Transform Academy

Convert CSV – XLSX

Transform Training
Project: CSV - XLSX
Version: V1.0
Author: Colin Avery
July, 2025

Legal Statement

CTA Consultancy Ltd

Information contained in this document is confidential and is supplied for internal purposes between CTA Consultancy Ltd and ClientName. Reproduction of this document and use of its contents is prohibited unless expressly approved.

Amendments and Changes

CTA Consultancy Ltd reserves the right to modify, update, or amend this document and its contents at any time without prior notice. Any changes made to this document will supersede previous versions. It is the responsibility of the recipient to ensure they are working with the most current version of this document. CTA Consultancy Ltd shall not be liable for any consequences arising from the use of outdated or superseded versions of this document.

Compliance

All recipients and users of this document must comply with applicable laws, regulations, and industry standards in their jurisdiction. The recipient acknowledges their responsibility to ensure that their use of this document and its contents complies with all relevant legal, regulatory, and contractual obligations. CTA Consultancy Ltd makes no warranty regarding compliance with specific regulatory requirements and disclaims any liability for non-compliance by the recipient. Any breach of these terms or applicable compliance requirements may result in immediate termination of access to this document and potential legal action.

This document and all information contained herein remains the exclusive property of CTA Consultancy Ltd. Any unauthorized disclosure, distribution, or use of this document or its contents may result in legal action. By accessing this document, the recipient acknowledges and agrees to maintain the confidentiality of all information contained within.

Table of Contents

About This Guide

This guide provides information on how to build a Transform Foundation Server project to convert Comma Separated Files (CSV) to an Excel XLSX workbook with formatting.

Intended Audience

This guide is intended for Transform developers responsible for, maintaining, and supporting the Transform Foundation server branch files.

This guide assumes the reader understands the associated technologies and standards, including the Transform Designer, Microsoft Excel, and VBScript language.

Related Documentation

The following documents are related to this guide:

- *https://www.activexperts.com/admin/vbscript-collection/msoffice/excel/*

Acronyms

The following acronyms are used in this document:

- **BT** – Bottomline Technologies
- **TFS** – Transform Foundation Server
- **CSV** – Comma Separated Values

Time to complete the Exercise

To complete the exercise will take approximately 30 minutes.

Exercise Resources (click resource to open attachments and save)

- **csvData.csv** – Sample CSV data file
- **CSV2XLSX.txt** – VBScript to convert CSV to XLSX
- **PitvotTable.txt** – VBScript to create workbook with Pivot Table summary worksheet

Acknowledgements

I would like to extend our gratitude to my cats, who's support during the creation of this exercise was invaluable keeping my keyboard warm.

1 Introduction

The purpose of this exercise is to demonstrate how you can use a TFS branch to convert a **CSV** file into a formatted **Excel xlsx** file. Following this exercise, you will be able to export a branch container to an Excel workbook. Transform Designer has an **ActiveX** Script object that allows you to create and execute an ActiveX script at runtime inline within your branch process.

The Scripts we use in this exercise are **VBScripts**, the scripts allow us to call and execute Microsoft Excel in the background. Whilst it would be advantageous to have knowledge of VBScript, it is not strictly necessary to complete the exercise, as we will supply you with the required scripts which you can paste into your branch Script object.

The guide will walk you through the required steps to build the branch from scratch so you can create it and enjoy the surprise of Transform performing its magic.

1.1 What will I learn?

In this exercise you will learn how to create a branch from scratch making use the following Transform objects.

- *Action List*
- *Bag*
- *Catch*
- *Check Directory*
- *Check File*
- *Container Shortcut*
- *Copy Number*
- *Copy Text*
- *Copy Yes/No*
- *Count*
- *Create Directory*
- *Decision*
- *Delimited Data Parser*
- *File Queue*
- *Folder*
- *Link*
- *Log Event*
- *Memory*
- *Number*
- *Queue Process*
- *Replace*
- *Script*

- *Sentence*
- *System Information*
- *Text*
- *Unicode From Text*
- *Unicode Text*
- *Unique Identifier*
- *Write File*
- *Yes/No*

1.2 What will I not learn?

Whilst we will make use of VBScripts to execute the Microsoft Excel calls, you will not learn how to program using VBScript in this exercise.

2 What to expect

Once you build the project and run it, then you can expect to see the following files in a Windows folder.

Name	Date modified	Type	Size
32AB361A-777B-EC49-97DB-CD42B3E351E7_PivotTable.xlsx	23/03/2022 10:06	Microsoft Excel W...	63 KB
32AB361A-777B-EC49-97DB-CD42B3E351E7_Output.xlsx	23/03/2022 10:06	Microsoft Excel W...	48 KB
csvData.csv	23/03/2022 10:05	Microsoft Excel C...	64 KB

2.1 csvData

The input CSV file, which can be an external file or from a Container copied through the **Delimited Data to Text** object within a branch. This file is available as an attachment.

```
ReportName,LegalEntity,Username,ProcessProject,DeliveryMethod,EmailAddressTo,PrinterNetworkId,AX_AOSId,Time,ProcessTime
WMSPickingList_OrderPick,bot,bot\teresa.churchill,,PrecisionForms screen,,,bot01AOS01@2712,7:21:02 AM,95.515
retaillabel,bot,bot\ashley.gomer,,PrecisionForms screen,,,bot01AOS01@2712,7:35:19 AM,459.7258
retaillabel,bot,bot\ashley.gomer,,PrecisionForms screen,,,bot01AOS01@2712,7:54:03 AM,120.8695
retaillabel,bot,bot\ashley.gomer,,PrecisionForms screen,,,bot01AOS01@2712,7:54:21 AM,126.0756
retaillabel,bot,bot\ashley.gomer,,PrecisionForms screen,,,bot01AOS01@2712,7:55:52 AM,133.6661
retaillabel,bot,bot\ashley.gomer,,PrecisionForms screen,,,bot01AOS01@2712,8:12:17 AM,131.6633
retaillabel,bot,bot\jonathan.giddings,,PrecisionForms screen,,,bot01AOS01@2712,8:41:03 AM,534.383
retaillabel,bot,bot\ashley.gomer,,PrecisionForms screen,,,bot01AOS01@2712,8:53:26 AM,148.2745
retaillabel,bot,bot\jonathan.giddings,,PrecisionForms screen,,,bot01AOS01@2712,8:57:58 AM,574.8327
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,9:08:16 AM,599.0997
retaillabel,bot,bot\mike.clark,,PrecisionForms screen,,,bot01AOS07@2712,9:08:37 AM,359.7601
retaillabel,bot,bot\mike.clark,,PrecisionForms screen,,,bot01AOS07@2712,9:09:44 AM,95.4396
PurchPurchaseOrder,bot,bot\claire.bradfield,,PrecisionForms e-mail,,,bot01AOS02@2712,9:10:40 AM,761.4601
PurchPurchaseOrder,bot,bot\claire.bradfield,,PrecisionForms e-mail,,,bot01AOS02@2712,9:10:46 AM,1339.9902
PurchPurchaseOrder,bot,bot\claire.bradfield,,PrecisionForms e-mail,,,bot01AOS02@2712,9:11:09 AM,822.9454
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,9:11:45 AM,541.9016
retaillabel,bot,bot\jonathan.giddings,,PrecisionForms screen,,,bot01AOS01@2712,9:15:22 AM,173.3461
SalesInvoice,bot,AX Batch PROD,,PrecisionForms e-mail,,,bot01AOS04@2712,9:16:07 AM,503.3677
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,9:17:32 AM,579.8639
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,9:20:03 AM,625.6041
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,9:21:14 AM,555.9063
retaillabel,bot,bot\sarah.burnett,,PrecisionForms screen,,,bot01AOS08@2712,9:21:27 AM,788.7045
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,9:28:20 AM,531.8223
retaillabel,bot,bot\mike.clark,,PrecisionForms screen,,,bot01AOS07@2712,9:37:01 AM,238.5304
retaillabel,bot,bot\helen.brown,,PrecisionForms screen,,,bot01AOS07@2712,9:38:52 AM,964.7991
retaillabel,bot,bot\helen.brown,,PrecisionForms screen,,,bot01AOS07@2712,9:39:09 AM,248.8123
SalesPackingSlip,bot,bot\andrew.clarke,,PrecisionForms printer,,,bot01AOS08@2712,9:39:55 AM,602.0124
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,9:44:24 AM,578.7085
PurchPurchaseOrder,bot,bot\shula.hutter,,PrecisionForms e-mail,,,bot01AOS07@2712,9:45:09 AM,726.4137
WMSPickingList_OrderPick,bot,bot\lauren.frost,,PrecisionForms screen,,,bot01AOS01@2712,9:53:10 AM,127.5831
WMSPickingList_OrderPick,bot,bot\stephanie.peel,,PrecisionForms screen,,,bot01AOS08@2712,9:53:35 AM,134.9674
WMSPickingList_OrderPick,bot,bot\lauren.frost,,PrecisionForms screen,,,bot01AOS01@2712,9:53:48 AM,82.9148
SalesPackingSlip,bot,bot\stephanie.peel,,PrecisionForms screen,,,bot01AOS08@2712,9:54:43 AM,331.2193
retaillabel,bot,bot\tim.hiscock,,PrecisionForms screen,,,bot01AOS03@2712,10:00:50 AM,203.0302
retaillabel,bot,bot\daniel.houghton,,PrecisionForms screen,,,bot01AOS08@2712,10:01:55 AM,1456.6939
WMSPickingList_OrderPick,bot,bot\evie.wilkins,,PrecisionForms screen,,,bot01AOS03@2712,10:05:56 AM,104.6661
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,10:06:07 AM,550.8403
SalesPackingSlip,bot,bot\andrew.clarke,,PrecisionForms printer,,,bot01AOS08@2712,10:07:17 AM,690.9549
SalesPackingSlip,bot,bot\andrew.clarke,,PrecisionForms printer,,,bot01AOS08@2712,10:08:37 AM,608.3386
SalesPackingSlip,bot,bot\jay.dossantos,,PrecisionForms printer,,,bot01AOS08@2712,10:09:13 AM,599.62
```

2.2 UUID_Output.xlsx

UUID_Output.xlsx Workbook with a single Worksheet using the following formatting.

- Data filters
- Header column color
- Alternate column color
- Autofit columns

ReportName	LegalEntity	Username	ProcessProject	DeliveryMethod	EmailAddressTo	PrinterNetworkId	AX_AOSId	Time	ProcessTime
WMSPickingList_OrderPick	bot	bot\teresa.churchill		PrecisionForms screen			bot01AOS01@2712	7:21:02 AM	95.515
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	7:35:19 AM	459.7258
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	7:54:03 AM	120.8695
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	7:54:21 AM	126.0756
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	7:55:52 AM	133.6661
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	8:12:17 AM	131.6633
retaillabel	bot	bot\jonathan.giddings		PrecisionForms screen			bot01AOS01@2712	8:41:03 AM	534.383
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	8:53:26 AM	148.2745
retaillabel	bot	bot\jonathan.giddings		PrecisionForms screen			bot01AOS01@2712	8:57:58 AM	574.8327
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:08:16 AM	599.0997
retaillabel	bot	bot\mike.clark		PrecisionForms screen			bot01AOS07@2712	9:08:37 AM	359.7601
retaillabel	bot	bot\mike.clark		PrecisionForms screen			bot01AOS07@2712	9:09:44 AM	95.4396
PurchPurchaseOrder	bot	bot\claire.bradfield		PrecisionForms e-mail			bot01AOS02@2712	9:10:40 AM	761.4601
PurchPurchaseOrder	bot	bot\claire.bradfield		PrecisionForms e-mail			bot01AOS02@2712	9:10:46 AM	1339.9902
PurchPurchaseOrder	bot	bot\claire.bradfield		PrecisionForms e-mail			bot01AOS02@2712	9:11:09 AM	822.9454
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:11:45 AM	541.9016
retaillabel	bot	bot\jonathan.giddings		PrecisionForms screen			bot01AOS01@2712	9:15:22 AM	173.3461
SalesInvoice	bot	AX Batch PROD		PrecisionForms e-mail			bot01AOS04@2712	9:16:07 AM	503.3677
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:17:32 AM	579.8639
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:20:03 AM	625.6041
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:21:14 AM	555.9063
retaillabel	bot	bot\sarah.burnett		PrecisionForms screen			bot01AOS08@2712	9:21:27 AM	788.7045
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:28:20 AM	531.8223
retaillabel	bot	bot\mike.clark		PrecisionForms screen			bot01AOS07@2712	9:37:01 AM	238.5304

2.3 UUID_PivotTable.xlsx

UUID_PivotTable.xlsx created as a summary of the source CSV file. The sheet contains two Worksheets **csvData** and **Summary**.

NoOfReports	Column Labels			
Row Labels	PrecisionForms e-mail	PrecisionForms printer	PrecisionForms screen	Grand Total
FreeTextInvoice	114			114
PurchPurchaseOrder	4		6	10
retaillabel			117	117
SalesInvoice	307		2	309
SalesPackingSlip		62	25	87
WMSPickingList_OrderPick			31	31
Grand Total	**425**	**62**	**181**	**668**

csvData | Summary

PivotTable Fields
Choose fields to add to report:
Search
ReportName
LegalEntity
Username
ProcessProject
DeliveryMethod
EmailAddressTo
PrinterNetworkId
AX_AOSId
Time
ProcessTime
More Tables...
Drag fields between areas below:
Filters
Columns
DeliveryMethod
Rows
ReportName
Values
NoOfReports
Defer Layo...
Update

3 Pre-Requisites

Before we embark on creating this exercise you will require the following.

- **Transform Designer**
- **Microsoft Excel**

To run the branch once deployed, you will need to set the **Microsoft Excel Application** so it can run as an interactive user, perform the steps below to configure the Windows application.

1. From Start menu type **DCOMCNFG run as administrator**
2. Expand **Component Services** | **My Computer** | **DCOM Config**
3. Find **Microsoft Excel Application** right-click and open the **Properties**
4. Under the **Identity tab** set it to **"The interactive user".**

4 Create Transform branch

If you are ready, let's start building the branch to perform the CSV to XLSX conversion.

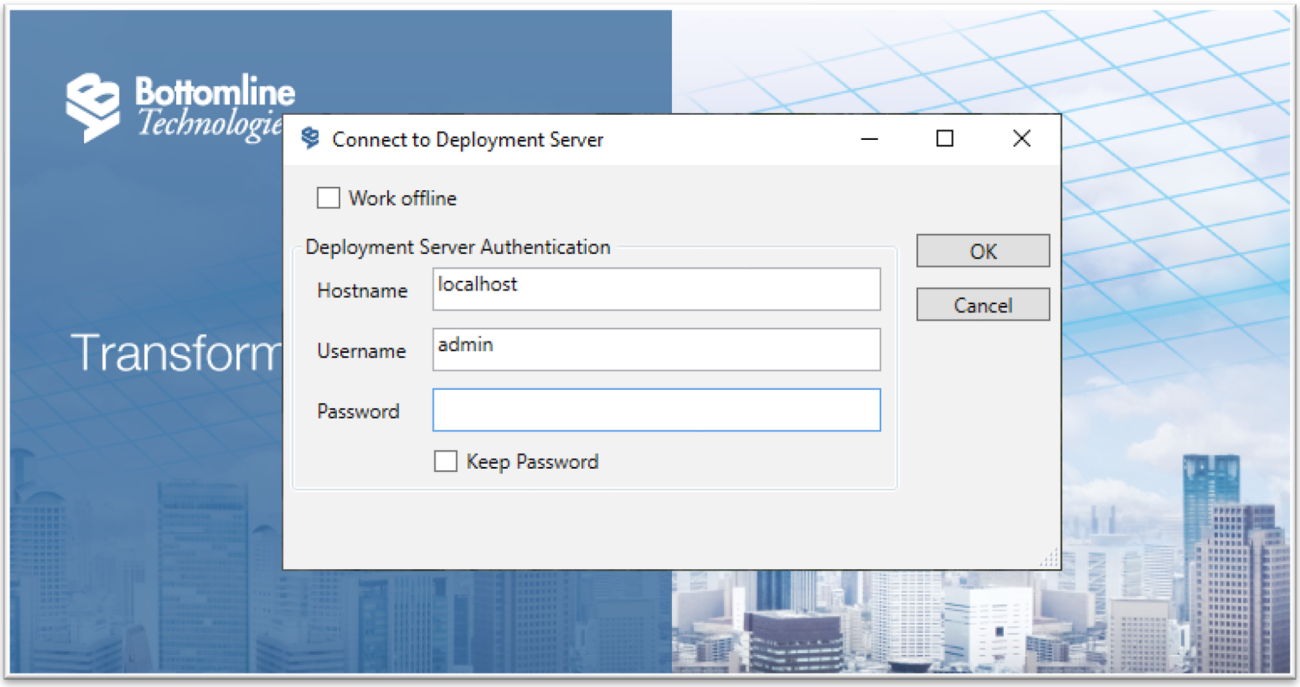
4.1 Open Transform Designer

From Windows Start menu navigate to **Bottomline Technologies** and select **Transform Designer**. You might be presented with the **Connect to Deployment Server** window, this is because to open Transform Designer you must have an available Designer license. The default credentials if the Transform Foundation Deployment server is installed locally is the following:

Hostname: localhost

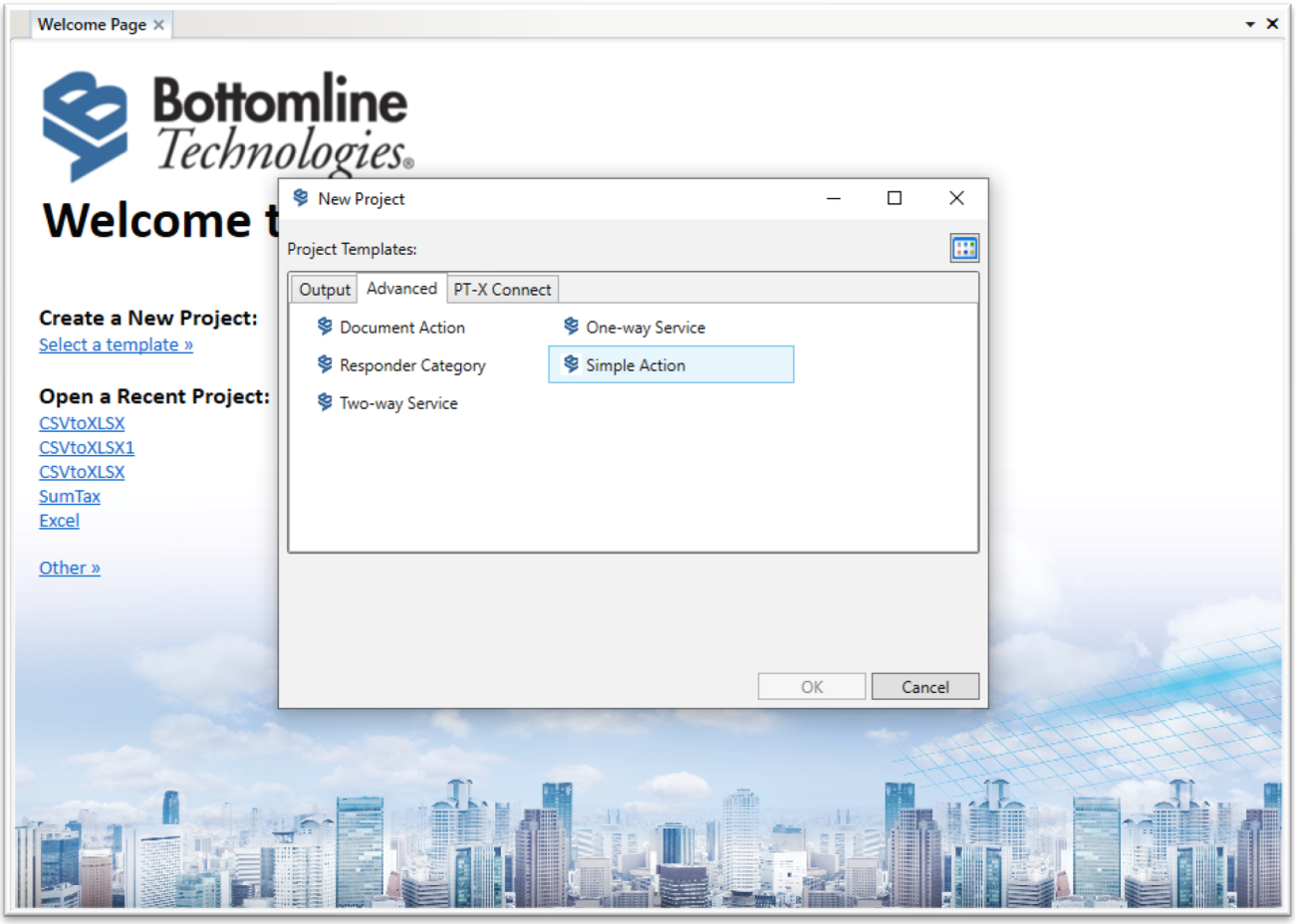
Username: admin

Password: administrator



4.2 Simple Action

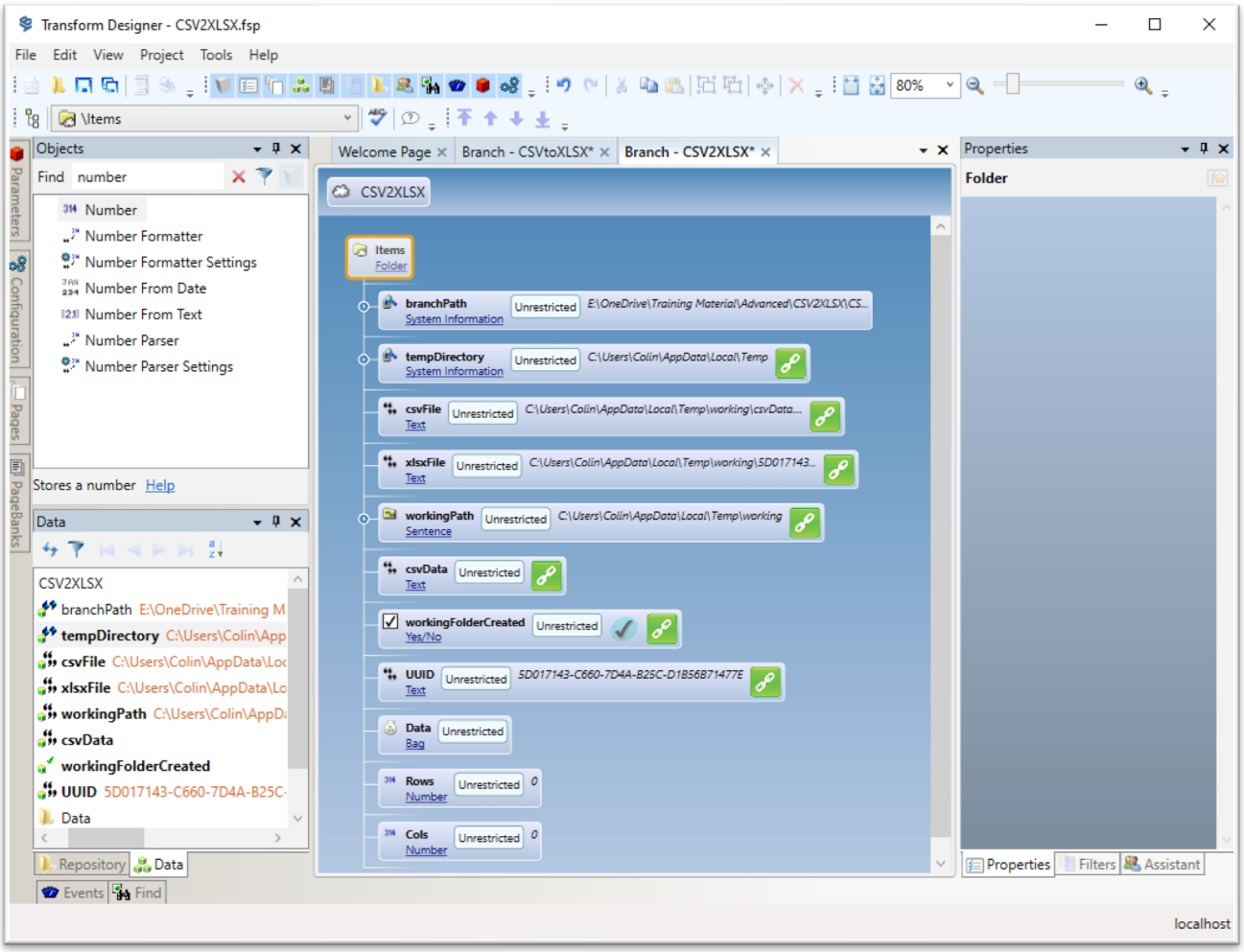
Once open, from Welcome page, under **Create a New Project** click Select a template >>, this will open the **New Project** window, navigate to the **Advanced** TAB and click **Simple Action**.



4.3 Memory objects

We will be using memory objects in the branch, from the Object Palette add the following objects under the Items Folder and label them as shown. To rename any object you can perform one of the following tasks, double-click the label, right-click and select rename, or press F2.

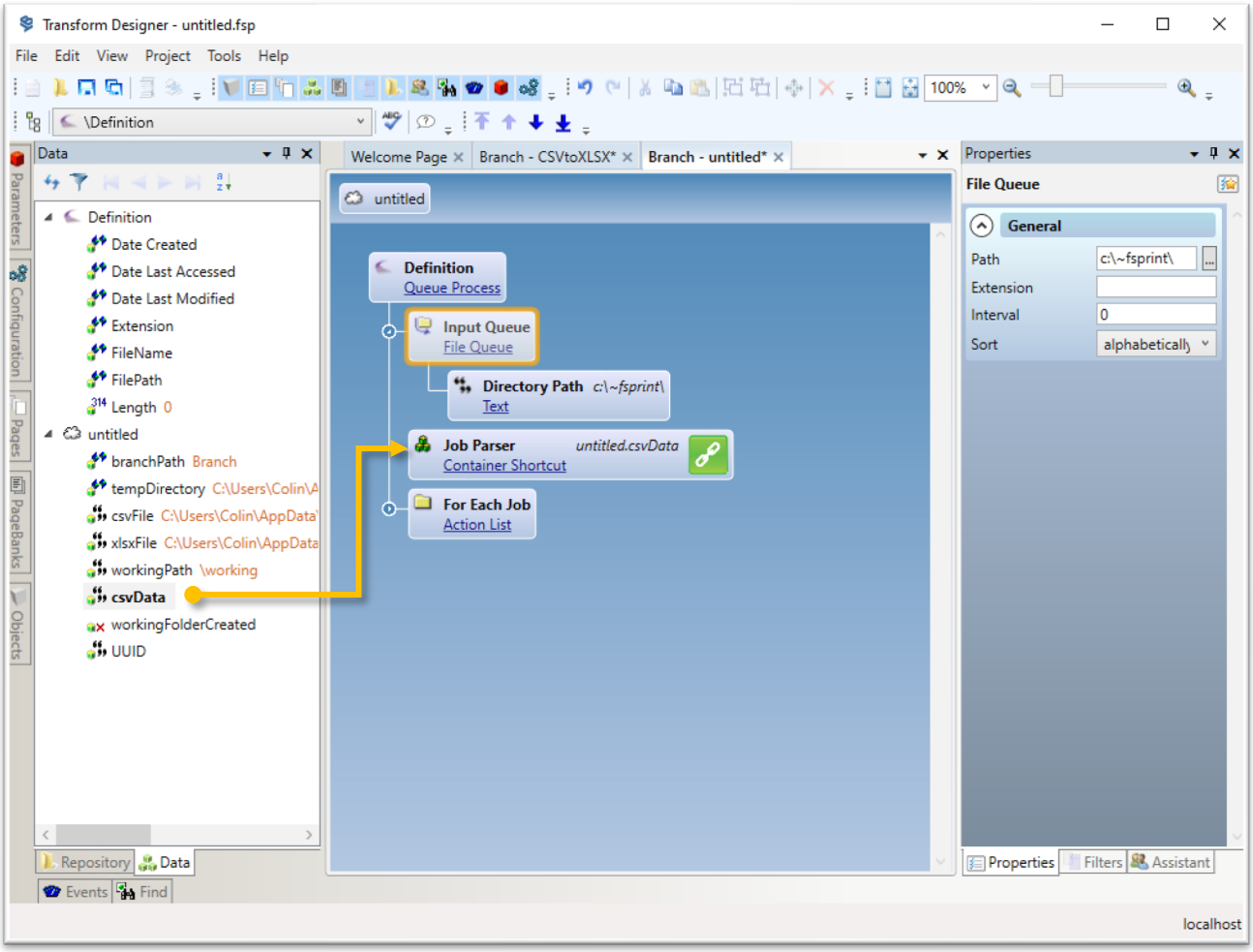
Object	Label	Value
System Information	**branchPath**	Set Properties Name to **branch.path** (returns the Transform branch path)
System Information	**tempDirectory**	Set Properties Name to **env.temp** (returns user Windows temp path)
Text	**csvFile**	Path and name of input CSV file
Text	**xlsxFile**	Path and name of output XLSX file
Sentence	**workingPath**	Path of working folder Link to tempDirectory & hard coded text \working
Text	**csvData**	Source raw CSV data
Yes/No	**workingFolderCreated**	No (boolean used to identify if working folder created)
Text	**UUID**	Unique identifier for used for output files
Bag	**Data**	Container to store parsed CSV data
Number	**Rows**	Number of RECORDS in the RECORDSET
Number	**Cols**	Number of fields in each RECORD



4.4 Queue Process

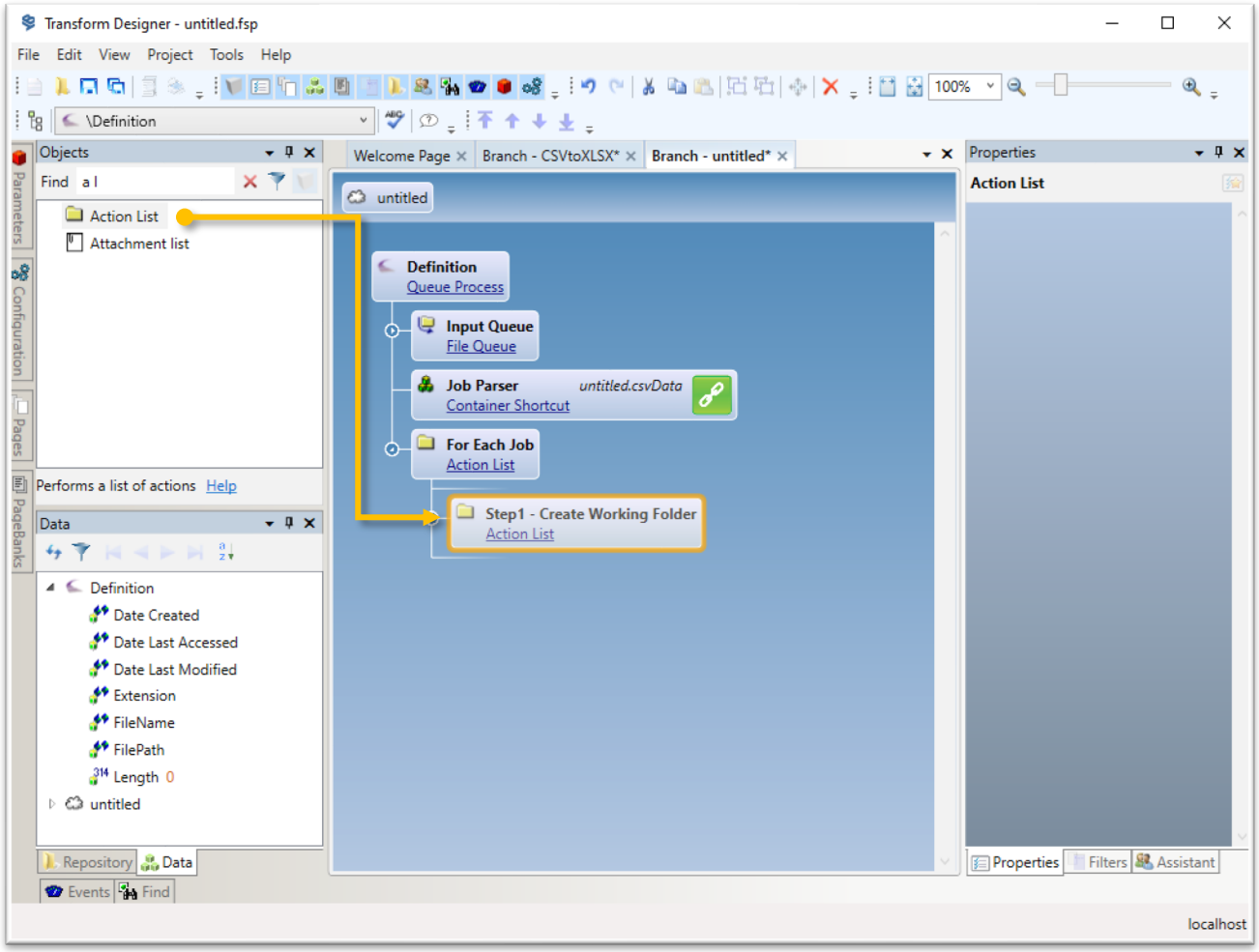
If you would like the branch to run as a centralized process, then replace the Definition Action List object with a Queue Process. Then change the Queue Process child objects to the following.

Object	Label	Value
File Queue	Input Queue	**Directory Path** – Watch folder you would like the process to pick up CSV files from
Container Shortcut	Job Parser	**Container Shortcut** – mapped to **csvData**



4.5 Step 1 – Create Working Folder

Expand the **For Each Job** Action List and from the Object Palette add a child Action List, rename it to **Step1 – Create Working Folder**.

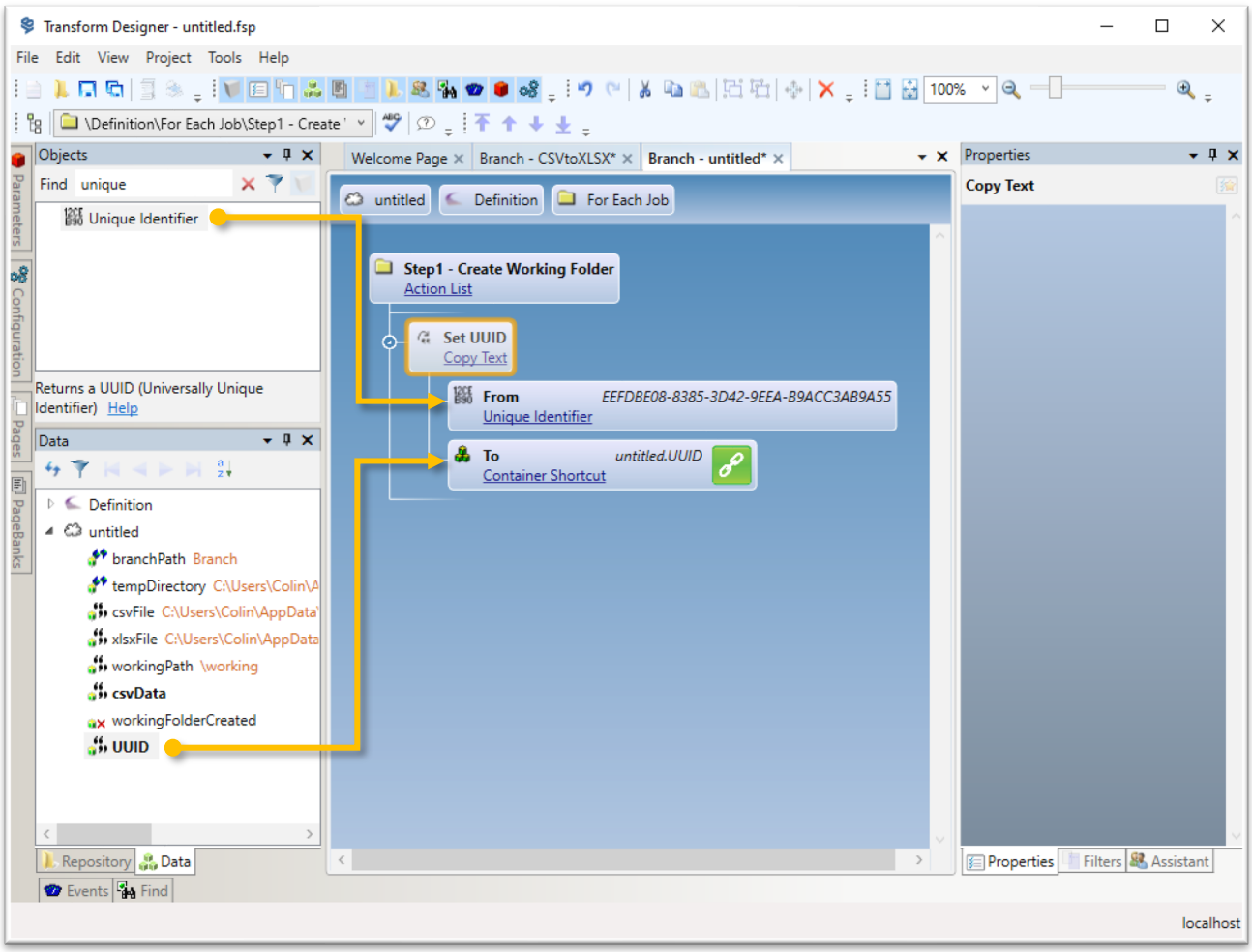


Right-click the Action List labelled **Step1 – Create Working Folder** and select Focus here so we can concentrate on this object in the branch editor, docking the ancestors as breadcrumbs across to top of the branch editor.

4.5.1 Set UUID

The first thing we need to do is set the value of the **UUID** memory Text object that will be used for the output filenames to ensure they are unique and do not overwrite the XLSX files each time the process runs. From the object palette select and add the following object:

Object	Name	Value
Copy Text	Set UUID	**From** – Unique Identifier object **To** – Container shortcut mapped to **UUID**

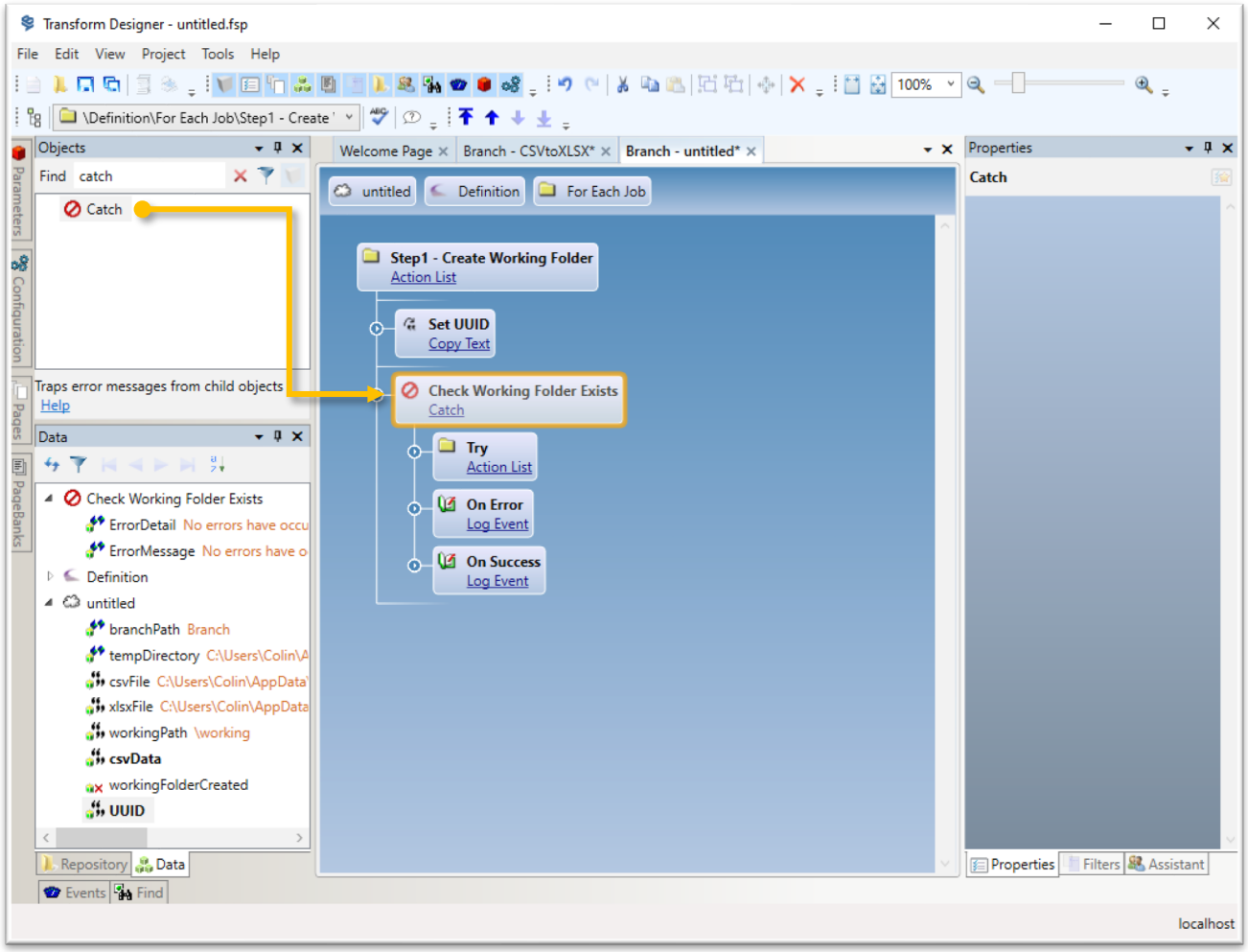


4.5.2 Create Working Folder

To ensure Excel can access the CSV and XLSX files to process them, we need to create a working folder on the local file system. To perform this action, we are going to make use of the Catch object that allows us to check if the Directory already exists and if it doesn't, we will create it.

On the add point beneath the **Set UUID** Copy Text object add the following:

Object	Label	Value
Catch	Check Working Folder Exists	N/A



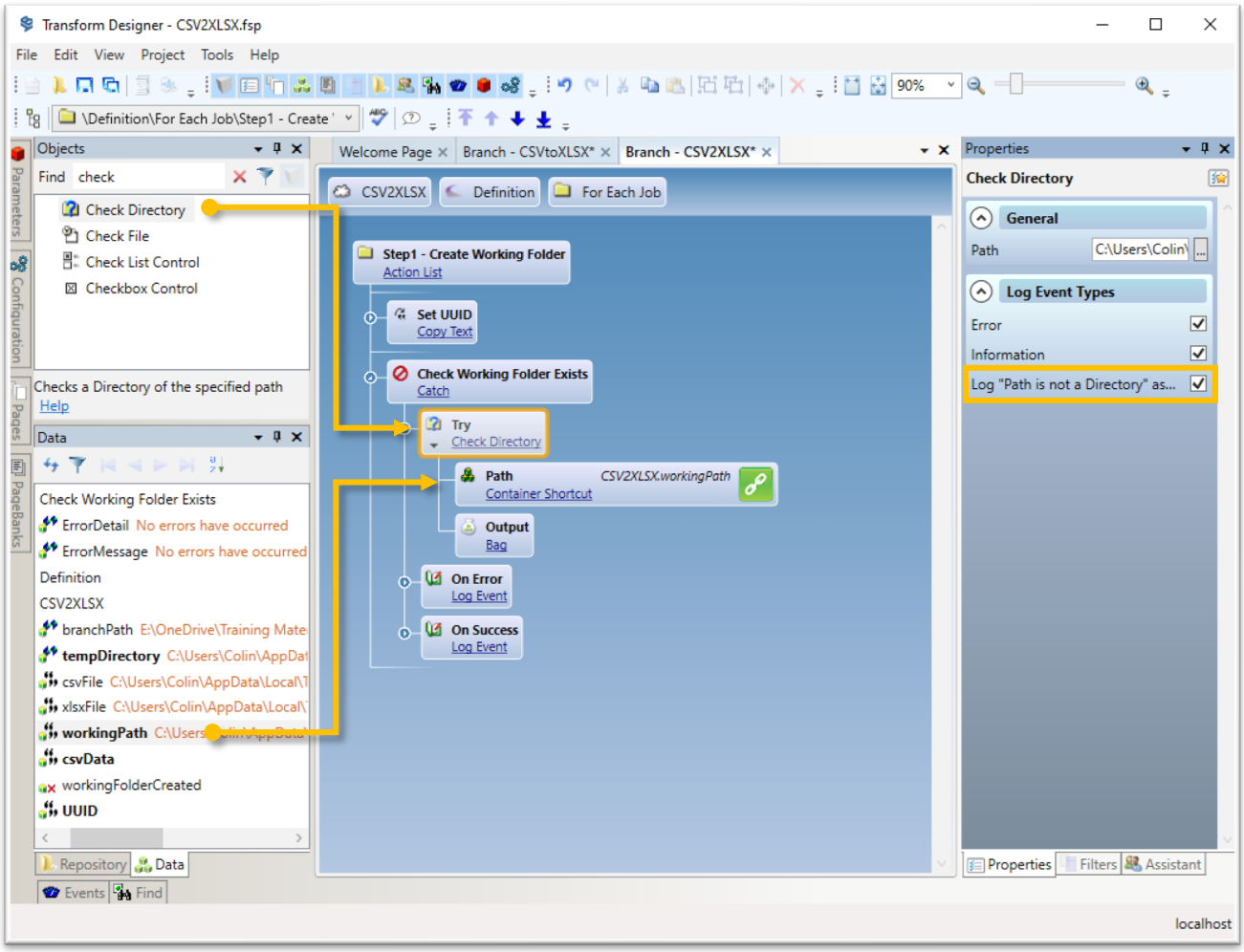
4.5.3 Check Directory

Replace the **Try** with the following object:

Object	Label	Value
Check Directory	Try	**Path** – Container shortcut mapped to **workingPath**

So that we can utilize the Catch object we must check **Log"Path is not a Directory" as error** within the Check Directory **Log Event Types** properties.

Note: When this option is unselected (the default), logs the Path is not a directory event as an informational message in the Events window or log file. Which means we would **NOT** be able to use the Catch parent object to trap the error to perform the next action.

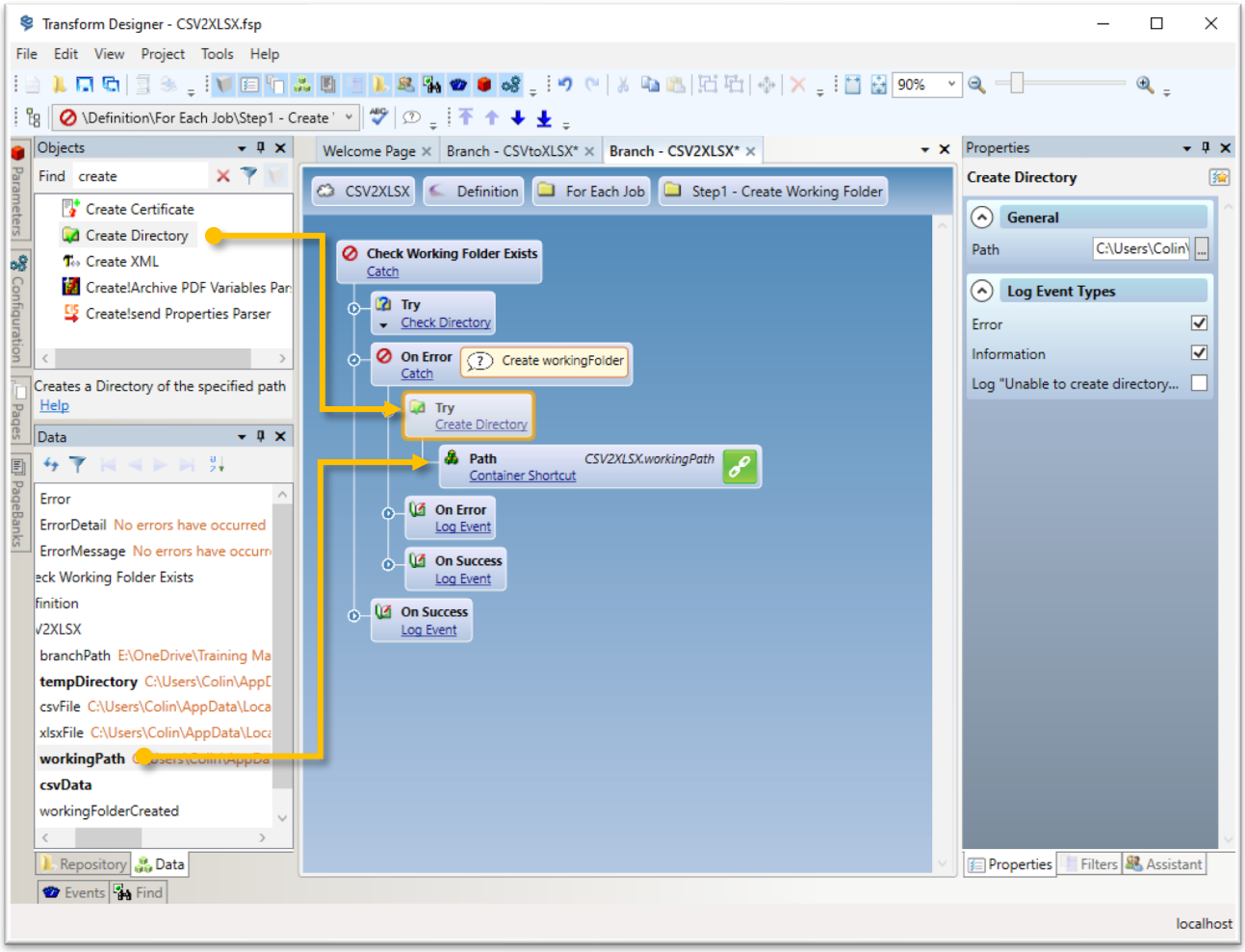


4.5.4 Create Directory

If the Check Directory returns an error, then the parent Catch object will execute the **On error** Log Event. Replace this Log Event object with another Catch object so you can attempt to Create the working folder using a Create Directory object.

Replace the Catch child Log Event object with the following:

Object	Label	Value
Create Directory	Try	**Path** – Container shortcut to **workingPath**

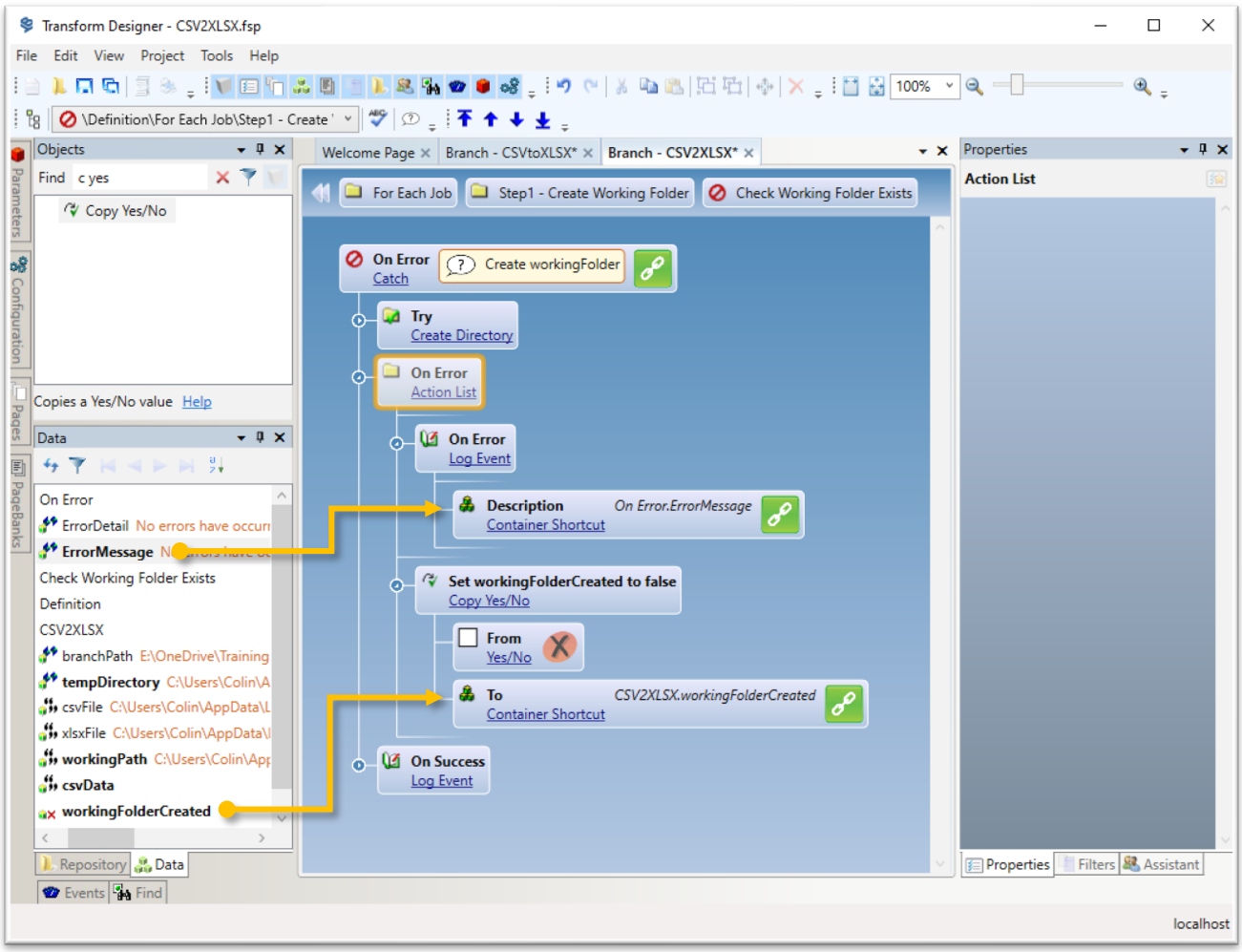


4.5.5 Create Directory – Error

To avoid the branch from attempting to execute the script if the working folder was not created change the **On Error** Log event to an Action List.

Expand the Action List and add the following:

Object	Label	Value
Log Event (adopted)	On Error	**Description** – Container shortcut mapped to **On Error.ErrorMessage**
Copy Yes/No	Set workingFolderCreated	**From** – No **To** – Container shortcut mapped to **workingFolderCreated**

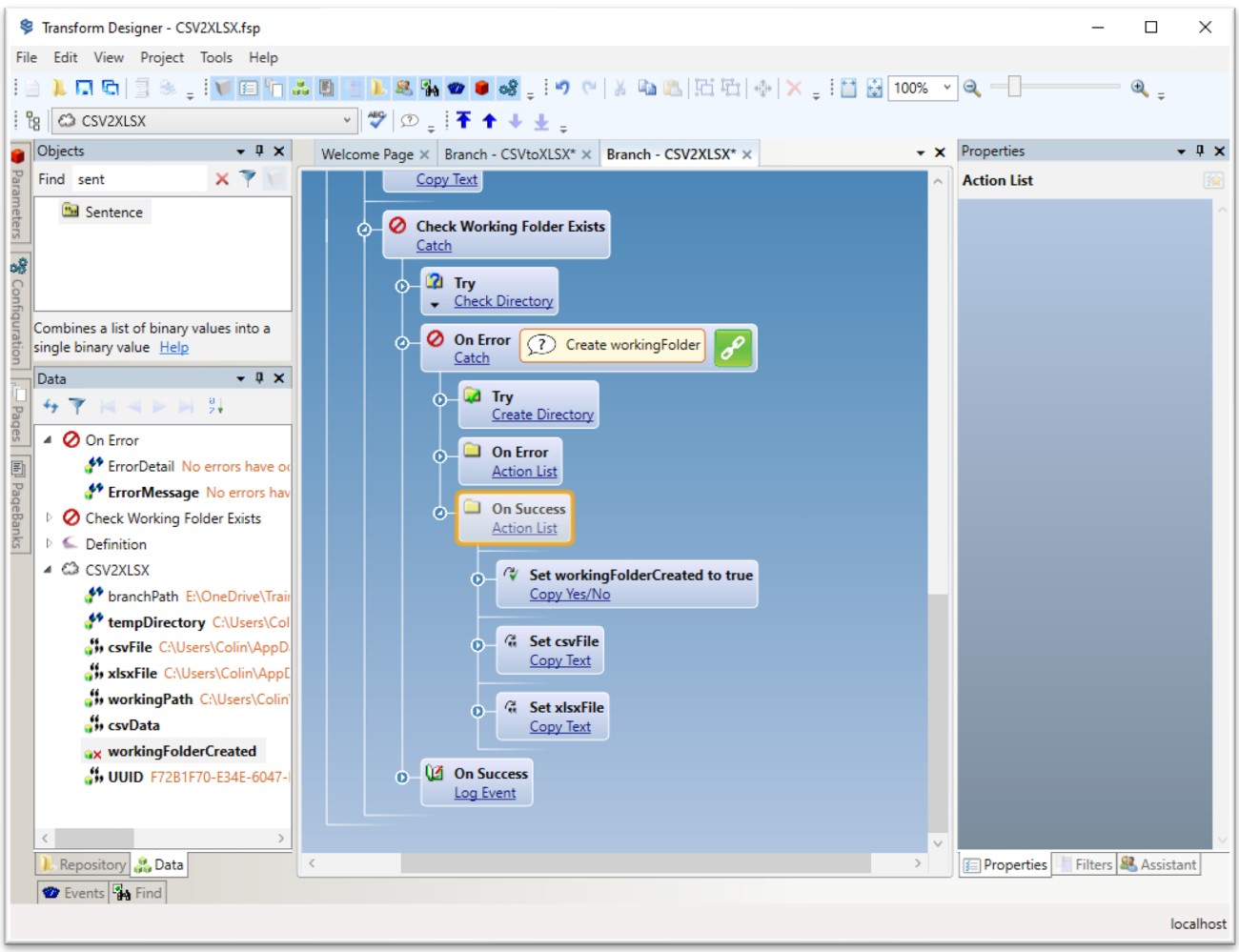


4.5.6 Create Directory - On Success

If the working folder is created successfully then the parent Catch will execute the **On Success** Child.

When the folder is created, we need update some of the memory objects, so replace the Log Event with an Action List and add the following objects.

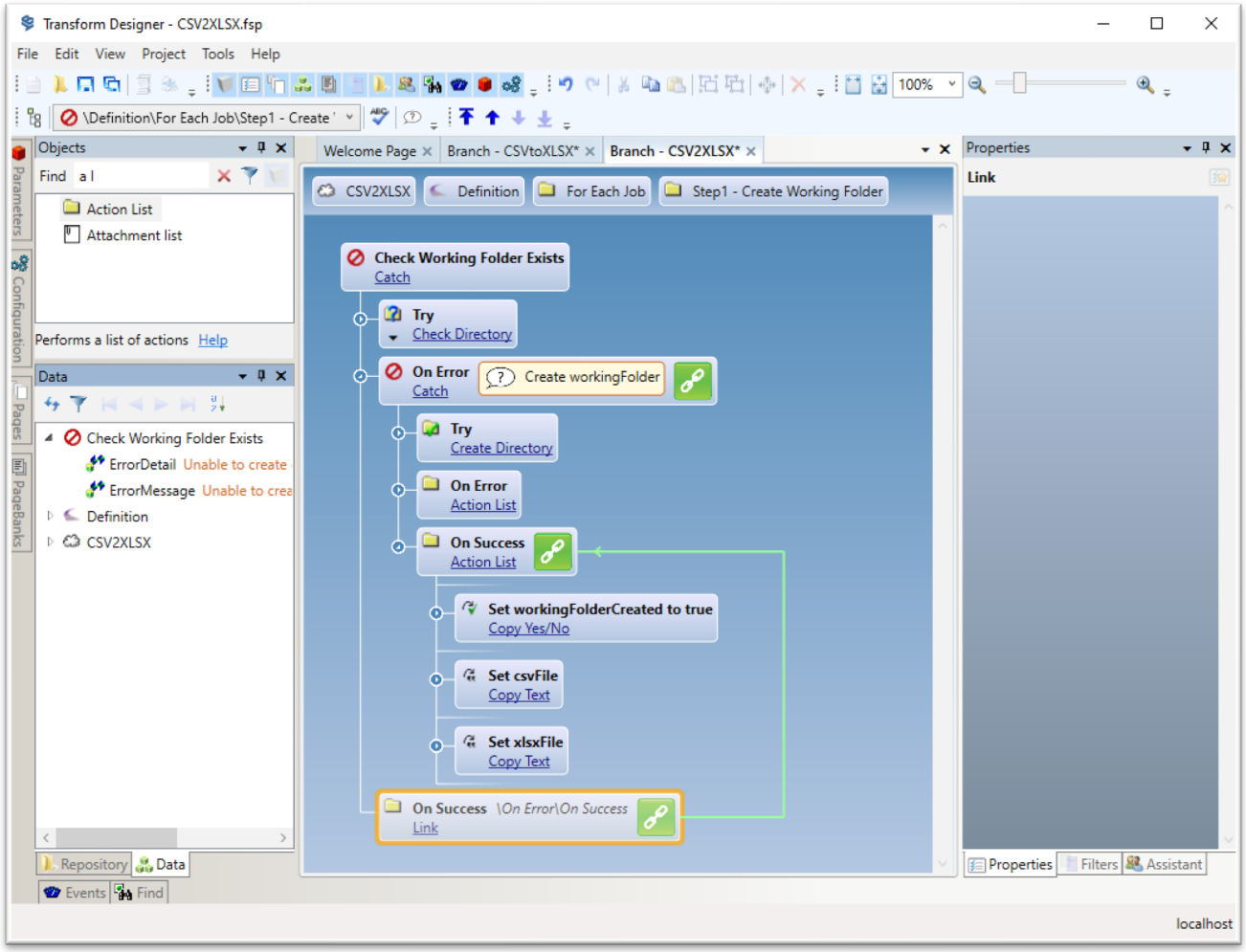
Object	Label	Value
Copy Yes/No	Set workingFolderCreated to true	**From** – Yes **To** – Container shortcut to **workingFolderCreated**
Copy Text	Set csvFile	**Sentence object** **Piece1** - Container shortcut to **workingPath** **Piece2** – Text "\csvData.csv"
Copy Text	Set xlsxFile	**Sentence object** **Piece1** – Container shortcut to **workingPath** **Piece2** – Text "\" **Piece3** – Container shortcut to **UUID** **Piece4** – Text "_Output.xlsx"



4.5.7 Check Directory – On Success

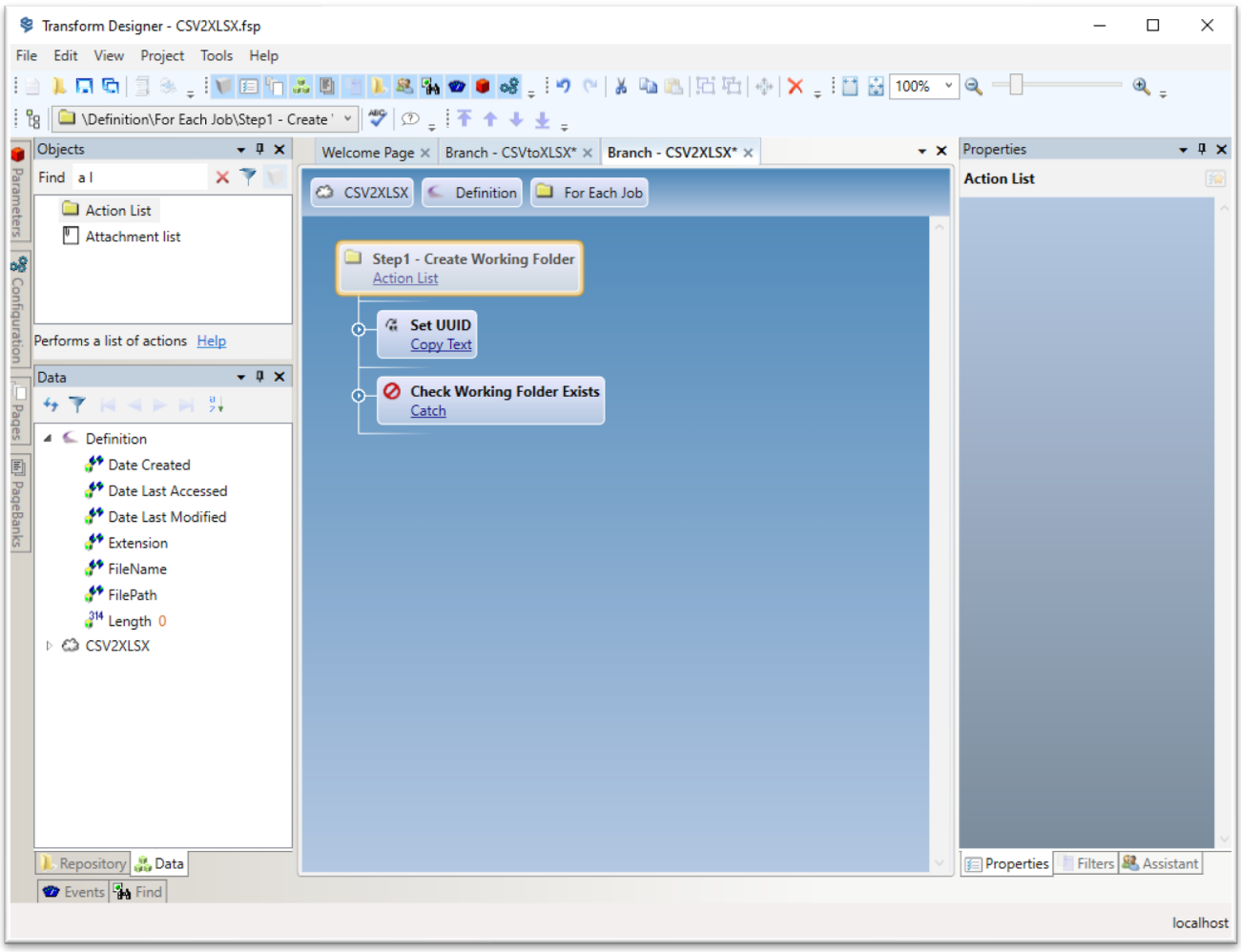
If the **workingFolder** already exists then the parent Catch object labelled **Check Working Folder Exists** will execute the **On Success** Log Event, we need to perform the same actions we created in the child Catch object **On Success** Action List.

To avoid having to add the same objects again we can create a link to the other actions in the branch. If you right-click the Inner **On Success** Action List and holding down the right mouse button drag it onto the parent **On Success** Log Event and select **Create Link**.

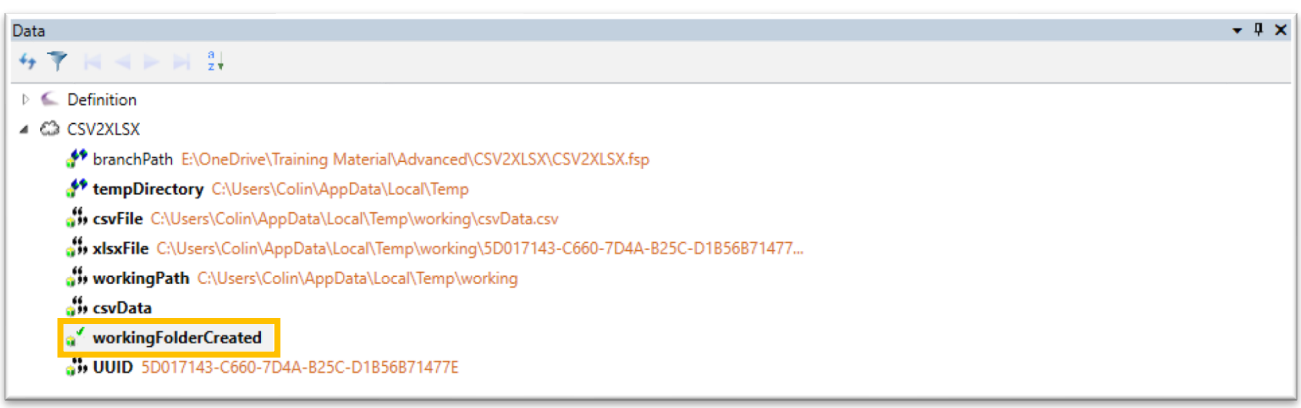


4.5.8 Run Step1

Now we have completed the routine to check and create the working directory, so let's go ahead and test it. Navigate the branch to expose **Step1 – Create Working Folder** Action List and run it. To run any Action object/s you can either right-click and select **Run** from the right click menu or select the object and press F5.

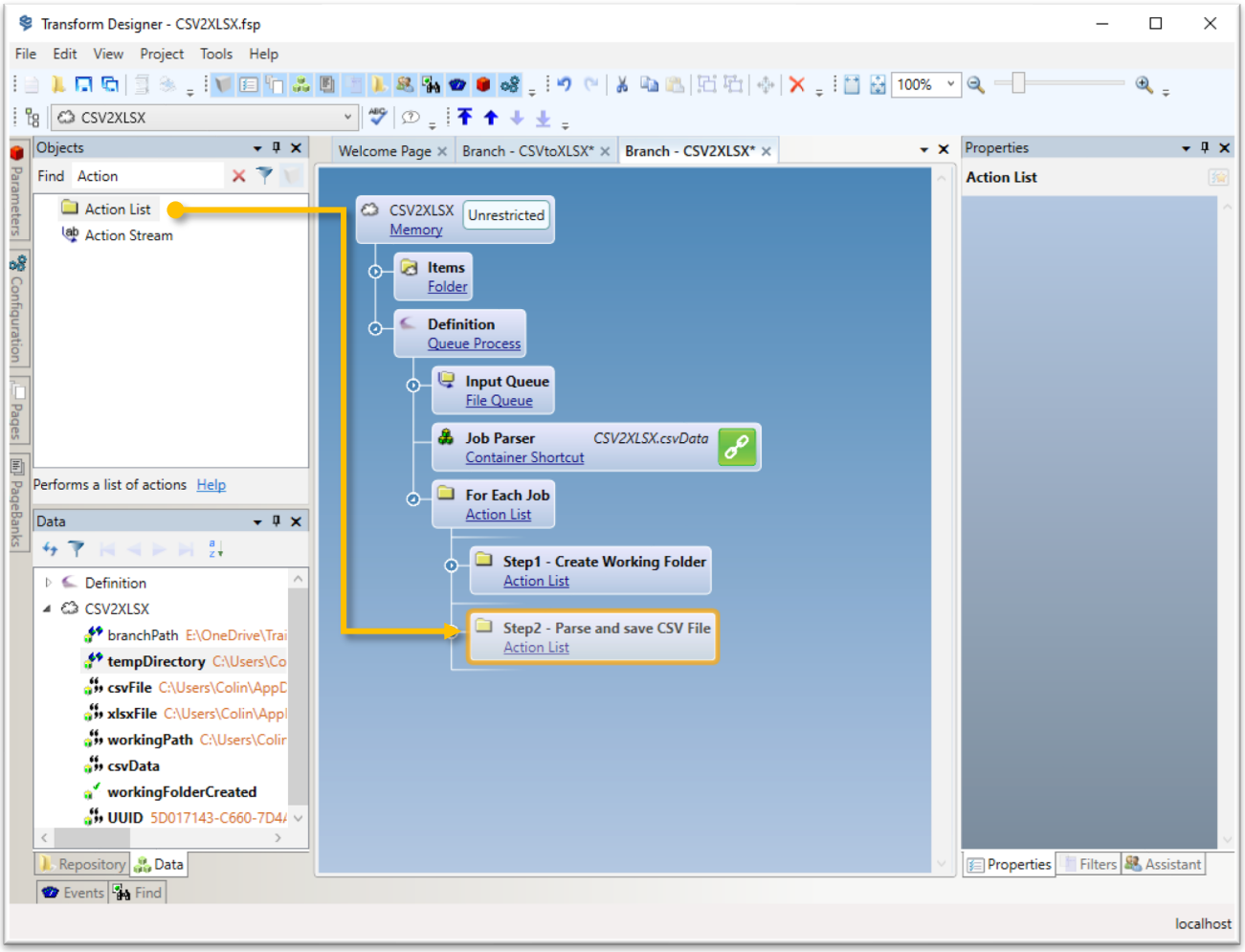


Then open Windows File Explorer and check to see if the working directory has been created. Also, you can check to see that the **workingFolderCreated** Yes/No object is set to True.



4.6 Step2 – Parse and save CSV file

At the add point beneath **Step1 – Create Working Folder** Action List add another Action List and rename to **Step2 – Parse and save CSV file**.

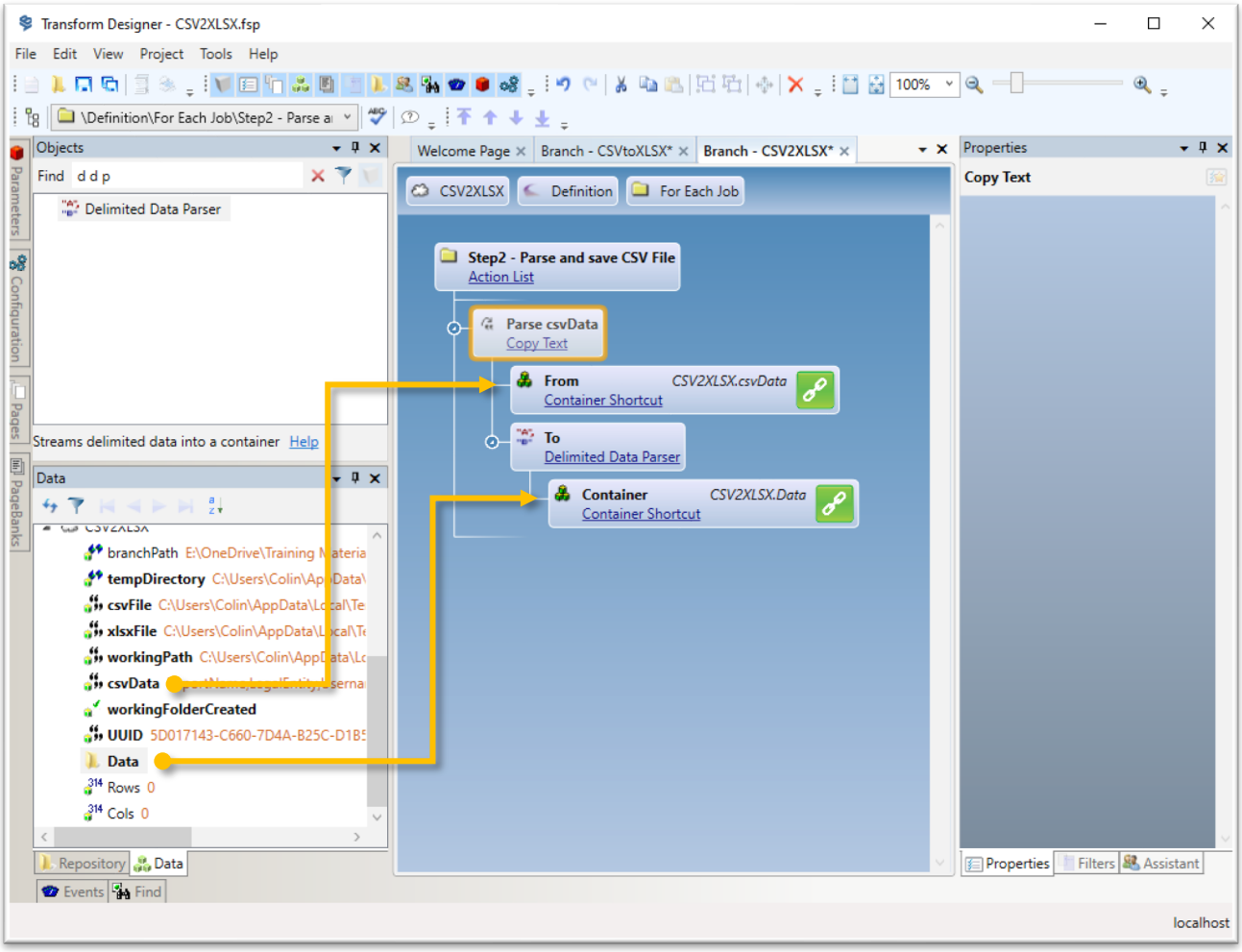


Right-click the Action List labelled **Step2 – Parse and save CSV File** and select Focus here so we can concentrate on this object on the branch editor.

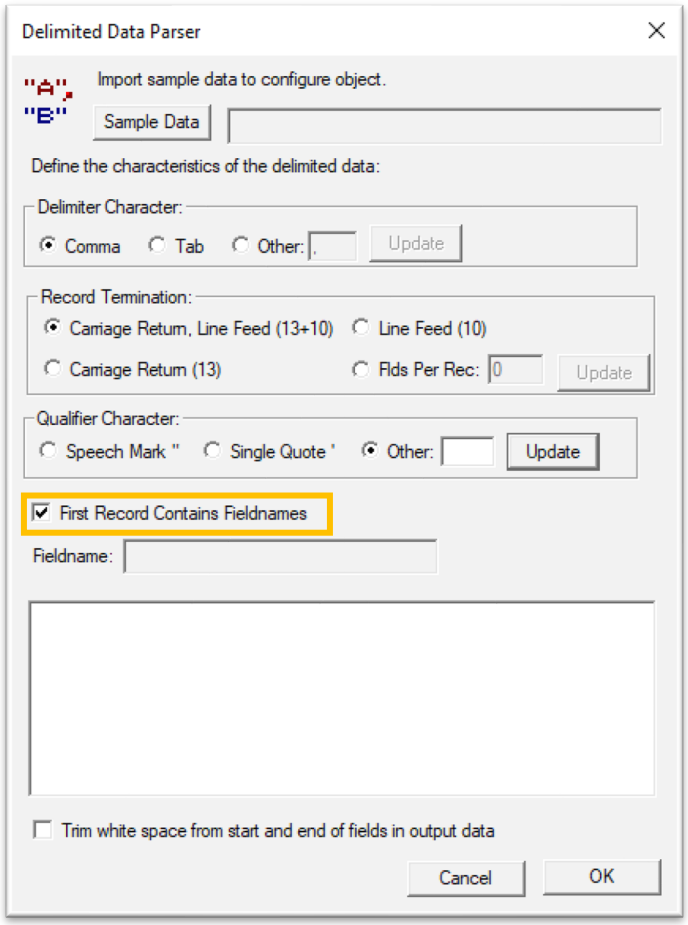
4.6.1 Parse csvData

First, we will parse the **csvData** to a Container Bag object labelled **Data**. Expand the Action List and add the following object

Object	Label	Value
Copy Text	Parse csvData	**From** – Container shortcut map to **csvData** **To** – Delimited Data Parser object with child Container shortcut map to **Data**



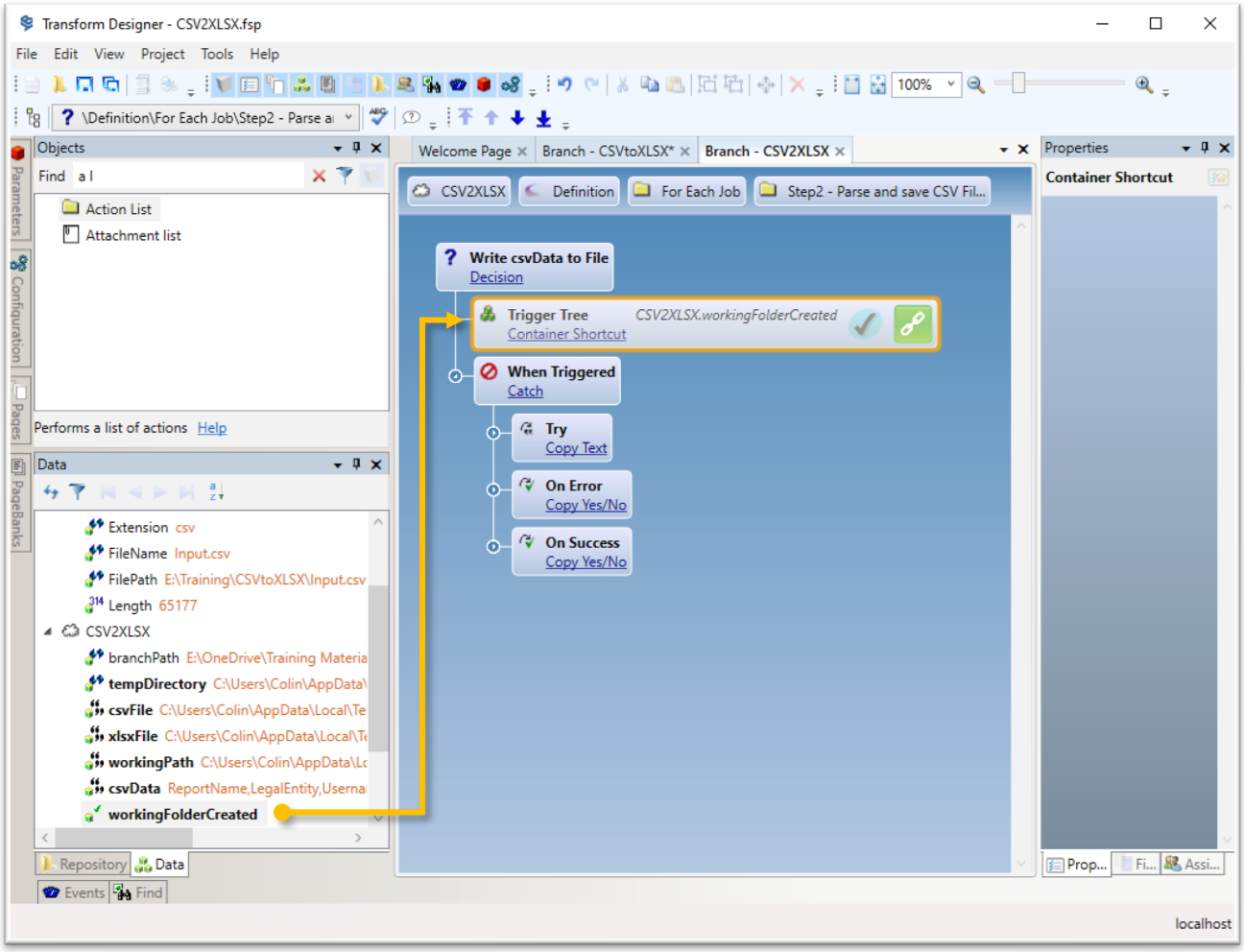
We need to configure the Delimited Data Parser object. To configure the object either click on the Delimited Data Parser object icon or click the Advanced Properties tool in the Properties window. This opens the Delimited Data Parser window where you can Define the characteristics of the delimited data, just check **First Record Contains Fieldnames,** and click **OK** to commit the settings.



4.6.2 Decision - Write csvData to File

Beneath the Action List labelled **Parse csvData** add a Decision object to decide if we should attempt to write the **csvData** to the working folder we created in Step1.

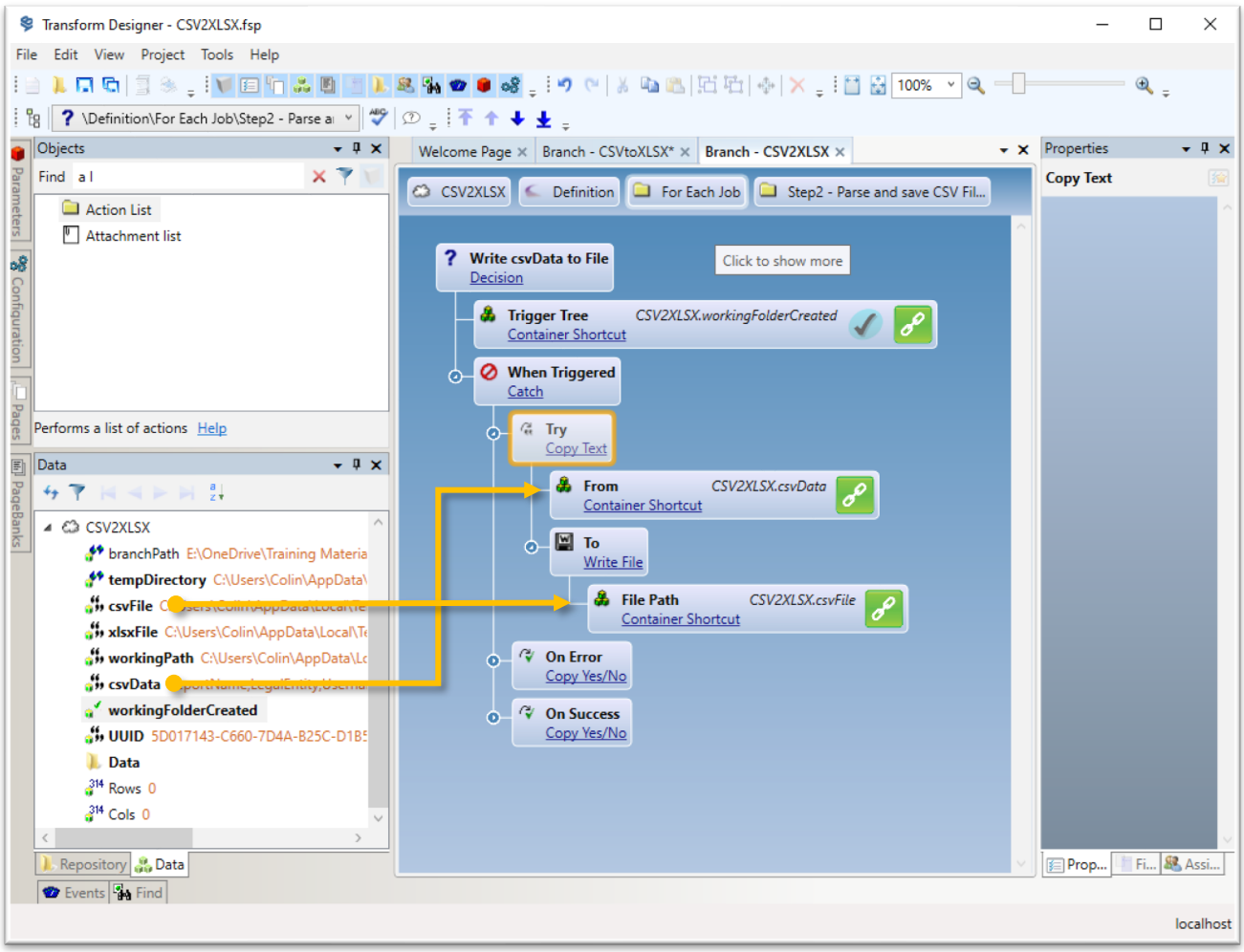
Object	Label	Value
Decision	Write csvData to File	**Trigger Tree** – Container shortcut mapped to **workingFolderCreated**



4.6.3 Catch – When Triggered

When the **workingFolderCreated** returns true then we will attempt to write the **csvData** to the filesystem within another Catch object so we can trap any errors to stop us attempting to execute the VBScript. Replace the object labelled **When Triggered** with a Catch object. Then replace the object labelled **Try** to the following.

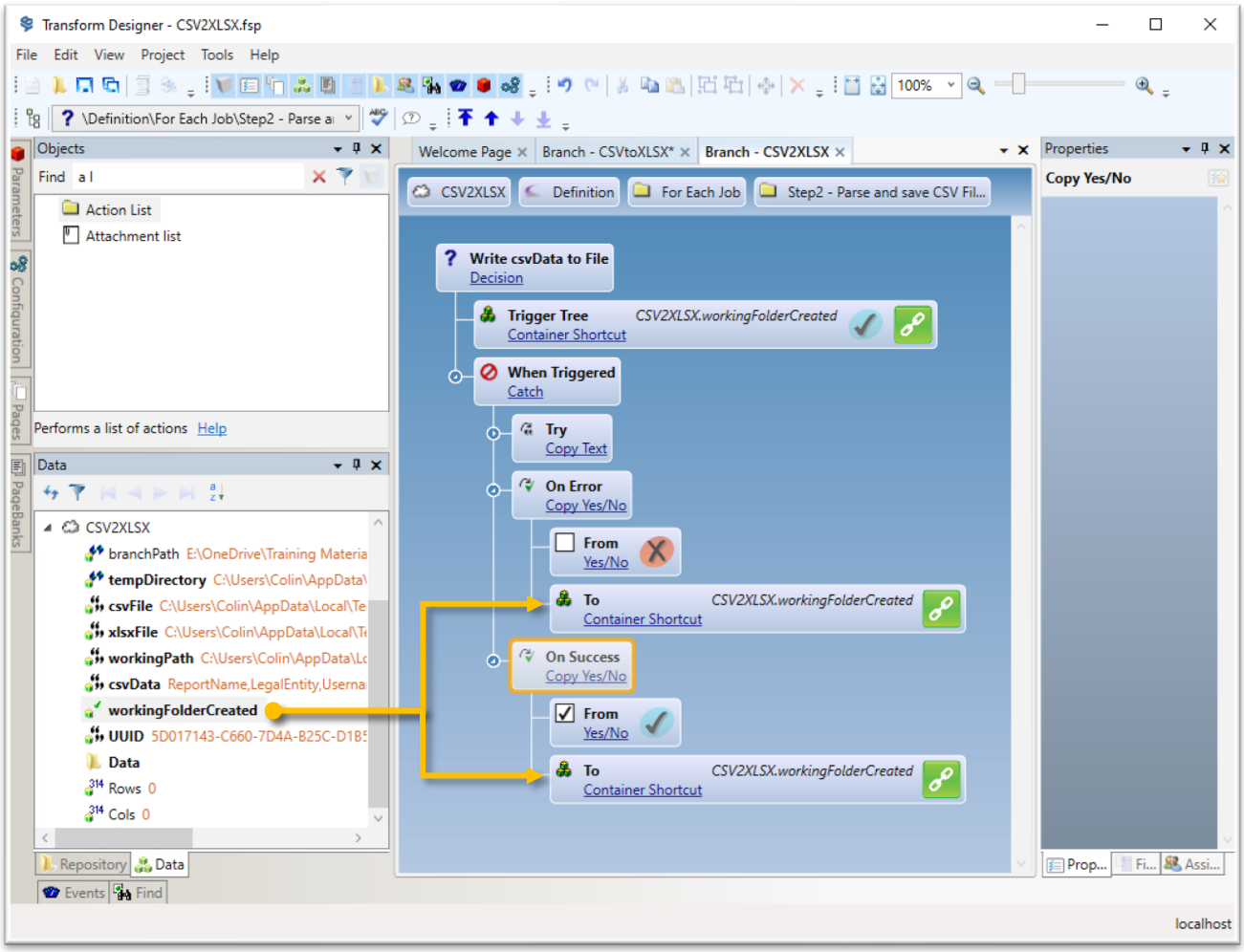
Object	Label	Value
Copy Text	Write csvData to File	**From** – Container shortcut mapped to **csvData** **To** – Write File object with child path Container shortcut mapped to **csvFile**



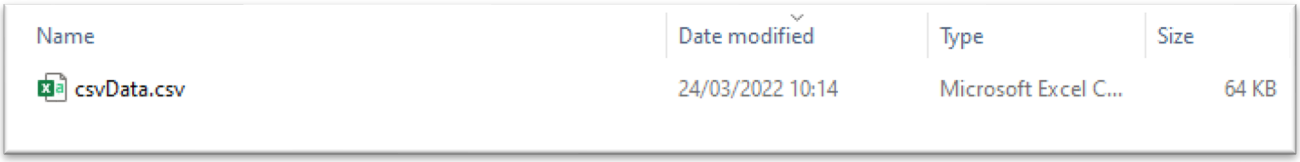
4.6.4 Catch – On Error and On Success

Replace the **On Error** and **On Success** Log Event objects to the following

Object	Label	Value
Copy Yes/No	On Error	**From** – No **To** – Container shortcut mapped to **workingFolderCreated**
Copy Yes/No	On Success	**From** – Yes **To** – Container shortcut mapped to **workingFolderCreated**



Navigate to **Step2 – Parse and save CSV File** and Run the Action List, then from Windows File Explorer browse to the working folder and see if the **csvData.csv** file has been written.

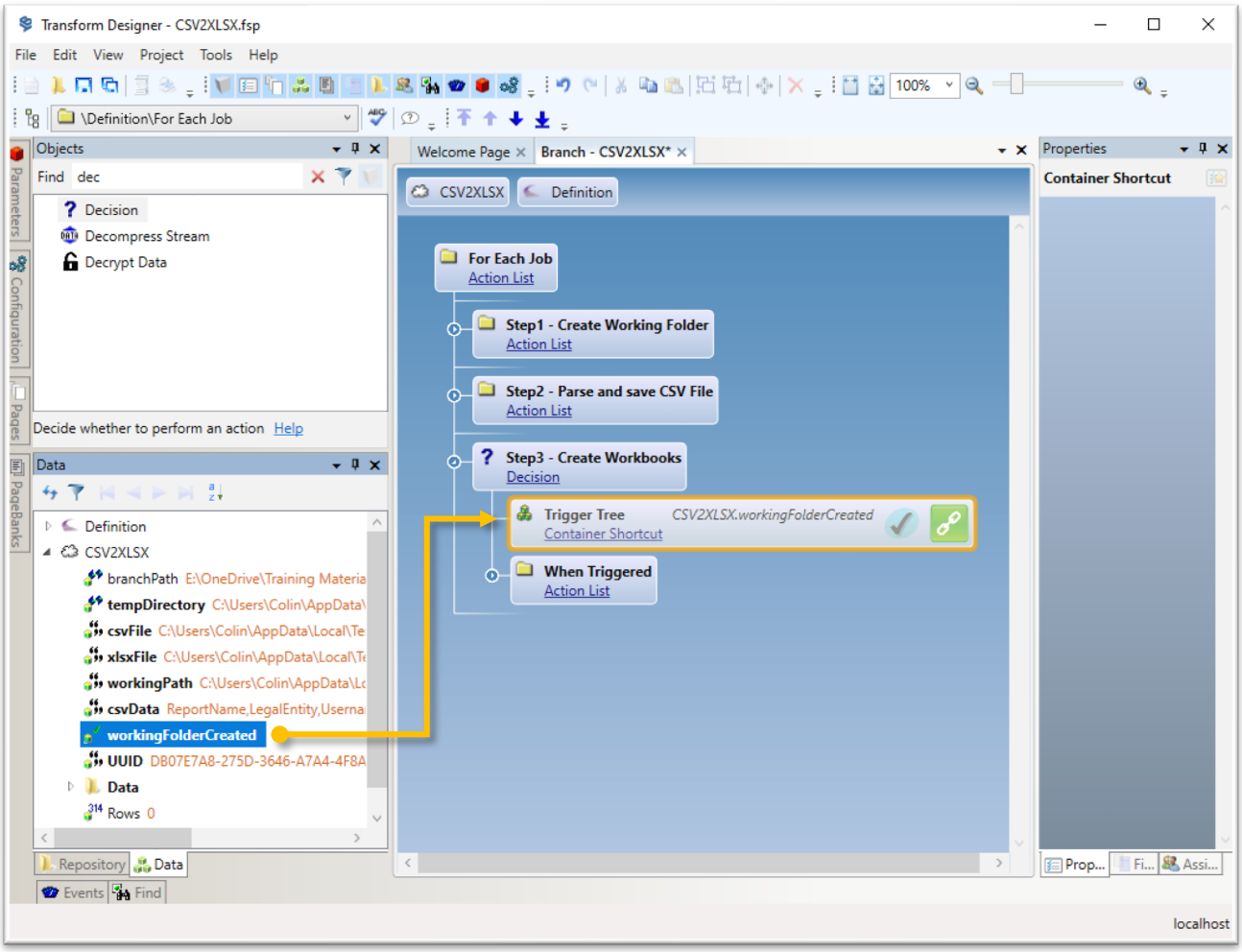


4.7 Step3 – Create Workbooks

Now we have prepared the environment we are now ready to execute our VBScripts to call Microsoft Excel. It is important that we do not attempt to call the Excel Application if the source **csvData.csv** file does not exist. Well, we could use the Check File object that checks if a specific file exists on the filesystem, but we already made use of the Catch object in **Step2** when we attempted to write the **csvData.csv** file and set the value of **workingFolderCreated** Yes/No object in memory depending on the outcome of the Catch. So, let's make use of this memory object to decide if we should execute the Scripts.

At the add point under the **Step2 – Parse and save CSV File** Action List and a Decision Object.

Object	Label	Value
Decision	Step3 – Create Workbooks	**Trigger Tree** – Container shortcut mapped to **workingFolderCreated**

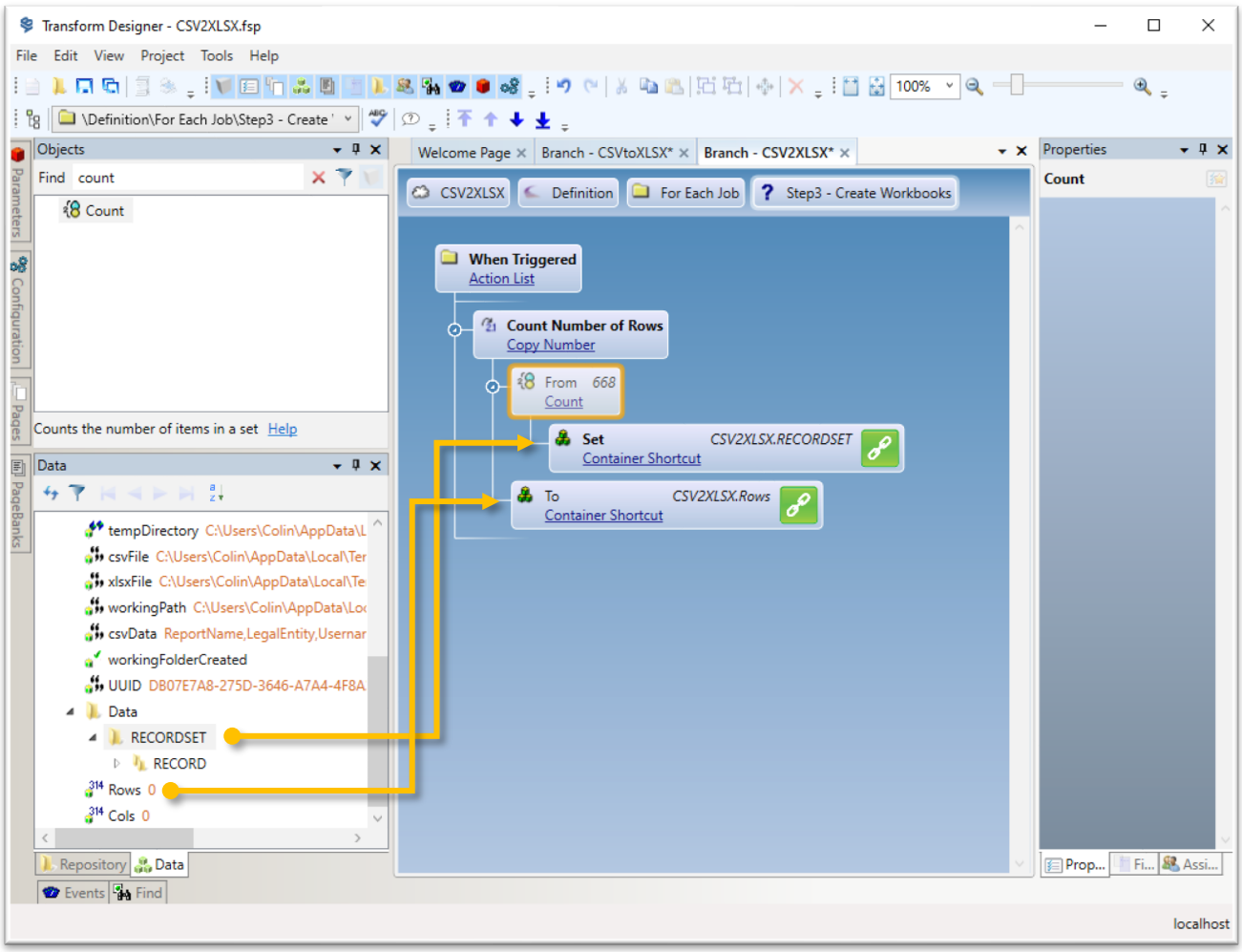


4.7.1 Count Rows

So that we can add formatting to the rows in the Worksheet we need to know how many RECORDS are in the RECORDSET within the **Data** container Bag object we parsed the **csvData** to.

Expand the **When Triggered** Action List and add the following.

Object	Label	Value
Copy Number	Count Number of Rows	**From** – Count **Set** mapped to **Data > RECORDSET** **To** – Container shortcut to **Rows**



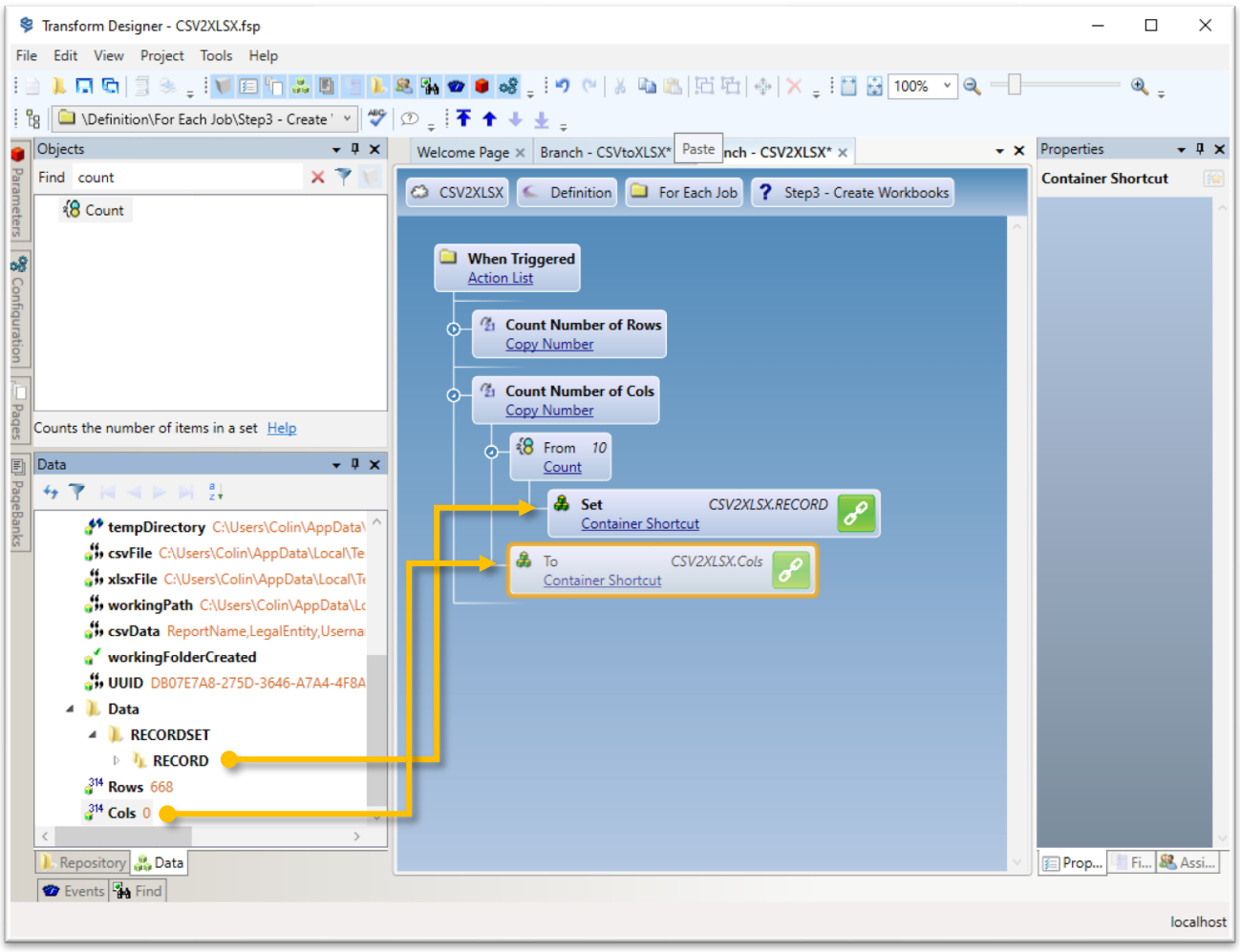
Right click and Run the Copy Number **Count Number of Rows** to update the **Rows** Number memory object.

4.7.2 Count Columns

So that we can add formatting to the columns in the Worksheet we need to know how many Fields are in each RECORD within the **Data** container Bag object we parsed the **csvData** to.

At the add point under the Copy Number labeled **Count Number of Rows** add the following.

Object	Label	Value
Copy Number	Count Number of Cols	**From** – Count **Set** mapped to **Data > RECORDSET > RECORD** **To** – Container shortcut to **Cols**



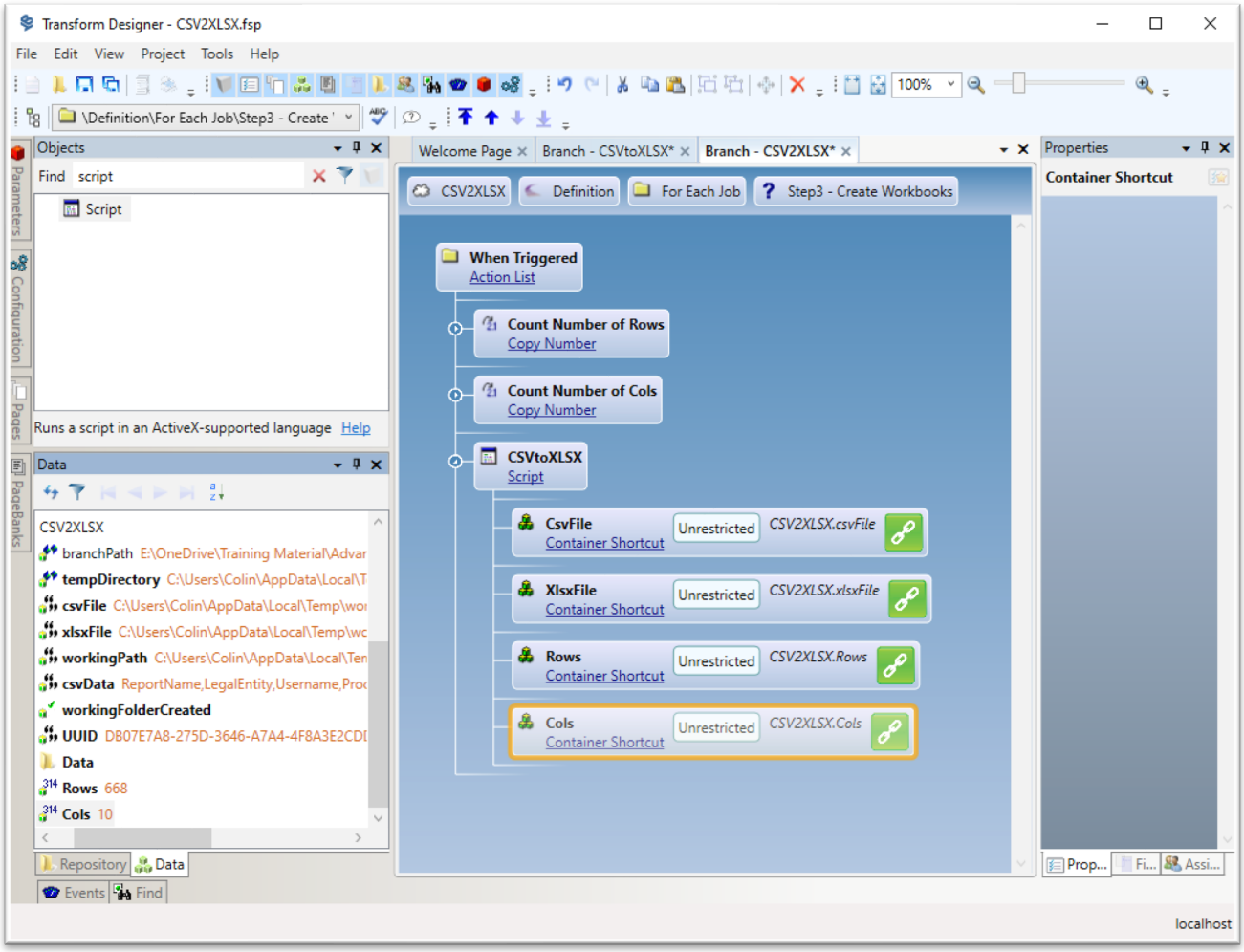
Right click and Run the Copy Number **Count Number of Cols** to update the **Rows** Number memory object

4.7.3 Convert CSV to XLSX

Well done, you have made it to the exciting bit. It is time to execute the Script to convert our **csvData.csv** file into a xlsx file with formatting.

At the add point under the Copy Number labeled **Count Number of Cols** add the following

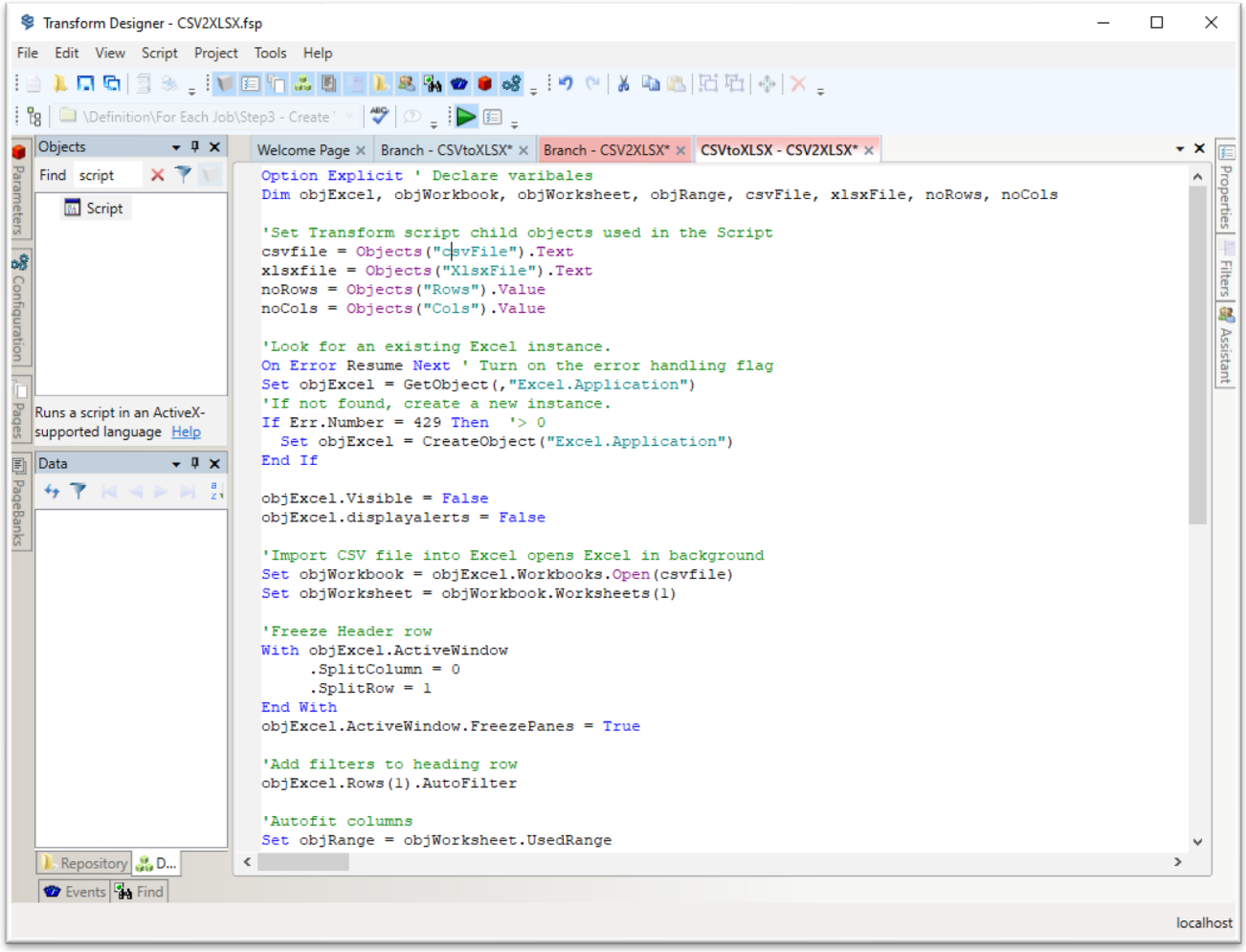
Object	Label	Value
Script	CSVtoXLSX	Script Variables **CsvFile** - Container shortcut – mapped to **csvFile** **XlsxFile** - Container shortcut – mapped to **xlsxFile** **Rows** - Container shortcut – mapped to **Rows** **Cols** - Container shortcut – mapped to **Cols**



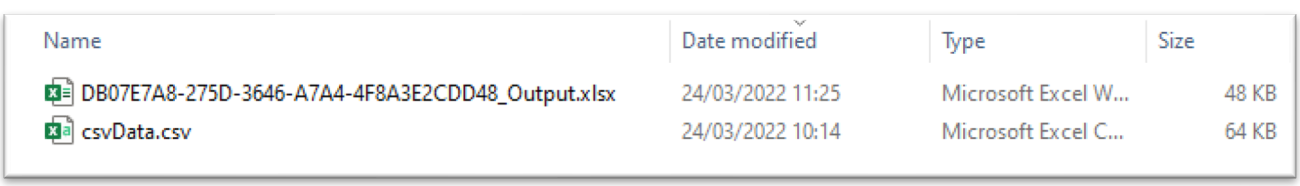
Open the Script editor, either click on the Script object icon or use the Advanced properties tool. Once open paste the VBScript below into the Script editor window.

```
Option Explicit ' Declare varibales
Dim objExcel, objWorkbook, objWorksheet, objRange, csvFile, xlsxFile, noRows, noCols

'Set Transform script child objects used in the Script
csvfile = Objects("CsvFile").Text
xlsxfile = Objects("XlsxFile").Text
noRows = Objects("Rows").Value
noCols = Objects("Cols").Value

'Look for an existing Excel instance.
On Error Resume Next ' Turn on the error handling flag
Set objExcel = GetObject(,"Excel.Application")
'If not found, create a new instance.
If Err.Number = 429 Then  '> 0
  Set objExcel = CreateObject("Excel.Application")
End If

objExcel.Visible = false
objExcel.displayalerts = false

'Import CSV file into Excel opens Excel in background
Set objWorkbook = objExcel.Workbooks.Open(csvfile)
Set objWorksheet = objWorkbook.Worksheets(1)

'Freeze Header row
With objExcel.ActiveWindow
    .SplitColumn = 0
    .SplitRow = 1
End With
objExcel.ActiveWindow.FreezePanes = True

'Add filters to heading row
objExcel.Rows(1).AutoFilter

'Autofit columns
Set objRange = objWorksheet.UsedRange
objRange.EntireColumn.Autofit()

'Set Alternate row color
'Change the colors by setting the background color index value
Dim i
For i = 1 To noRows + 1
   If i Mod 2 = 0 Then
      Set objRange = objWorksheet.Columns(1-noCols)
      objRange.Rows(i).Interior.ColorIndex = 0 ' background color white
      objRange.Rows(i).Borders.ColorIndex = 15 ' border color gray
   Else
      Set objRange = objWorksheet.Columns(1-noCols)
      objRange.Rows(i).Interior.ColorIndex = 20 ' background color light blue
      objRange.Rows(i).Borders.ColorIndex = 15 ' border color gray
   End If
   intNewRow = objWorksheet.Row + 1
Next

'set header row color and change font style to bold
Set objRange = ObjWorksheet.Columns(1-noCols)
objRange.Rows(1).Interior.ColorIndex = 11
objRange.Rows(1).Font.ColorIndex = 2
objRange.Rows(1).Font.Bold = True

'Save Worksheet as XLSX, 51 = Excel xlsx
objWorksheet.SaveAs xlsxfile, 51

'Release Lock on Spreadsheet
objExcel.Quit()
Set objRange = Nothing
Set objWorksheet = Nothing
Set objWorkbook = Nothing
Set ObjExcel = Nothing
```

Once pasted you can test the Script in the script editor by clicking the Run tool on the Script toolbar. Please note that to format the worksheet takes Excel around 35 seconds to finish, and the Transform Designer will not respond until the script completes.



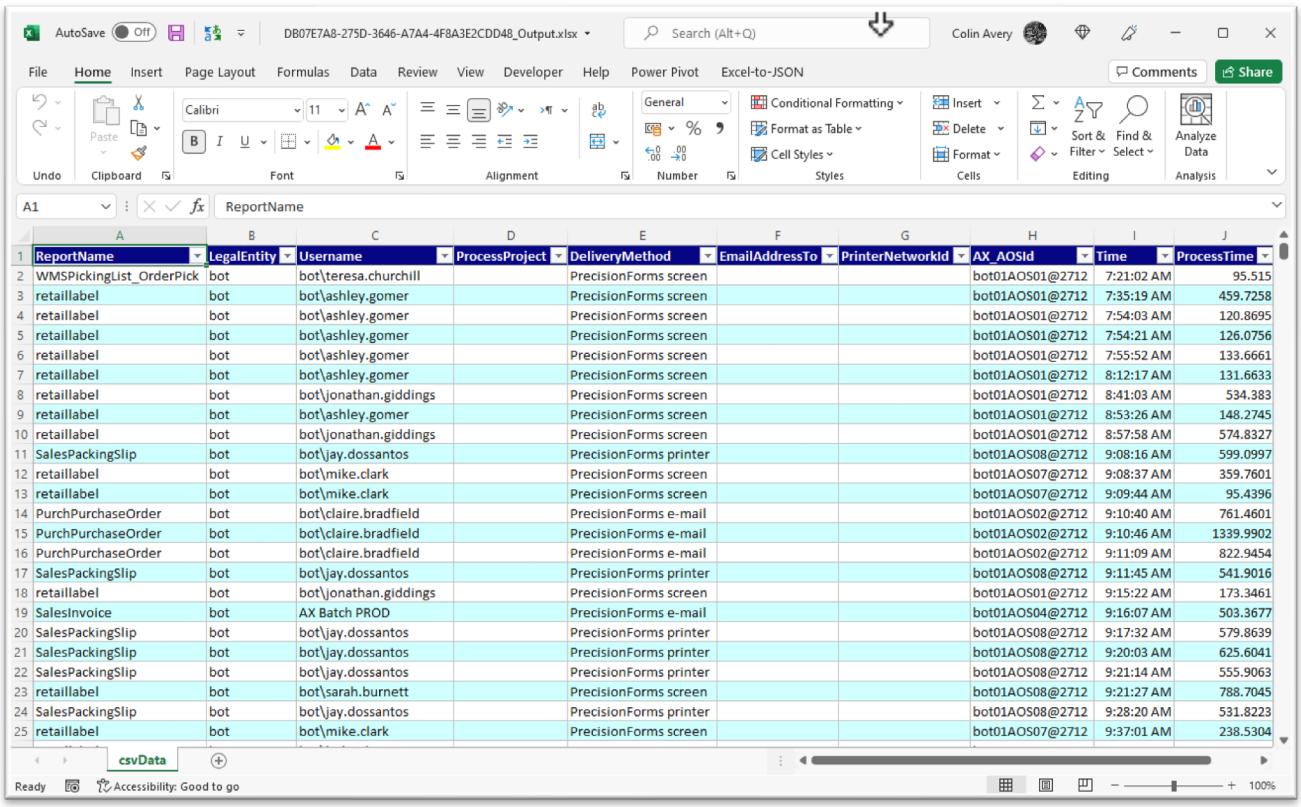
And if all went well then you should see a new Excel workbook in the working folder, with the naming convention UUID_Output.xlsx



Name	Date modified	Type	Size
DB07E7A8-275D-3646-A7A4-4F8A3E2CDD48_Output.xlsx	24/03/2022 11:25	Microsoft Excel W...	48 KB
csvData.csv	24/03/2022 10:14	Microsoft Excel C...	64 KB

Which if you open in Microsoft Excel has a single Worksheet labeled **csvData** and has the following formatting.

- Header row frozen
- Header row data filter
- Header row colored
- Autofit columns
- Alternate row color

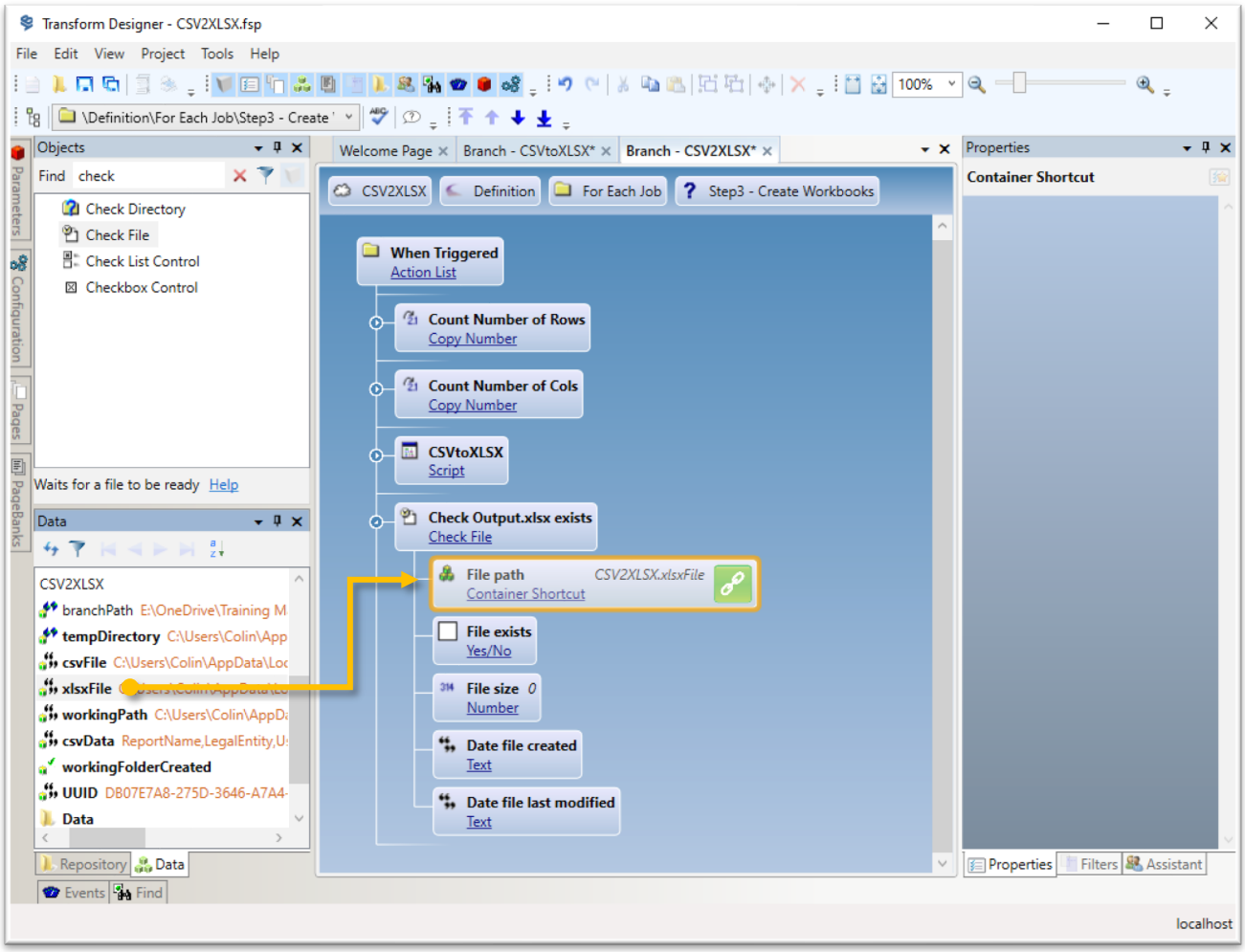


ReportName	LegalEntity	Username	ProcessProject	DeliveryMethod	EmailAddressTo	PrinterNetworkId	AX_AOSId	Time	ProcessTime
WMSPickingList_OrderPick	bot	bot\teresa.churchill		PrecisionForms screen			bot01AOS01@2712	7:21:02 AM	95.515
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	7:35:19 AM	459.7258
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	7:54:03 AM	120.8695
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	7:54:21 AM	126.0756
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	7:55:52 AM	133.6661
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	8:12:17 AM	131.6633
retaillabel	bot	bot\jonathan.giddings		PrecisionForms screen			bot01AOS01@2712	8:41:03 AM	534.383
retaillabel	bot	bot\ashley.gomer		PrecisionForms screen			bot01AOS01@2712	8:53:26 AM	148.2745
retaillabel	bot	bot\jonathan.giddings		PrecisionForms screen			bot01AOS01@2712	8:57:58 AM	574.8327
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:08:16 AM	599.0997
retaillabel	bot	bot\mike.clark		PrecisionForms screen			bot01AOS07@2712	9:08:37 AM	359.7601
retaillabel	bot	bot\mike.clark		PrecisionForms screen			bot01AOS07@2712	9:09:44 AM	95.4396
PurchPurchaseOrder	bot	bot\claire.bradfield		PrecisionForms e-mail			bot01AOS02@2712	9:10:40 AM	761.4601
PurchPurchaseOrder	bot	bot\claire.bradfield		PrecisionForms e-mail			bot01AOS02@2712	9:10:46 AM	1339.9902
PurchPurchaseOrder	bot	bot\claire.bradfield		PrecisionForms e-mail			bot01AOS02@2712	9:11:09 AM	822.9454
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:11:45 AM	541.9016
retaillabel	bot	bot\jonathan.giddings		PrecisionForms screen			bot01AOS01@2712	9:15:22 AM	173.3461
SalesInvoice	bot	AX Batch PROD		PrecisionForms e-mail			bot01AOS04@2712	9:16:07 AM	503.3677
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:17:32 AM	579.8639
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:20:03 AM	625.6041
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:21:14 AM	555.9063
retaillabel	bot	bot\sarah.burnett		PrecisionForms screen			bot01AOS08@2712	9:21:27 AM	788.7045
SalesPackingSlip	bot	bot\jay.dossantos		PrecisionForms printer			bot01AOS08@2712	9:28:20 AM	531.8223
retaillabel	bot	bot\mike.clark		PrecisionForms screen			bot01AOS07@2712	9:37:01 AM	238.5304

4.7.4 Create Pivot Table – Check File

Before we run the script to create the Pivot Table, we first need to check that the **UUID_Output.xlsx** file exists. This time we will use the Check File object instead of the Catch to demonstrate another technique. At the add point beneath the Script labeled **CSVtoXLSX** add the following.

Object	Label	Value
Check File	Output.xlsx exists	**File path** – Container shortcut mapped to **xlsxFile**

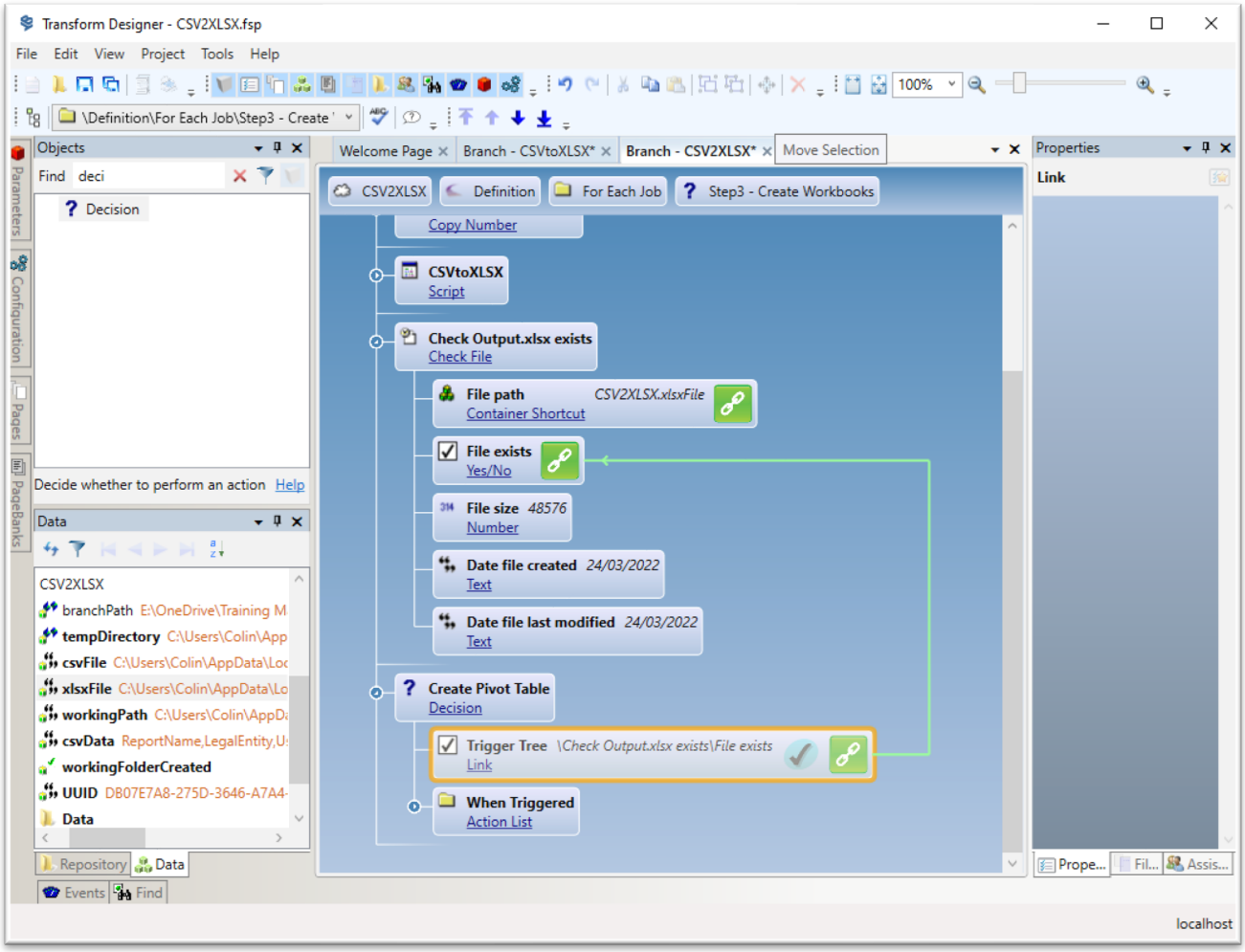


Right-click and Run the Check File object

4.7.5 Create Pivot Table – Decision

Under the Check File object add a Decision Object

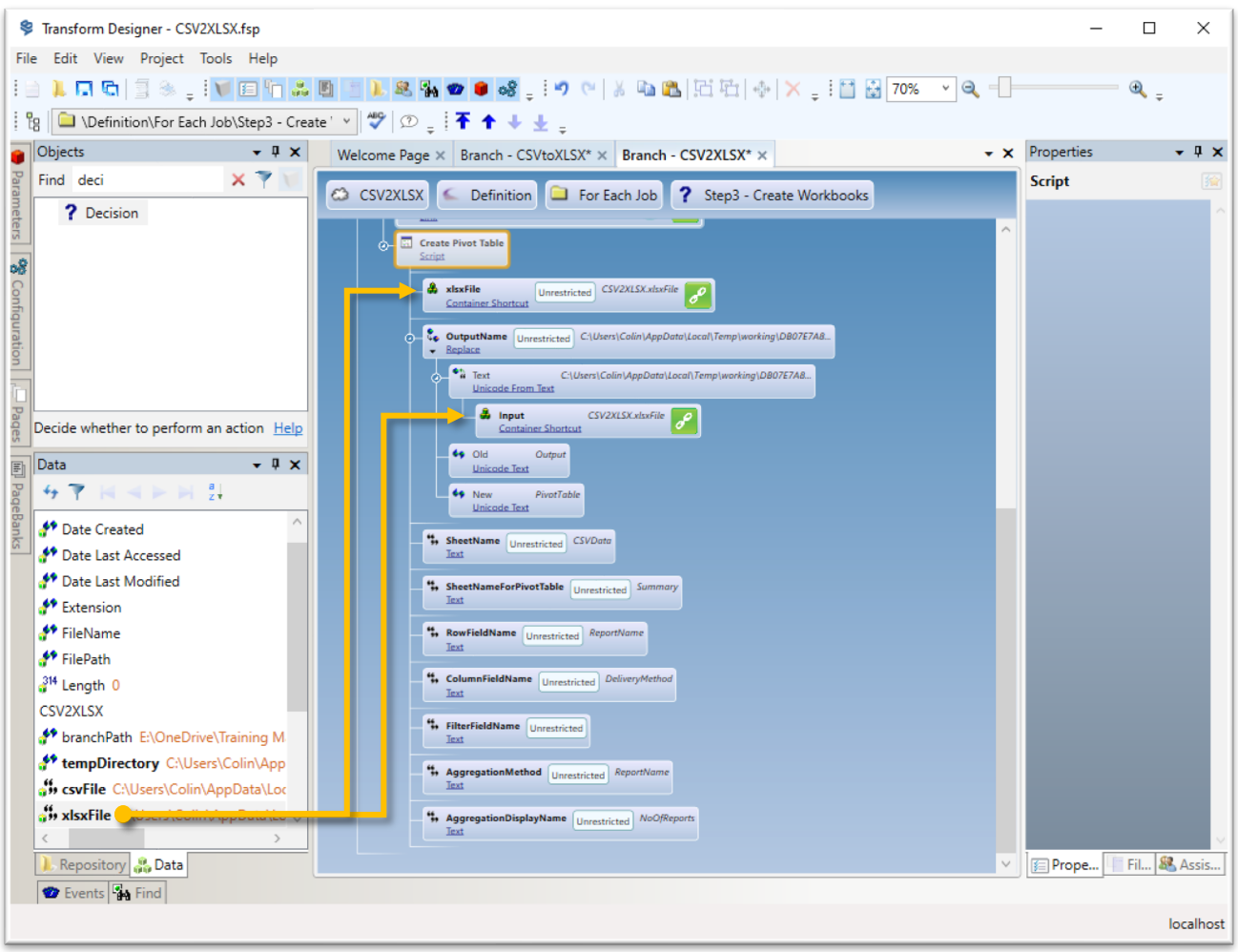
Object	Label	Value
Decision	Create Pivot Table	**Trigger Tree** – Link to Check File **exists** Yes/No object (right click and drag onto Trigger Tree and select Create Link)



4.7.6 Create Pivot Table – Script

Replace the **When Triggered** Action List with the following

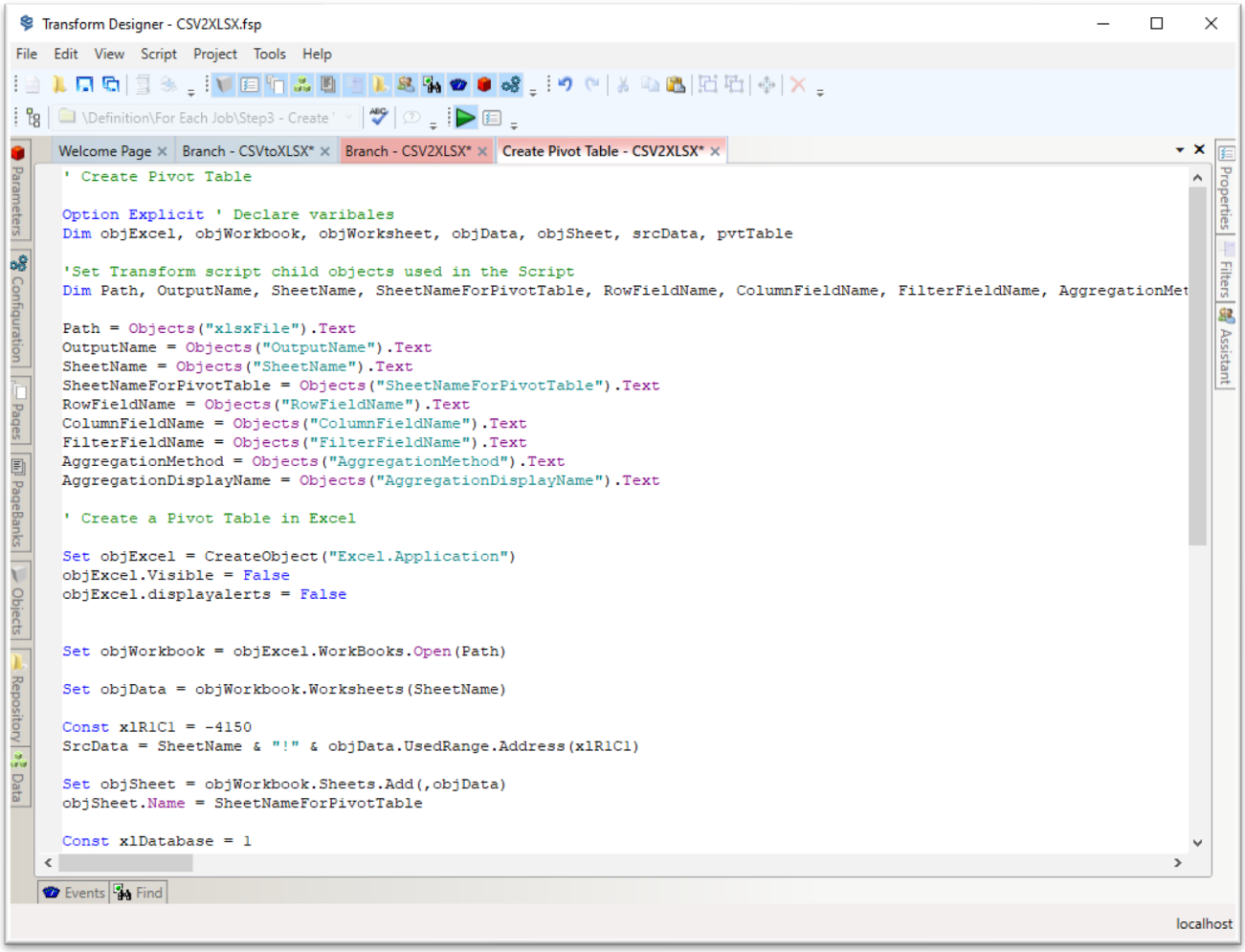
Object	Label	Value
Script	Create Pivot Table	Script Variables **xlsxFile** - Container shortcut – mapped to **xlsxFile** **OutputName** – Replace object Text Container shortcut mapped to **xlsxFile** **SheetName** – Text "csvData" **SheetNameForPivotTable** – Text "Summary" **RowFieldName** – Text "ReportName" **ColumnFieldName** – Text "DeliveryMethod" **FilterFieldName** – Text "" **AggregationMethod** – Text "ReportName" **AggregationDisplayName** – Text "NoOfReports"



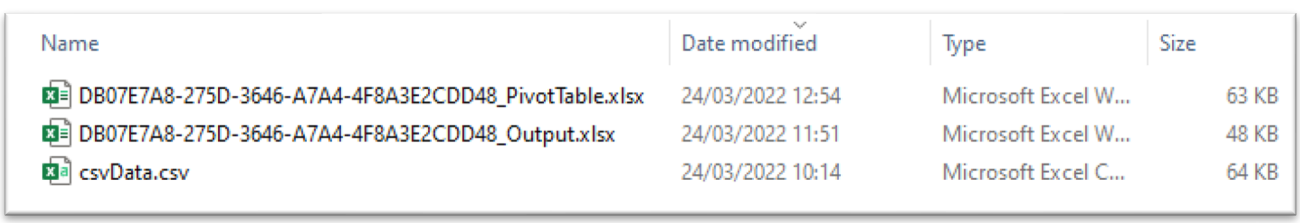
Open the Script editor, either click on the Script object icon or use the Advanced properties tool. Once open paste the VBScript below into the Script editor window.

```
' Create Pivot Table
Option Explicit ' Declare varibales
Dim objExcel, objWorkbook, objWorksheet, objData, objSheet, srcData, pvtTable

'Set Transform script child objects used in the Script
Dim Path, OutputName, SheetName, SheetNameForPivotTable, RowFieldName, ColumnFieldName, FilterFieldName, AggregationMethod,
AggregationDisplayName

Path = Objects("xlsxFile").Text
OutputName = Objects("OutputName").Text
SheetName = Objects("SheetName").Text
SheetNameForPivotTable = Objects("SheetNameForPivotTable").Text
RowFieldName = Objects("RowFieldName").Text
ColumnFieldName = Objects("ColumnFieldName").Text
FilterFieldName = Objects("FilterFieldName").Text
AggregationMethod = Objects("AggregationMethod").Text
AggregationDisplayName = Objects("AggregationDisplayName").Text

' Create a Pivot Table in Excel
Set objExcel = CreateObject("Excel.Application")
objExcel.Visible = False
objExcel.displayalerts = False

Set objWorkbook = objExcel.WorkBooks.Open(Path)
Set objData = objWorkbook.Worksheets(SheetName)

Const xlR1C1 = -4150
SrcData = SheetName & "!" & objData.UsedRange.Address(xlR1C1)

Set objSheet = objWorkbook.Sheets.Add(,objData)
objSheet.Name = SheetNameForPivotTable

Const xlDatabase = 1
Set pvtTable = objWorkbook.PivotCaches.Create(xlDatabase,SrcData).CreatePivotTable(SheetNameForPivotTable & "!R1C1","PivotTable1")

Const xlColumnField = 2
pvtTable.pivotFields(ColumnFieldName).orientation = xlColumnField

'Const xlFilterField = 3
'pvtTable.pivotFields("").orientation =  xlFilterField

'https://docs.microsoft.com/en-us/office/vba/api/excel.xlconsolidationfunction

Const xlSum = -4112
pvtTable.AddDataField pvtTable.PivotFields(RowFieldName), AggregationDisplayName, xlSum

Const xlRowField = 1
pvtTable.pivotFields(RowFieldName).orientation = xlRowField

'Save Workbook
objWorkbook.SaveAs OutputName, 51

'Release Lock on Spreadsheet
objExcel.Quit()
Set objData = Nothing
Set srcData = Nothing
Set objSheet = Nothing
Set pvtTable = Nothing
Set objWorksheet = Nothing
Set objWorkbook = Nothing
Set ObjExcel = Nothing
```

Once pasted you can test the Script in the script editor by clicking the Run tool on the Script toolbar. Please note whilst the script is running the Transform Designer will not respond.

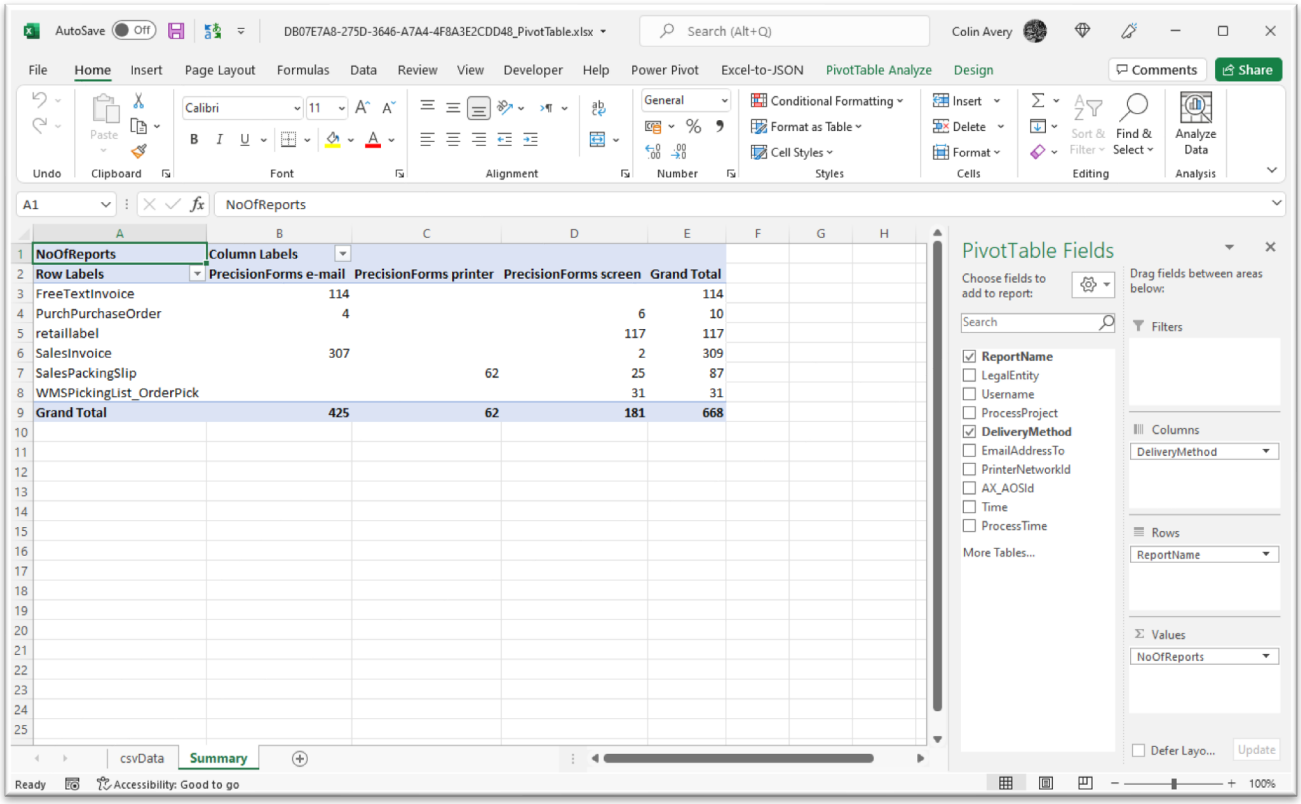


If all went well then you should see another Workbook has been created in your working folder named **UUID_PivotTable.xlsx**.



Name	Date modified	Type	Size
DB07E7A8-275D-3646-A7A4-4F8A3E2CDD48_PivotTable.xlsx	24/03/2022 12:54	Microsoft Excel W...	63 KB
DB07E7A8-275D-3646-A7A4-4F8A3E2CDD48_Output.xlsx	24/03/2022 11:51	Microsoft Excel W...	48 KB
csvData.csv	24/03/2022 10:14	Microsoft Excel C...	64 KB

If you open the Workbook in Microsoft Excel, you will see that it contains two Worksheets the original csvData sheet created in the first Script, and a new Worksheet called Summary that contains the Pivot Table.



NoOfReports	Column Labels			
Row Labels	PrecisionForms e-mail	PrecisionForms printer	PrecisionForms screen	Grand Total
FreeTextInvoice	114			114
PurchPurchaseOrder	4		6	10
retaillabel			117	117
SalesInvoice	307		2	309
SalesPackingSlip		62	25	87
WMSPickingList_OrderPick			31	31
Grand Total	425	62	181	668

5 Objects used in Branch

The following objects are you used in the branch file.

- Action List
- Bag
- Catch
- Check Directory
- Check File
- Container Shortcut
- Copy Number
- Copy Text
- Copy Yes/No
- Count
- Create Directory
- Decision
- Delimited Data Parser
- File Queue
- Folder
- Link
- Log Event
- Memory
- Number
- Queue Process
- Replace
- Script
- Sentence
- System Information
- Text
- Unicode From Text
- Unicode Text
- Unique Identifier
- Write File
- Yes/No

6 Feedback

Thank you for taking the time to complete this exercise, we welcome your feedback and it helps us improve our training materials, please forward any comments or enhancements to this exercise to cavery@ctaconsultancy.com

Every day is a learning day!