close the wizard.

The created file metadata is shown in the **Repository** tree view.



You now have the movies file metadata ready for use. Next, you need to apply the created metadata to the component that reads the source file.

5.1.4. Configuring and executing your Job

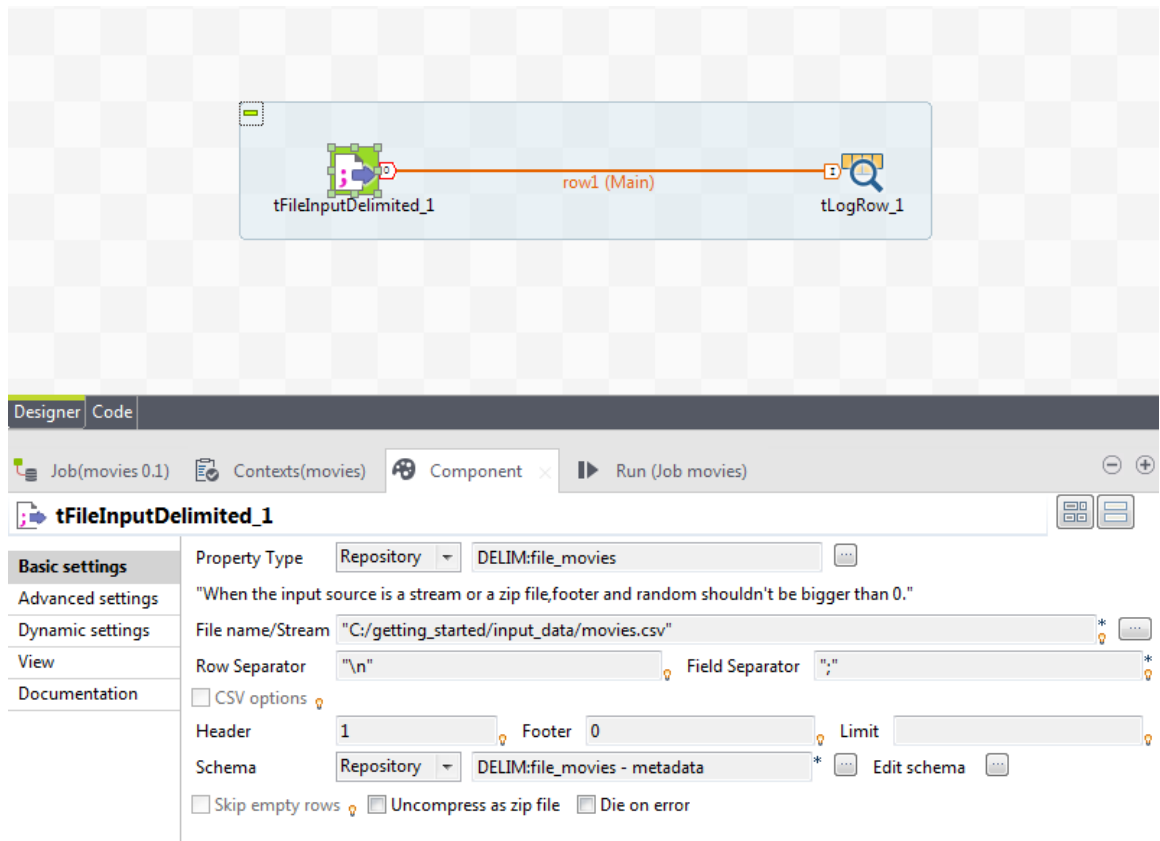
This example describes how to configure the components using the metadata created in the previous procedure and run your Job.

- In the **Repository** tree view, double-click the Job *movies* to open it in the design workspace.

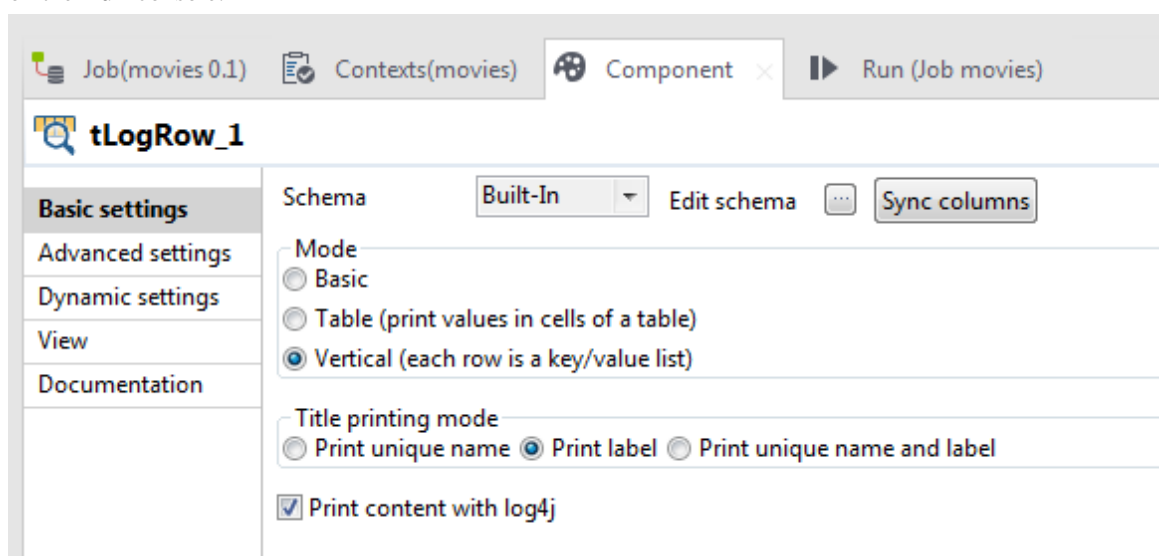
You can skip this step if the Job is already open and active in the design workspace.

2. In the **Repository** tree view, expand **Metadata > File delimited**, and drag and drop the file connection *movies* or its schema *movies_schema* onto the **tFileInputDelimited** component in the design workspace. When asked whether to propagate the changes to the output component, click **Yes**.

In the **Basic settings** tab of the **Component** view, you'll find that all the parameters of the component have been automatically filled.



3. Double-click the **tLogRow** component to open its **Basic settings** tab view.
4. In the **Mode** area, select the **Vertical (each row is a key/value list)** option for better readability of long fields on the **Run** console.



5. Press **F6** or click the **Run** button on the **Run** view to execute your Job.

The screenshot shows the Talend Open Studio interface with the 'Run' view active. The 'Job movies' console displays the following data:

```
[statistics] connected
```

#1. tLogRow_1	
key	value
movieID	315
title	Apt Pupil
releaseYear	1998
url	http://us.imdb.com/Title?Apt+Pupil+(1998)
directorID	26

#2. tLogRow_1	
key	value
movieID	1294
title	Ayn Rand: A Sense of Life
releaseYear	1998
url	http://us.imdb.com/Title?Ayn+Rand%3A+A+Sense+of+Life+(1997)
directorID	123

The **Run** console displays the movies information read from the source file.

5.2. Filtering the movies information

This scenario will extend the Job described in [Reading movies information from a CSV file](#) to filter the data flow to get only those movies with valid director information.

This scenario demonstrates:

- How to duplicate a Job. See [Duplicating the existing Job](#) for details.
- How to add a component by typing its name on a connection or on the design workspace. See [Adding a mapping component](#) for details.
- How to drop a metadata item or its schema as a component on the design workspace. See [Adding a lookup component](#) for details.
- How to perform basic processing to data flows using **tMap**. See [Configuring mappings and executing the Job](#) for details.

5.2.1. Preparing directors file metadata

This procedure shows how to set up the metadata of the reference file *directors.txt* in the **Repository**. This metadata item will be used to add and set up the lookup input in this scenario.

Prerequisites:

- You have the file *directors.txt* ready in the directory *C:\getting_started\input_data*.

1. In the **Repository** tree view, expand the **Metadata** node, right-click **File delimited**, and select **Create file delimited** from the contextual menu to open the **[New Delimited File]** wizard.
2. In the **[New Delimited File]** wizard enter a name for the file metadata, *directors* in this example, and other useful information to better describe your file metadata. Then, click **Next** to go to the next step and define the general properties of the file.

New Delimited File

File - Step 1 of 4

Add a Metadata File on repository
Define the properties

Name: directors

Purpose: Centralize the metadata of directors info

Description: Metadata of the directors dataset

Author: user@talend.com

Locker:

Version: 0.1 M m

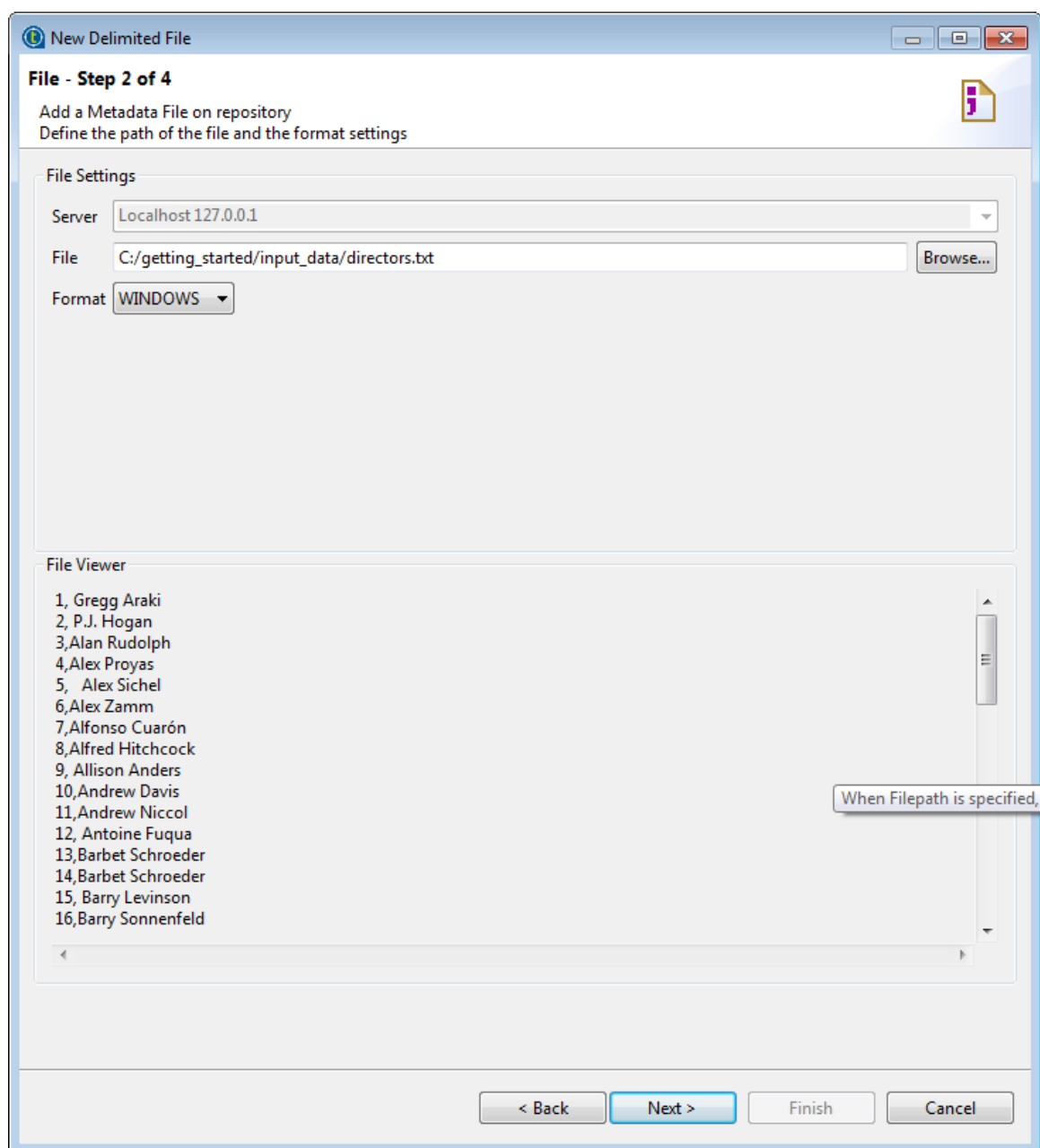
Status:

Path: Select

< Back Next > Finish Cancel

3. In the **File** field specify the path of the source file, *C:/getting_started/input_data/directors.txt* in this example, or click **Browse** to browse to the file.

The **File Viewer** area displays an abstract of the file, allowing you to check the file consistency, the presence of a header and the structure of the file.



4. Select **Windows** from the **Format** list, and click **Next** to parse the file.
5. From the **Field Separator** list of the **File Settings** area, select **Comma**.

New Delimited File

File - Step 3 of 4
Add a Metadata File on repository
Define the setting of the parse job

File Settings
 Encoding: UTF-8
 Field Separator: Comma Corresponding Character: " "
 Row Separator: Standard EOL Corresponding Character: "\n"

Escape Char Settings
☐ CSV ☒ Delimited
 Escape Char: Empty
 Text Enclosure: Empty
☐ Split row before field

Rows To Skip
 If any rows must be ignored, specify the following parameters
 Header: ☐
 Footer: ☐
☐ Skip empty row

Limit Of Rows
 If the number of lines must be limited, specify this number
 Limit: ☐

Preview **Output**
☐ Set heading row as column names **Refresh Preview**

Column 0
1, Gregg Araki
2, P.J. Hogan
3, Alan Rudolph
4, Alex Proyas
5, Alex Sichel
6, Alex Zamm

Export as context **Revert Context**

< Back **Next >** Finish Cancel

- Click **Next** to go to retrieve the file schema.

The **Description of the Schema** table displays the generated file schema.

- Name the schema *directors_schema* and rename the columns to *directorID* and *directorName* respectively, and change the data type of the *directorID* columns from Integer to String.

File - Step 4 of 4
Add a Schema on repository
Define the Schema

Name:
Comment:

Schema
Click to update schema preview Guess

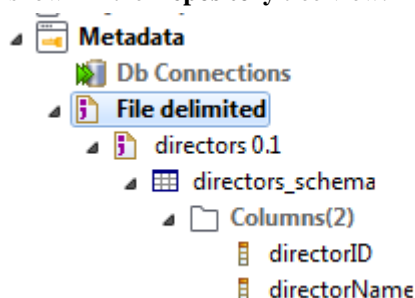
Description of the Schema

Column	Key	Type	N..	Date Pattern (Ctrl+Sp...	Length	Precision	Default	Comment
directorID	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>		2	0		
directorName	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>		20	0		

< Back Next > Finish Cancel

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

The created file metadata is shown in the **Repository** tree view.



You now have the directors file metadata ready for use when you set up the component to read the reference file.

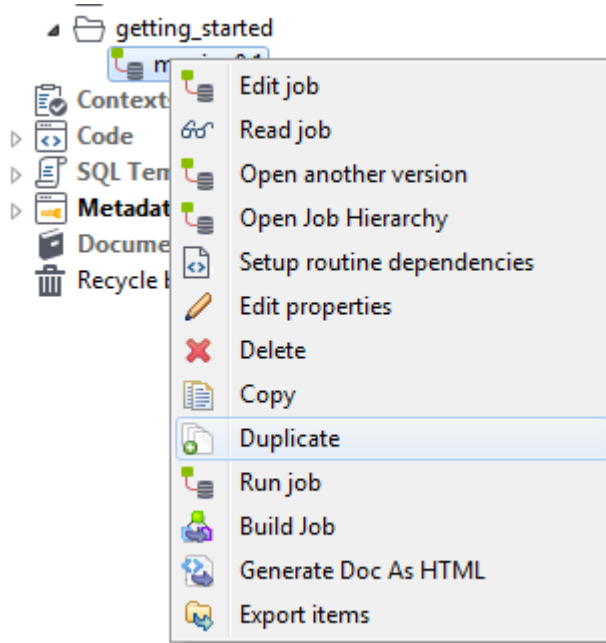
5.2.2. Duplicating the existing Job

This procedure shows how to create a Job based on an existing Job.

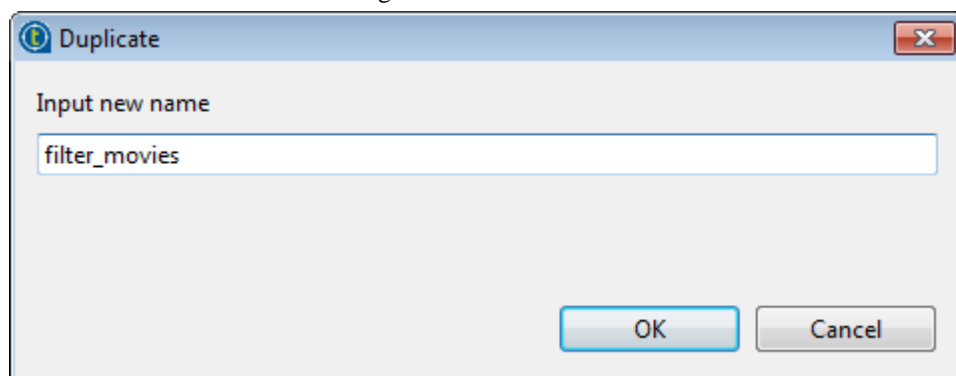
Prerequisites:

- You have created and successfully executed the Job *movies* as described in [Reading movies information from a CSV file](#).

1. In the **Repository** tree view, right-click the Job named *movies* and select **Duplicate** from the contextual menu.



2. In the **[Duplicate]** dialog box, enter a name for the new Job, *filter_movies* in this example, and click **OK** to validate the Job creation and close the dialog box.

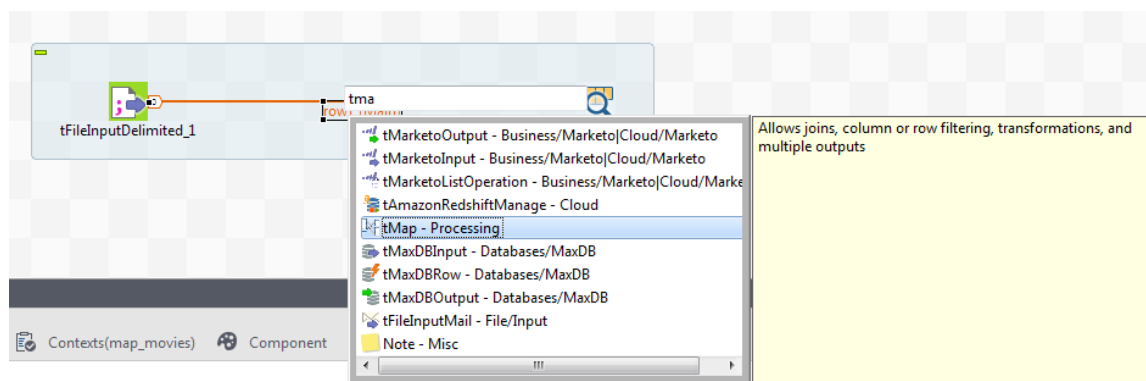


The Job *filter_movies* is created, which is a duplicate of the Job *movies*.

5.2.3. Adding a mapping component

The procedure below shows how to add a mapping component by typing the component name directly on the existing connection.

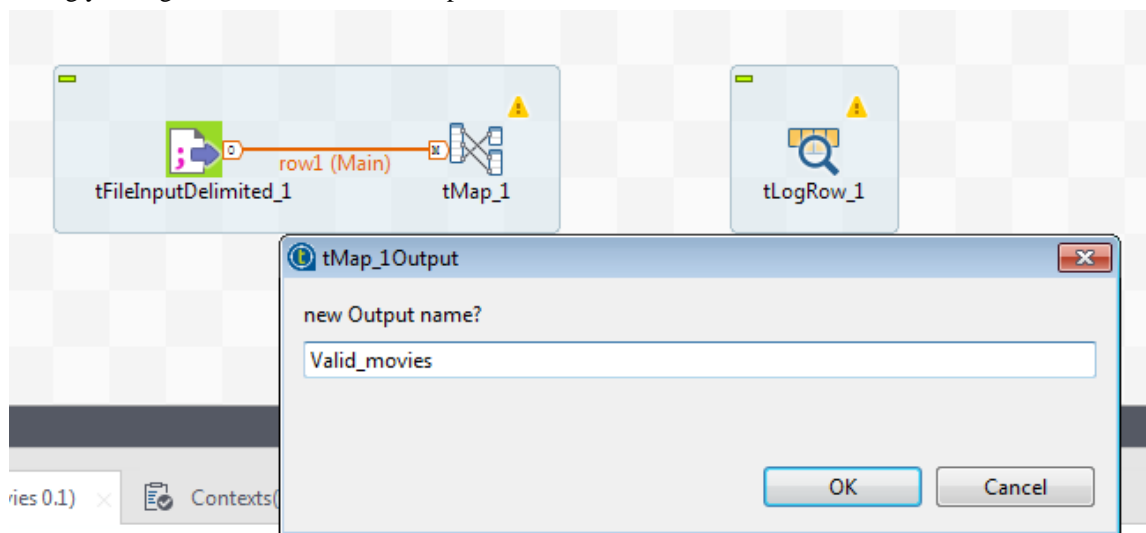
1. In the new Job *filter_movies*, select the **Row** connection linking the **tFileInputDelimited** and **tLogRow** components, and type name of **tMap** or part of it.



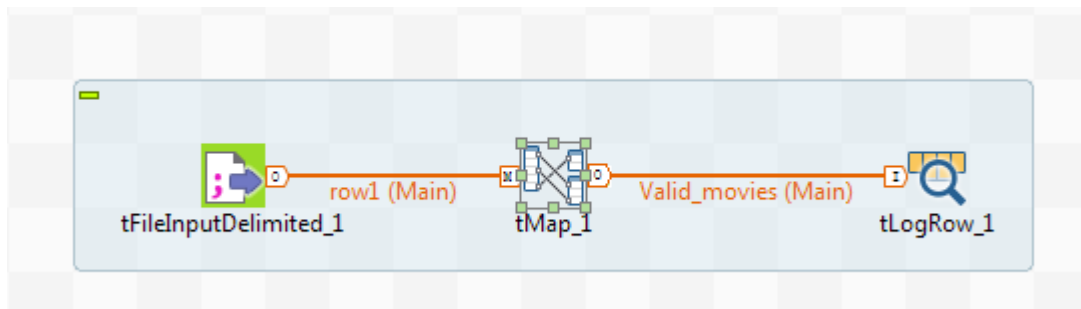
When you start typing the component name, a list of components that match your input appears. You can select a component to view its description besides the component list.

2. Double-click **tMap** on the list to add it onto the connection.

The newly added **tMap** component is now connected with the input component, and a dialog box opens asking you to give a name to the new output connection.



3. Enter a name for the new output connection, *Valid_movies* in this example, and click **OK**. When asked whether you want to propagate the input schema to the target output component, click **Yes**.



The **tMap** component is now added to the Job and connected with the two existing components via **Row > Main** connections.

5.2.4. Adding a lookup component

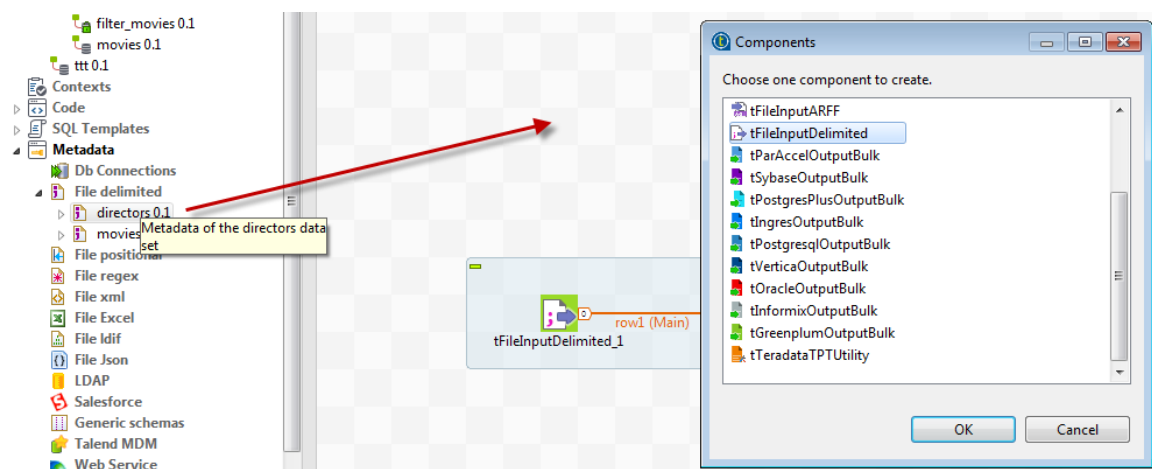
The procedure below shows how to add a lookup input component from the **Repository**, connect it to the **tMap**, and enable column trimming in the component.

Prerequisites:

- You have centralized the metadata for *directors.txt* in the **Repository** as described in [Preparing directors file metadata](#).

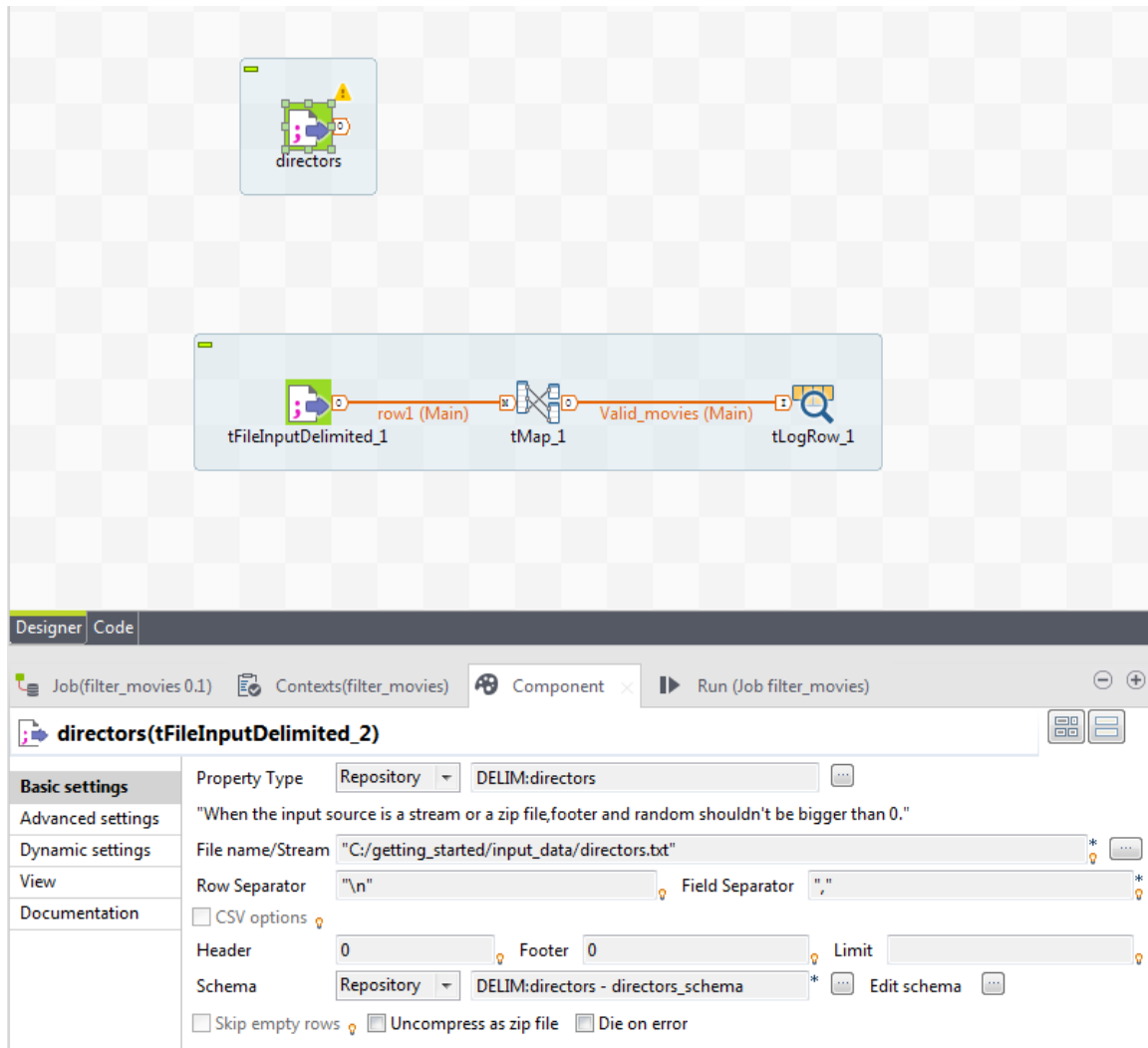
- In the **Repository** tree view, expand **Metadata > File delimited**, drag and drop the file connection *directors* or its schema *directors_schema* onto the design workspace.

The **[Components]** dialog box opens, showing a list of components you can add to the Job from this metadata item.



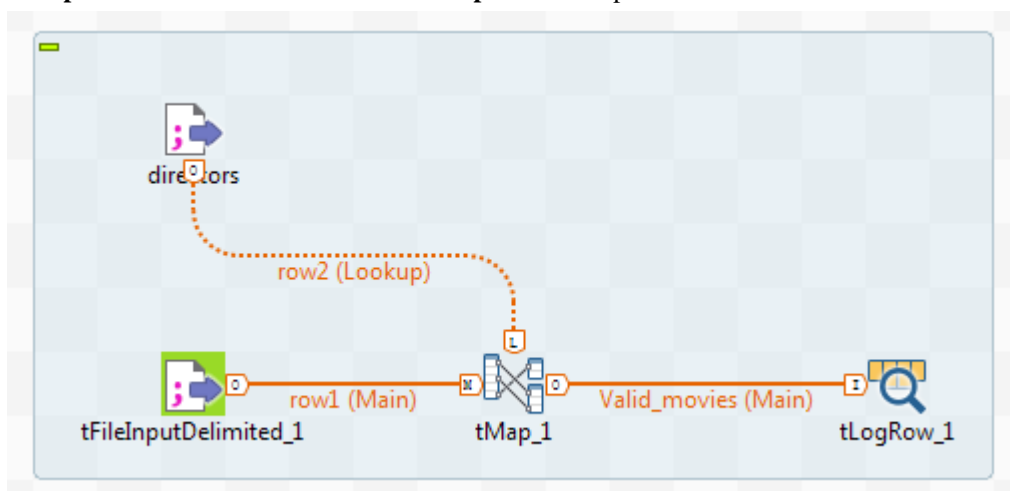
- Select **tFileInputDelimited** and click **OK**.

A **tFileInputDelimited** labelled *directors* is added to the design workspace, with its basic settings automatically filled.



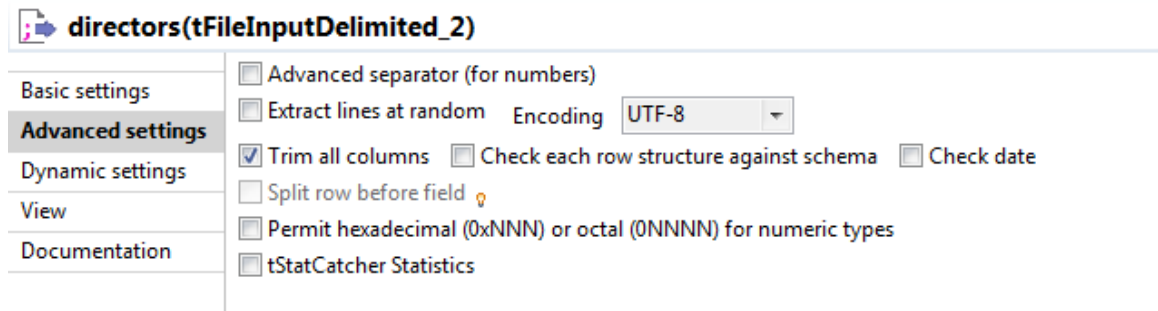
- Right-click the newly added **tFileInputDelimited** component, select **Row > Main** from the contextual menu, and click the **tMap** component.

The **tFileInputDelimited** is connected to the **tMap** via a lookup connection now.



- In the **Advanced settings** tab of the new **tFileInputDelimited** component, and select the **Trim all columns** check box.

Some records of the reference input file *directors.txt* contains leading white spaces. This option allows you to remove such white spaces from the lookup input flow when the Job is executed.



directors(tFileInputDelimited_2)

- Basic settings
- Advanced settings**
- Dynamic settings
- View
- Documentation

☐ Advanced separator (for numbers)
☐ Extract lines at random Encoding: UTF-8
☒ Trim all columns ☐ Check each row structure against schema ☐ Check date
☐ Split row before field
☐ Permit hexadecimal (0xNNN) or octal (0NNNN) for numeric types
☐ tStatCatcher Statistics

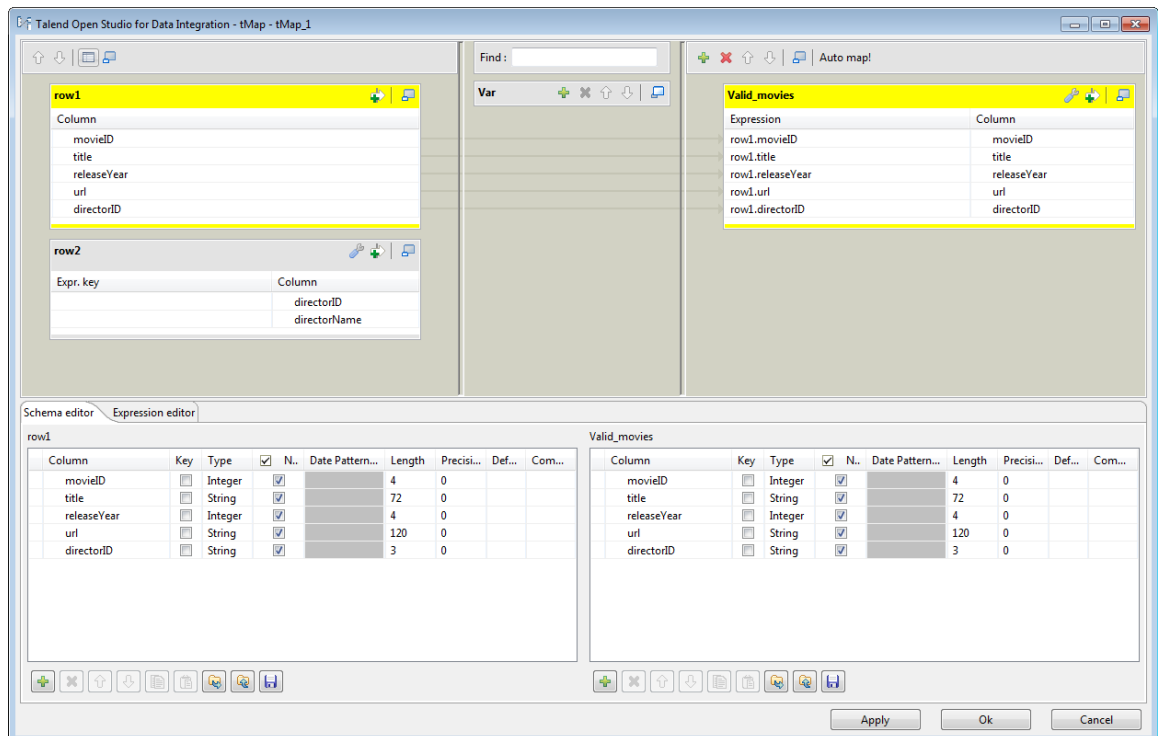
You have now all the components in the Job needed for filtering the movies information. Next you'll need to configure mappings in the **tMap** component to filter the main input flow against the lookup flow and output the desired information.

5.2.5. Configuring mappings and executing the Job

The procedure below shows how to configure mappings and an inner join to output movies information with valid director IDs.

1. Double-click the **tMap** component to open the map editor.

The map editor shows three tables, named *row1*, *row2* and *Valid_movies* in this example, corresponding respectively to the movies file schema, the directors file schema, and the schema of the output for valid movies information, and columns in the *row1* table are already mapped to the columns in the *Valid_movies* table.



Talend Open Studio for Data Integration - tMap - tMap_1

Find: Var: Auto map!

row1

Column
movieID
title
releaseYear
url
directorID

row2

Expr. key	Column
	directorID
	directorName

Valid_movies

Expression	Column
row1.movieID	movieID
row1.title	title
row1.releaseYear	releaseYear
row1.url	url
row1.directorID	directorID

Schema editor Expression editor

row1

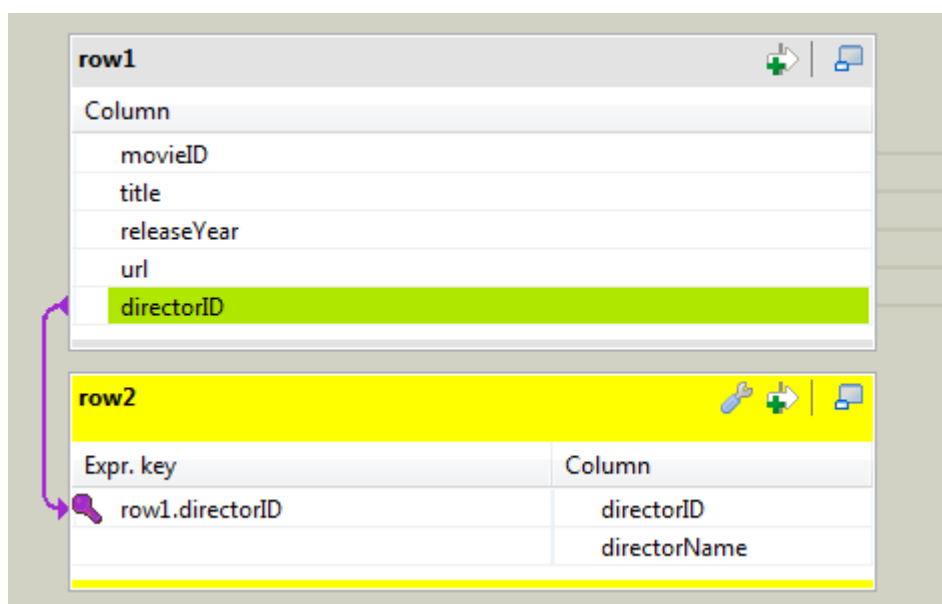
Column	Key	Type	✓	N..	Date Pattern...	Length	Precisi...	Def...	Com...
movieID	<input type="checkbox"/>	Integer	<input checked="" type="checkbox"/>			4	0		
title	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			72	0		
releaseYear	<input type="checkbox"/>	Integer	<input checked="" type="checkbox"/>			4	0		
url	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			120	0		
directorID	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			3	0		

Valid_movies

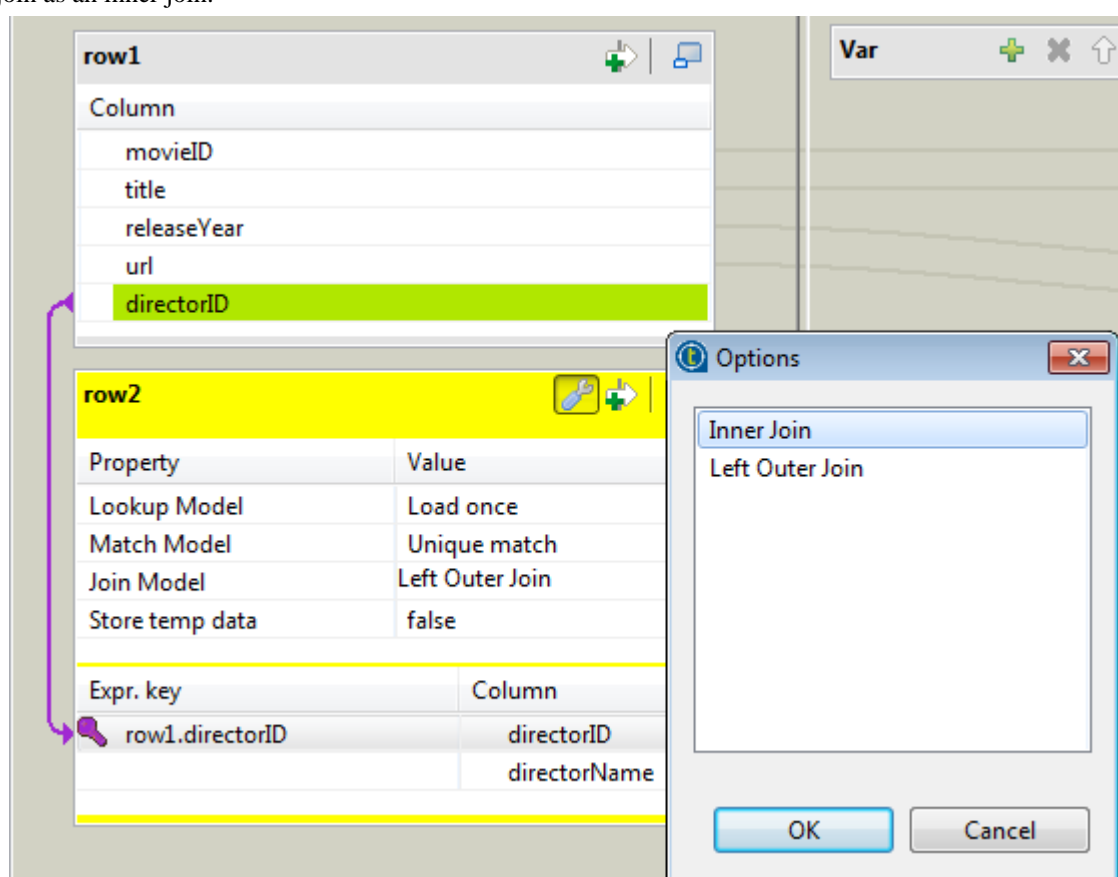
Column	Key	Type	✓	N..	Date Pattern...	Length	Precisi...	Def...	Com...
movieID	<input type="checkbox"/>	Integer	<input checked="" type="checkbox"/>			4	0		
title	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			72	0		
releaseYear	<input type="checkbox"/>	Integer	<input checked="" type="checkbox"/>			4	0		
url	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			120	0		
directorID	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			3	0		

Apply Ok Cancel

2. Select the *directorID* column in the *row1* table, and drop it onto the *directorID* column in the *row2* table to create a join between the two input data sets based on the director IDs.



- Click the **tMap settings** button, then click **Value** field for **Join Model**, and then click the [...] button that appears to open the **[Options]** dialog box. In the dialog box, select **Inner Join** and click **OK** to define the join as an inner join.



With this setting, only the movie records with the director IDs matching with those in the reference file will be passed to the output.

- In the **Schema editor** at the bottom of the map editor, select *directorID* column of the output schema, *Valid_movies* in this example, and click the **[X]** button to remove it.

- Click the [+] button beneath the output table to add a new column, name it *directedBy*, set its length to 20, and move it up so that it's between the *title* and *releaseYear* columns.

Valid_movies

Column	Key	Type	<input checked="" type="checkbox"/>	N..	Date Pattern...	Length	Precisi...	Def...	Com...
movieID	<input type="checkbox"/>	Integer	<input checked="" type="checkbox"/>			4	0		
title	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			72	0		
directedBy	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			20			
releaseYear	<input type="checkbox"/>	Integer	<input checked="" type="checkbox"/>			4	0		
url	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			120	0		

- Select the *directorName* column in the *row2* table, and drop it to the **Expression** field corresponding to the *directedBy* column in the output table.

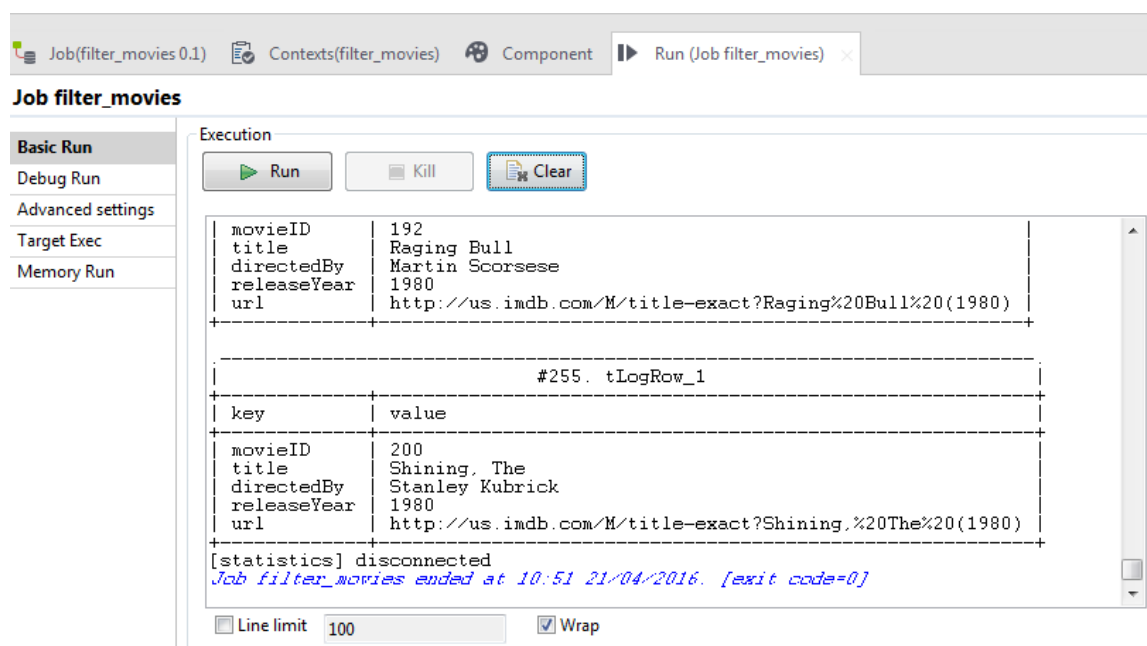
A new mapping is created between lookup table and the output table.

The screenshot shows the Talend mapping editor with three panels. On the left, the 'row1' table has columns: movieID, title, releaseYear, url, directorID. Below it, the 'row2' table has columns: Expr. key, Column, with row2.directorName selected. In the center, a 'Find:' search bar and a 'Var' panel are visible. On the right, the 'Valid_movies' table has columns: Expression, Column, with row2.directorName mapped to directedBy. A yellow arrow indicates the mapping from row2.directorName to Valid_movies.directedBy.

- Click **OK** to validate the mappings and close the map editor, and click **Yes** when asked whether to propagate the changes.

The mapping configurations are saved and the output schema is synchronized to the output component **tLogRow**.

- Press **F6** or click the **Run** button on the **Run** view to execute your Job.



Only movie records with valid director information are displayed on the **Run** console.

5.3. Gathering rejected movies information and saving processing results to a database

Based on the scenario described in [Filtering the movies information](#), this scenario further extends the Job to gather movies data missing director information and writes both valid and invalid data to a MySQL database.

This scenario demonstrates:

- How to add a component by typing on the design workspace or dragging from an existing component. See [Adding database output components to your Job](#) for details.
- How to configure mappings for rejected information in **tMap**. See [Configuring mappings for rejected data](#) for details.
- How to configure database outputs. See [Configuring MySQL database outputs](#) for details.

5.3.1. Adding database output components to your Job

In the example below we will create a new Job from the Job *filter_movies* and add two **tMySqlOutput** components. These components will be used to write the processed movies information to the specified database tables.

Prerequisites:

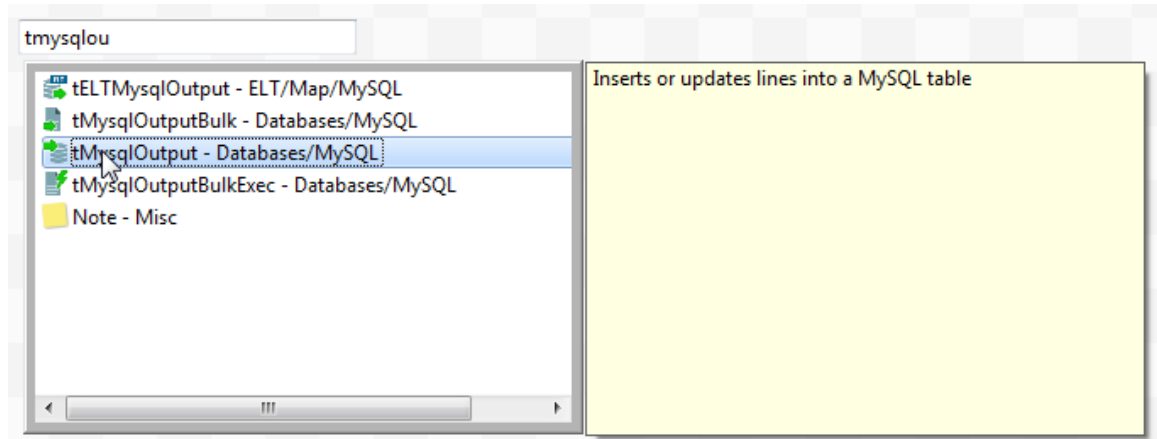
- You have created and successfully executed the Job *filter_movies* as described in [Filtering the movies information](#).

1. Create a new Job by duplicating the Job created in the previous scenario, and name the new Job *write_movies_to_db*, and then double-click the Job to open it in the design workspace.

For more information about how to duplicate a Job, see [Duplicating the existing Job](#).

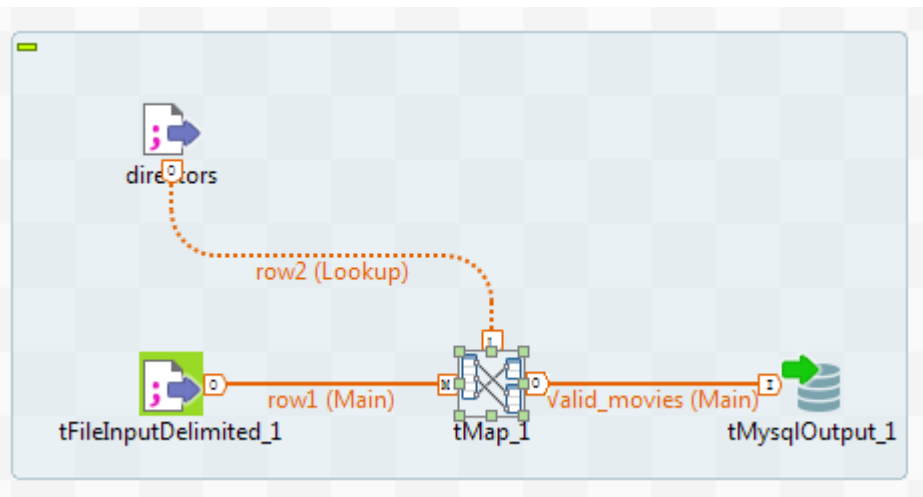
2. Right-click the **tLogRow** component and select **Delete** from the contextual menu to delete it.
3. Click where the **tLogRow** was on the design workspace and type the name of **tMySQLOutput** or part of it, and then select and double-click **tMySQLOutput** on the list to add it onto the design workspace.

When you start typing the component name, a list of components that match your input appears. You can select a component to view its description besides the component list.



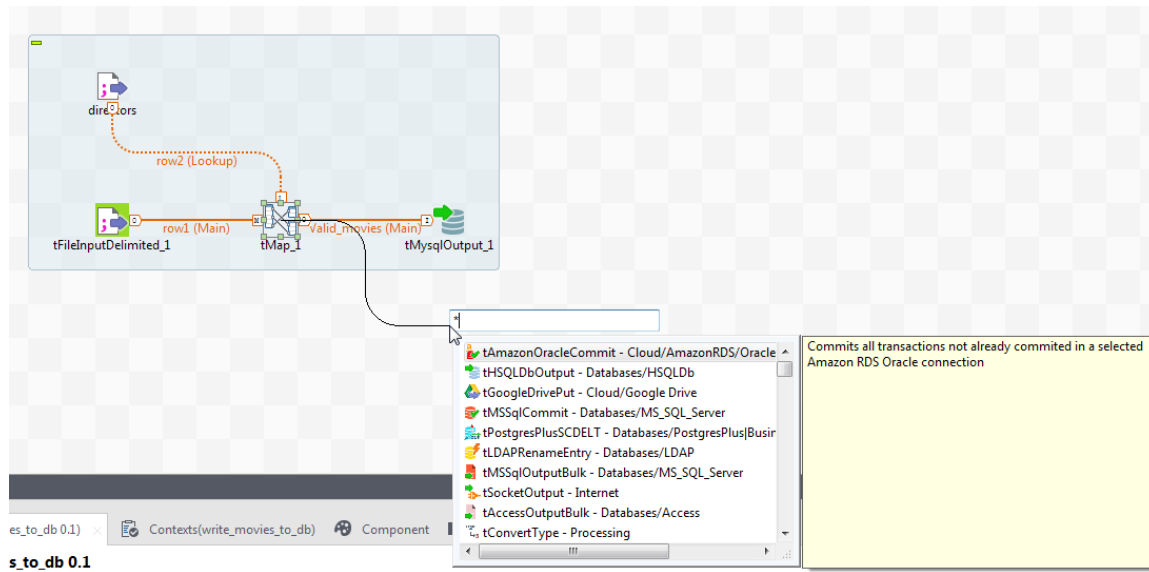
4. Right-click the **tMap** component, select **Row > Valid_movies** from the context menu, and click the **tMySQLOutput** to link it with the **tMap**.

The connection name *Valid_movies* corresponds to the name of the existing output table in **tMap**.



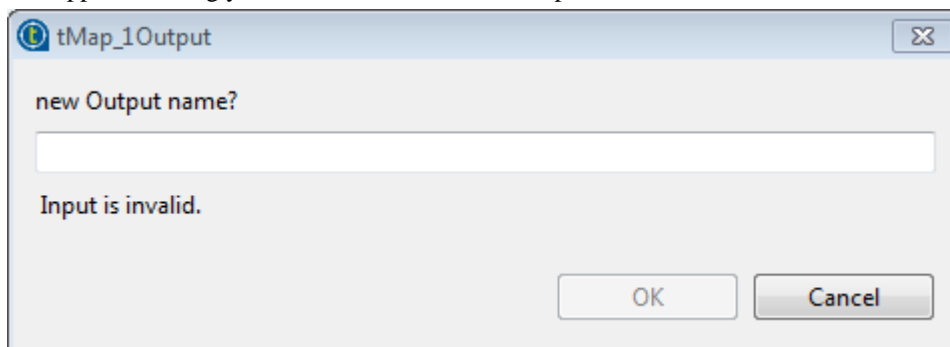
5. Click the **tMap** component, and drag and drop the **o** icon onto the design workspace.

A text field and a list of suggested components appear. You can select a component to view its description besides the component list.

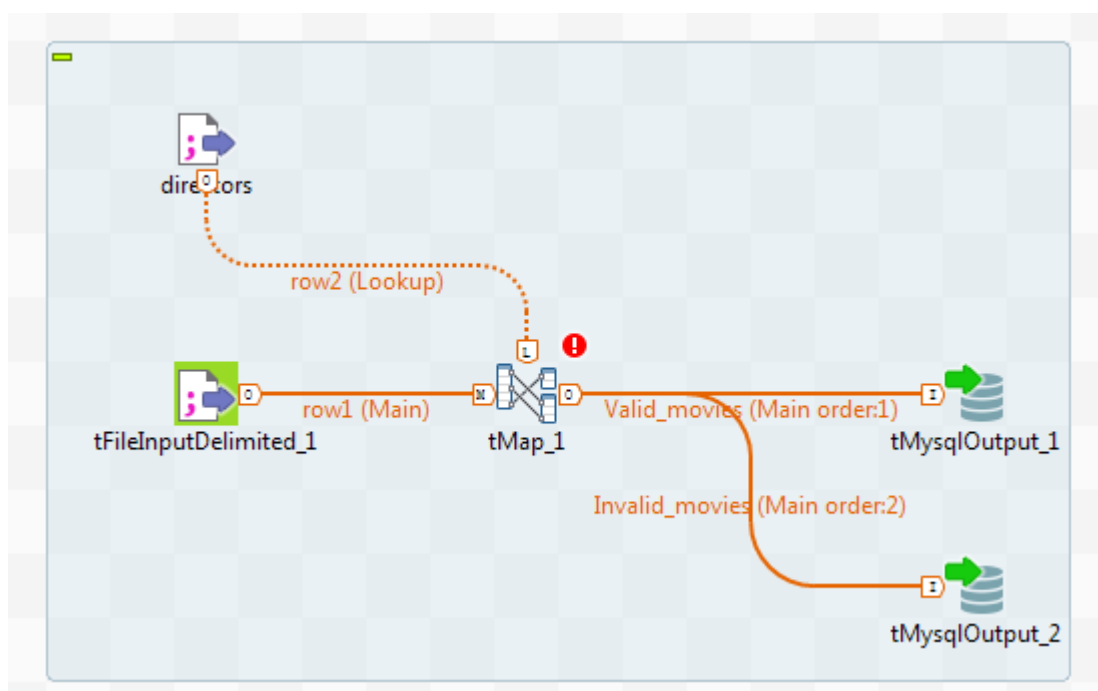


6. In the text field, type the name of **tMySQLOutput**, select the component on the list, and press **Enter** to add another **tMySQLOutput** component onto the design workspace.

A dialog box appears, asking you to enter a name for the output connection.



7. In the dialog box, enter *Invalid_movies* and click **OK** to connect **tMap** to the second **tMySQLOutput** component.

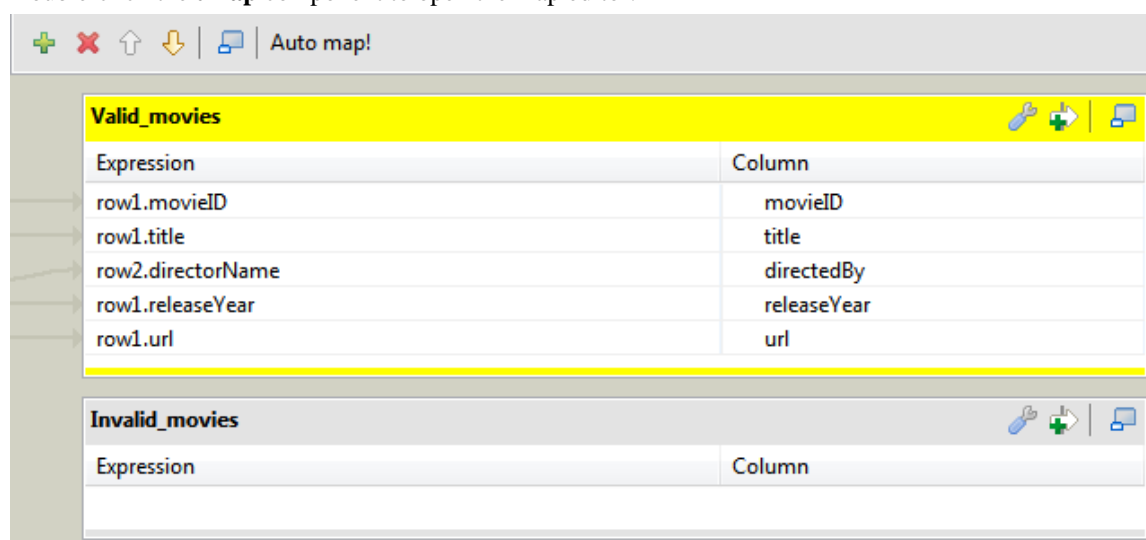


Now you have added and connected the database output components you need to write the processed movies information to a MySQL database. Next, you'll need to configure new mappings in the **tMap** and database settings in the **tMySQLOutput** components.

5.3.2. Configuring mappings for rejected data

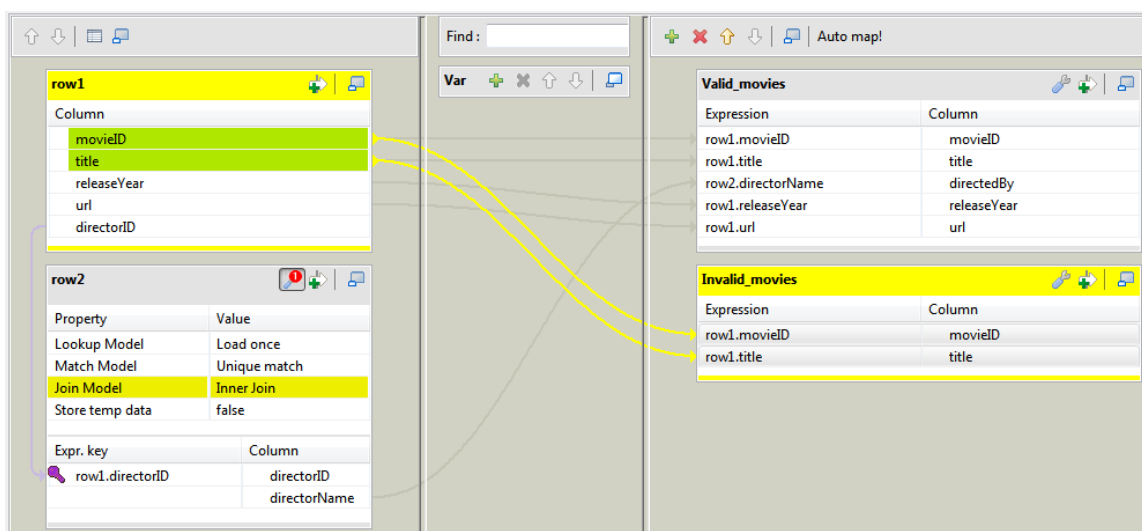
This procedure shows how to configure mappings to gather rejected information.

1. Double-click the **tMap** component to open the map editor.



An second output table named *Invalid_movies* has been automatically created.

2. Drop the *movieID* and *title* columns from the *row1* table to the *Invalid_movies* table.



- Click the **tMap settings** button on the *Invalid_movies* table, click the **Value** field for **Catch lookup inner join reject**, and then click the [...] button that appears to open the [Options] dialog box. In the dialog box, select **true** and click **OK**.

The screenshot shows the 'Invalid_movies' table configuration dialog box. The 'Property' table has 'Catch output reject' set to 'false' and 'Catch lookup inner join reject' set to 'true'. The 'Schema Type' is 'Built-In'. Below the 'Property' table, the 'Expression' table shows 'row1.movieID' mapped to 'movieID' and 'row1.title' mapped to 'title'.

Property	Value
Catch output reject	false
Catch lookup inner join reject	true
Schema Type	Built-In

Expression	Column
row1.movieID	movieID
row1.title	title

With this setting, any records without director IDs or with director IDs that do not match with those in the reference file will be passed to this output.

- Click **OK** to validate the mappings and close the map editor, and click **Yes** when asked whether to propagate the changes.

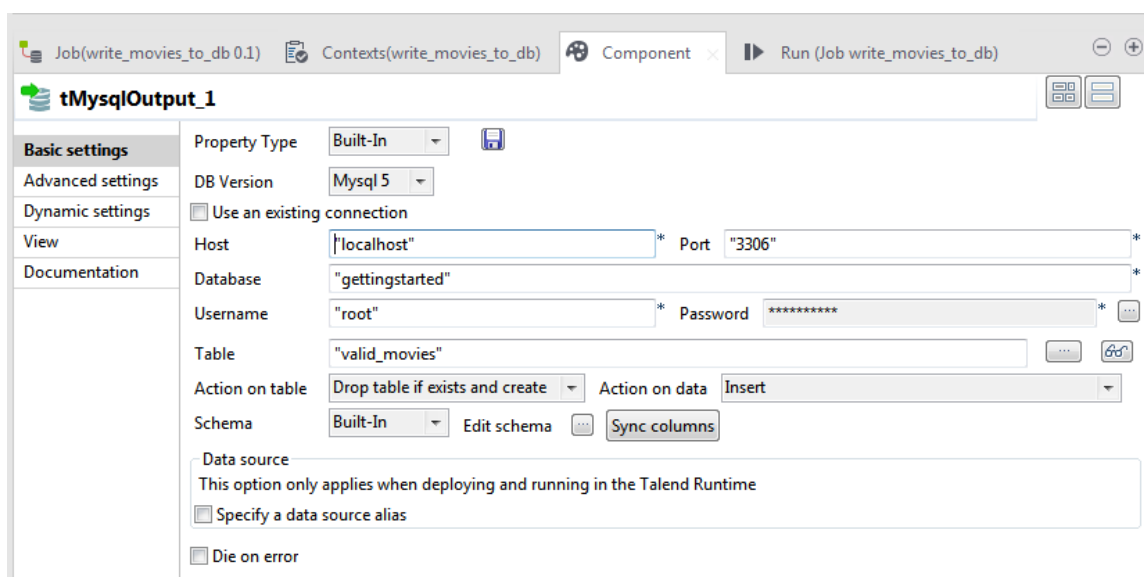
The mapping configurations are saved and the output schema is synchronized to the output component.

Now you have configured mappings for the rejected output. Next, you'll need to configure the output components to write the output flows to database tables.

5.3.3. Configuring MySQL database outputs

This procedure shows how to configure database output components to write movies information to MySQL database tables.

- Double-click the first **tMySQLOutput** component to open its **Basic settings** in the **Component** view.



2. Provide the connection details needed to access your database, including the host name or IP address, port number, database name, user name and password, in the relevant fields.

When entering your password, you need first to click the [...] button next to the **Password** field to open a dialog box, enter your password between double quotation marks in the text field, and then click **OK**.

3. In the **Table** field, enter the name of the target database table.

In this example, the table for valid movies information is *valid_movies*.

4. Select the **Action on table** and **Action on data** options according to your needs.

In this example, we want to remove the table first if it already exists and then create an empty one, and use the default option for the action on data.

5. In the **Basic settings** of the second **tMySQLOutput** component, use the same settings as in the first **tMySQLOutput** except the name for the target database table.

In this example, the table for invalid movies information is *invalid_movies*.

6. Press **F6** or click the **Run** button on the **Run** view to execute your Job.

The movies records with valid director information are saved to the database table named *valid_movies*, and those without valid director information are saved to the database table named *invalid_movies*.

5.4. What's next?

You have seen how *Talend Studio* helps you manage your data using *Talend Jobs*. You have learned how to access your data via *Talend Studio*, filter and transform your data, and store the filtered and transformed data in a database. Along the way, you have learned how to centralize frequently used connections in the **Repository** and easily reuse these connections in your Jobs.

To learn more about *Talend Studio*, see:

- *Talend Studio User Guide*
- *Talend Open Studio Components Reference Guide*

To ensure that your data is clean, you can try *Talend Open Studio for Data Quality* and *Talend Data Preparation*.

To learn more about *Talend* products and solutions, visit www.talend.com.