

## Working with Files

This chapter discusses the following.

Working with Files .....	42
Reading an Input File .....	43
Transforming Data .....	53
Running the Job .....	61
Combining Columns .....	64
Duplicating a Job .....	68
Challenges .....	70
Solutions .....	71
Wrap-Up .....	72
Next step .....	72

## Working with Files

### Lesson Overview

In this lesson you will start working with files.

First you will read a delimited file storing customer information from your local file system. Then you will use a data transformation component to apply a basic change to the data: transform the customer's state name to upper case. In order to test your transformation, you will write the transformed data to an output file.

Finally, you will consolidate your knowledge on data transformation by combining the first name and last name columns into one column containing the full name of the customer.

Here is high-level view of the Job you will build in this lesson.



### Objectives

After completing this lesson, you will be able to:

- Read and Write a Delimited Text File
- Read Warning and Error messages
- Transform data using the tMap component
- Build simple expressions
- Explore data using the Data Viewer in the Studio
- Duplicate an existing Job as the basis for a new Job

### Before you begin

Be sure that you are working in an environment that contains the following:

- A properly installed copy of Talend Studio
- The supporting file for this lesson: **Custs.csv** under **C:/StudentFiles**

### Next Step

The first step is to create a new Job that [reads a delimited file](#)

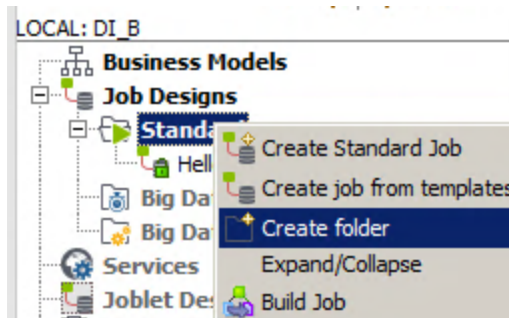
## Reading an Input File

### Overview

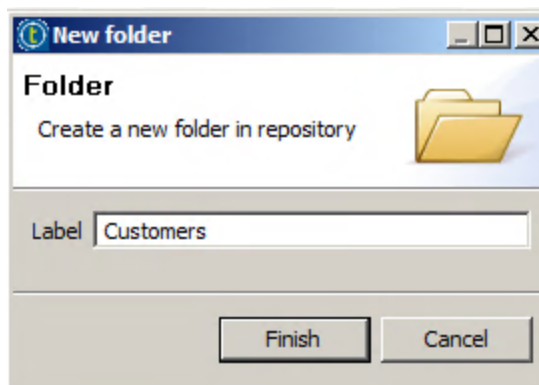
In the first lesson you already created a simple first Job. In this section you will create a new Job that reads data from a CSV delimited file containing customer data. The Job will be created in a specific folder that will contain all Jobs related to the Customer use case.

### Create New Job

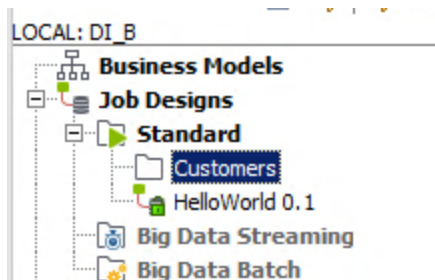
1. Create a new folder for your Jobs. Under **Job Designs**, extend the **[+]** sign in front of **Job Designs**, right click on **Standard** and then select **Create folder**:



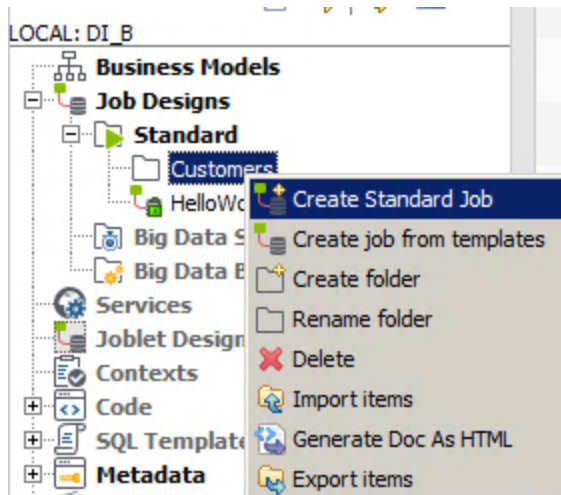
Set the **Label** to *Customers* and then click **Finish** :



The new folder appears under **Job Designs - Standard**:



2. To create the next Job, in the **Repository**, right-click **Customers**, under **Standard** folder from **Job Designs** and then click **Create Standard Job**:



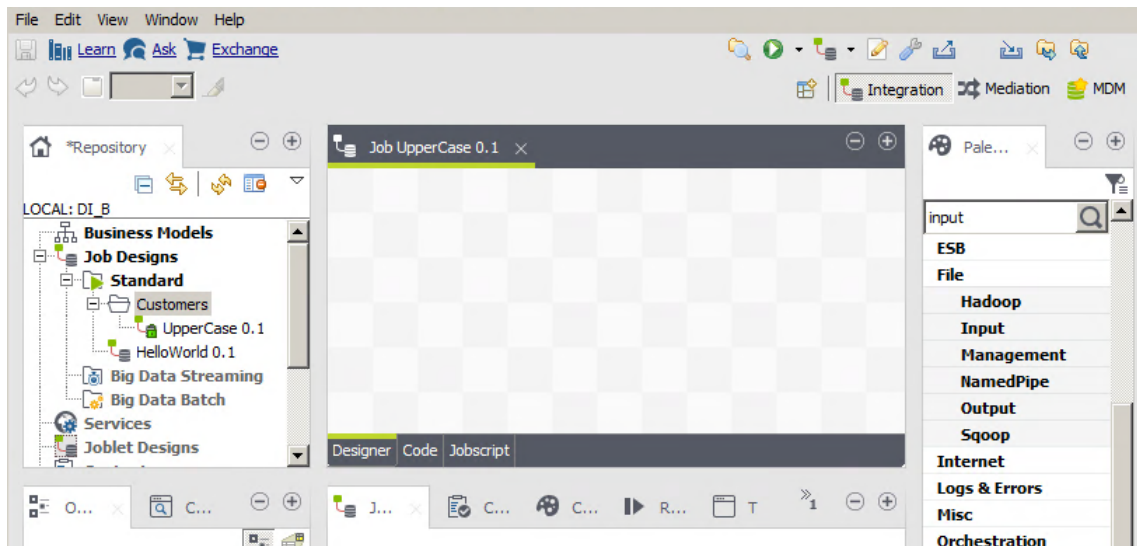
3. Enter *UpperCase* into the **Name** box, and fill in the **Purpose** and **Description** boxes as follows. Then click **Finish**:

Name	UpperCase	
Job Type	Standard	Framework
Purpose	Convert to uppercase	
Description	Reads customer file, converts the state column to uppercase and then writes the result to an output file	
Author	user@talend.com	
Locker	user@talend.com	
Version	0.1	M m
Status		
Path	Customers	

Finish Cancel

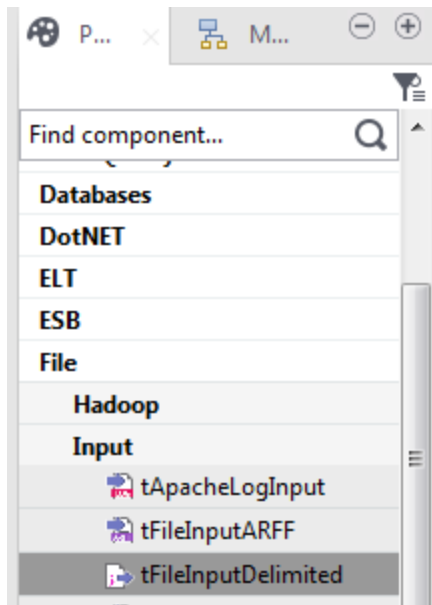
Remember that this Job will convert the text in the State column to upper case. While only the name is required, it is good practice to complete the remaining fields to help document the Job.

4. The new Job opens, ready for you to build:



## Add Read File Component

1. In the **Palette**, click **File**, followed by **Input**, and then scroll down until you locate the component **tFileInputDelimited**:



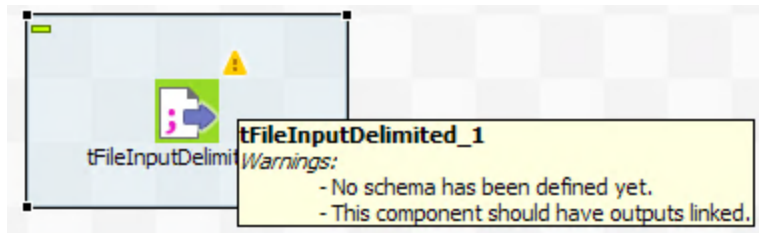
Components in the **Palette** are grouped by function. This particular component reads input from a delimited file.

2. Click **tFileInputDelimited** in the **Palette**, and then click in the design workspace to place the component.



Note the yellow triangle with an exclamation point (!) above the component, indicating a warning.

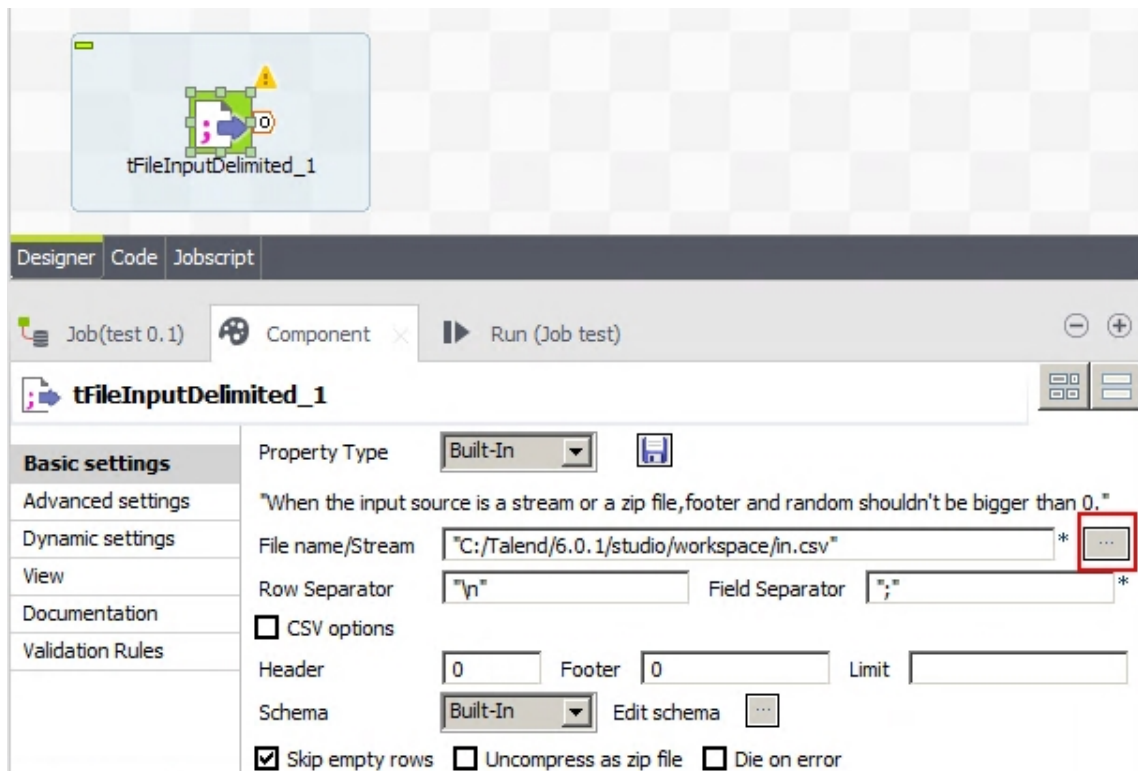
- Place the mouse over it to read the warnings:



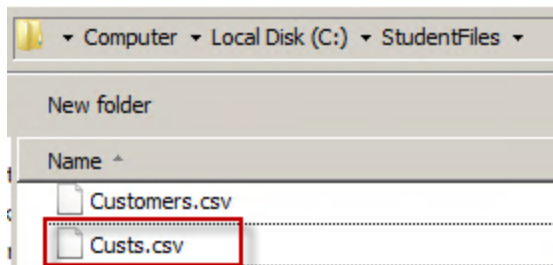
The warning specifies that the component requires a schema and should be attached to another component, items you will provide as you build the Job.

## Configure

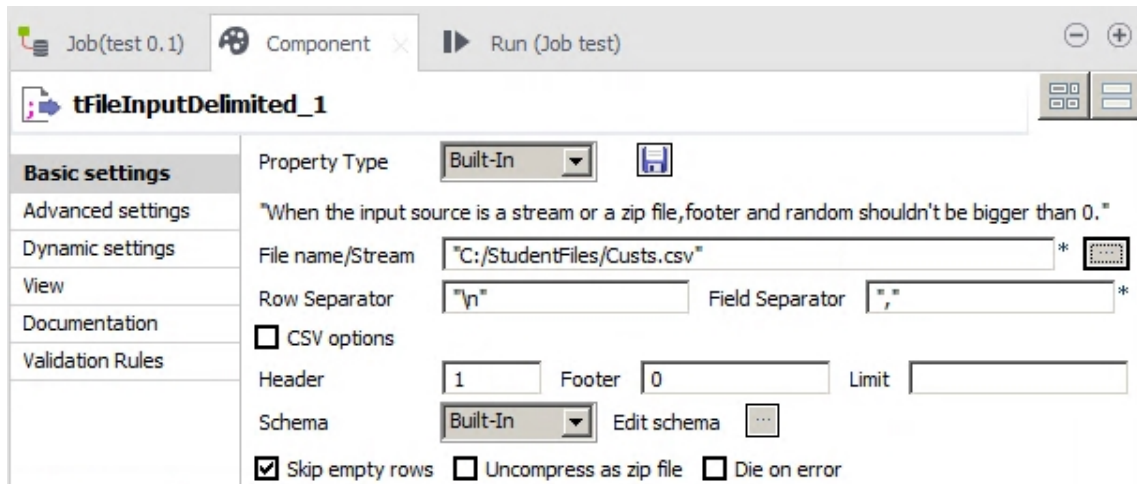
- Double-click **tFileInputDelimited** to open the **Component** view beneath the design workspace. Click the button marked with an ellipsis [...] next to the **File Name / Stream** box, to browse for the input file:



2. Locate the file **C:/StudentFiles/Custs.csv** and then click **Open**:

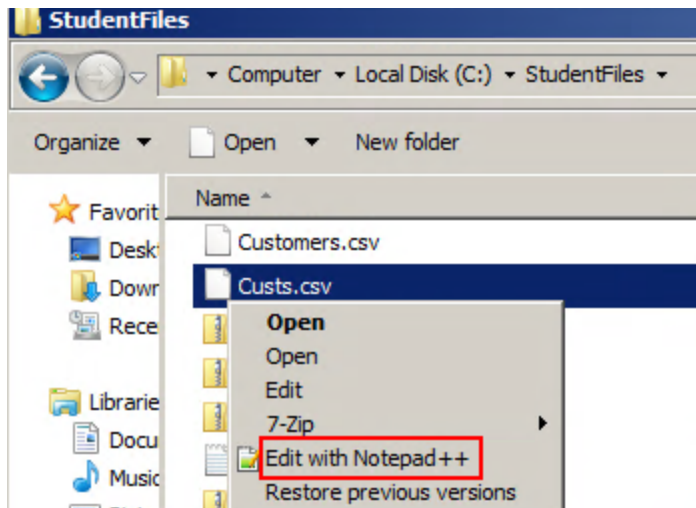


3. In the **Component** view, change the value in the **Field Separator** box to **,** to indicate that a comma separates column values in the file, rather than the default semicolon. Then change the value in the **Header** box to **1** to specify that the first row of the input file contains column names:



**Note:** Here is where you specify all the information necessary for the component to perform its tasks. Note the tabs along the left, such as **Advanced settings** and **View**, that allow you to access additional configuration parameters.

4. Using Windows Explorer, locate the **Custs.csv** file, right click on it and select **Edit with Notepad++**:



5. Explore the file content:

```

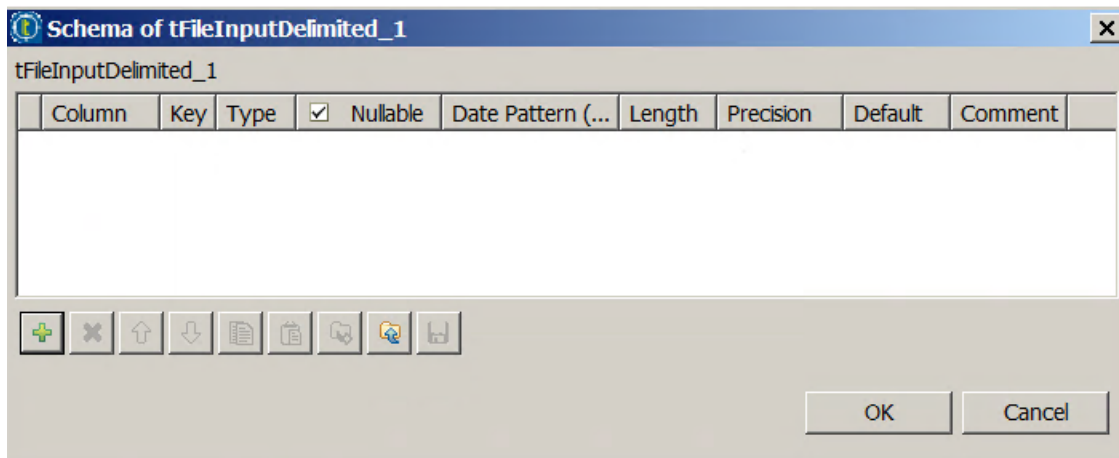
1 First, Last, Number, Street, City, State
2 Bill, Coolidge, 85013, Via Real, Austin, IL
3 Thomas, Coolidge, 63489, Lindbergh Blvd, Springfield, ca
4 Harry, Ford, 97249, Monroe Street, Salt Lake City, ca
5 Warren, McKinley, 82589, Westside Freeway, Concord, ak
6 Andrew, Taylor, 29886, Padre Boulevard, Madison, CA
7 Ulysses, Coolidge, 98646, Bayshore Freeway, Columbus, MN
8 Theodore, Clinton, 12292, San Marcos, Bismarck, NY
9 Benjamin, Jefferson, 82077, Carpinteria North, Sacramento, ca
10 William, Van Buren, 21712, Tully Road East, Albany, IL
11 Calvin, Washington, 50742, Richmond Hill, Charleston, ca
12 Jimmy, Polk, 76143, Richmond Hill, Salt Lake City, AK
13 Calvin, Adams, 52386, Lake Tahoe Blvd., Montgomery, NY
14 Ulysses, Monroe, 70511, Jones Road, Trenton, IL
15 Zachary, Tyler, 45040, Santa Rosa North, Carson City, AK

```

Note that the file separator is comma as you previously configured and that the first line contains the Header. In the Header you can see the list of fields that you'll define as the schema of your file in the following steps.

## Schema

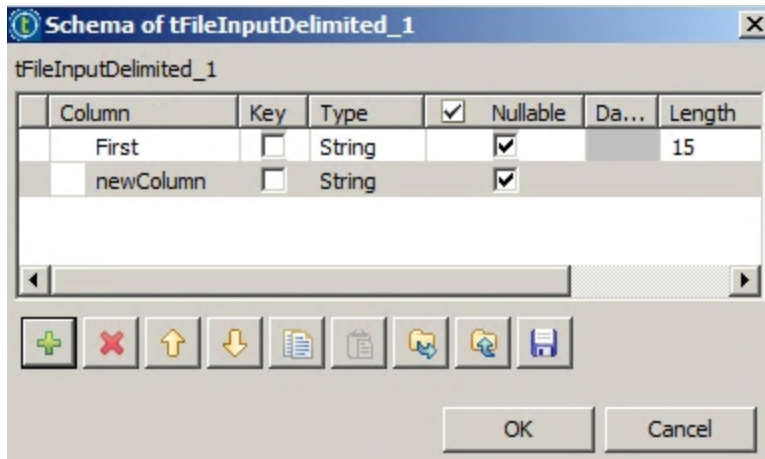
1. Near the bottom of the **Component** view, click the **Edit schema** button marked with an ellipsis [...]. The **Schema** window appears:



**Note:** This is where you specify the format of the data in the *Custs.csv* file. The information about the structure of this particular file is provided for you in this lesson; in your own projects you need to know the schema for the data in your source files.

2. Click the button marked with a **[+]** sign to add a new column to the schema. Replace *newColumn* with *First* and then enter **15** into the **Length** column:





**Note:** This specifies that the first column of each row of data in the file *Custs.csv* contains a 15-character string value, corresponding to the first name of the customer.

3. Add five more columns of type **String** and set the Length as follows:

Last, 15

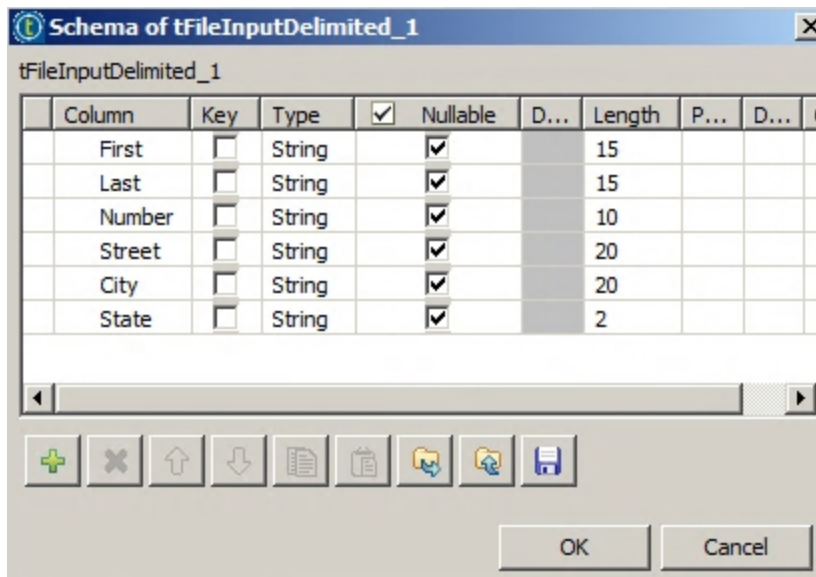
Number, 10

Street, 20

City, 20

State, 2

When finished, your schema should look like this:

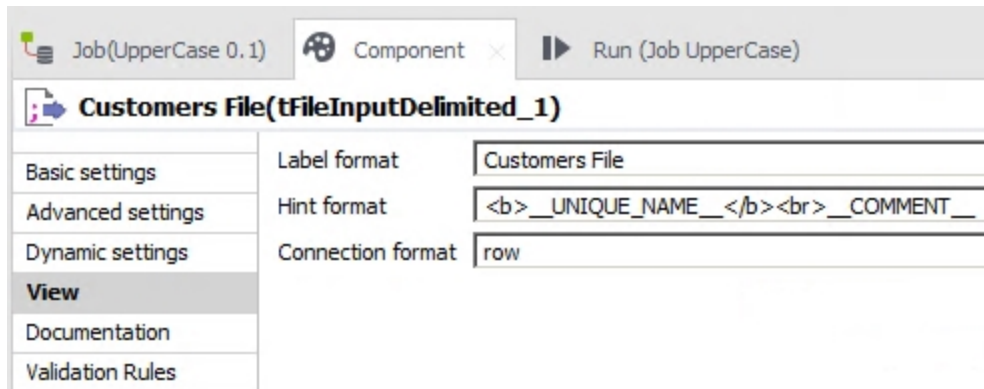


Then click **OK**.

As you can see from this schema, the file **Custs.csv** contains basic identifying and address information for a fictional group of customers organized into six columns.

## Display

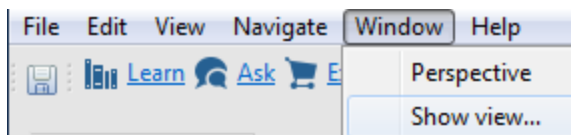
1. Click the **View** section on the left of the **Component** tab and then enter *Customers File* into the **Label format** box:



**Note:** The set of parameters on the **View** tab affects how the component displays in the design workspace. Note that the text you entered here is duplicated as the component label in the design workspace. You can enter HTML tags as well as text in the format boxes to change the style of the label and hint text.

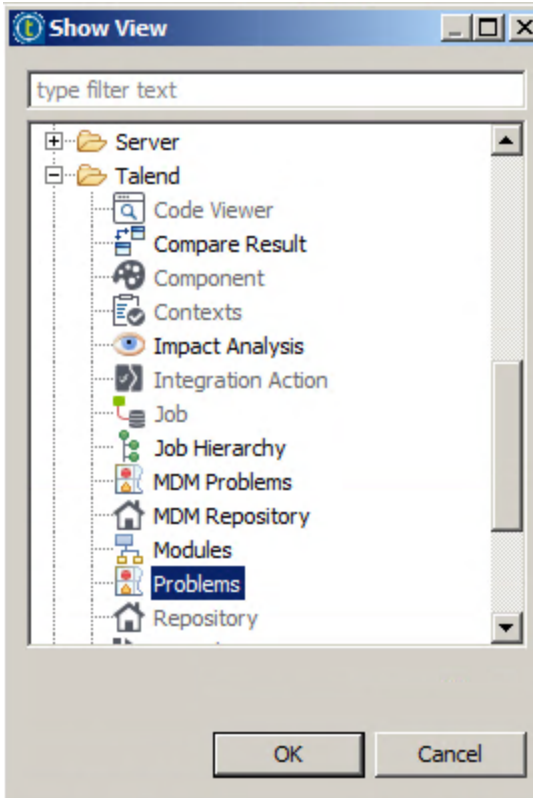
## Warnings

1. On the **Window** menu, click **Show View**:



This is where you choose which views to display in the application.

2. Expand **Talend** and click **Problems**. Then click **OK**:



3. The **Problems** view now appears under the design workspace.

Click to expand **Warnings**:

Description	Resource
Errors (0 items)	
Warnings (1 item)	
⚠ This component should have outputs linked.	Job:UpperCase (component:)
Infos (0 items)	

Note: The **Problems** view is where you can find information about issues with the Job or individual components. In this case, the warning tells you that your input component needs to be linked to some kind of output. Note that one of the warnings you saw earlier is no longer displayed because you defined a schema for the component.

**Next**

The next step is to [send the data from the input file to a component](#) that performs the capitalization.

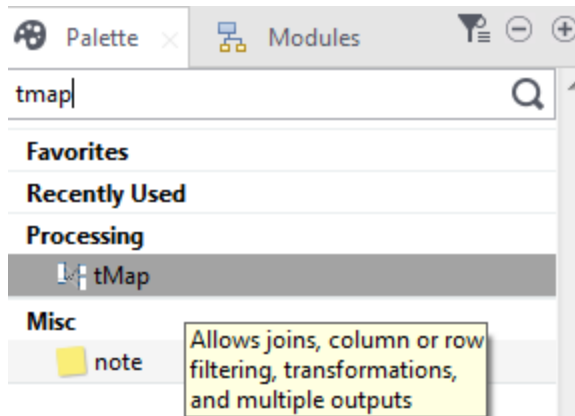
## Transforming Data

### Overview

Your **UpperCase** Job contains a component to provide the input data, now you need to add a component to perform the transformation: converting any lower-case state abbreviation to upper case.

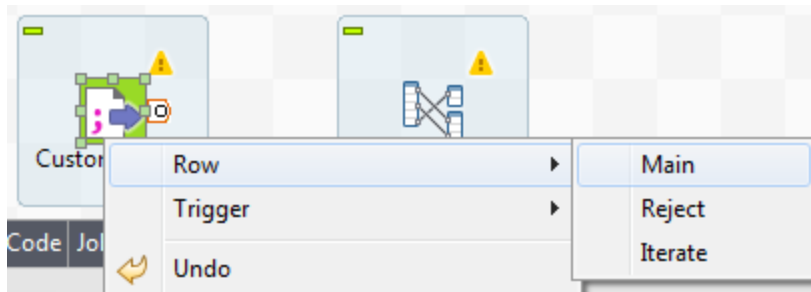
### Add Component

1. Enter **tMap** into the text box near the top of the **Palette** and then either click the magnifying glass button or press Enter:



Note: You can use the **Palette** search feature to locate a component quickly, even if you only remember part of the component name.

2. Place the **tMap** component on the design workspace to the right of the existing component and Right-click the **Customers File** component, click **Row** followed by **Main**, and then click the **tMap** component. This creates a connection between the two components:

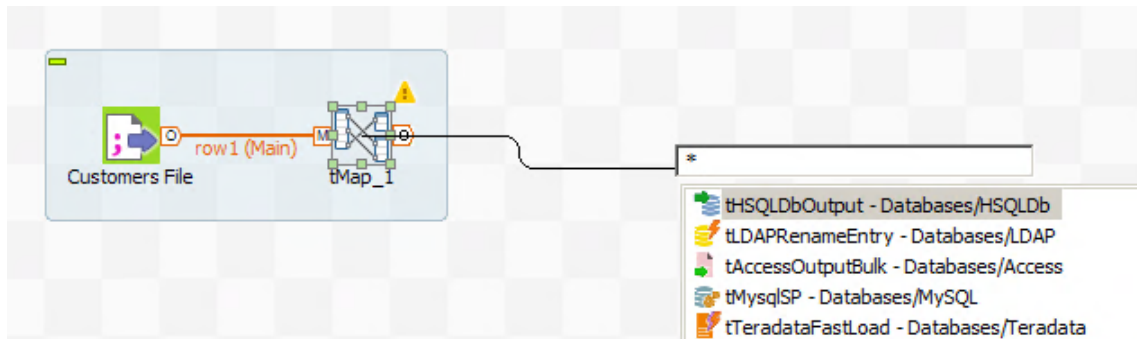


The **tMap** component maps data from input to output, with the capability to perform a variety of transformations on the data. You will find yourself using it as the center piece of almost all of your Talend Data Integration Jobs.

Note the warning displayed for the **tMap** component. The **tMap** component transforms data, so it needs an output destination.

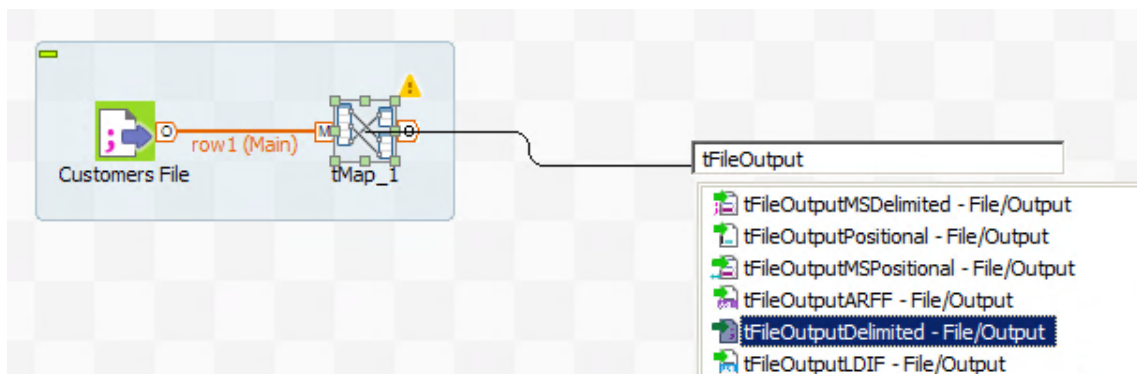
## Add Output Component

1. Right-click the **tMap** component, hold the right-click on and move the mouse outside the component.



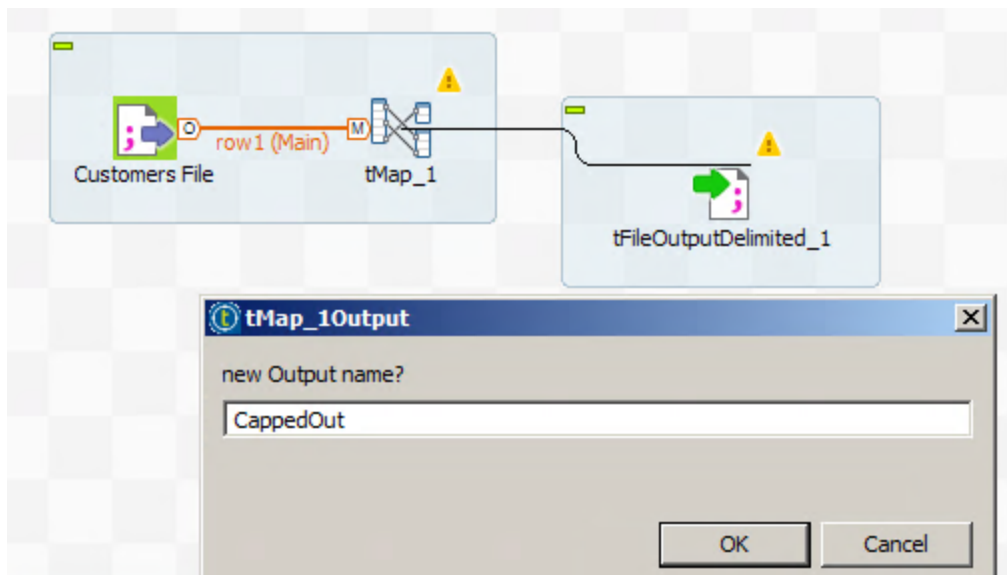
A list of components are then suggested.

Type the first characters from *tFileOutputDelimited* in the search box and then double-click the component to add it to your Job:

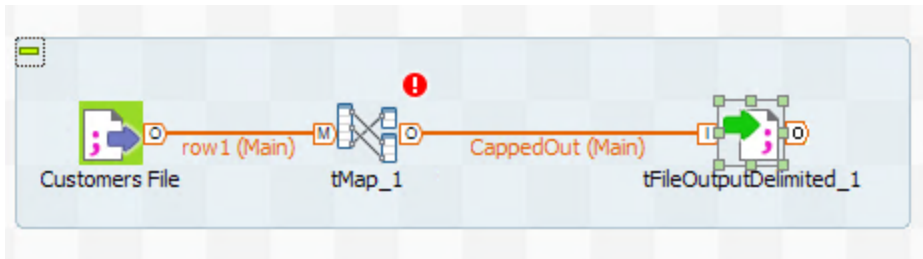


As its name implies, a **tFileOutputDelimited** component writes rows to a file in delimited format.

2. This window asks you to supply a name for the output connection. Enter *CappedOut* (no spaces allowed) into the text box and then click **OK**:



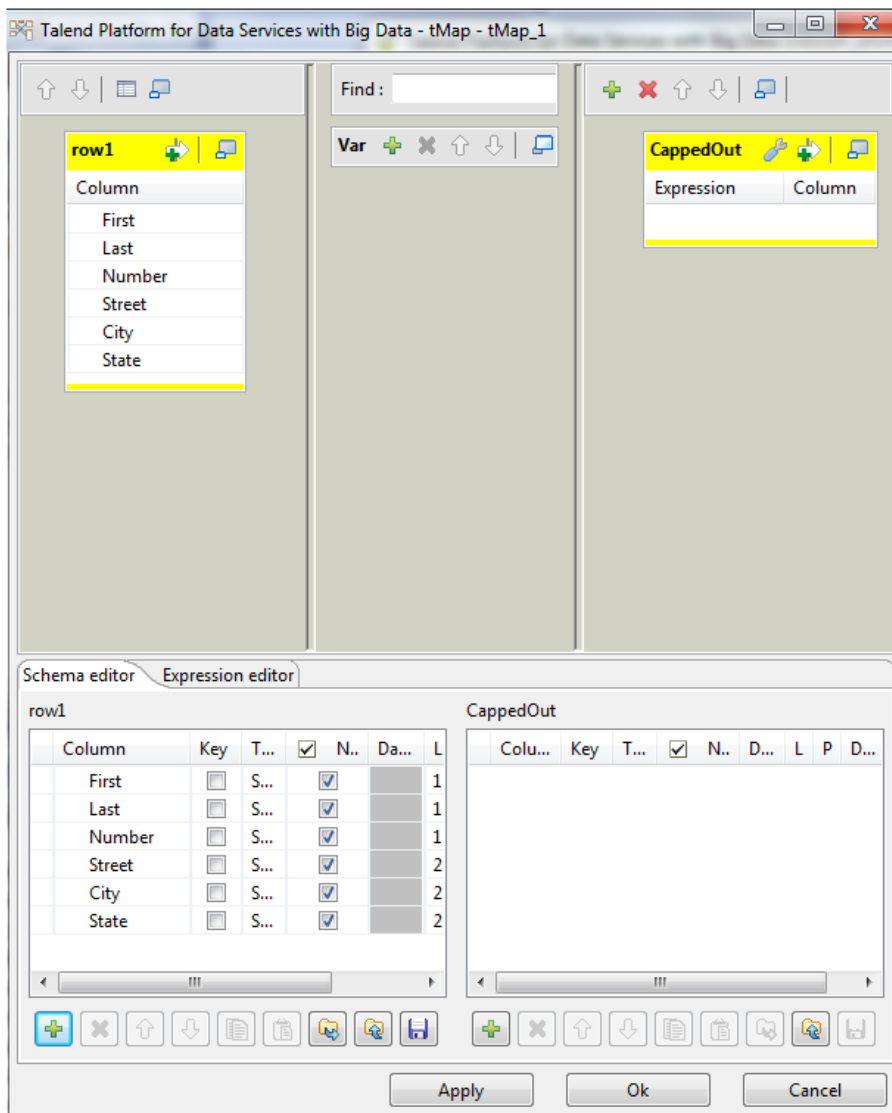
3. The new output connection is a **Main** row, just like the input connection.



Either hover over the **tMap** error or view the **Problems** window to see what the error is. There is an issue with the output schema, which will get resolved next.

## Map Rows

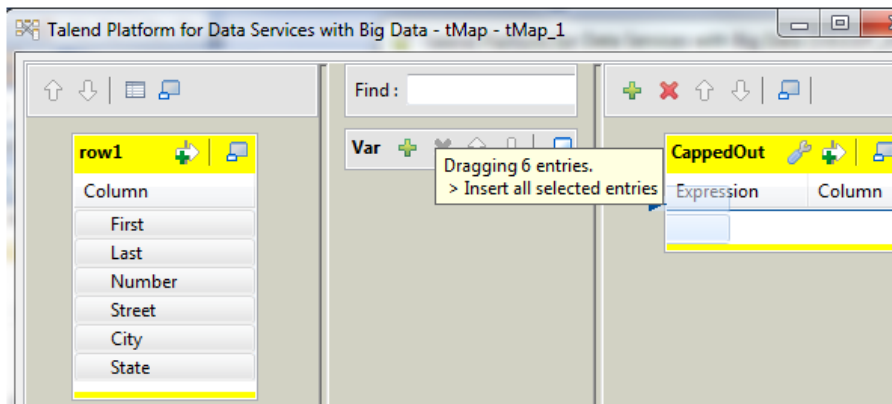
1. Double-click the **tMap** component to open the configuration window:



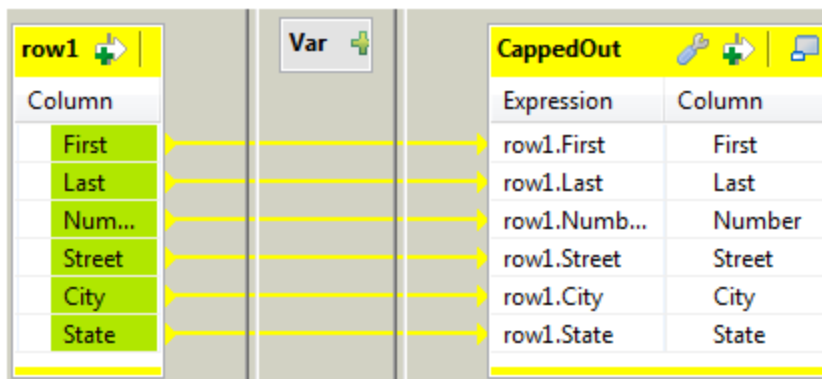
Because the **tMap** component provides so much functionality, this window has many elements. Notice the table on the left, labeled **row1**, and the table on the right labeled **CappedOut**. Those tables represent the schema of the input and output connections, with names that match the rows. The schema for **row1** is copied from the input component and should look familiar. The schema for the new row **CappedOut** is not defined yet.

The goal of this Job is to pass all the data from **row1** to **CappedOut**, simply capitalizing the state abbreviations.

2. In **row1**, click on the top row and then shift-click on the bottom row, to select all the rows.



3. Hold the Shift key and then drag all the rows to **CappedOut**:

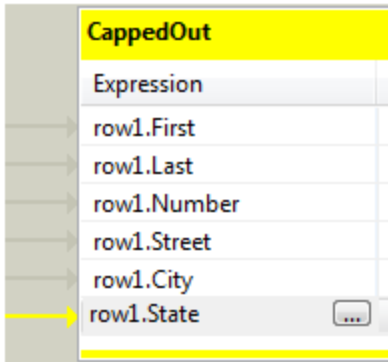


First notice the arrows indicating the mapping of columns from one schema to the other. Next, in the **CappedOut** table, note the **Expression**, which defines the data content, and then the column names. At this point, the data and column names are an exact match to those in **row1** (the expression **row1.<name>** specifies the data in the name column of **row1**).

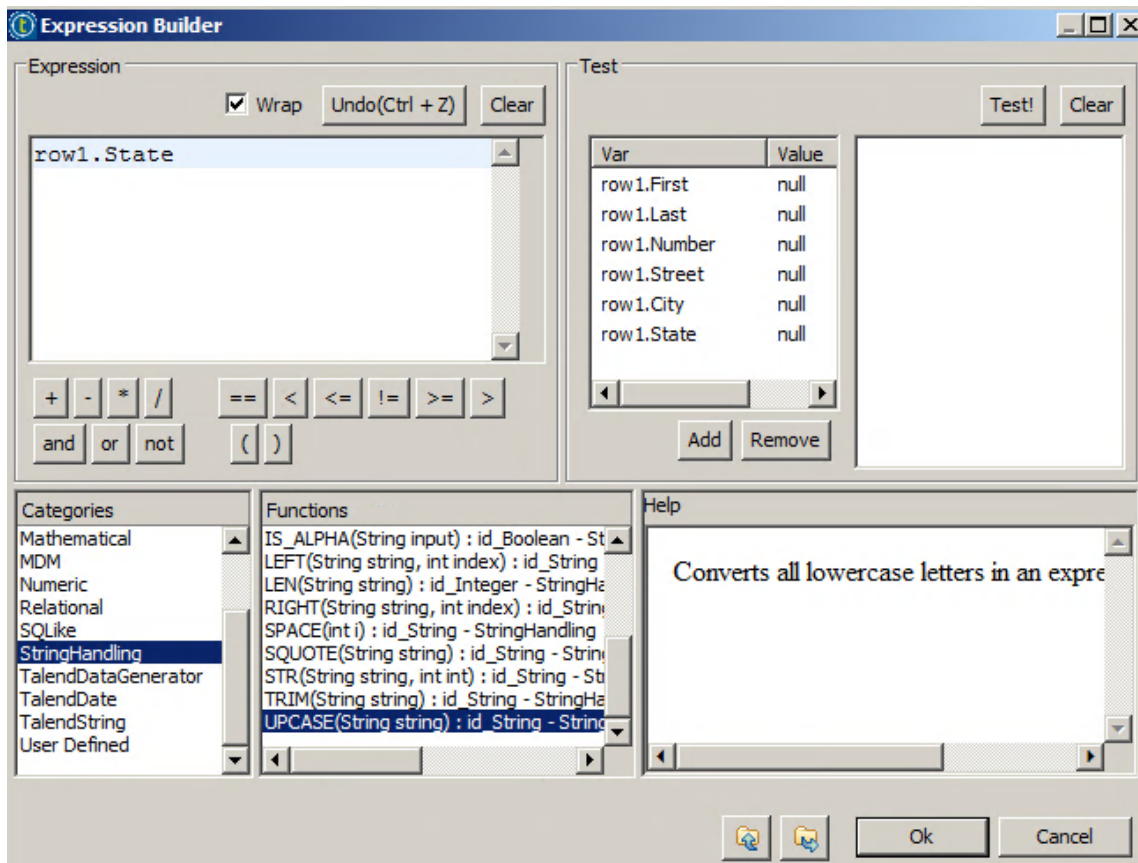
## Build Expression

1. Because you want to modify the data in the **State** column, click **row1.State** in the **CappedOut** output table, and then click the button marked with an ellipsis [...] that appears.



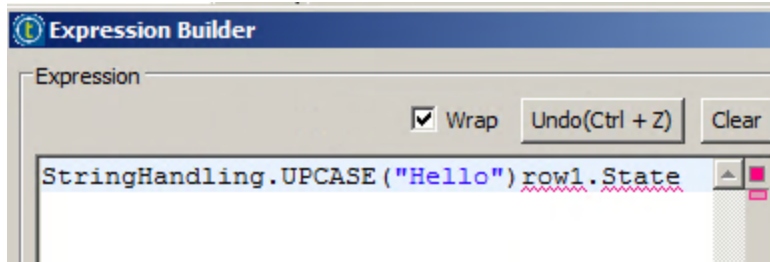


2. In the **Categories** list, click **StringHandling**, and then scroll down in the **Functions** list until you find and click **UPCASE**:



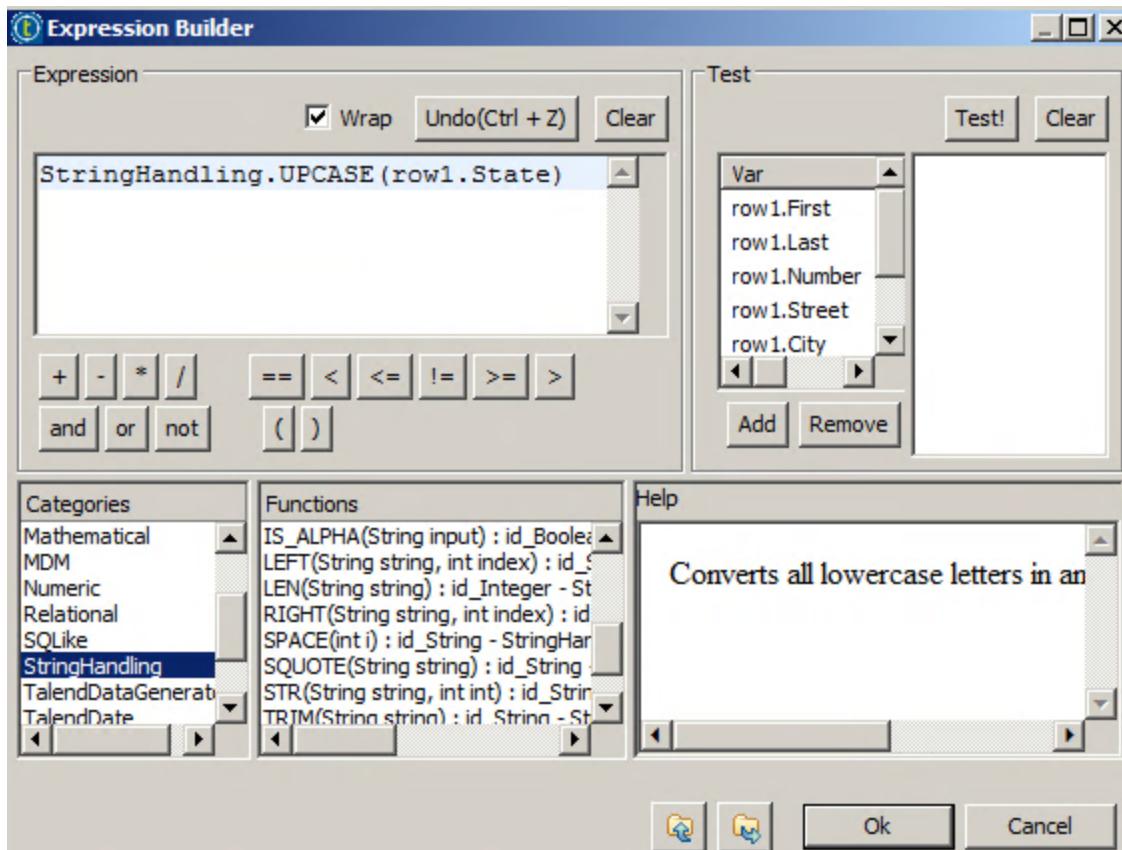
Note: The **Expression Builder** allows you to construct expressions with Java expression syntax. You can just enter the expression as text yourself, but if you are not completely familiar with the syntax, this tool makes the process easier. At this point, the expression is still **row1.State**. This area of the **Expression Builder** provides organized access to a variety of functions. Note the documentation provided for each function in the **Help** area.

3. Double-click **UPCASE** to insert that syntax into the expression at the insertion point (yours may be at the beginning of the expression rather than the end):



Note that the expression is the category followed by a period and then the function name, with arguments enclosed in parentheses. This function converts the string within the parentheses to upper case. As a default example, the function contains the literal string "Hello". What you want to convert is not a literal string, but the contents of the **State** column.

4. Move *row1.State* in the expression so that it replaces "Hello" and click **Ok**:



**Note:** Now the expression acts on the correct argument. Make sure that you deleted the quotes inside the parenthesis!

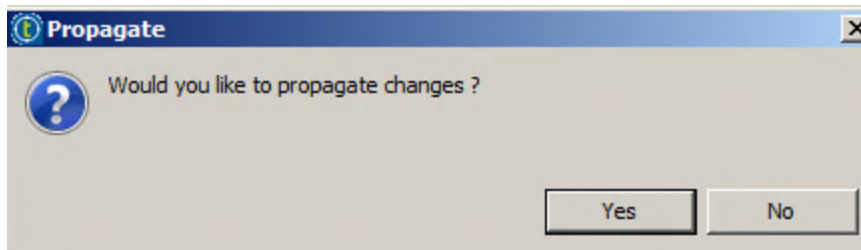
5. The new expression appears in the **CappedOut** table:

CappedOut	
Expression	Column
row1.First	First
row1.Last	Last
row1.Number	Number
row1.Street	Street
row1.City	City
StringHandling.UPCASE(row1.State)	State

Now the output column **State** will contain the string from the column of the same name in the **row1** table, converted to upper case.

## Save

1. You are finished with your **tMap** component configuration, so click **Ok**. A prompt appears:

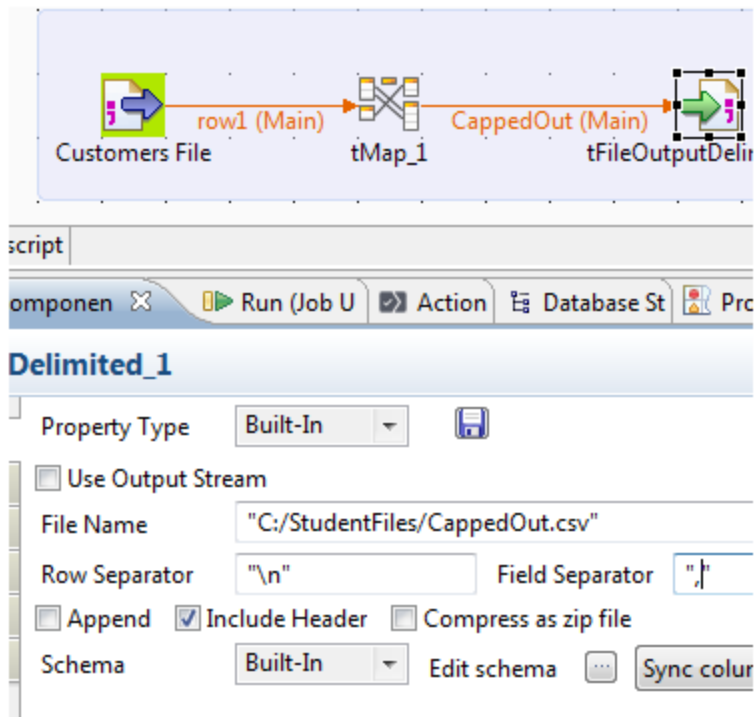


You have made changes to the schema of the output connection, and this prompt asks if you want those changes to be reflected in the schema of the component on the other end of the connection.

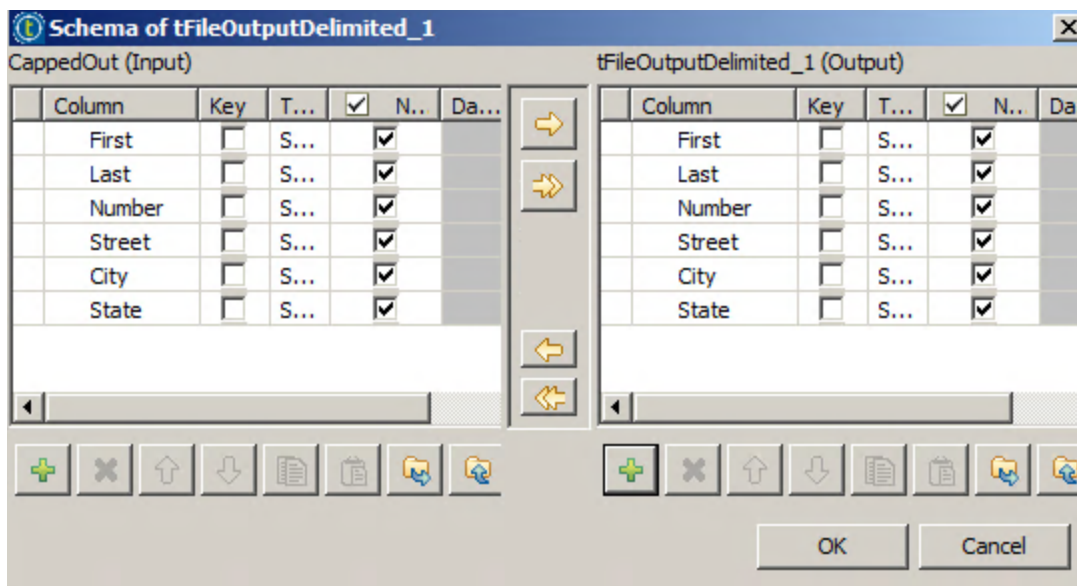
2. Click **Yes**.

## Configure Output

1. Double-click the output component **tFileOutputDelimited** to open the **Component** view.
2. Change the name of the output file to **C:/StudentFiles/CappedOut.csv**, change the **Field Separator** to a comma, and then select the **Include Header** check box to include the column names as the first row of the output file :



3. Click the **Edit schema [...]**(ellipsis) button:



Note that the schema of the component matches the schema you specified in the **tMap** component, which matches the schema of the input connection ( **CappedOut** ).

There is no need to make any changes here, so click **OK**.

## Next

Your Job is complete and you are now [ready to run it](#).

## Running the Job

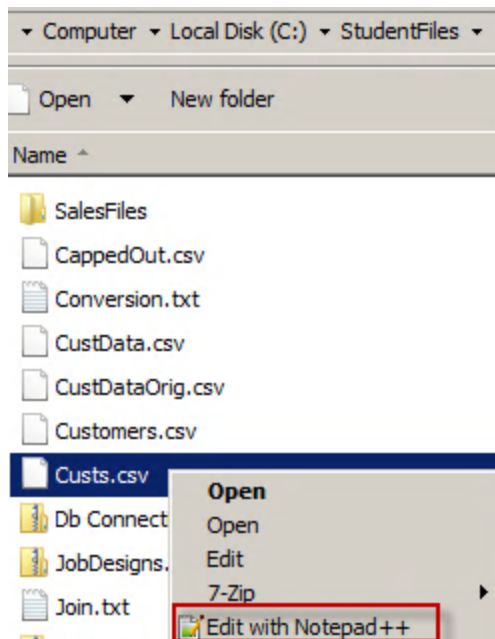
### Overview

The construction of your Job is complete, and now you are ready to run it and see the results.

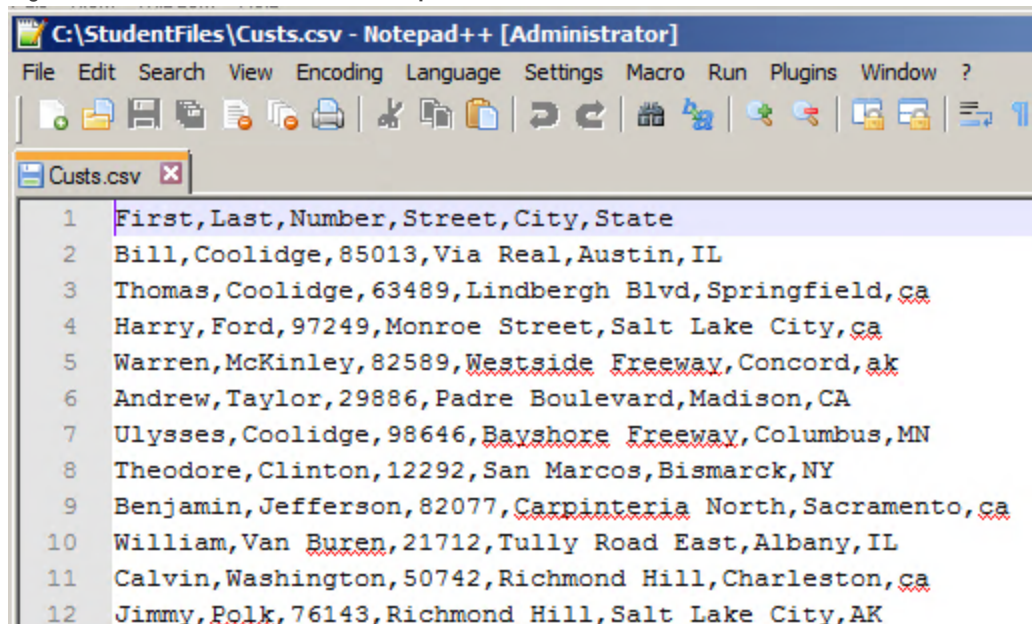
### Examine input file

Before running your Job, examine the input file again..

1. Use the File Browser to locate the **Custs.csv** file in the **C:/StudentFiles** folder :



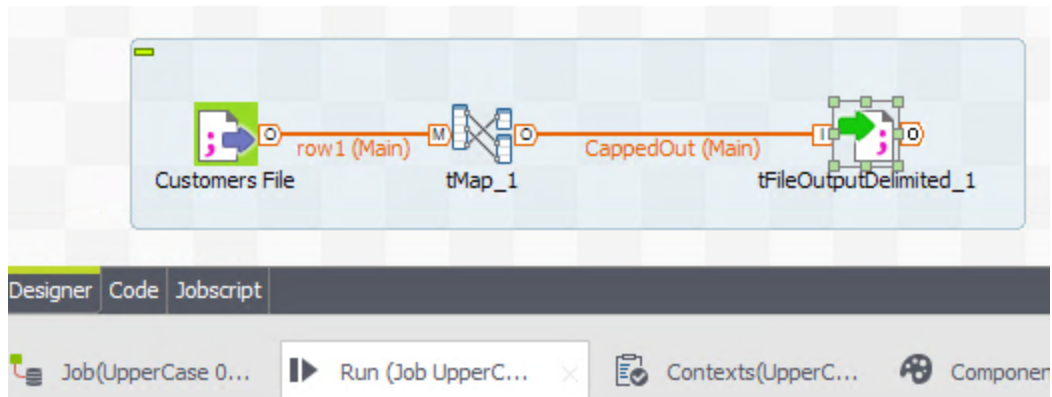
2. Right-click the file and select **Edit with Notepad++**:



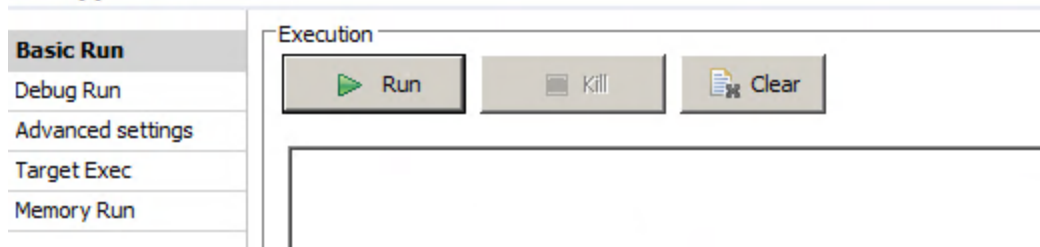
You can notice that several states are in lower case. Your Job is designed to change all states in upper case. Now, you can run your Job and check the output.

## Run

1. Click the **Run** view under the design space:



### Job UpperCase



This is where you execute a Job. Note that in the design workspace, the name of the Job in the tab at the top is preceded by an asterisk, indicating unsaved changes.

2. Click the **Run** button:

Note that statistics about the Job execution display in the design workspace:

The screenshot shows the 'Job UpperCase 0.1' window. The top section displays a data flow diagram with three components: 'Customers File' (input), 'tMap\_1' (transform), and 'tFileOutputDelimited\_1' (output). The flow is labeled 'row1 (Main)' and 'CappedOut (Main)'. Performance metrics show '100 rows in 0.02s' and '6250 rows/s' for both input and output. Below the diagram are tabs for 'Designer', 'Code', and 'Jobscript'. The bottom section, titled 'Job UpperCase', contains a 'Basic Run' tab and an 'Execution' panel. The 'Execution' panel shows a 'Run' button, a 'Kill' button, and a 'Clear' button. The execution log displays the following text:

```
Starting job UpperCase at 02:25 15/02/2016.
[statistics] connecting to socket on port 3534
[statistics] connected
[statistics] disconnected
Job UpperCase ended at 02:25 15/02/2016. [exit code=0]
```

This shows that 100 rows passed through each of the connections.

3. In the File Browser, locate and then open the output file **CappedOut.csv** in **C:/StudentFiles**. Notice that all state abbreviations are now in uppercase and then close the file:

```
First, Last, Number, Street, City, State
Bill, Coolidge, 85013, Via Real, Austin, IL
Thomas, Coolidge, 63489, Lindbergh Blvd, Springfield, CA
Harry, Ford, 97249, Monroe Street, Salt Lake City, CA
Warren, McKinley, 82589, Westside Freeway, Concord, AK
Andrew, Taylor, 29886, Padre Boulevard, Madison, CA
Ulysses, Coolidge, 98646, Bayshore Freeway, Columbus, MN
Theodore, Clinton, 12292, San Marcos, Bismarck, NY
Benjamin, Jefferson, 82077, Carpinteria North, Sacramento, CA
William, Van Buren, 21712, Tully Road East, Albany, IL
Calvin, Washington, 50742, Richmond Hill, Charleston, CA
Jimmy, Polk, 76143, Richmond Hill, Salt Lake City, AK
```

## Next

As mentioned earlier, the **tMap** component provides a great deal of transformation functionality, so your next step is to [explore some different kinds of transformations](#).

## Combining Columns

### Overview

The **tMap** component provides the capability to perform a variety of data mapping and transformation functions. Imagine that your destination data store requires the first and last names to be combined in a single column. Follow the steps below to achieve that.










### Edit Schema

1. Double-click the **tMap** component.
2. Click the **row1.Last** expression in the **CappedOut** table:

CappedOut		
Expression		Column
row1.First		First
row1.Last	...	Last
row1.Number		Number
row1.Street		Street
row1.City		City
StringHandling.UPCASE(row1.State)		State

In the **Schema editor** at the bottom of the window, notice that the same column is selected:

CappedOut			
	Column	Key	Type
	First	<input type="checkbox"/>	String
	Last	<input type="checkbox"/>	String
	Number	<input type="checkbox"/>	String
	Street	<input type="checkbox"/>	String
	City	<input type="checkbox"/>	String
	State	<input type="checkbox"/>	String



Because you are going to combine the data from two columns into a single column, this column is no longer necessary.

3. Click the **Remove selected items** button, marked with a red X (highlighted in the previous print screen ), to delete the **Last** column from the output schema:

**Important !** Make sure to click **Remove selected items** in the Schema Editor of CappedOut (lower right part of the Map Editor) and not the Remove table red X (in the upper right part of the Map Editor). If you click Remove Table, you will delete your mapping and will have to rebuild it from the beginning.

4. The column has disappeared from the **CappedOut** table at the top of the window as well.



## Combine Columns

1. Drag the column **Last** from the **row1** table on the left, dropping it onto **First** in the **CappedOut** table:

CappedOut	
Expression	Column
row1.First row1.Last	First
row1.Number	Number

Note that the expression now contains references to two columns from the **row1** table.

2. Insert a plus sign (+) between the two references:

CappedOut	
Expression	Column
row1.First + row1.Last	First
row1.Number	Number

Because the two columns contain strings, the plus sign concatenates the values.

3. To include a space between the first and last names, insert " " + after the first plus sign. Press the **Enter** key on your keyboard to save this expression :

CappedOut	
Expression	Column
row1.First + " " + row1.Last	First

4. In the **Schema editor** at the bottom of the window, click **First** for the **CappedOut** table and then change it to *Name*:

CappedOut						
	Column	Key	Type	<input checked="" type="checkbox"/> N..	Date ...	Length
	Name	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>		30
	Number	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>		10
	Street	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>		20

Then update the length of the **Name** field from 15 to 30.

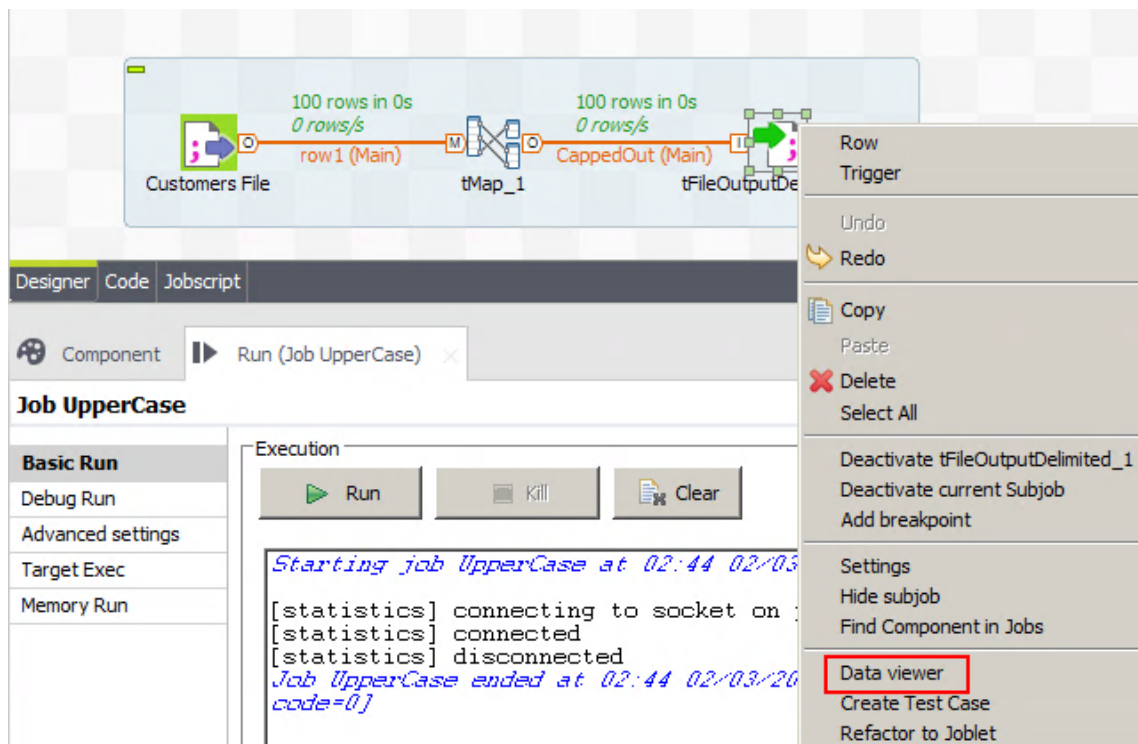
Now the schema column name in the **CappedOut** table is **Name** rather than **First**:

Expression	Column
row1.First + " " + row1.Last	Name

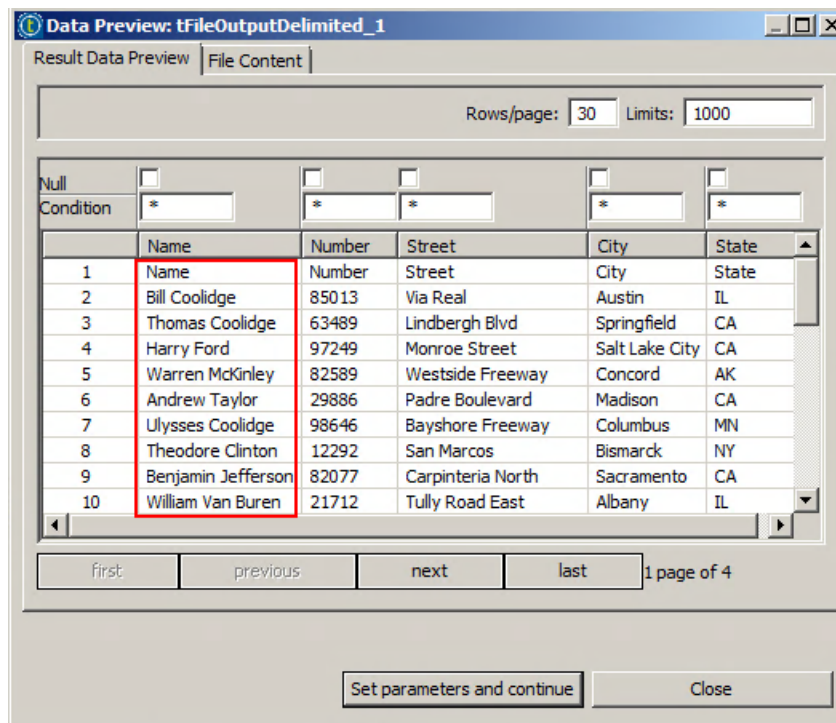
5. Click **Ok** to save your changes, and then propagate the changes when prompted.

## Run

1. In order to Run the Job you can either use the **Run** button or press the **F6** key on your keyboard.
2. Right-click the **tFileOutputDelimited** component and select **Data viewer**:

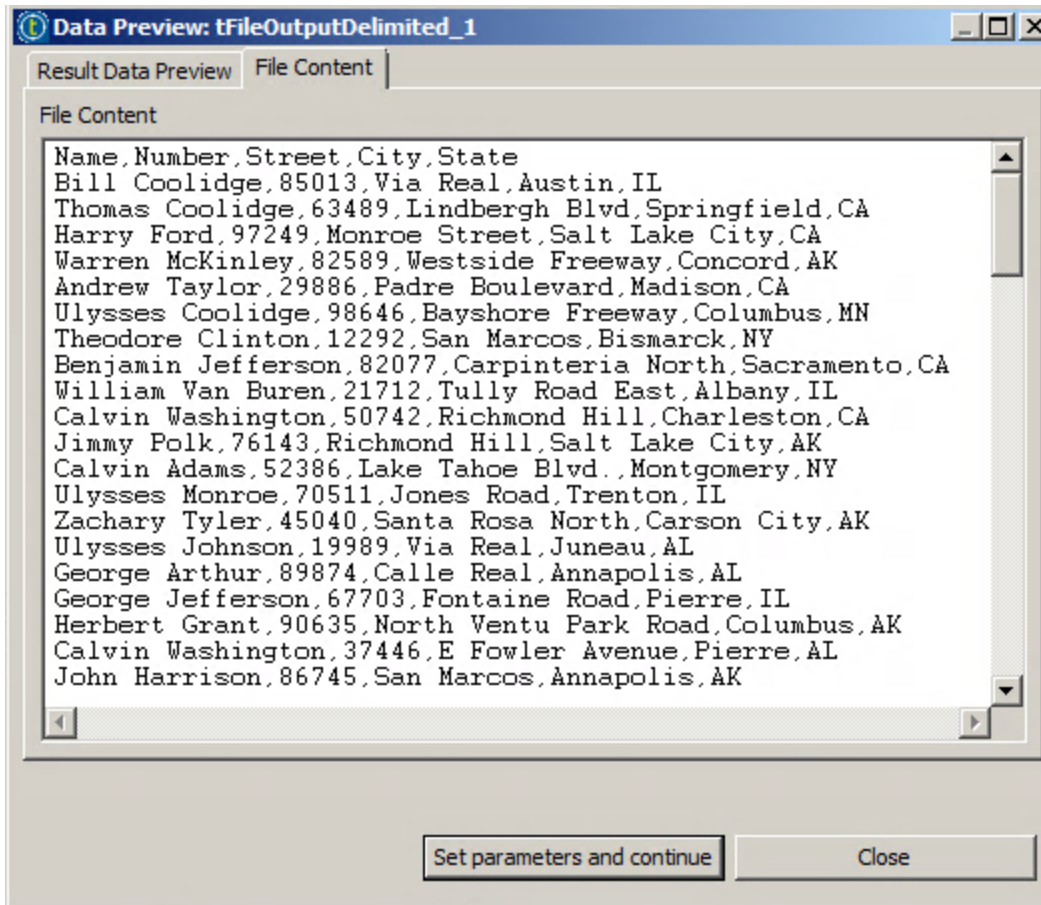


3. Explore the content of the output file in the **Data Preview** window:



Notice that the first column is now **Name** and is the combination of **First** and **Last** columns of the **Custs.csv** file separated by a space (as specified in the Expression Builder).

4. Click on the **File Content** tab to see the raw format of the file:



Click **Close**.

## Next

You have now finished this Job. Next you will [Duplicate the Job](#), modify it and then execute the new Job.

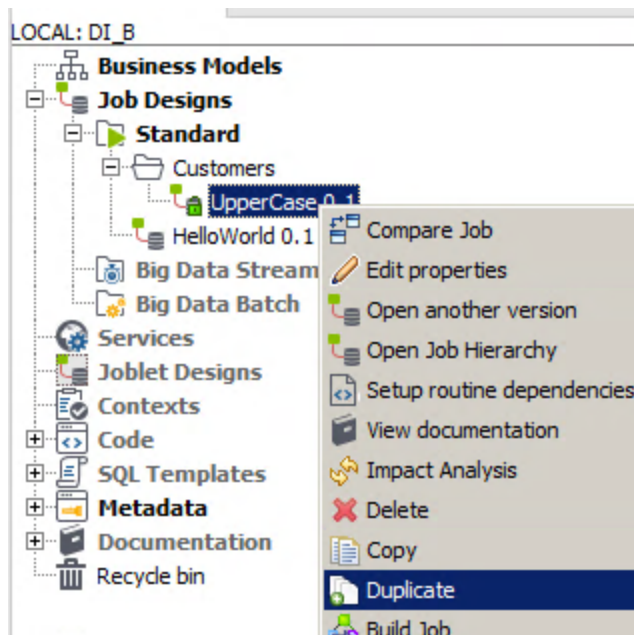
## Duplicating a Job

### Overview

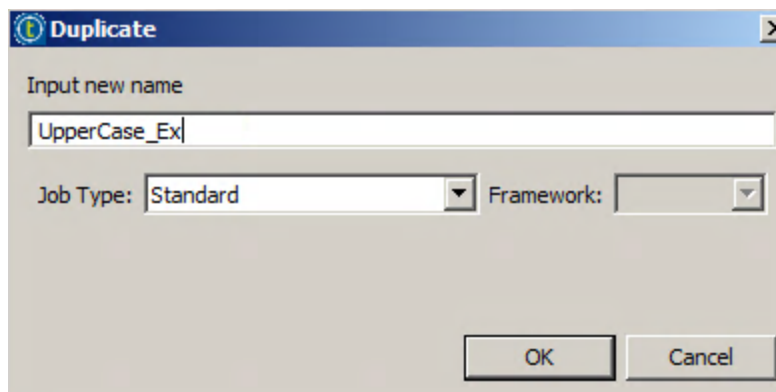
In the following Exercises you will be build an extension of the previous Job. Rather than starting from scratch with a new Job, or modifying the existing Job so that it no longer performs its intended function, you can duplicate it as the basis for the coming Exercises.

### Duplicate

1. In the **Repository**, under **Job Designs**, right-click **UpperCase** and then click **Duplicate**:



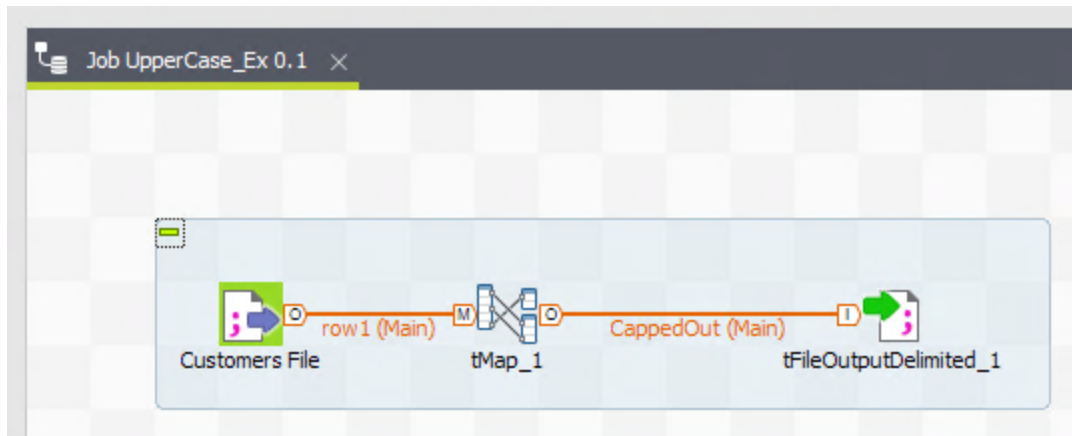
2. Enter **UpperCase\_Ex** for the **Input new name** and click **OK**:



Note: The new Job appears in the **Repository** but does not open.

## Open

1. Double-click **UpperCase\_Ex** in the **Repository** to open it:



## Next

You have now finished the lesson. [Complete the exercises](#) to reinforce your understanding of the topics covered.

## Challenges

### Overview

Complete these exercises to further explore the use of **tMap** in transforming data. See [Solutions](#) for possible solutions to these exercises.

### Combine Columns

Modify the **UpperCase\_Ex** Job previously created so that the output combines the number and street values into a single column named **Address**.

**Hint:** Review the [Combining Columns](#) lesson to see how you performed this action before.

### Add a Column

Modify the Job again, adding a new *integer* column named *id* to the output that contains an automatically generated index value.

**Hint:** After you add the column from the **tMap** configuration window, open the **Expression Builder** and then examine the functions in the **Numeric** category for one that generates a sequential number.

### Add a Zip Code Column

Modify the **tMap** configuration so that the output contains a new column named **Zip** containing a randomly generated five-digit number.

## Solutions

### Overview

These are possible solutions to the [challenges](#). Note that your solutions may differ and still be valid.

### Combine Columns

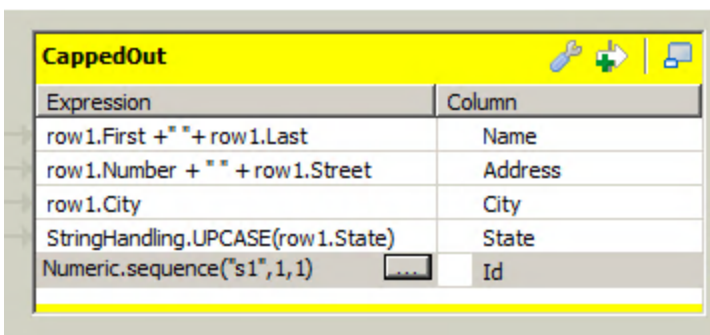
Configure the **CappedOut** table in the **tMap** component, deleting the **Number** column, and replacing the **Street** column with a column named **Address** that contains the following:

```
row1.Number + " " + row1.Street
```

### Add a Column

Add a column of type **Int** with the name *Id* to the **CappedOut** table containing the following expression:

```
Numeric.sequence("s1",1,1)
```



Expression	Column
row1.First + " " + row1.Last	Name
row1.Number + " " + row1.Street	Address
row1.City	City
StringHandling.UPCASE(row1.State)	State
Numeric.sequence("s1",1,1)	Id

### Add a Zip Code Column

Click the **CappedOut** table and then add a column in the **Schema editor** of type **int** named **Zip**. Open the **Expression Builder** for the new column expression and then create an expression like the following:

```
Numeric.random(10000,99999)
```

### Run

Run your Job and then open **CappedOut.csv** to check the result.

### Next

You have now completed this lesson. It's now time to [Wrap-Up](#)

## Wrap-Up

In this lesson you learned how to read a delimited file and how to define the related schema. You also learned how to apply simple transformations on the input data and write them into an output file.

## Next step

Congratulations! You have successfully completed this lesson. To save your progress, click **Check your status with this unit** below. To go to the next lesson, on the next screen, click **Completed. Let's continue >**.