LESSON 3

Joining Data Sources

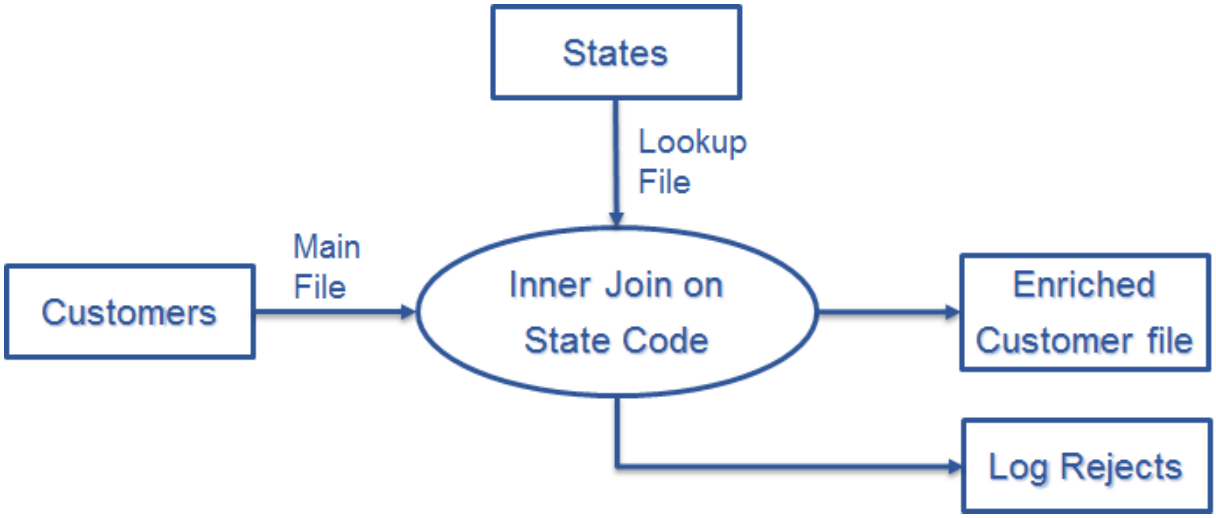
This chapter discusses the following.

Joining Data Sources

Lesson Overview

This lesson provides you with practice joining data from multiple sources. Most enterprises have data in multiple locations and need to combine that data, either to store it in a unified format or to process it consistently. The example you build in this lesson is based on the previous example that capitalizes US state codes / abbreviations, adding a file containing a list of US state codes and names as a lookup table so that the output contains both the abbreviation and the full name.

A common column (the State Code) is used for the **Join** between the two input sources.



Objectives

After completing this lesson, you will be able to:

- Store metadata centrally for use in other components and Jobs
- Use metadata
- Join two data sources
- Troubleshoot a join by examining rejects
- Log data rows rejected in the console

Before You Begin

Be sure that you are working in an environment that contains the lab files for this exercise in **C:/StudentFiles/** which are **Custs.csv** and **States.txt**.

Next Step

The first step will be to Create Metadata for the States file.

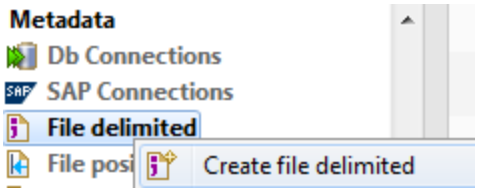
Creating Metadata

Overview

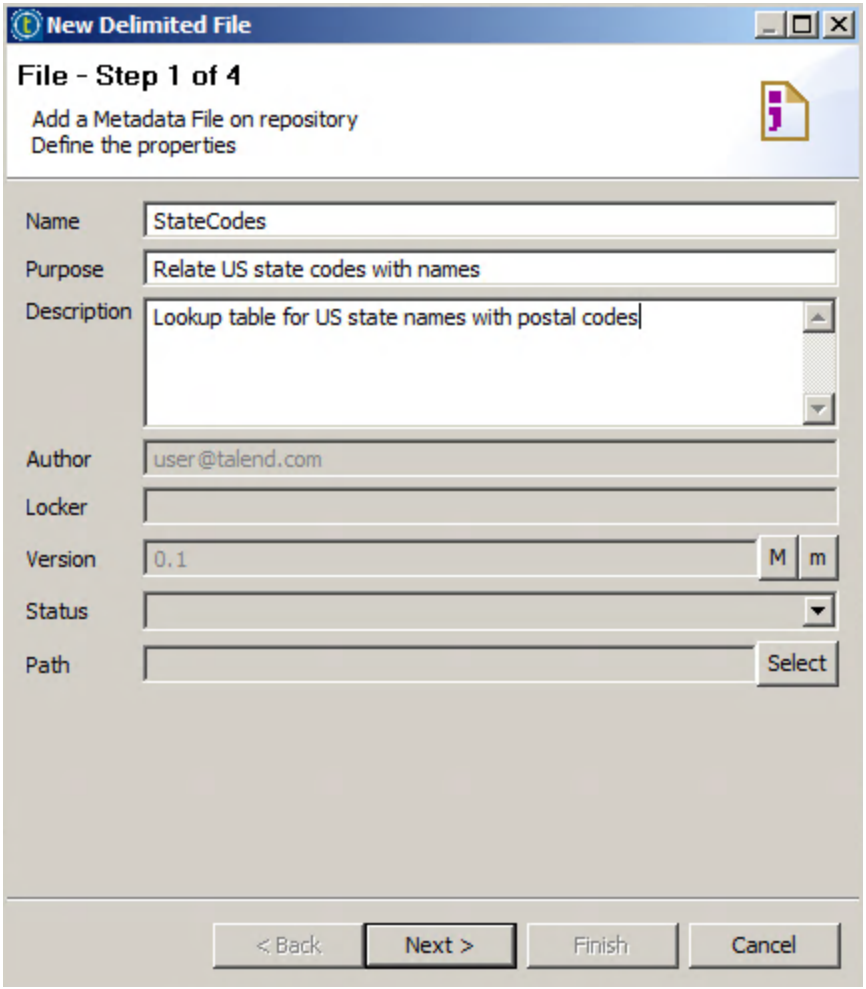
You previously configured a file input component with information about the source file and its schema. That information was local to that particular component. With Talend Studio, you can store such configuration information as **Metadata** so that you can reuse it for multiple components, whether in the same or different Jobs. You are creating information about a delimited file containing US state codes and names so that you can use the same information for multiple components in multiple Jobs.

Create Metadata

1. In the **Repository**, expand **Metadata** then Right-click **File delimited** , and then click **Create file delimited**:



2. Enter *StateCodes* in the **Name** box, and appropriate descriptions into the **Purpose** and **Description** boxes:

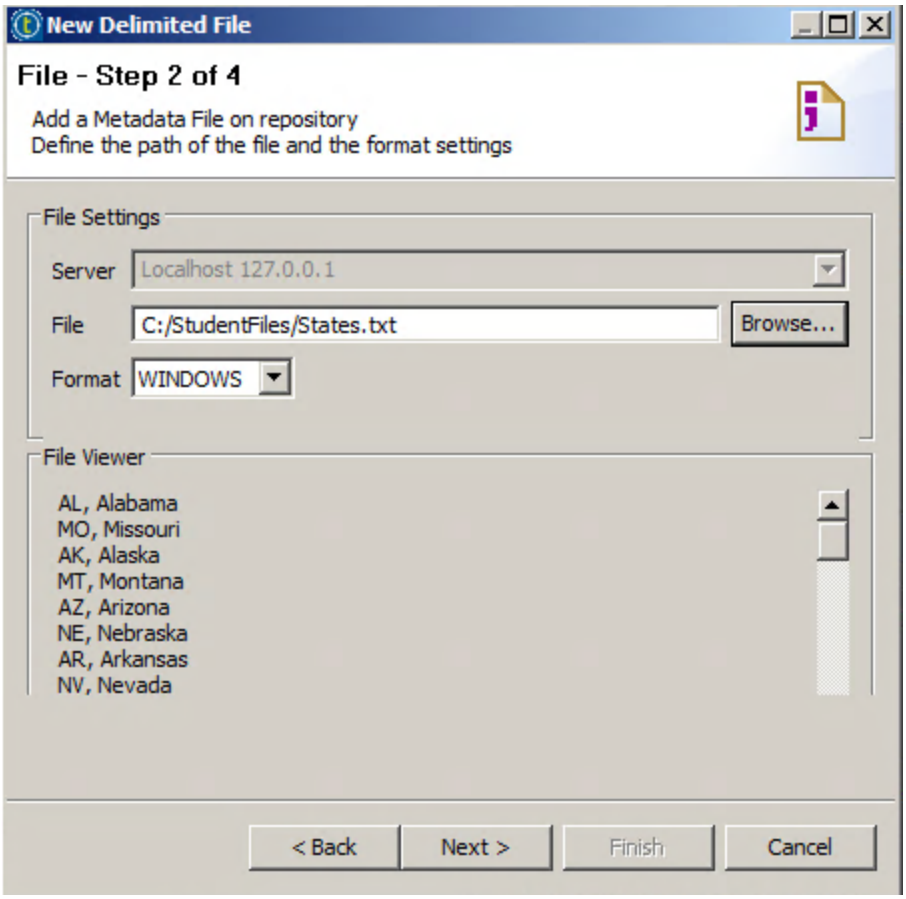


Note: While it is tempting to move quickly and only provide the required name, it is always a good idea to provide documentation about your Job and project elements so that others can understand their purpose.

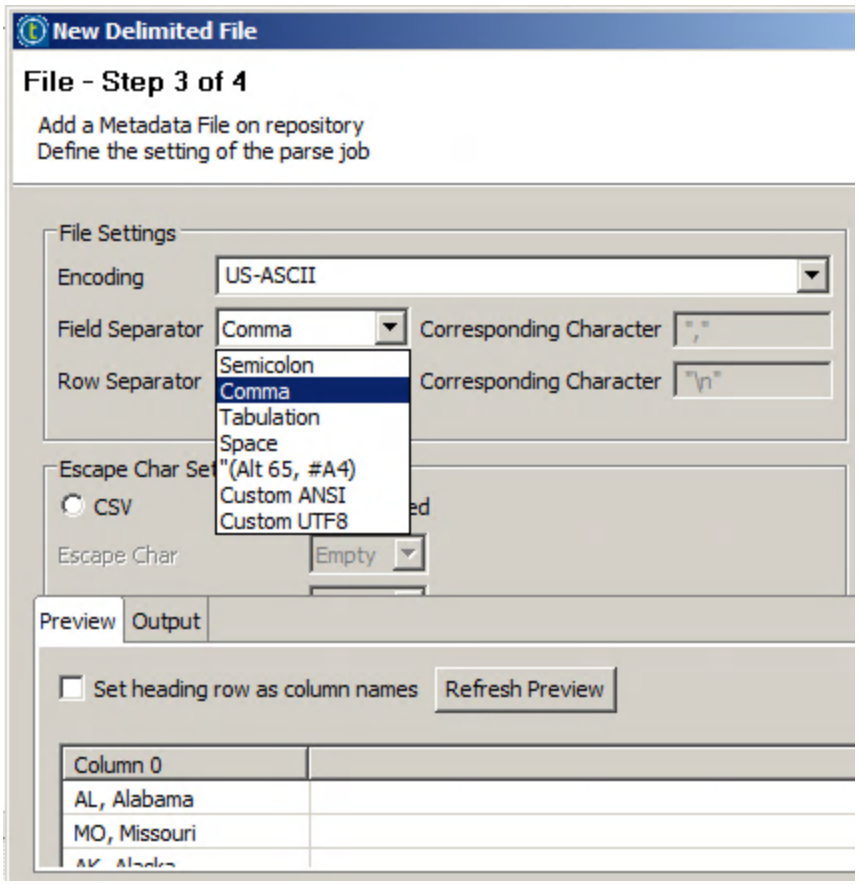
Then click **Next**.

3. In the second step, click **Browse**, select ***.txt** in the file type list, choose the file **States.txt** under **C:/StudentFiles** and then click **Open**.

   Change the **Format** to **WINDOWS** and then click **Next**:

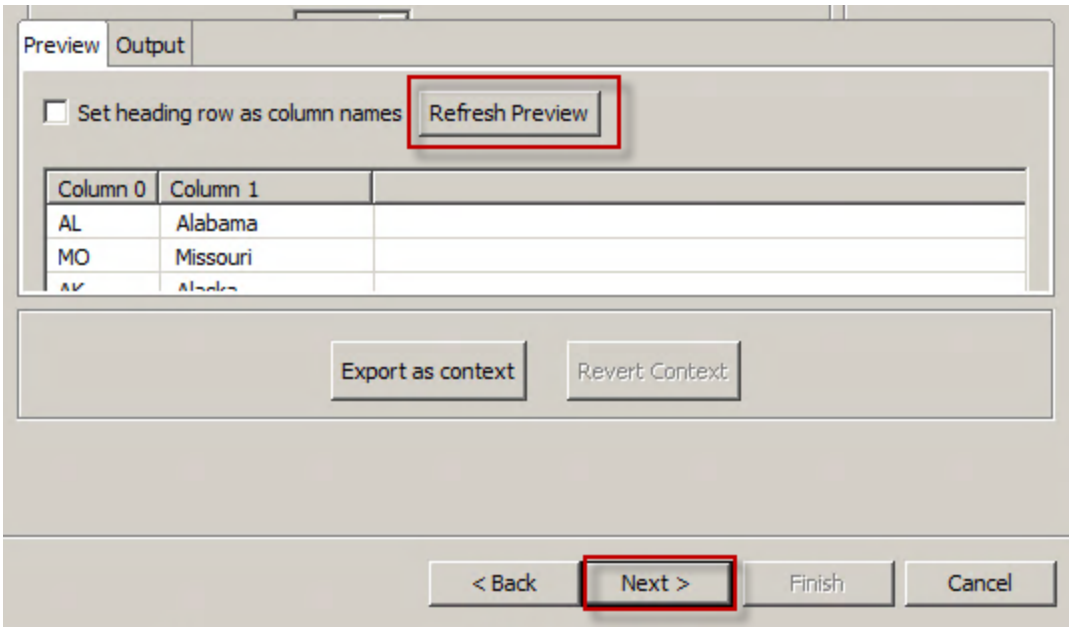


4. This page is where you provide configuration information about the file. Click **Comma** in the **Field separator** list to specify that the fields are delimited with a comma. Leave the other settings as they are, because this file has no header row:



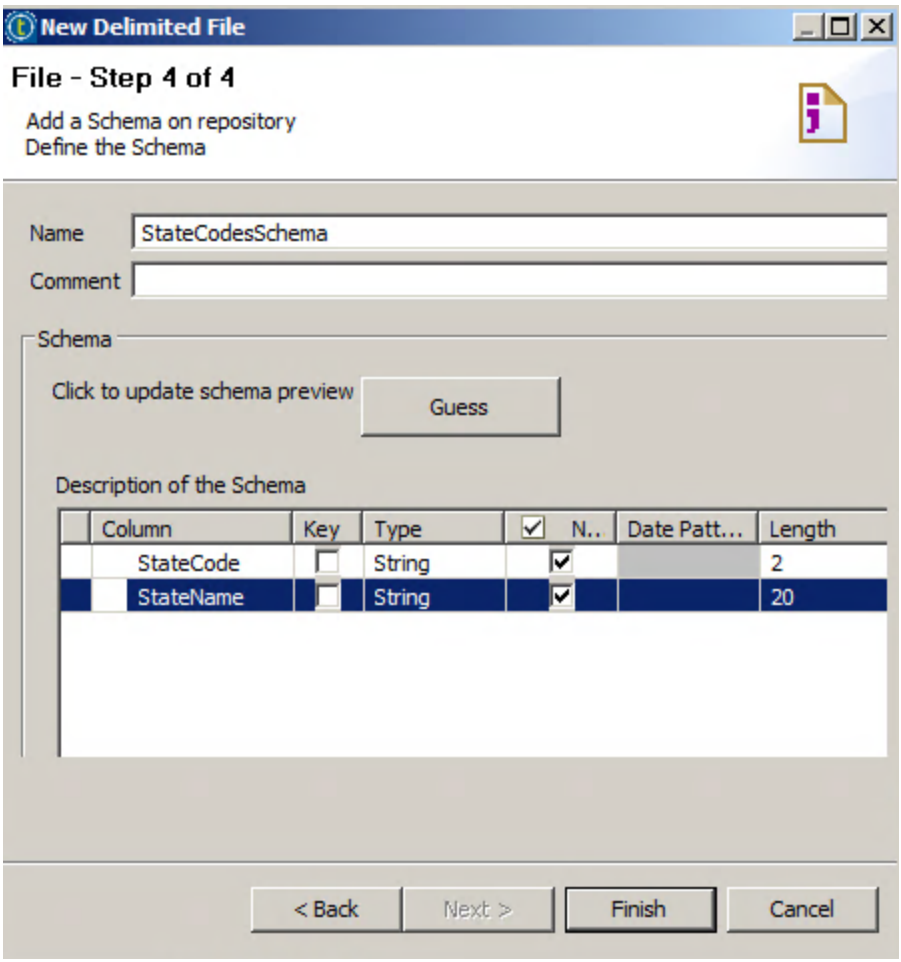
5. In the bottom area of the page, click **Refresh Preview**. Now that you have specified the correct delimiter, Talend Studio recognizes that the file contains two columns.

   Click **Next**:

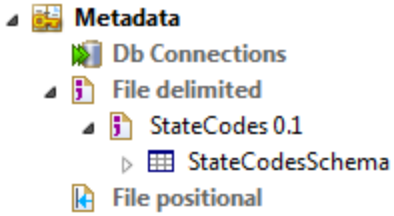


6. This Step is where you specify the schema for the file. Change the text in the **Name** box to **StateCodesSchema**. Change **Column0** to *StateCode*, set **Length** to *2* and replace **Column1** with *StateName* and set **Length** to *20*.

   Then click **Finish**:



7. Click **Finish**. The new metadata appears in the **Repository**:



Next

Now you are ready to use this metadata to create a component that will read State codes from a file in order to join them to customers information.

Creating a Join

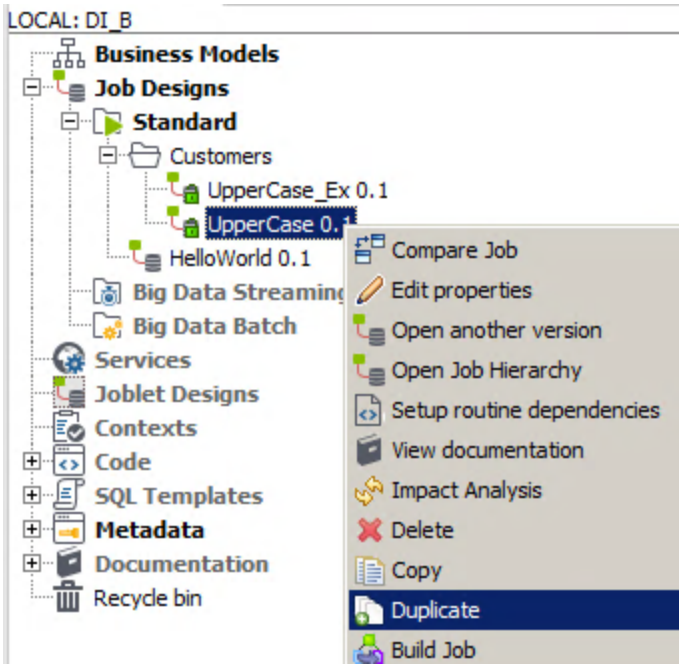
Overview

In this section, you will first duplicate the Job created in the previous lesson.

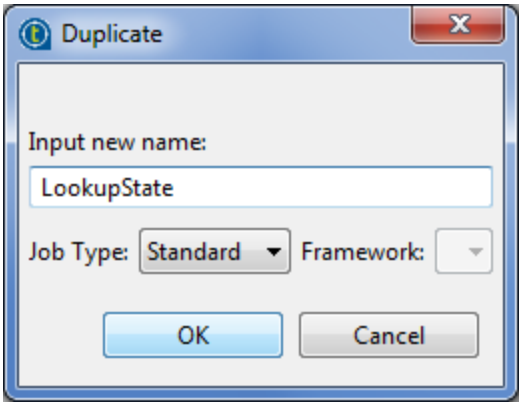
Then you will use the file containing state codes and names as a lookup table to include the full state name in the output.

Duplicate the Job

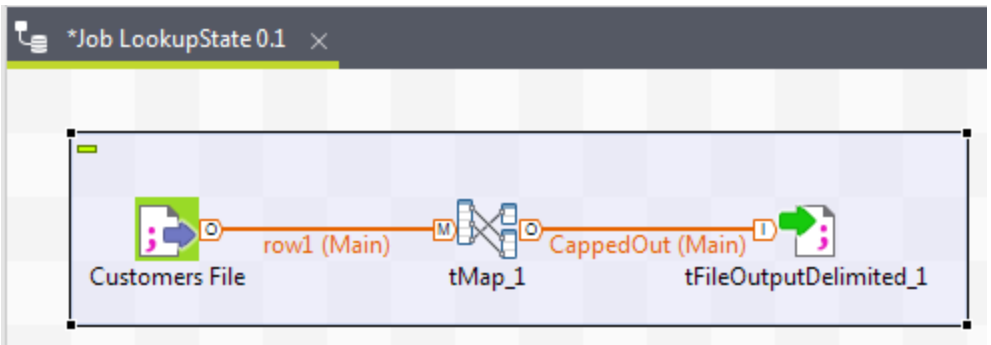
1. In the **Repository**, under **Job Designs**, right-click **UpperCase** and then click **Duplicate**:



2. Enter **LookupState** for the **Input new name** and click **OK**



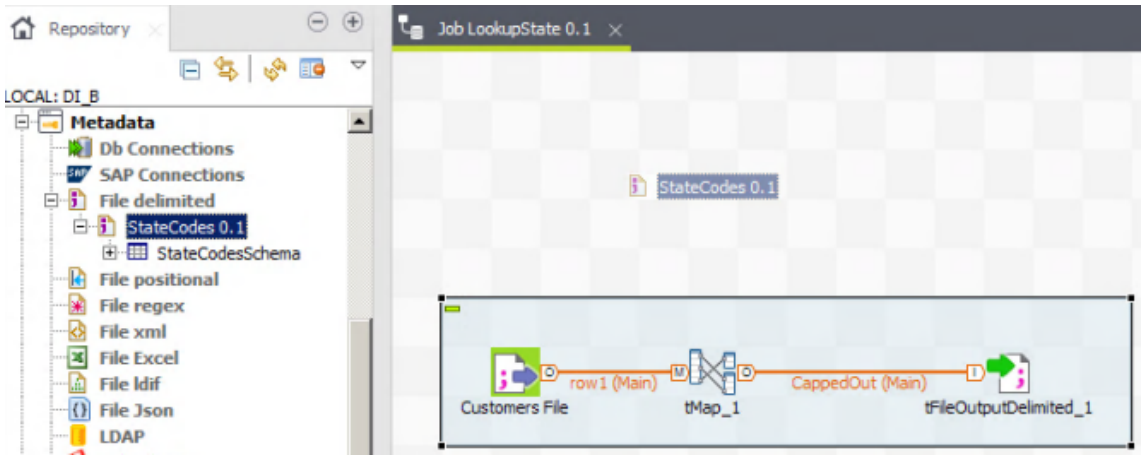
3. Double-click **LookupState** in the **Repository** to open it:



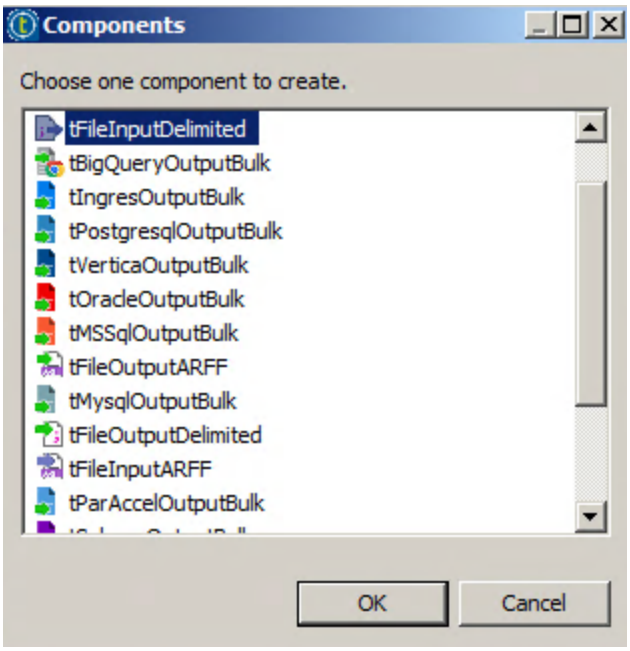
Take a moment to examine the components to remind yourself of the function of the current Job.

Add Second Source

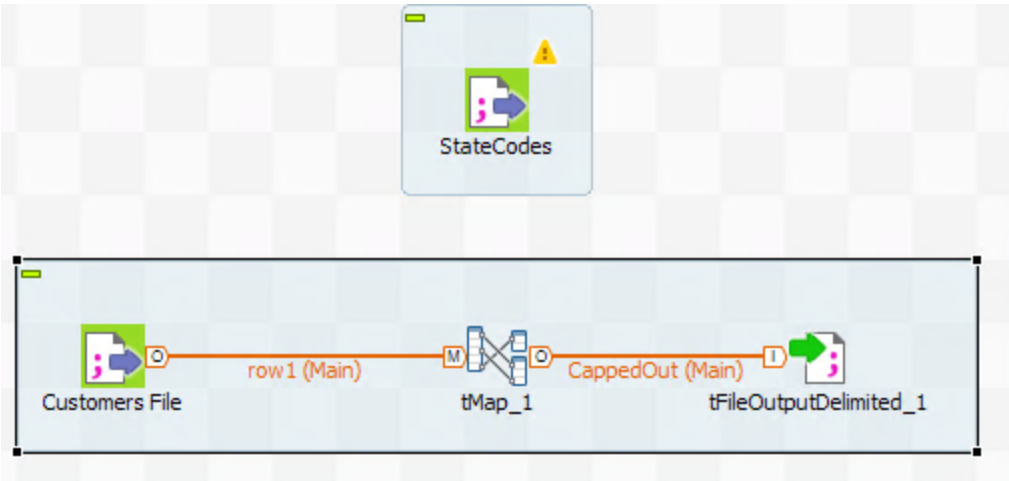
1. Drag the new **StateCodes** item from under **Metadata** in the **Repository** onto the design workspace above the **tMap** component.
   Hint: If you don't have room, you can left-click the group of components and simply drag and drop them lower in the design workspace.



2. This window allows you to choose which component you want to use with the metadata.
   Choose **tFileInputDelimited** and then click **OK**:

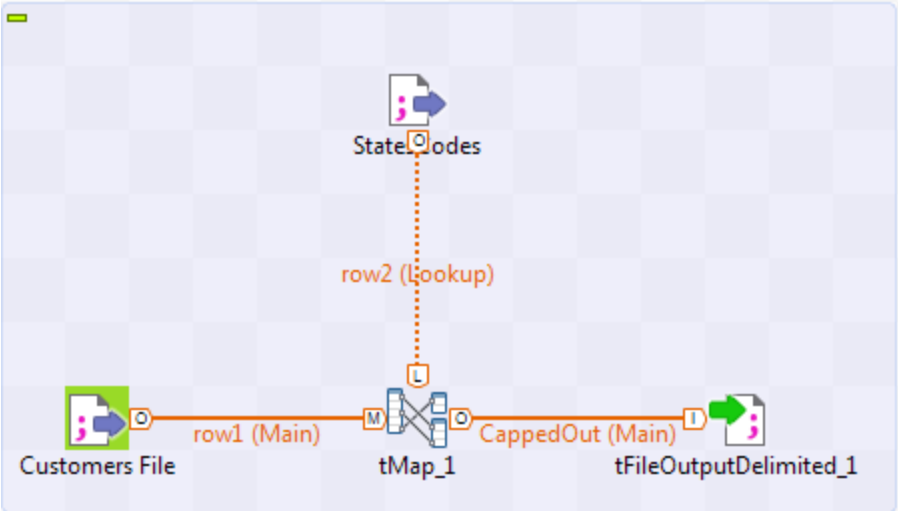


3. Your Job should look like this:



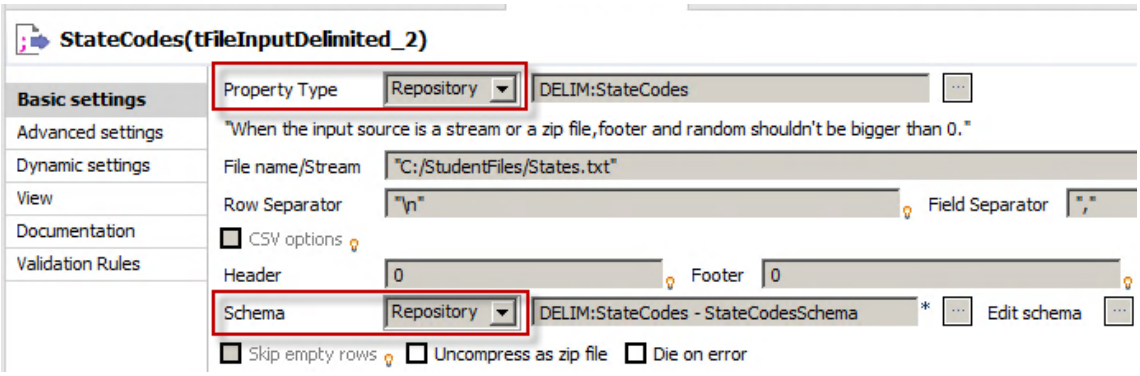
Notice that the component label is the same as the name of the metadata item. When you store configuration information as metadata in the **Repository**, you can use that metadata as the starting point, as you did here, rather than begin by placing a component and then adding the configuration.

4. Right-click **StateCodes**, click **Row** followed by **Main**, and then click the **tMap** component to create a row:



Note that the row is a **Lookup**, not a **Main**. You can provide multiple input sources for a **tMap** component, but only one can be the **Main** row. The others are **Lookups**.

5. Double-click **StateCodes** to open the **Component** view:

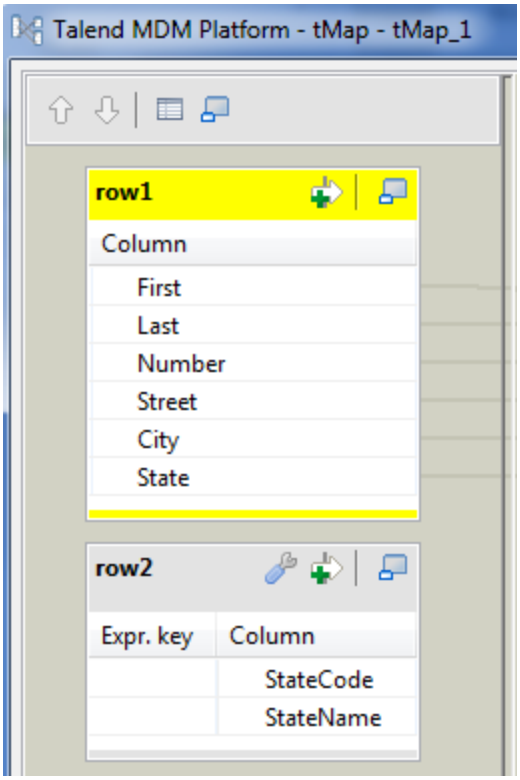


Note that the **Property Type** and **Schema** value is **Repository** and not **Built-In**.

6. Double-click the **CustomersFile** component to see that this component uses **Built-In** configuration information. So, **Built-in** property type is specific to a single component, while **Repository** property type is stored as Metadata and can be used by multiple components in multiple Jobs.

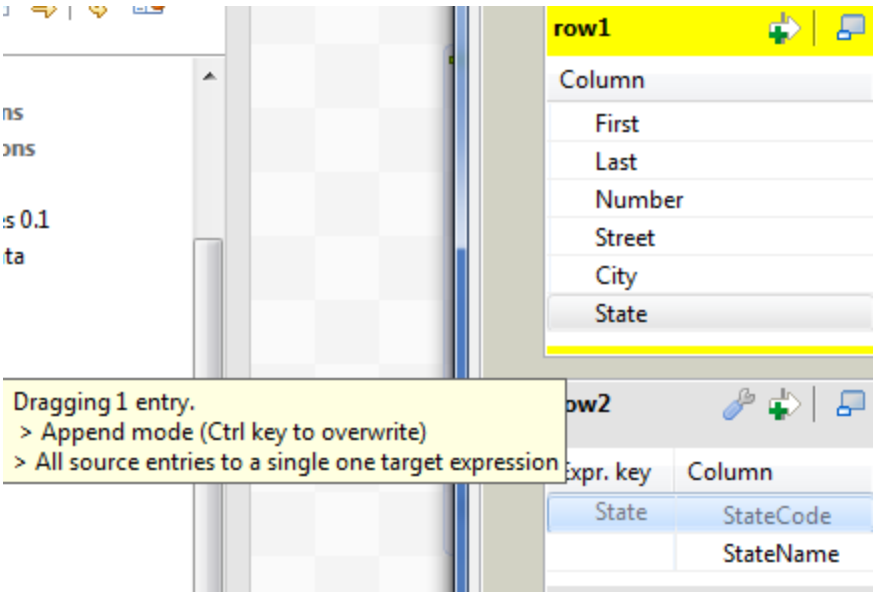
Configure tMap

1. Double-click the **tMap** component:



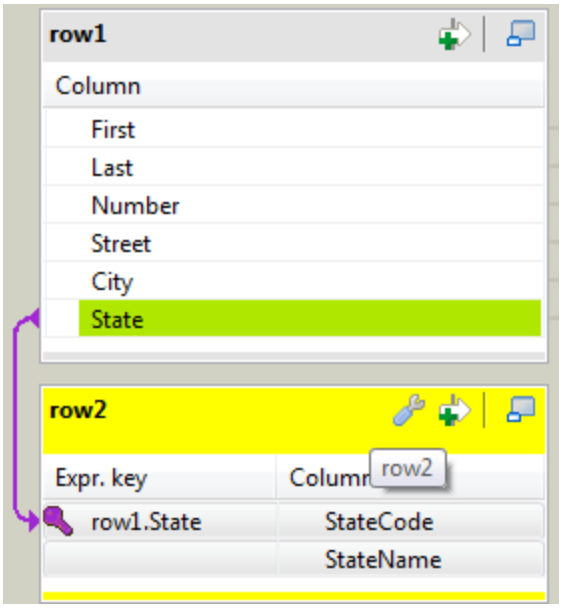
Note that there is now a second input table on the left, called **row2**, corresponding to the input row connecting **StateCodes** to the **tMap** component.

2. Drag **State** from the **row1** table to the **Expr. key** field for **StateCode** in the **row2** table:

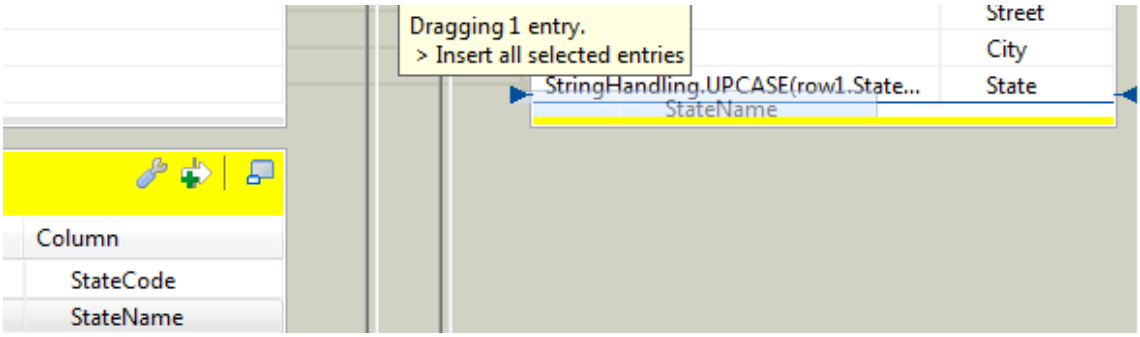


3. This creates the join, so that the value of the **State** column from **row1** will be compared to the value of the **StateCode**

column in **row2**

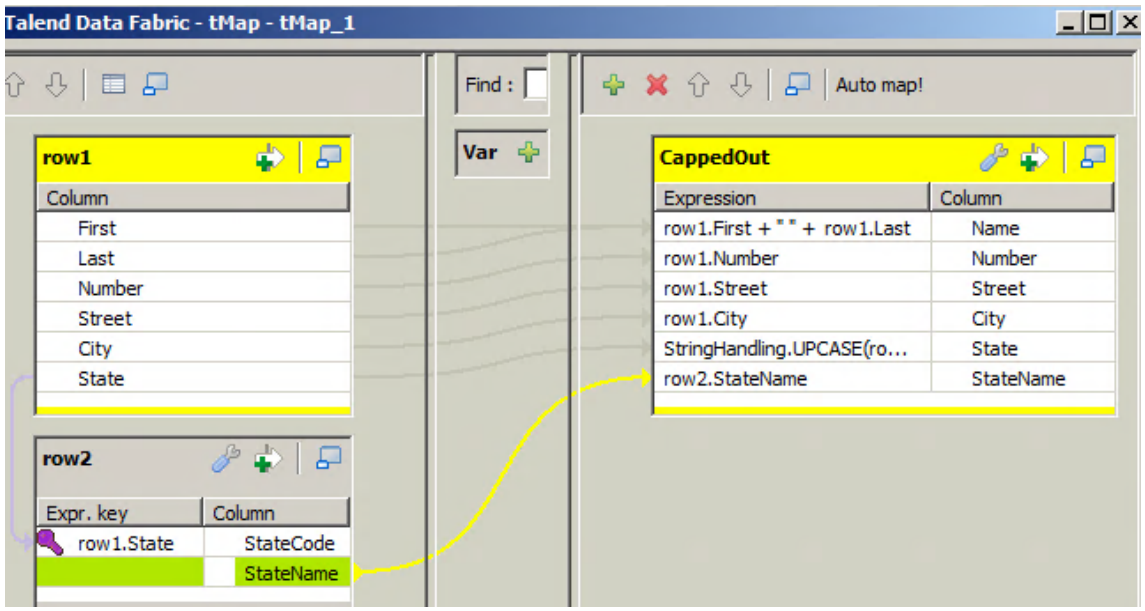


4. Drag **StateName** from the **row2** table to the bottom of the **CappedOut** table on the right:

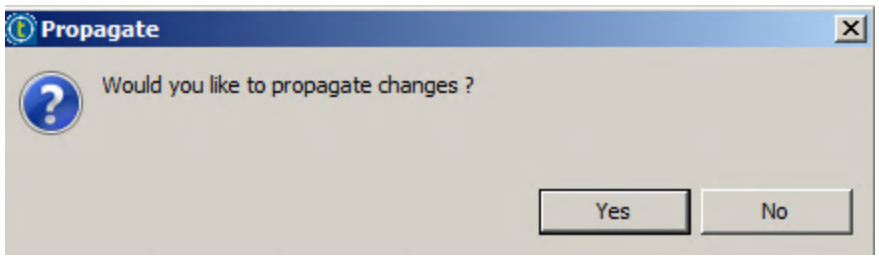


This adds the value of the **StateName** column to the output.

5. Click **Ok**



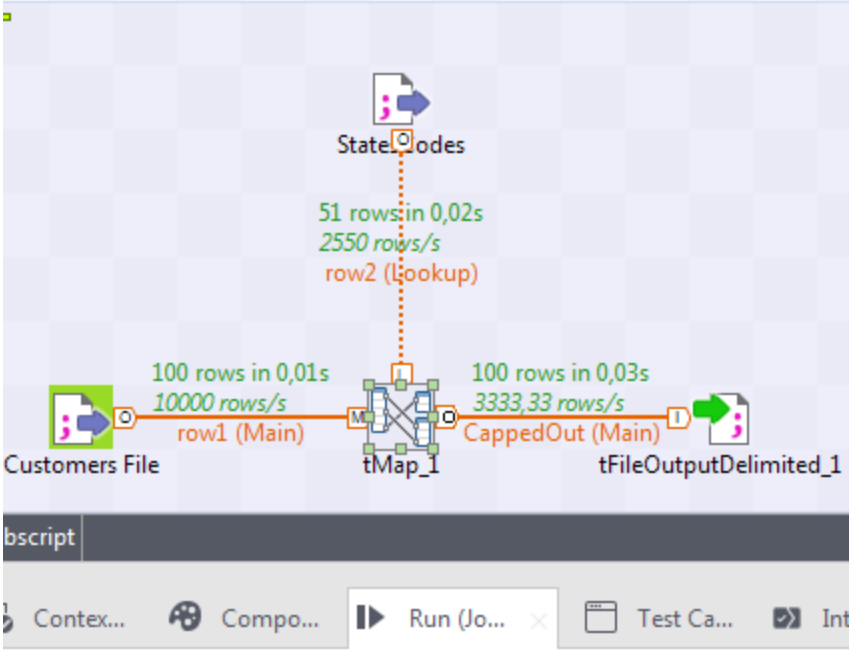
6. The system will ask you if you want to propagate the changes. Click **Yes**:

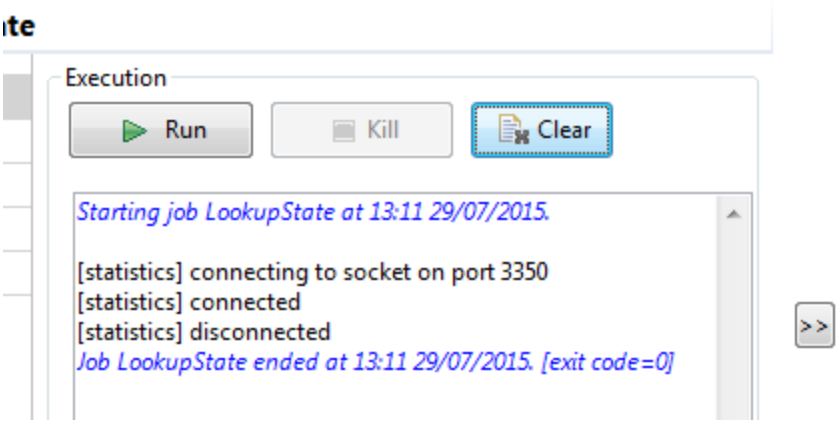


7. Save the Job

Run Job

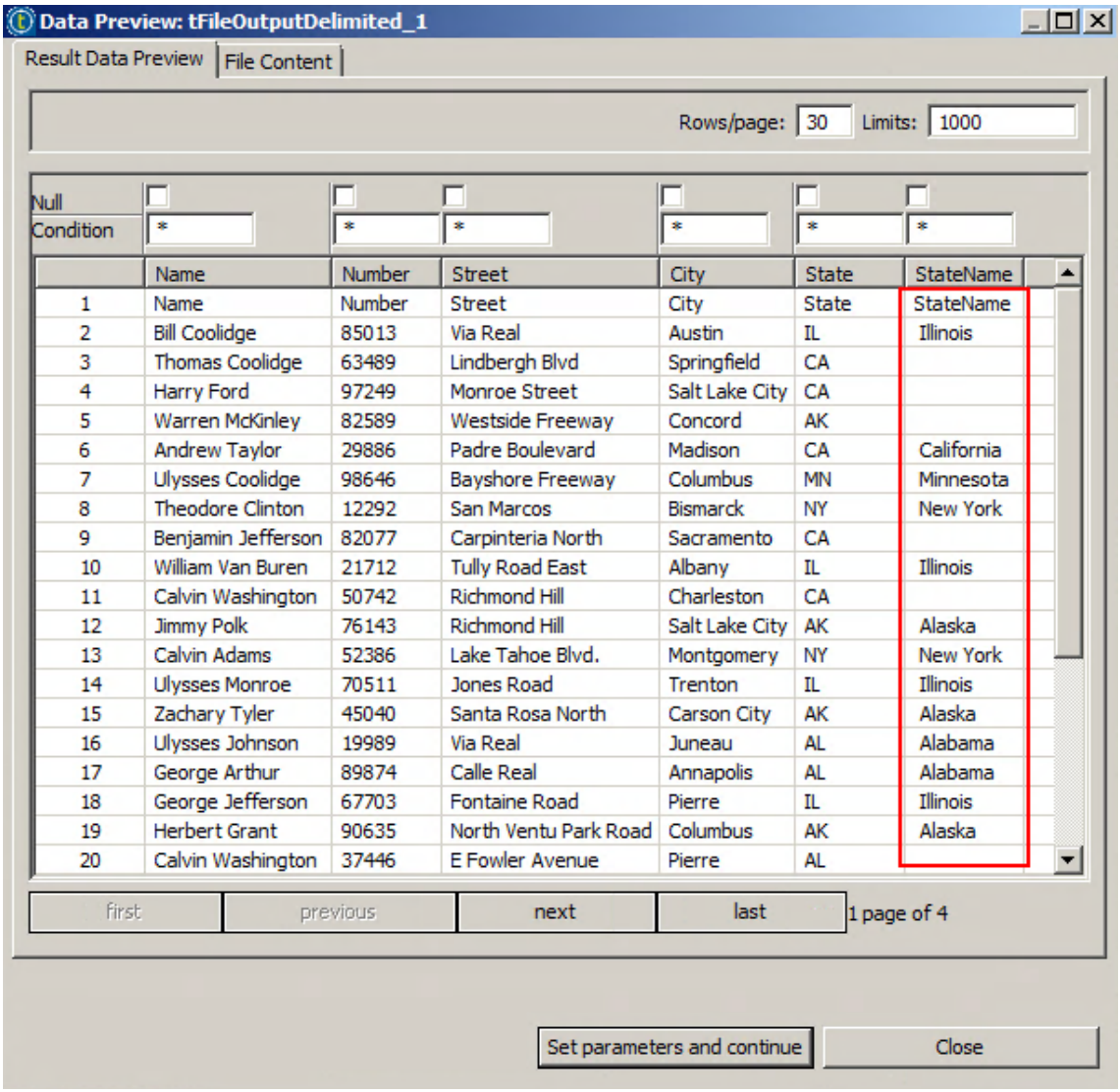
1. Run the Job.





2. Examine the output file.

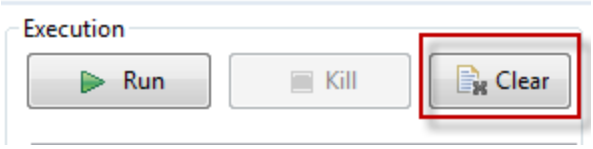
   Right-click on **tFileOutputDelimited** component and select **Data viewer**.
   Note that not all rows include the state name:



	Name	Number	Street	City	State	StateName
1	Name	Number	Street	City	State	StateName
2	Bill Coolidge	85013	Via Real	Austin	IL	Illinois
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	California
7	Ulysses Coolidge	98646	Bayshore Freeway	Columbus	MN	Minnesota
8	Theodore Clinton	12292	San Marcos	Bismarck	NY	New York
9	Benjamin Jefferson	82077	Carpinteria North	Sacramento	CA	
10	William Van Buren	21712	Tully Road East	Albany	IL	Illinois
11	Calvin Washington	50742	Richmond Hill	Charleston	CA	
12	Jimmy Polk	76143	Richmond Hill	Salt Lake City	AK	Alaska
13	Calvin Adams	52386	Lake Tahoe Blvd.	Montgomery	NY	New York
14	Ulysses Monroe	70511	Jones Road	Trenton	IL	Illinois
15	Zachary Tyler	45040	Santa Rosa North	Carson City	AK	Alaska
16	Ulysses Johnson	19989	Via Real	Juneau	AL	Alabama
17	George Arthur	89874	Calle Real	Annapolis	AL	Alabama
18	George Jefferson	67703	Fontaine Road	Pierre	IL	Illinois
19	Herbert Grant	90635	North Ventu Park Road	Columbus	AK	Alaska
20	Calvin Washington	37446	E Fowler Avenue	Pierre	AL	

This means that the Job is not doing exactly what you intended, so you need to investigate further.

3. Click **Clear** to remove the statistics display from the design workspace and the execution console:



Next

The next step is to determine why the Job is failing to perform as expected so that you can correct it.

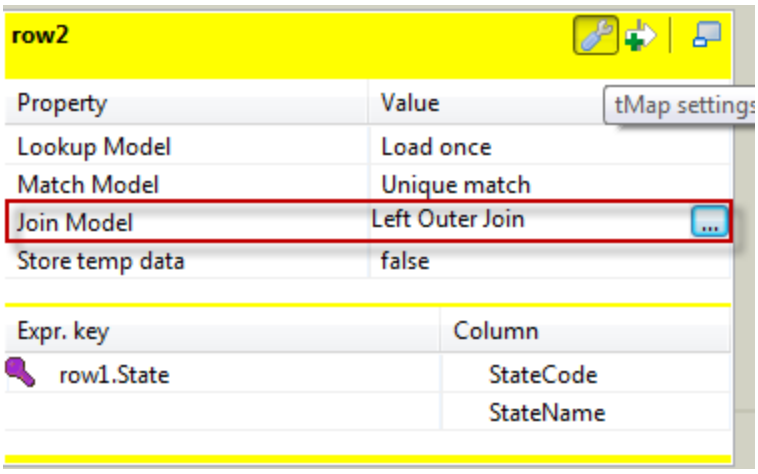
Capturing Join Failures

Overview

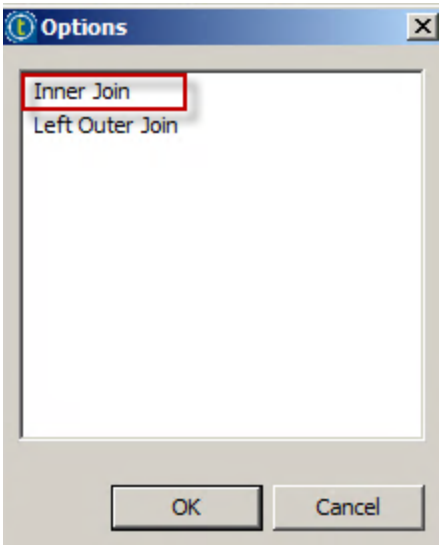
Configure Join Model in tMap

1. Double-click the **tMap** component. In the **row2** table, click the **tMap settings** button (marked with a wrench):

   This is where you can configure additional parameters about the table. Notice that the **Join Model** says **Left Outer Join**. A left outer join includes all rows from the primary table even if there is not a row in the lookup table with a matching value in the join column, which explains why some rows did not include the state name. But now you need to know why some lookup rows did not match, and in order to do that, you need to use an inner join.
   Click **Left Outer Join** in the Value column, next to **Join Model**, and then click the button marked with an ellipsis **[...]** to change the value:

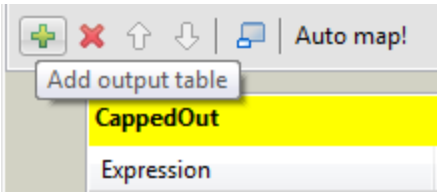


2. The **Options** wizard opens. Choose **Inner Join** and then click **OK**. By changing the join to an inner join, rows that don't match will be excluded from the output, and you can capture the rejects for troubleshooting purposes:



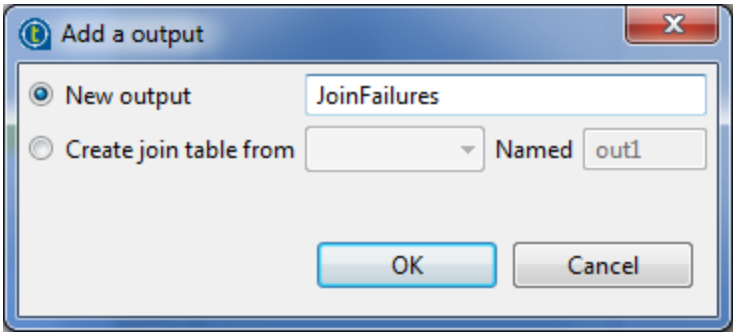
Add Rejects Table

1. Still in the **tMap** component, click the **Add output table** button above the **CappedOut** table (marked with a plus sign):

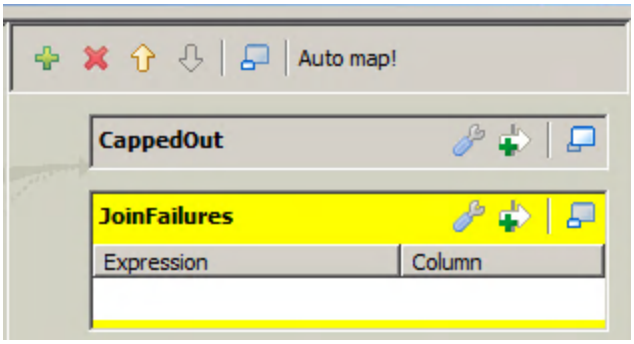


   You are creating a new table to capture rows that fail the inner join so that you can determine the problem.

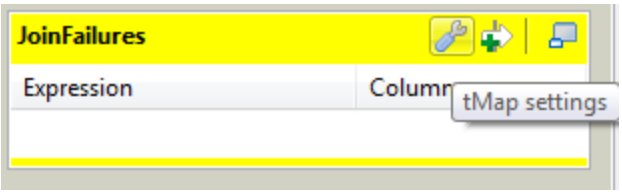
2. Enter **JoinFailures** into the text box to name the new output, and then click **OK**:



3. The new table appears below **CappedOut**:

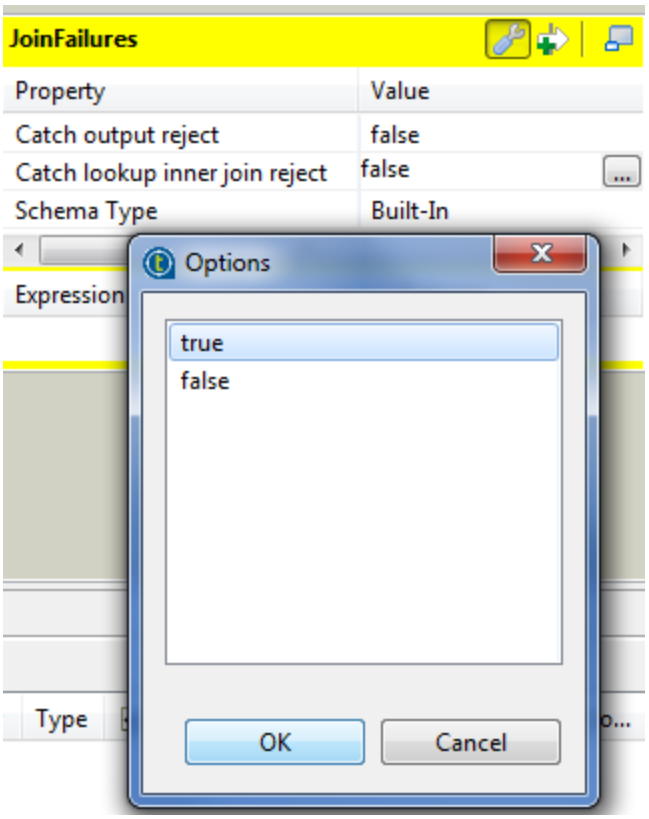


4. Click the **tMap settings** button (the wrench) in the **JoinFailures** table:

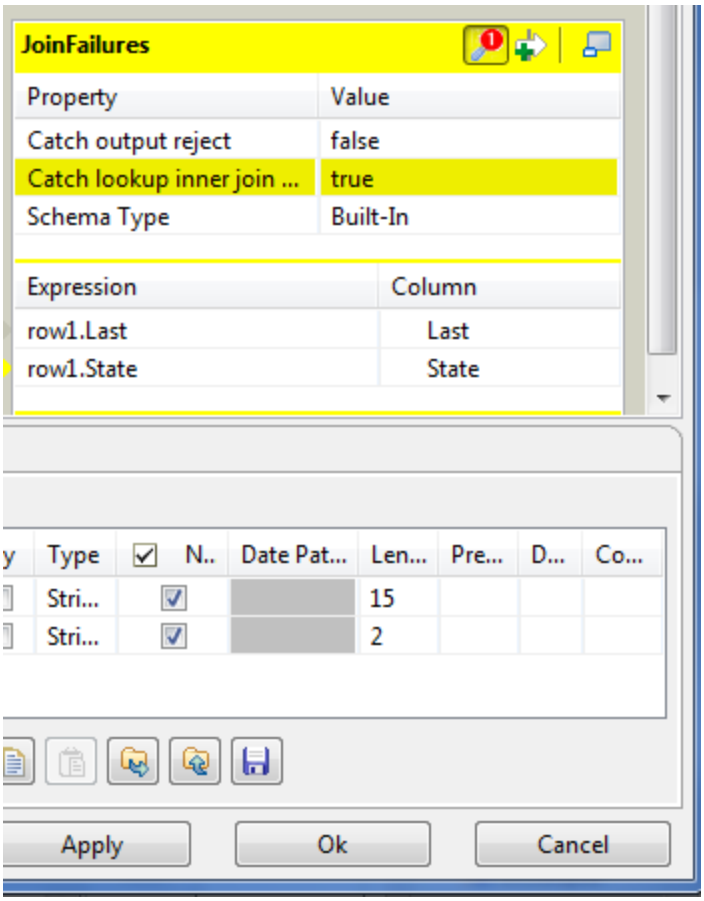


   Notice that the settings on an output table are different from those on an input table.

5. Click the value of **Catch lookup inner join reject** and change it to **true**. Then click **OK**:



6. Drag **Last** and **State** from the **row1** table to the **JoinFailures** table:



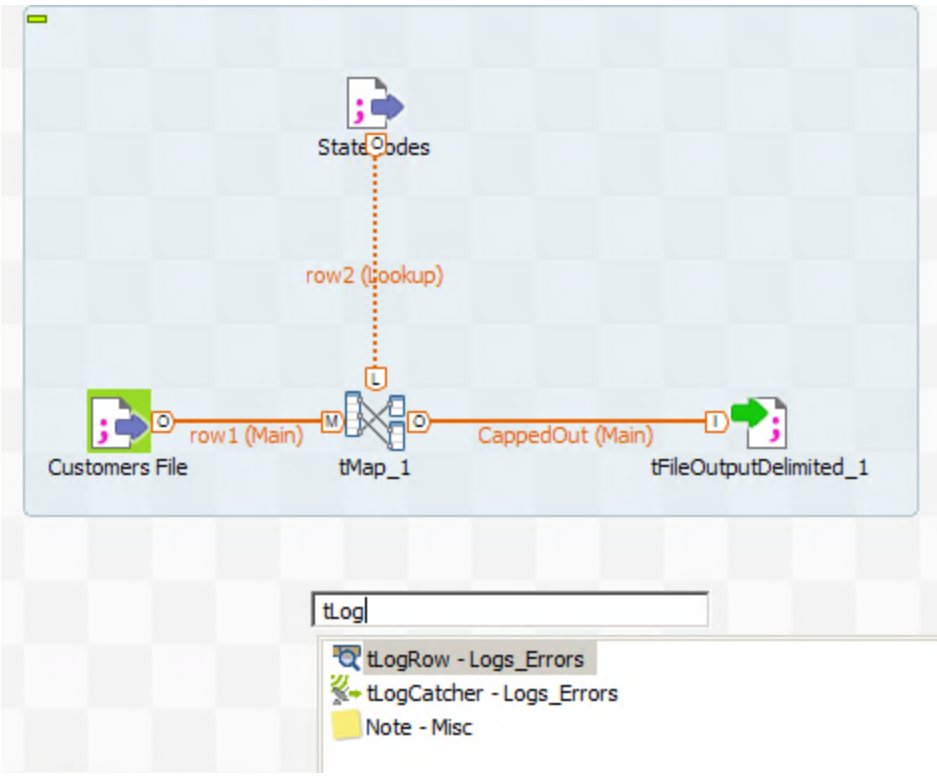
Now this output will hold the last name and state for rows that fail the inner join.

7. Click **Ok**.

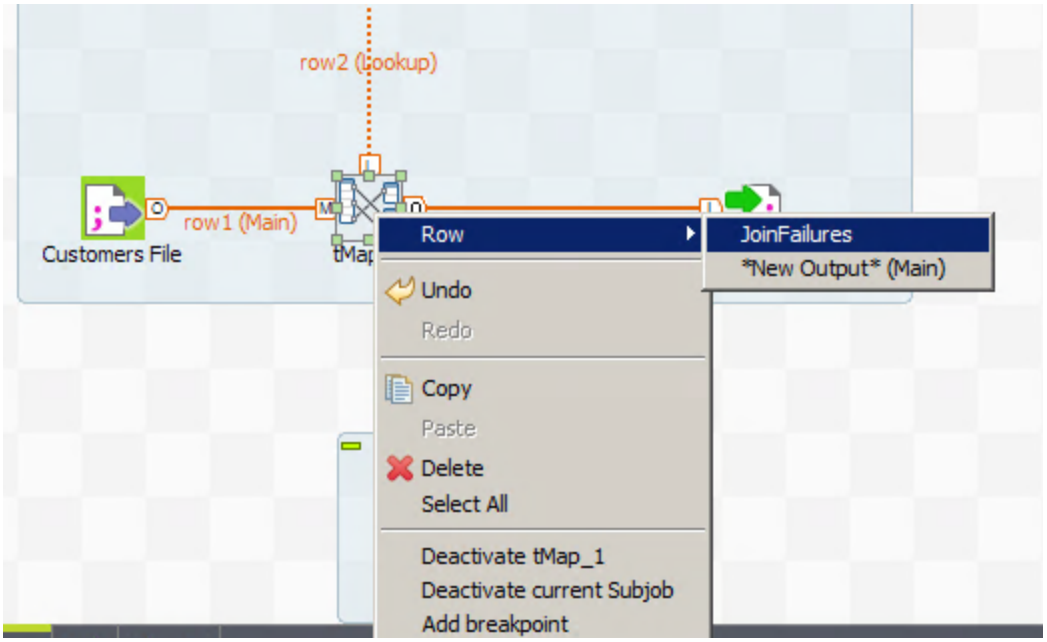
Log Output

1. Add a **tLogRow** component just below the **tMap** component.

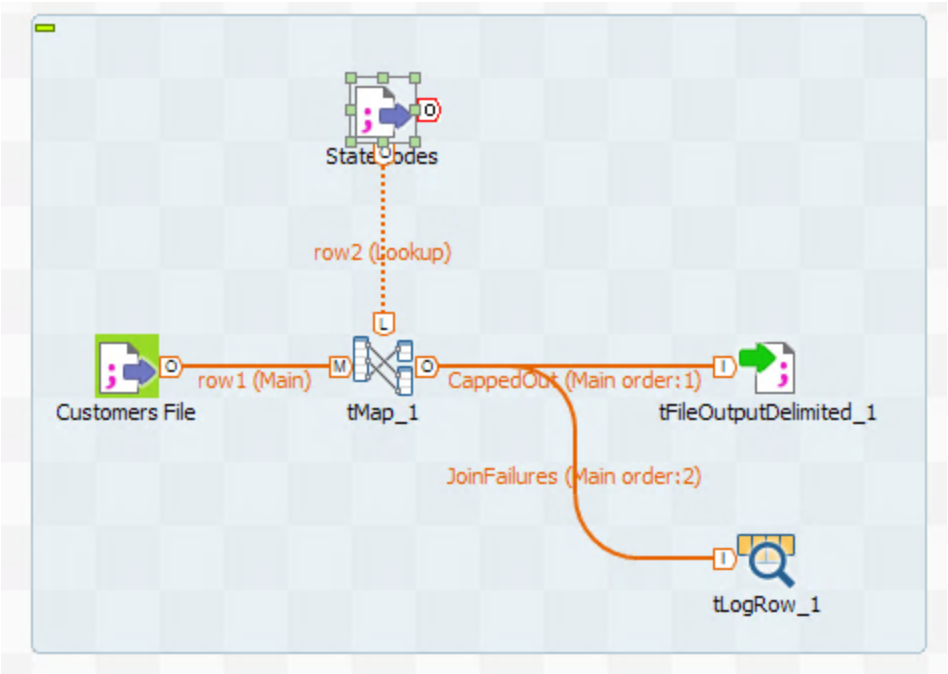
   Recall that this component writes rows to the execution console in the **Run** view:



2. Right-click the **tMap** component, click **Row** followed by **JoinFailures**, and then click **tLogRow** to connect them:

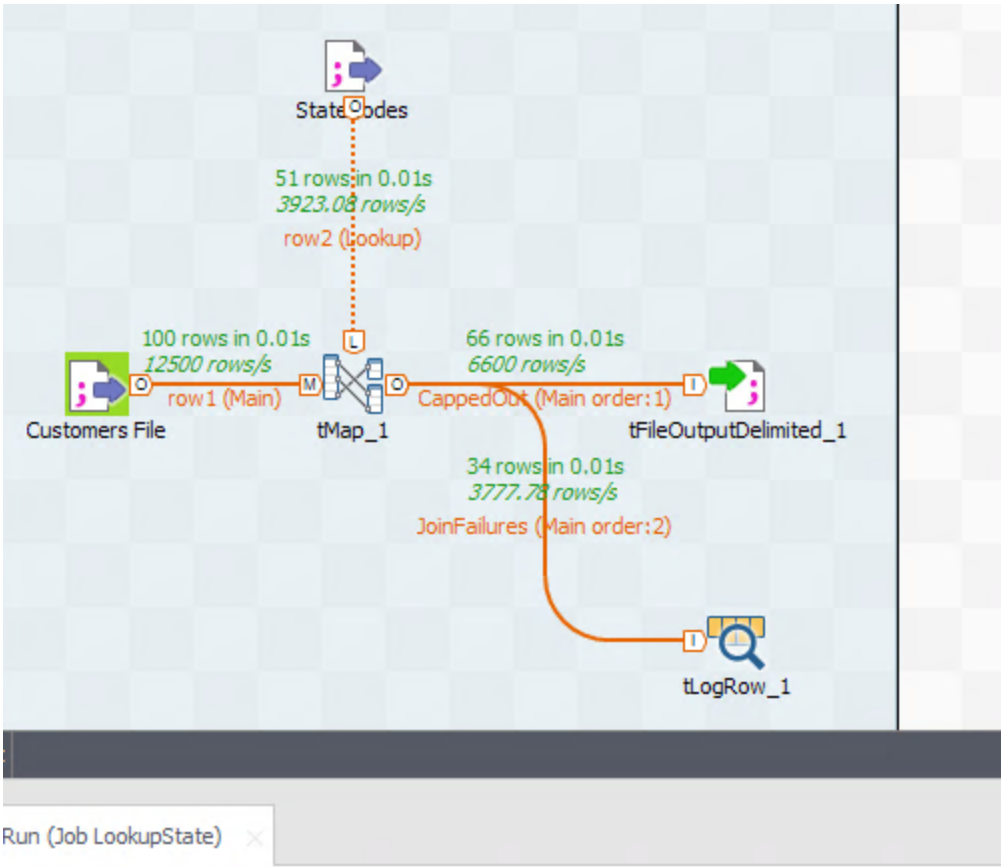


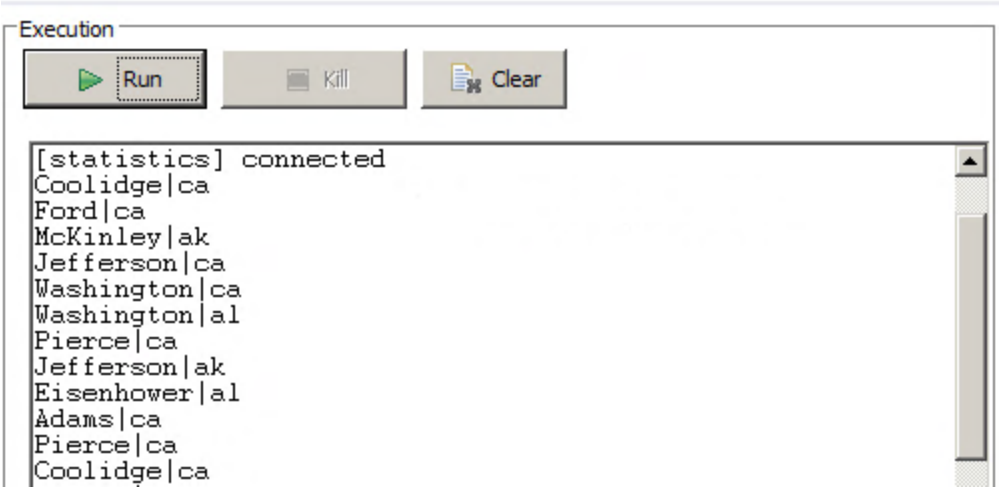
3. The **JoinFailures** row corresponds to the output table you added to the **tMap** component:


StateCodes
row2 (Lookup)
Customers File
row1 (Main)
tMap_1
CappedOut (Main order:1)
tFileOutputDelimited_1
JoinFailures (Main order:2)
tLogRow_1

Run the Job

1. Run the Job and then examine the output in the console:





Note that the state codes in the rejected rows are lower case (from the input file), which means they won't match the upper-case codes from the lookup file.

Next

Now that you know the problem with the join, you can make a correction so that the lookup works as planned.

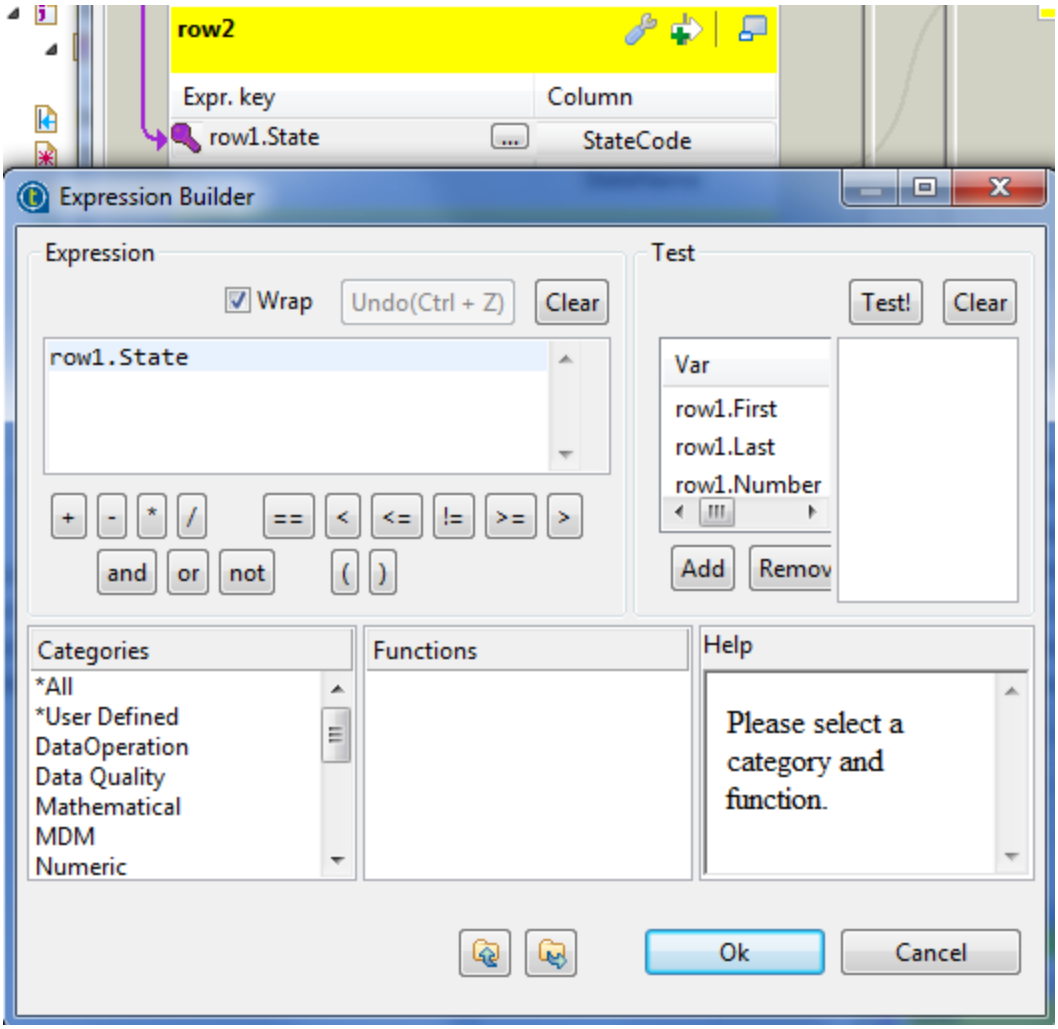
Correcting the Lookup

Overview

Now that you know the lookup is failing because of lower case string values, you can correct the configuration.

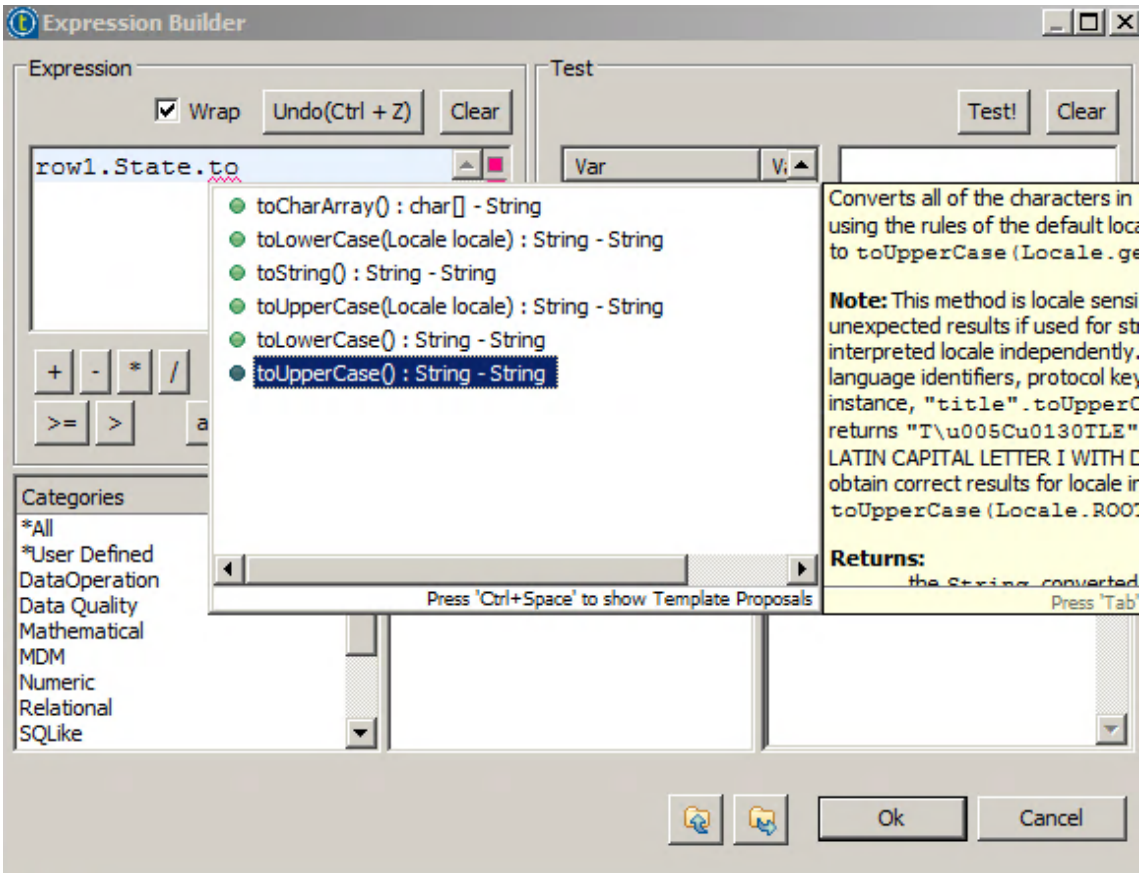
Configure tMap

1. Double-click the **tMap** component. Remember that **row2** is the lookup table for the join. In the **row2** table on the left, click **row1.State** and then click the button marked with an ellipsis **[...]** to open the **Expression Builder**:

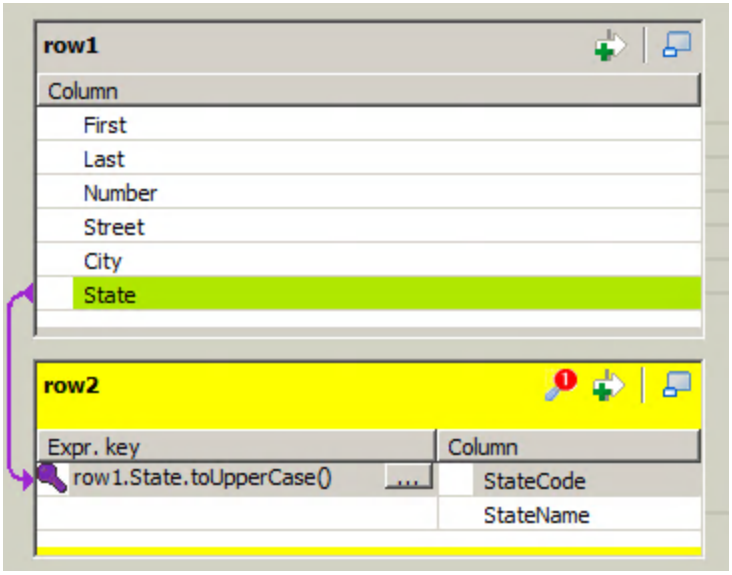


2. In a previous exercise, you used the **Expression Builder** to convert lowercase strings to uppercase. Now you need to do the same thing for this column. But let's do it differently. After the last character of *row1.State* type directly *.to* and you will get the options the editor's contextual auto-completion has to propose.

   Double-click *toUppercase(): String - String*in order to select it.



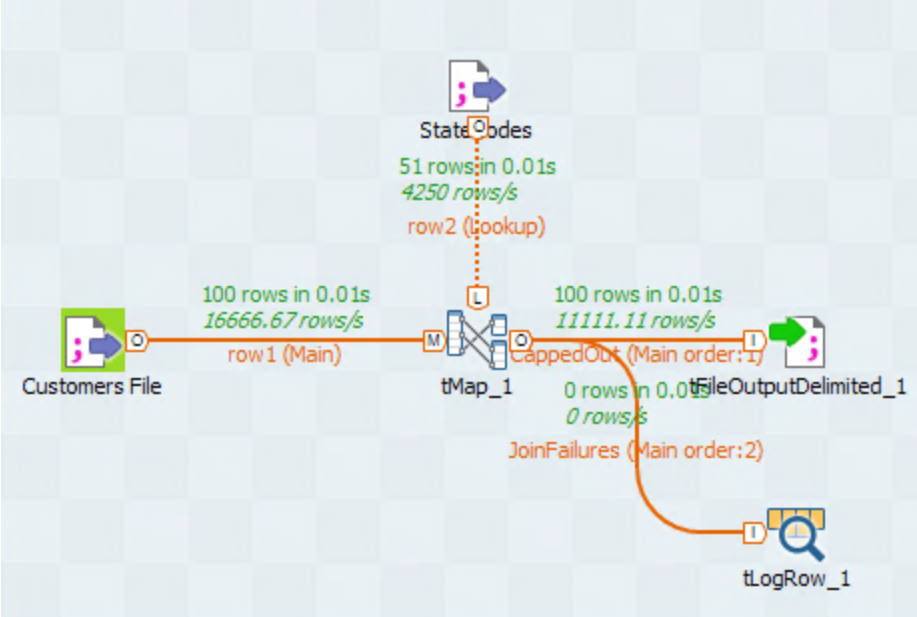
3. Click **Ok** to close the **Expression Builder**. Now the value of the **State** column is converted to upper case before being compared to the lookup table, so the join should work properly:



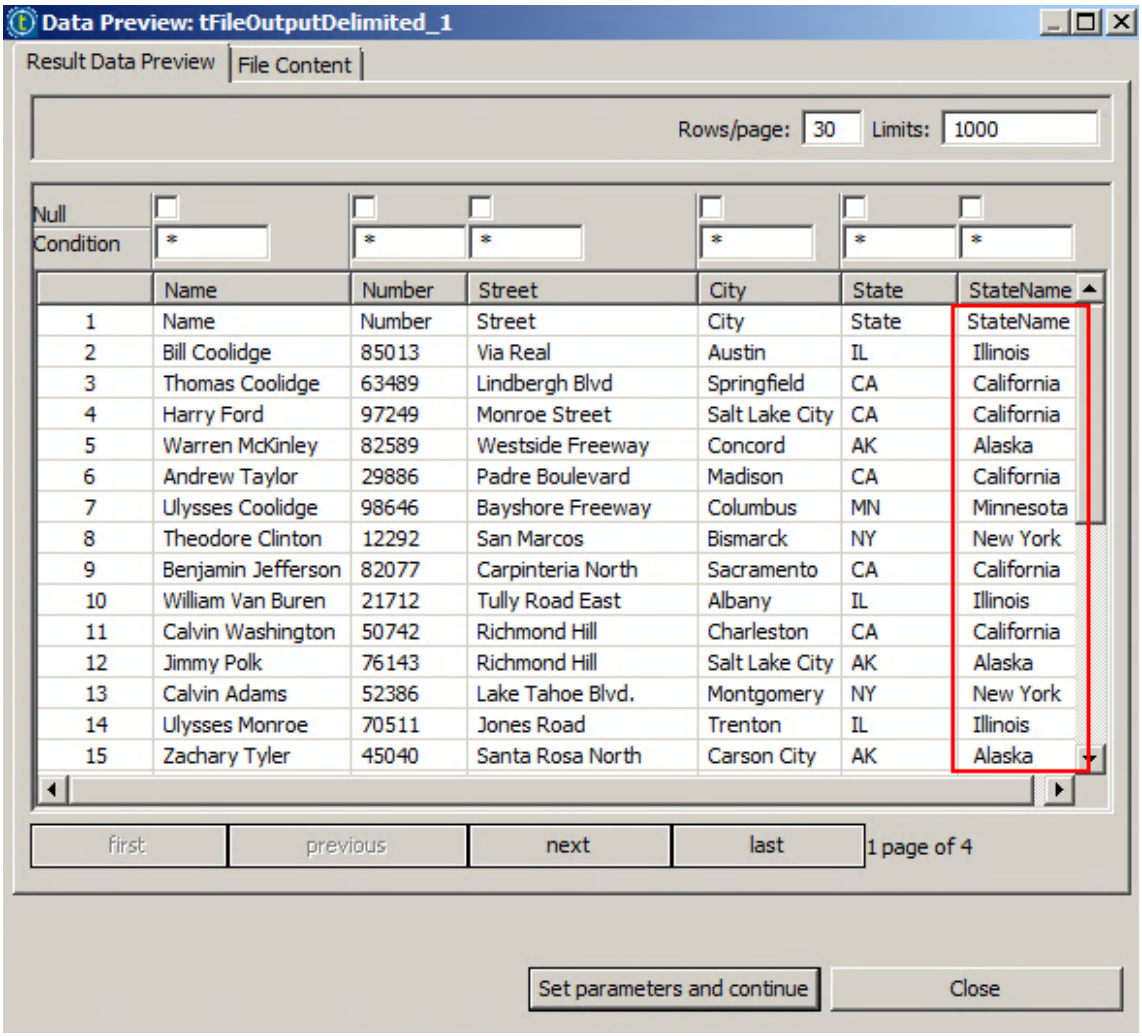
4. Click **Ok** to close the **tMap** component.

Run the Job

1. Run the Job . Note that the **tLogRow** component processed 0 rows this time, as there were no rejects found. The data flow to the output flow is back up to 100 rows processed again.



2. Examine the output file content using the **Data Preview** option in the Studio or **Notepad++** to make sure that all rows include the state name:

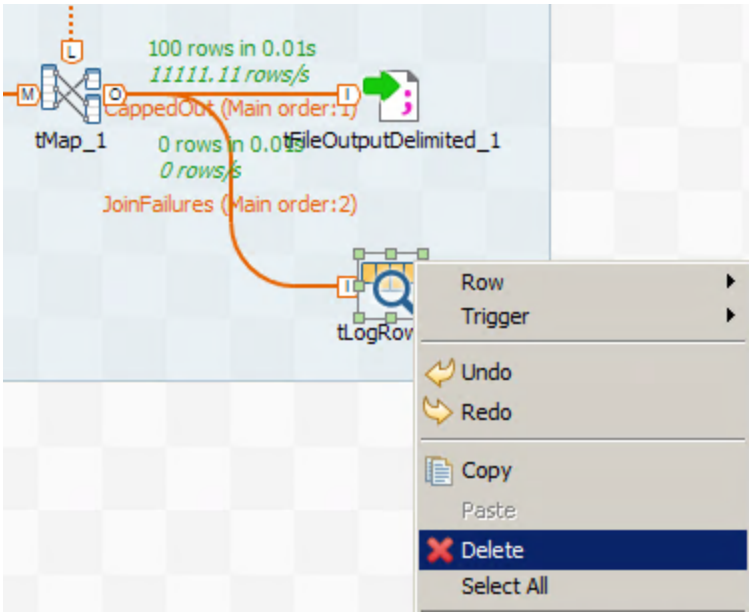


	Name	Number	Street	City	State	StateName
1	Name	Number	Street	City	State	StateName
2	Bill Coolidge	85013	Via Real	Austin	IL	Illinois
3	Thomas Coolidge	63489	Lindbergh Blvd	Springfield	CA	California
4	Harry Ford	97249	Monroe Street	Salt Lake City	CA	California
5	Warren McKinley	82589	Westside Freeway	Concord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

At this point, there is no longer a need to log the rejects, so you can clean up the Job by removing the **tLogRow** component.

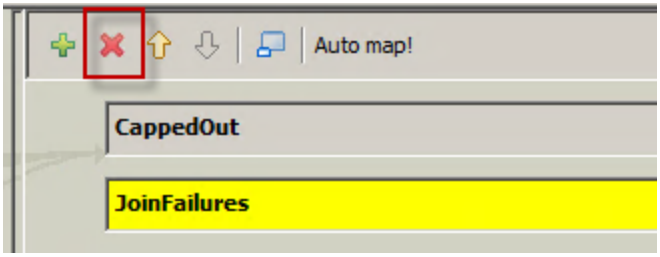
Clean up the Job

1. Right-click the **tLogRow** component in the design workspace and then select **Delete** to remove the component:

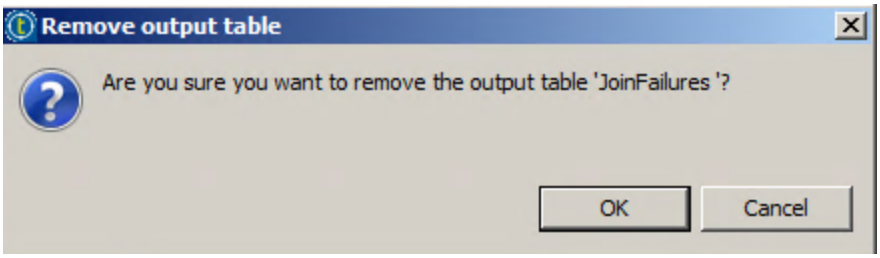


Note you can also delete a component by using the **Delete** key on your keyboard.

2. Double-click the **tMap** component to open the Mapping Editor.
Click the **JoinFailures** table on the right and then click the **Remove selected output table** button, marked with an X (near the top of the main window).

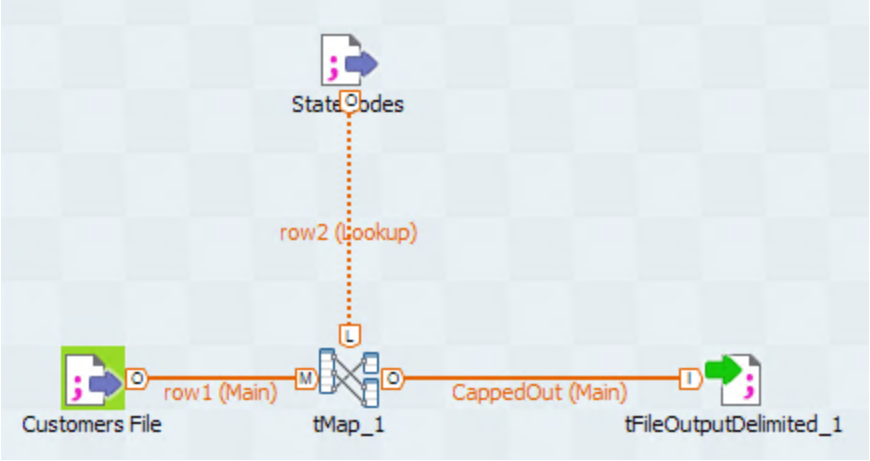


Answer **OK** when asked if you are sure you want to remove the **JoinFailures** table:



Click **Ok** to close the Mapping Editor.

3. Click **Clear** in the **Run** view to remove the statistics display from the design workspace:



4. Save the Job.

Next

You have completed this Job, you are now ready to read the Wrap-Up.

Wrap-Up

In this lesson, you extended the Job from the previous lesson to explore joining two data sources through a **tMap** component. You looked at how to troubleshoot data issues by capturing join failures, and practiced writing rows to the console. You also stored component configuration information as metadata in the **Repository** so that it could be used later by multiple components and other Jobs.

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 >**.