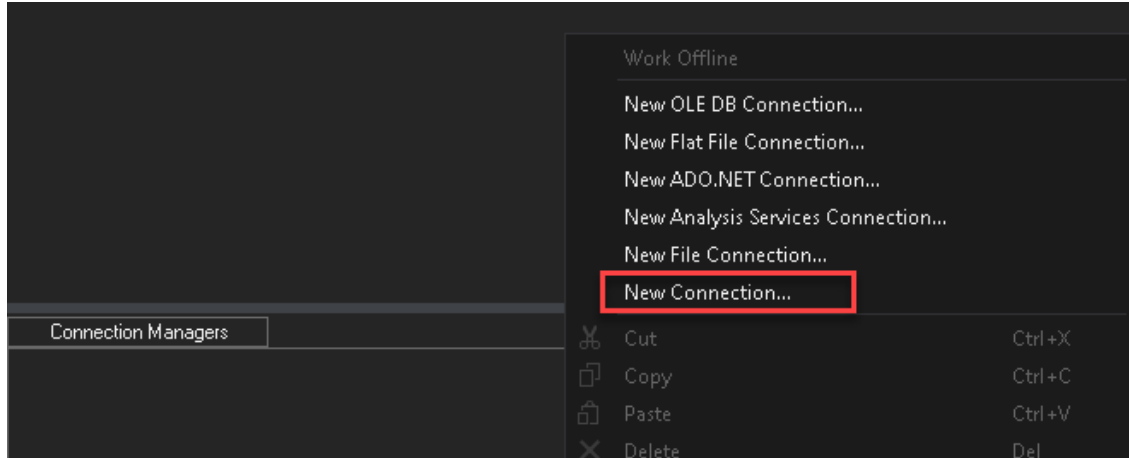


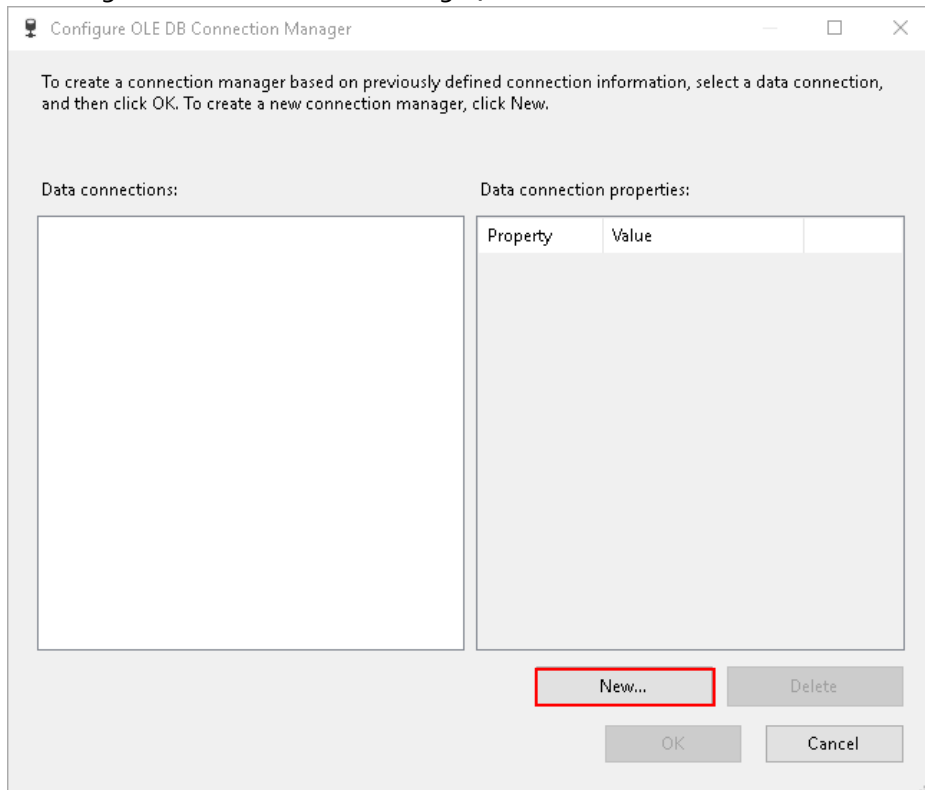
SQL SERVER INTEGRATION SERVICES

MODULE 03 – LAB 06: CONTROL FLOW: BRING IT TOGETHER

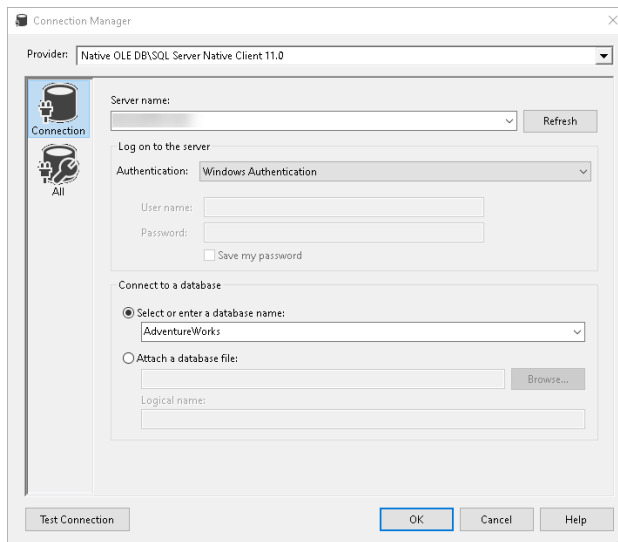
1. Launch Visual Studio.
2. Create new project.
Set up a connection manager to our database. In the bottom center pain under Connection Manger, right-click select New Ole-DB Connection.



3. In Configure OLE DB Connect Manager, click New.



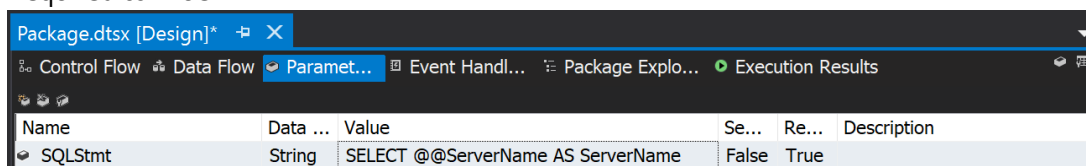
4. In connection manager, type the server's name, select the database "AdventureWorks" and click OK.



5. Click OK in Configure OLE DB Connection Manager. The new connection should show up in the list.

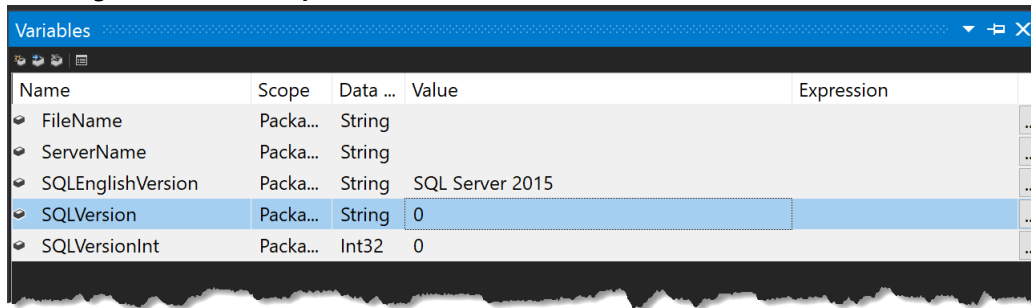
Hint: Rename the connection manager to OLEDB.ServerName.DatabaseName. This will make it easier to identify which driver is being used for the driver.

6. Setup a Package Parameter assign it value "SELECT @@ServerName AS ServerName". Set Required to True.



Name	Data ...	Value	Se...	Re...	Description
SQLStmt	String	SELECT @@ServerName AS ServerName	False	True	

- Setup variables for the project. Go to SSIS Menu > Variables. Then create the following variables in Package Scope and their respective types (Note: The is default value for SQLEnglishVersion is **required**).

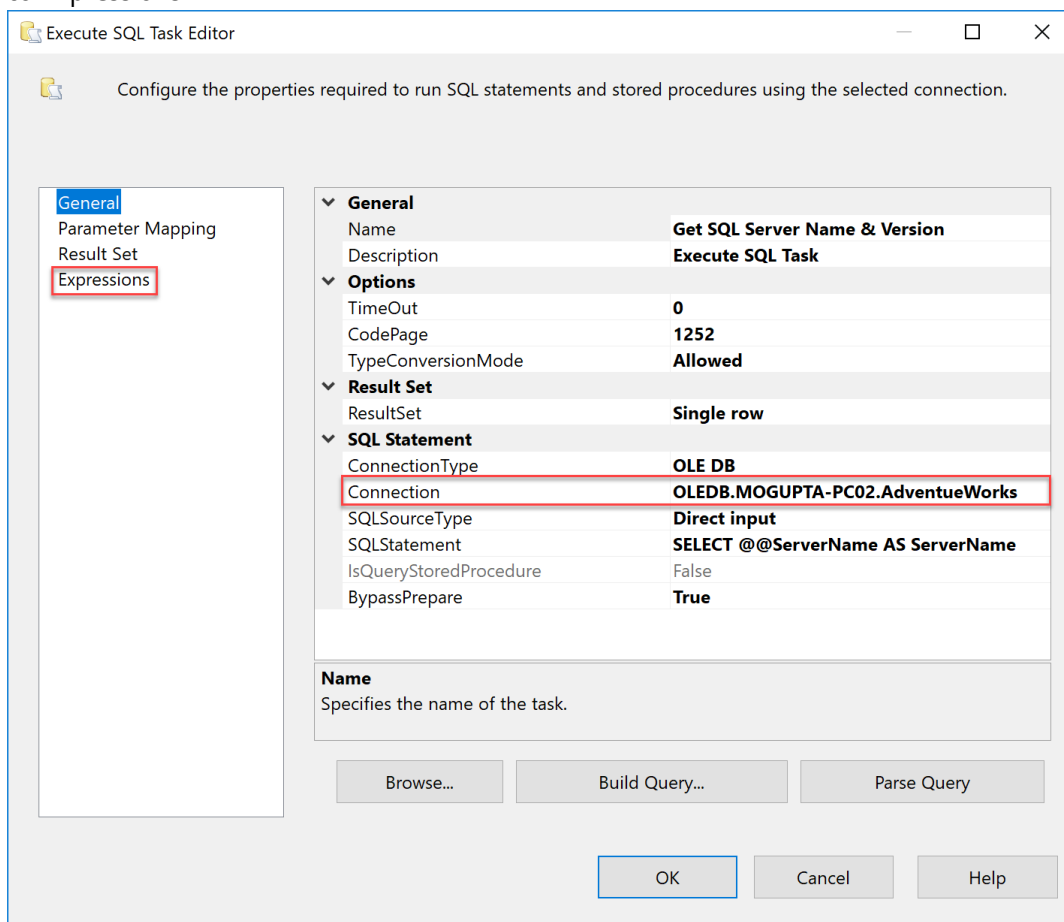


Name	Scope	Data ...	Value	Expression
FileName	Packa...	String		
ServerName	Packa...	String		
SQLEnglishVersion	Packa...	String	SQL Server 2015	
SQLVersion	Packa...	String	0	
SQLVersionInt	Packa...	Int32	0	

- Create an Execute SQL Task. Rename it to "Get SQL Server Name & Version".

Hint: It is possible to rename a task by double-clicking and updating the Name attribute. Or update the name by selecting the task and pressing F2.

- Connect the "Get SQL Server Name & Version" to OLEDB connection created in step #5 and go to Expressions.



Execute SQL Task Editor

Configure the properties required to run SQL statements and stored procedures using the selected connection.

General

Name: **Get SQL Server Name & Version**

Description: **Execute SQL Task**

Options

TimeOut: **0**

CodePage: **1252**

TypeConversionMode: **Allowed**

Result Set

ResultSet: **Single row**

SQL Statement

ConnectionType: **OLE DB**

Connection: **OLEDB.MOGUPTA-PC02.AdventureWorks**

SQLSourceType: **Direct input**

SQLStatement: **SELECT @@ServerName AS ServerName**

IsQueryStoredProcedure: **False**

BypassPrepare: **True**

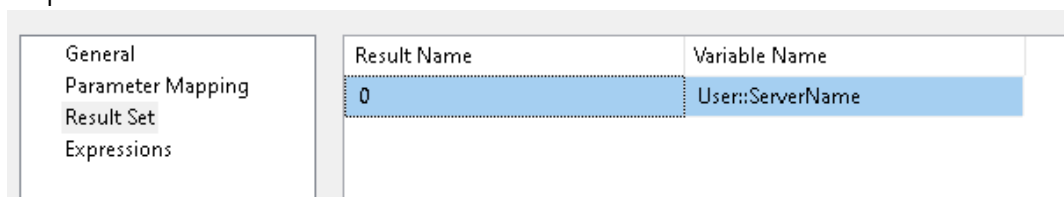
Name

Specifies the name of the task.

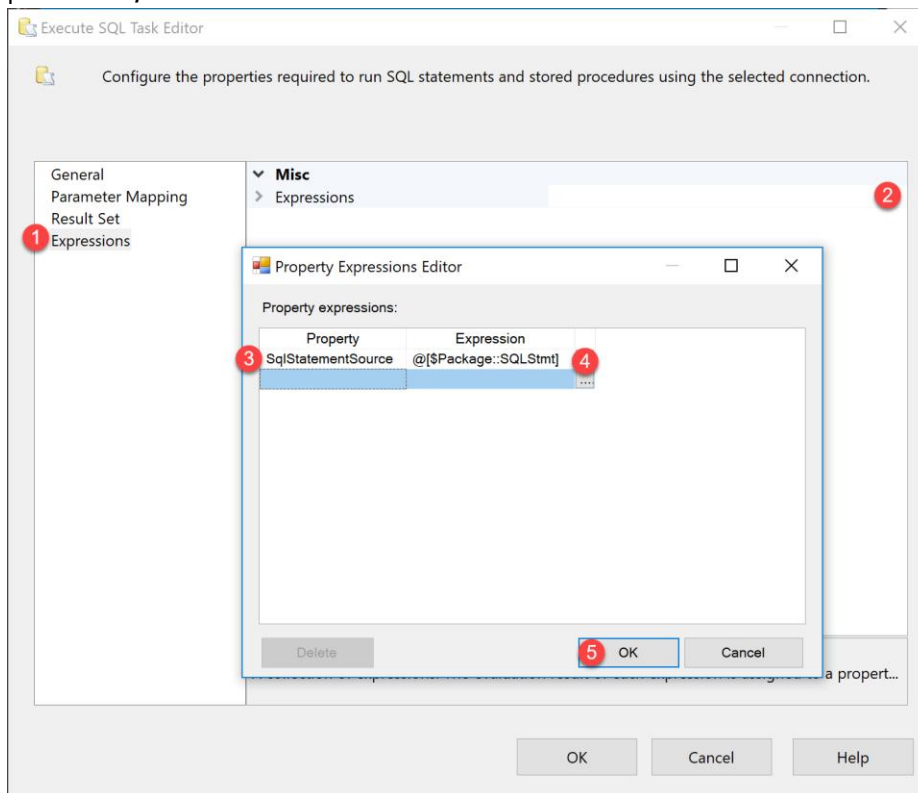
Browse... Build Query... Parse Query

OK Cancel Help

10. Map the result set to variable to ServerName.

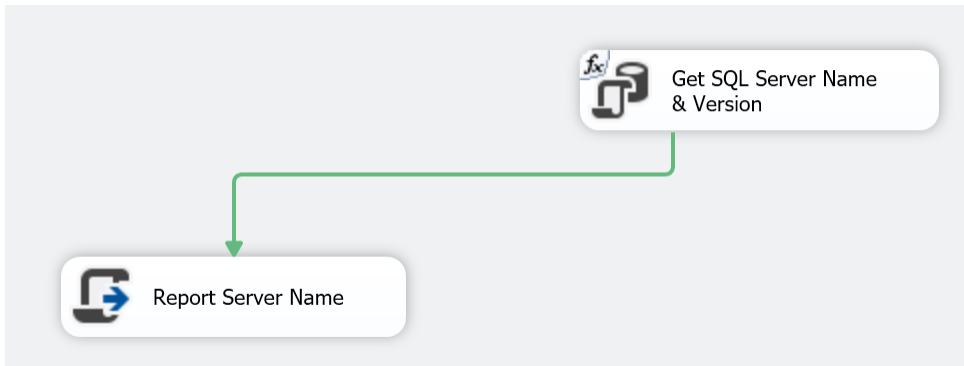


11. Under expressions, build an expression and attach the SqlStatementSource to package parameter, **SQLStmt**.

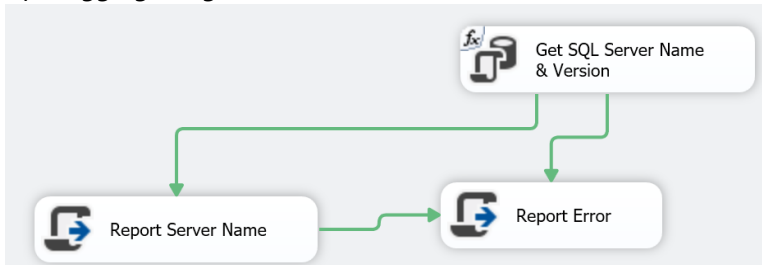


12. Create a Script Task. Rename it to "Report Server Name".

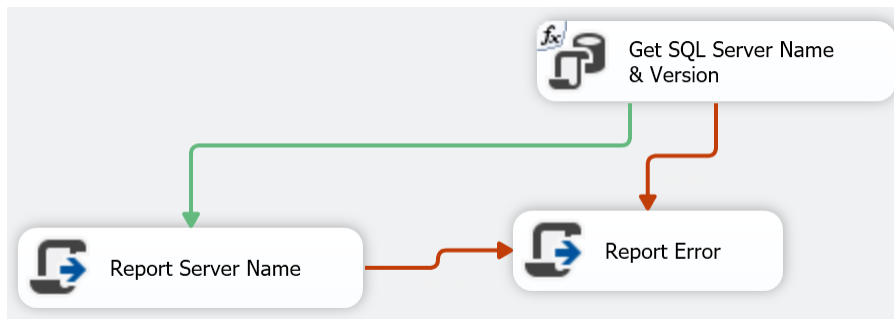
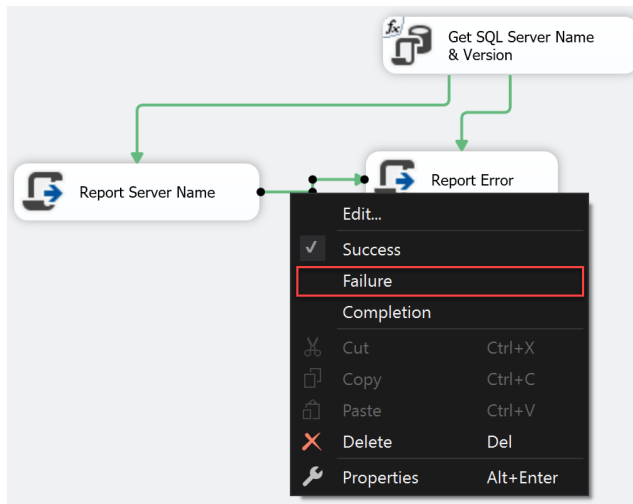
13. Connect the "Get SQL Server Name & Version" to "Report Server Name" by dragging the green arrow to task.



14. Create another Script Task. Rename it to "Report Error".
15. Connect both "Report Server Name" and "Get SQL Server Name & Version" to "Report Error" by dragging the green arrows.



16. However, since we want to report the error on a failure, we will change connections to on failure. Right click on the green line connecting each of the tasks to Report Error and Select Failure. After both lines are updated, it should be like below.



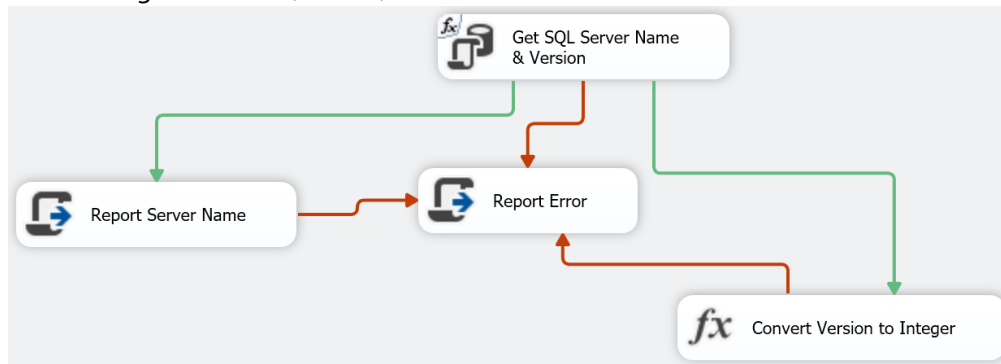
17. Update each Script task with their respect C# Code. Open the script task, assign variable, and type the following script in the Main() function.

Script Task Name	ReadOnlyVariables	Script
Report Server Name	User::ServerName	<code>MessageBox.Show("SQL Server Name: " + Dts.Variables["User::ServerName"].Value.ToString(), "Server Name");</code>
Report Error	<i>Blank-No-Assignment</i>	<code>MessageBox.Show("Uhhh! I think something went wrong!", "Help!");</code>

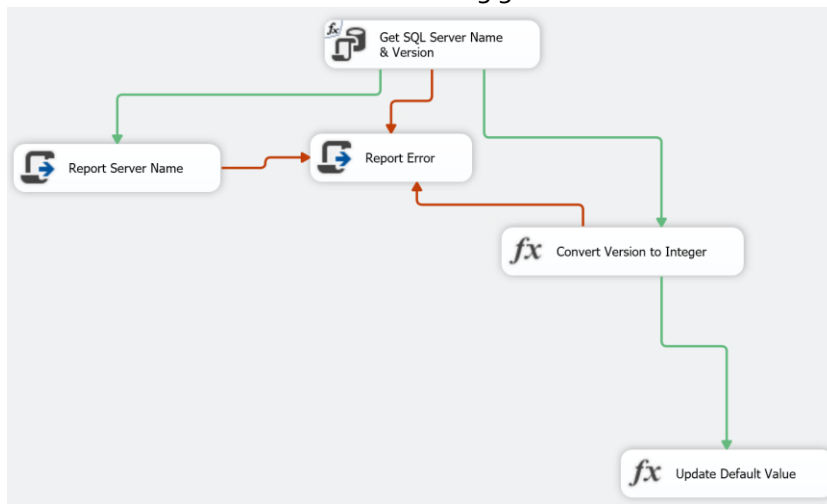
18. Next create an Expression Task, rename it to "Convert Version to Integer" and set the expression to "`@[User::SQLVersionInt] = (DT_I4) @[User::SQLVersion]`".



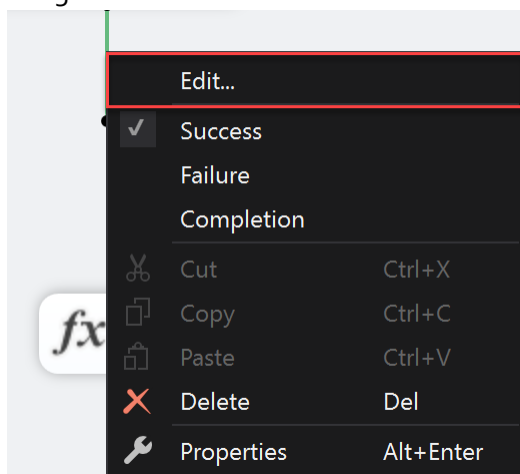
19. After building the express task, connect, "Get SQL Server Name & Version" to "Convert Version to Integer" with Green arrow (Success) and connect "Convert Version to Integer" to "Report Error" using Red arrow (Failure).



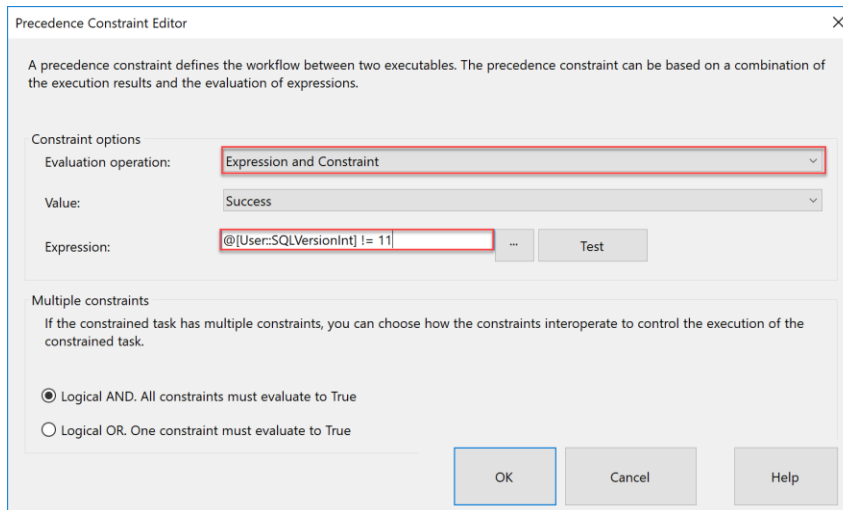
20. Next build another Expression Task, rename it to "Update Default Value". Set the expression to "@[User::SQLEnglishVersion] = "SQL Server 2017"". Connect the "Update Default Value" to "Get SQL Server Name & Version" using green arrow.



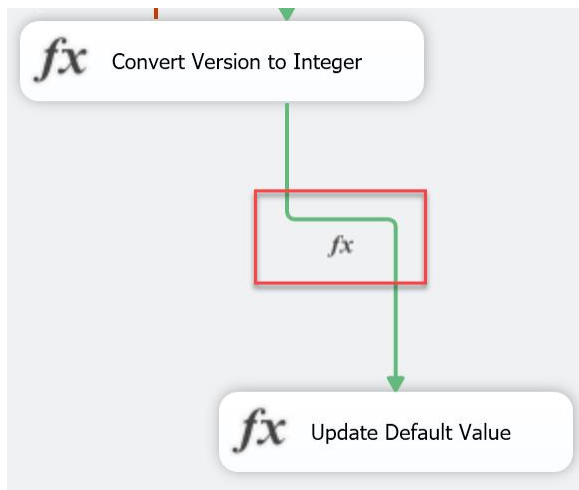
21. Update the expression, for the Update Default Value to make it a bit restrictive. Right click on the green line and select Edit.



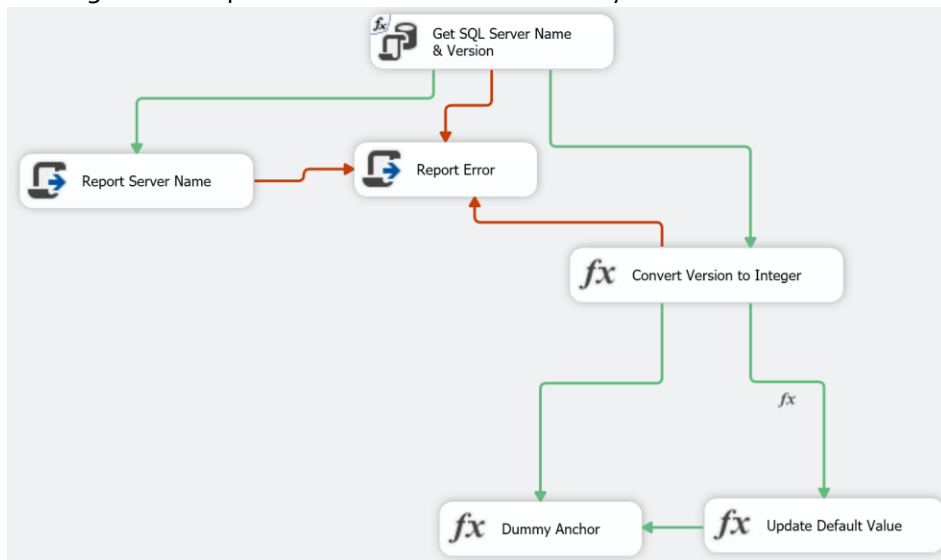
22. Change the Evaluation operation to Expression and Constraint and change the Expression to “@[User::SQLVersionInt] != 15”. Note after the precedence constraint is built an “fx” added to the constraint line.



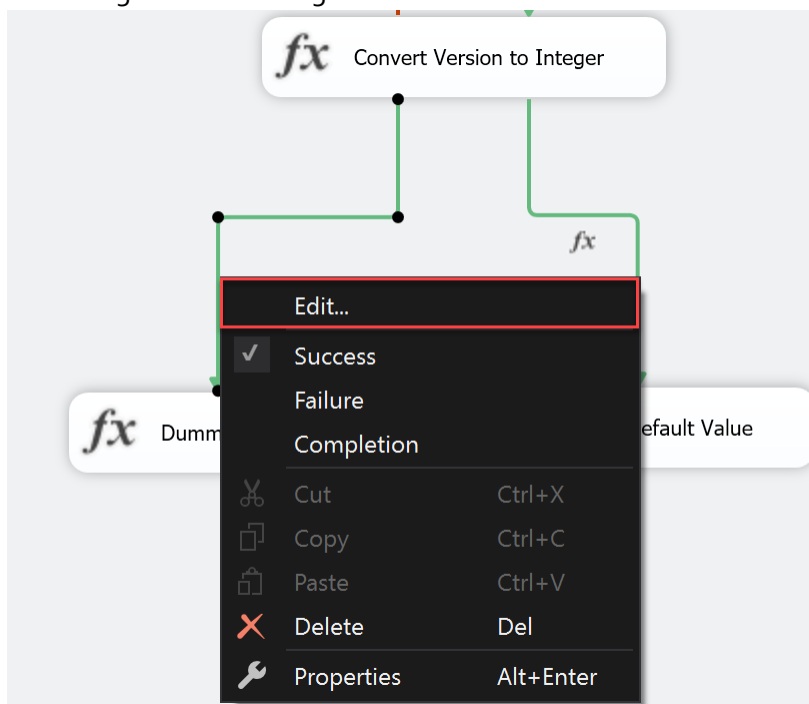
The image shows the 'Precedence Constraint Editor' dialog box. It has a title bar with a close button (X). The main text says: 'A precedence constraint defines the workflow between two executables. The precedence constraint can be based on a combination of the execution results and the evaluation of expressions.' Below this is the 'Constraint options' section. It contains three fields: 'Evaluation operation:' with a dropdown menu set to 'Expression and Constraint', 'Value:' with a dropdown menu set to 'Success', and 'Expression:' with a text box containing '@[User::SQLVersionInt] != 11'. To the right of the text box are two buttons: '...' and 'Test'. Below the 'Constraint options' section is the 'Multiple constraints' section. It contains the text: 'If the constrained task has multiple constraints, you can choose how the constraints interoperate to control the execution of the constrained task.' Below this text are two radio buttons: 'Logical AND. All constraints must evaluate to True' (which is selected) and 'Logical OR. One constraint must evaluate to True'. At the bottom right are three buttons: 'OK', 'Cancel', and 'Help'.



23. Create another Expression Task, rename it to “Dummy Anchor”. Set the expression to “@[User::SQLEnglishVersion] = @[User::SQLEnglishVersion]”. Connect the “Convert Version to Integer” and “Update Default Value” to “Dummy Anchor” task.



24. It is not possible to have both tasks from “Convert” and “Update” going to Dummy Anchor because of function constraints. Therefore, change the Multiple constraints logical operator from ADD Condition to OR condition. Right click on either of the green-lines connecting to the Dummy Anchor and select edit. In precedence constraint editor, select “Logical OR”. Notice the lines green line changed from solid to dotted.



Precedence Constraint Editor

A precedence constraint defines the workflow between two executables. The precedence constraint can be based on a combination of the execution results and the evaluation of expressions.

Constraint options

Evaluation operation: Constraint

Value: Success

Expression: ... Test

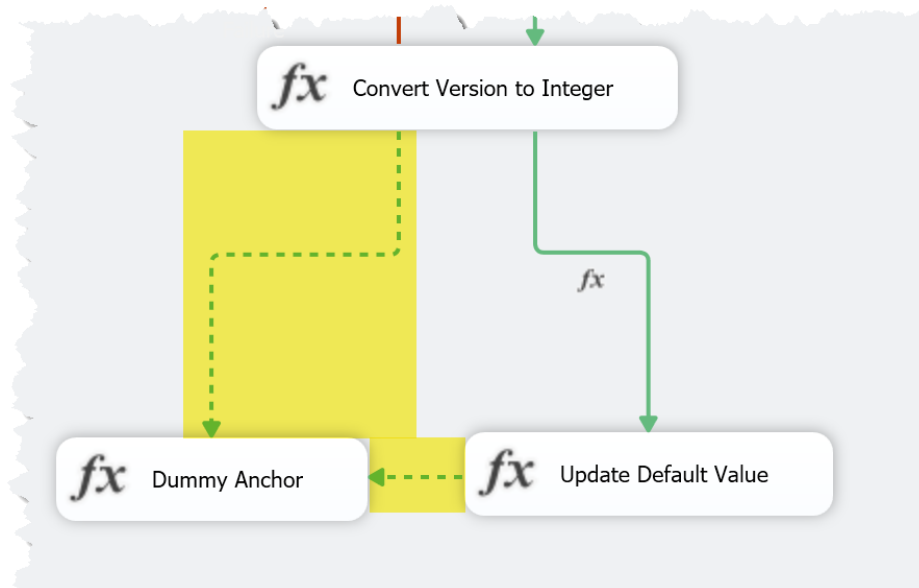
Multiple constraints

If the constrained task has multiple constraints, you can choose how the constraints interoperate to control the execution of the constrained task.

☐ Logical AND. All constraints must evaluate to True

☒ Logical OR. One constraint must evaluate to True

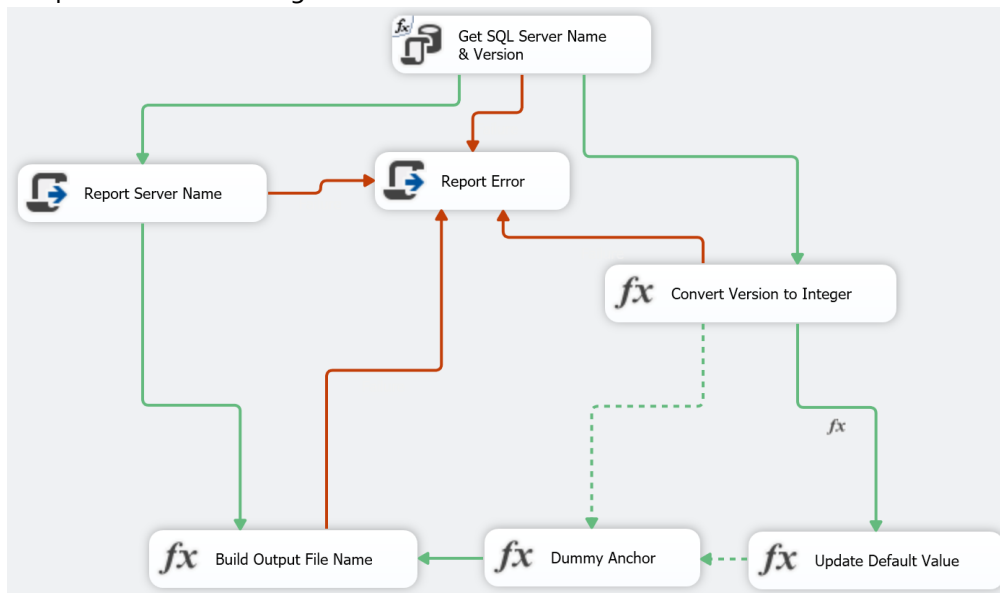
OK Cancel Help



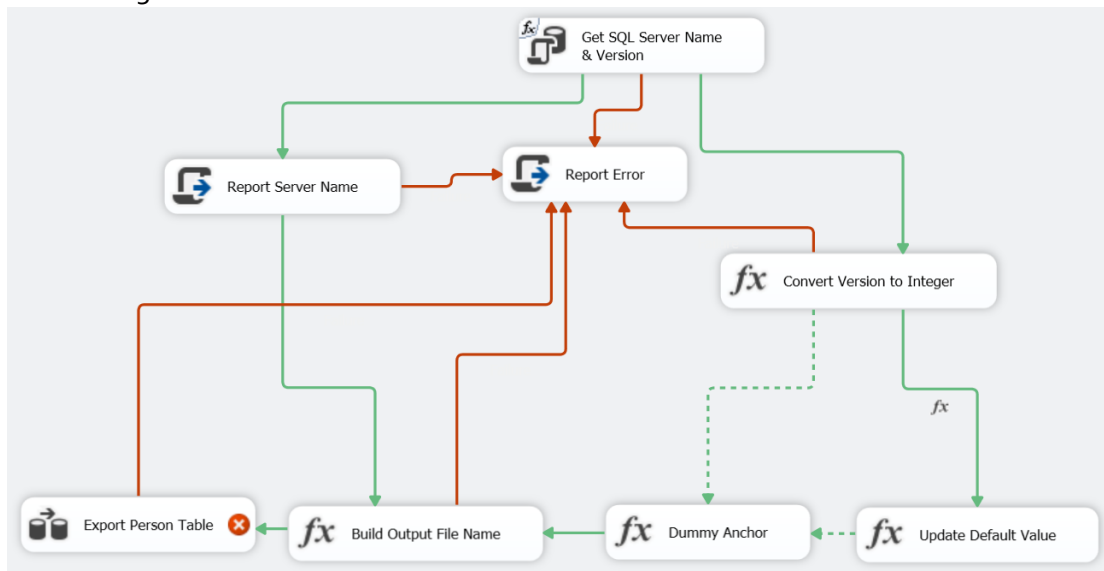
25. Create another Expression Task. Rename it "Build Output File Name". Set the expression to `"@[User::FileName] = "C:\\Temp\\" + @[User::ServerName] + "_" + @[User::SQLEnglishVersion] + "_Persons.txt"` (Please update the starting path C:\\Temp\\ to your respective directory. Also note to escape each \\ with \\).

```
@[User::FileName] = "C:\\Temp\\" + @[User::ServerName] + "_" + @[User::SQLEnglishVersion] + "_Persons.txt"
```

26. Connect the “Build Output File Name” task to “Report Error” with Red line. Connect “Dummy Anchor” to “Build Output File Name” with Green line. Connect “Report Server Name” to “Build Output File Name” using Green line.

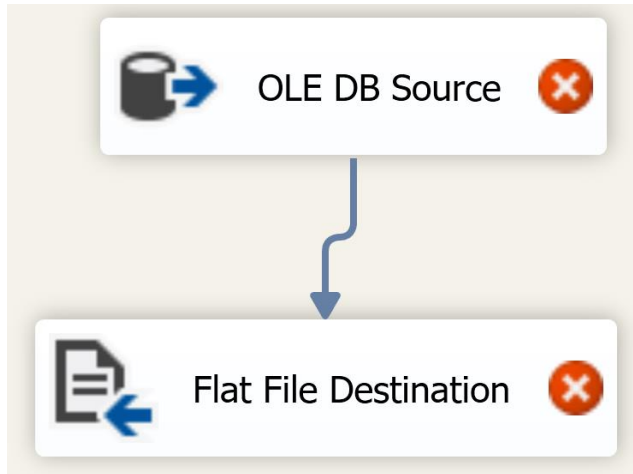


27. Next add a data flow task and rename it to “Export Person Table”. Connect “Build Output File Name” to “Export Person Table” using Green Line. Connect “Export Person Table” to “Report Error” using Red Line.

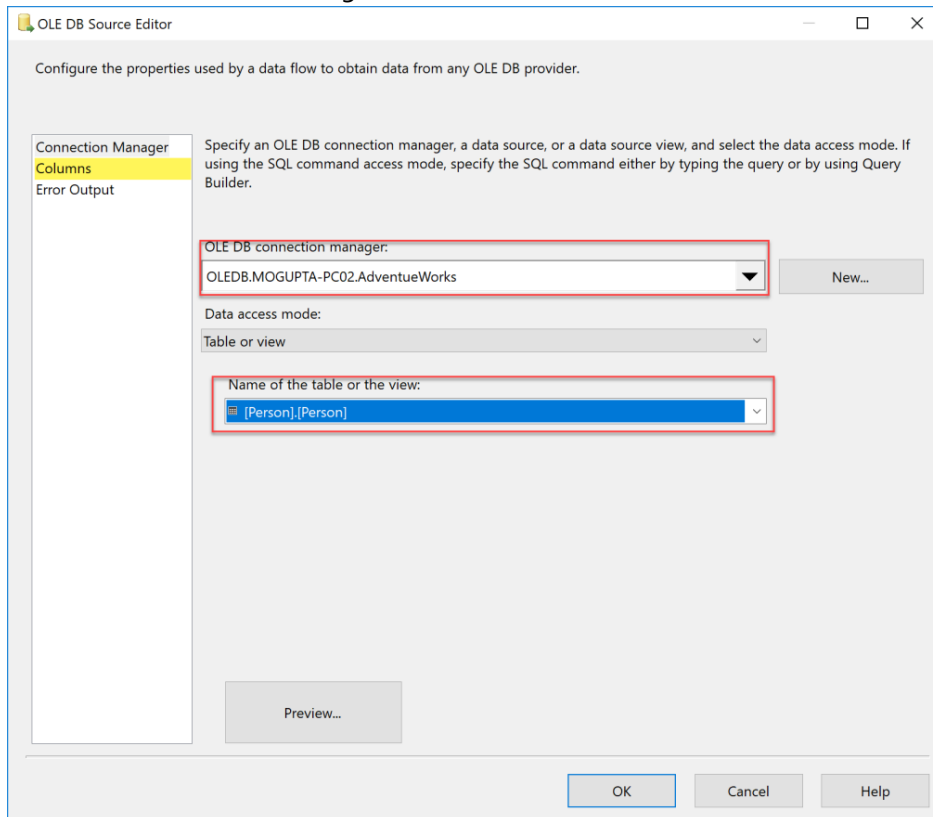


28. Double-click on the “Export Person Table” task to configure the data flow.

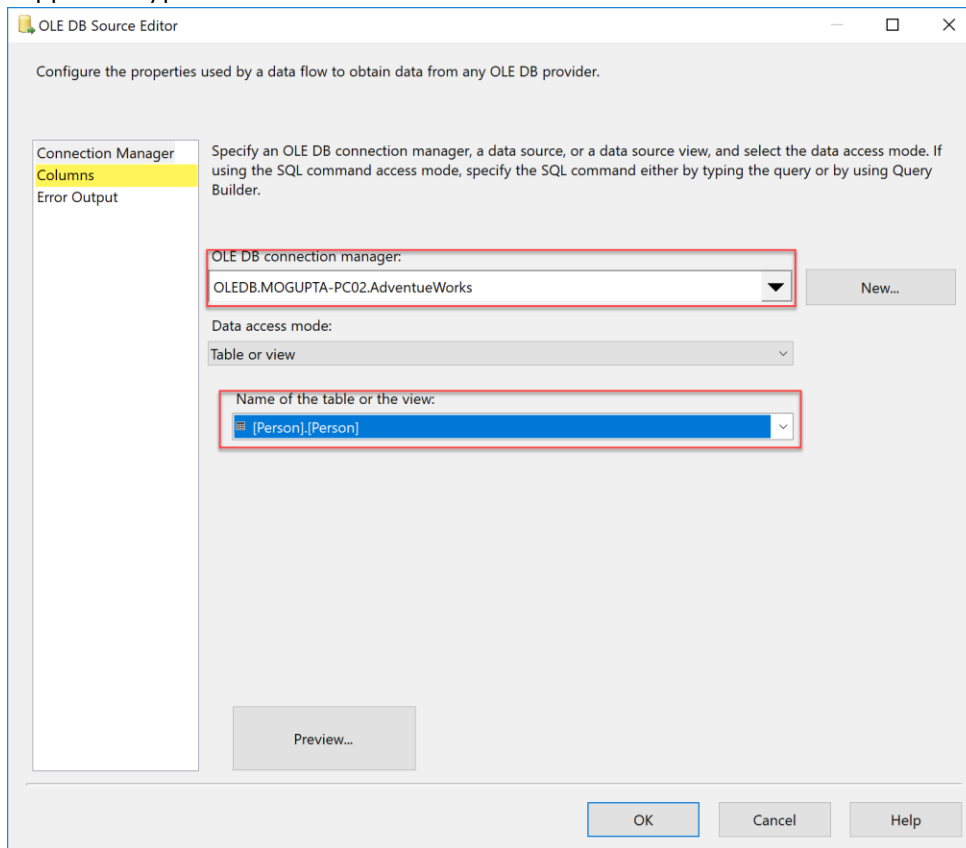
29. Create two tasks “OLE DB Source” and “Flat File Destination” and connect them.



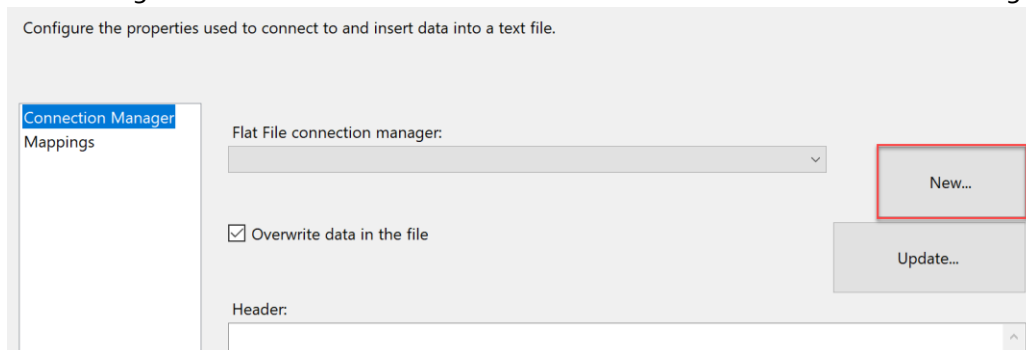
30. Configure the “OLE DB Source” to connect to our connection string and pull data from Person.Person. After setting those select Columns.



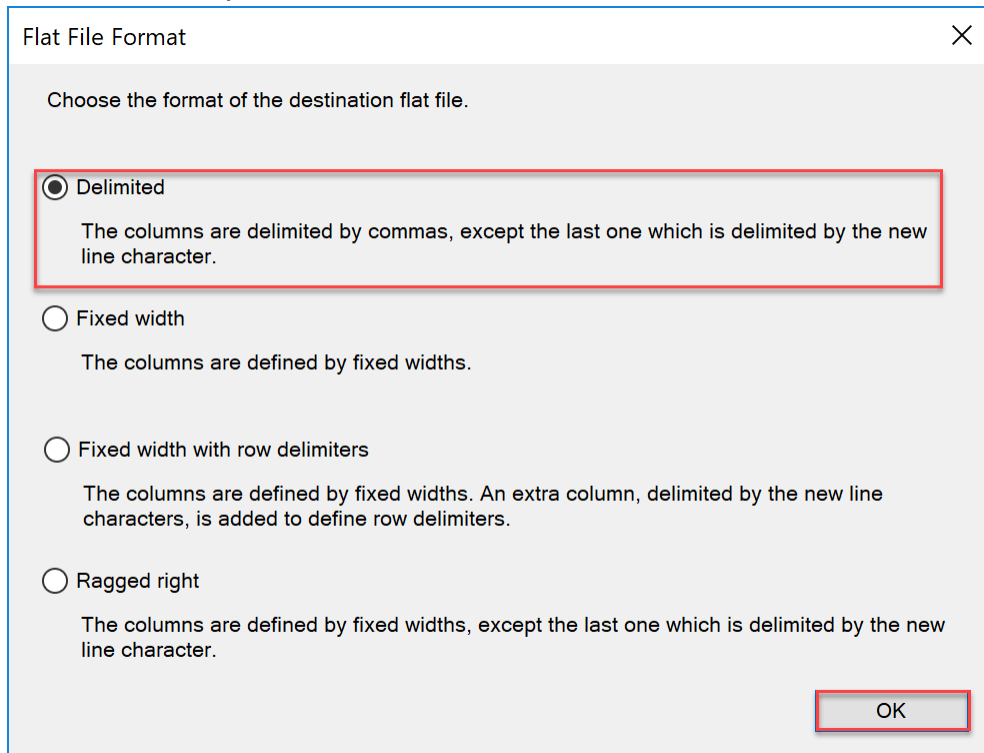
31. Only select columns shown below, we cannot export all columns to CSV because they are not supported types.



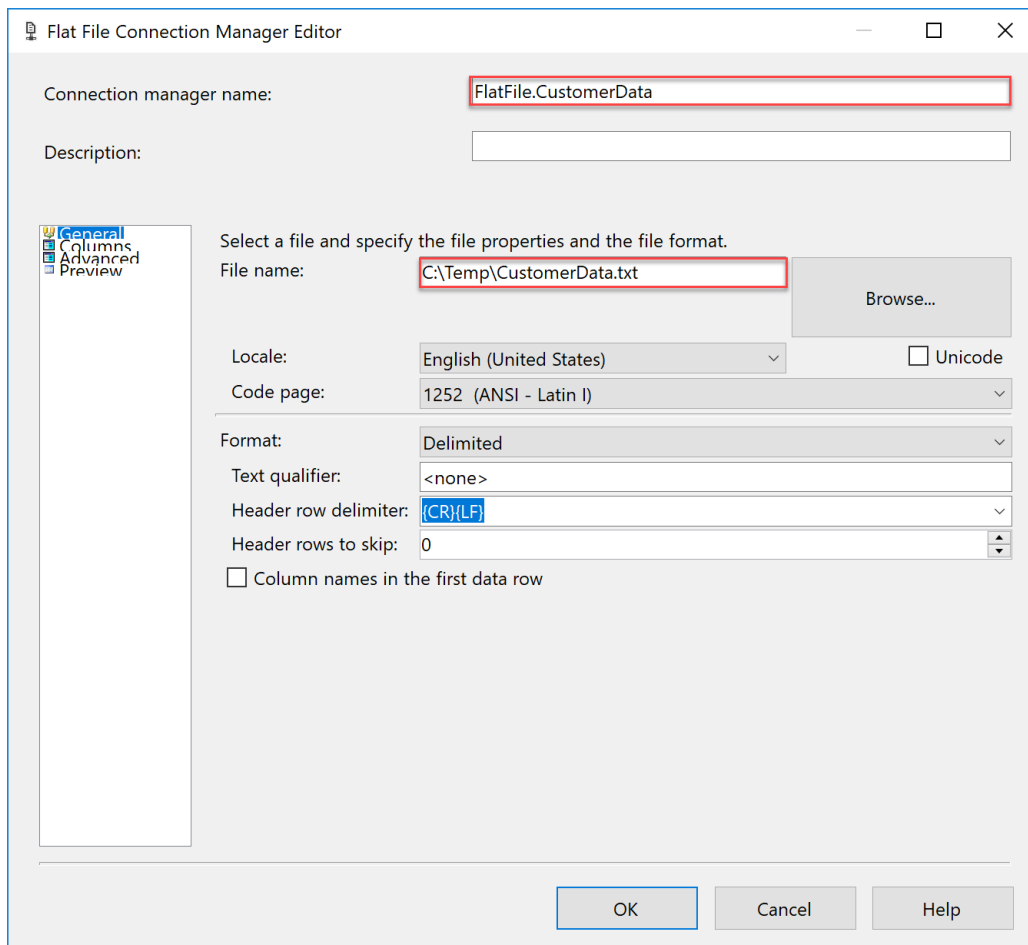
32. Next configure the "Flat File Destination". Create a New Flat File Connection Manager.



33. In Flat File Format, Select Delimited and click OK.



34. Rename the connection manager and set file name, doesn't have to be valid as we will be changing it via expressions.

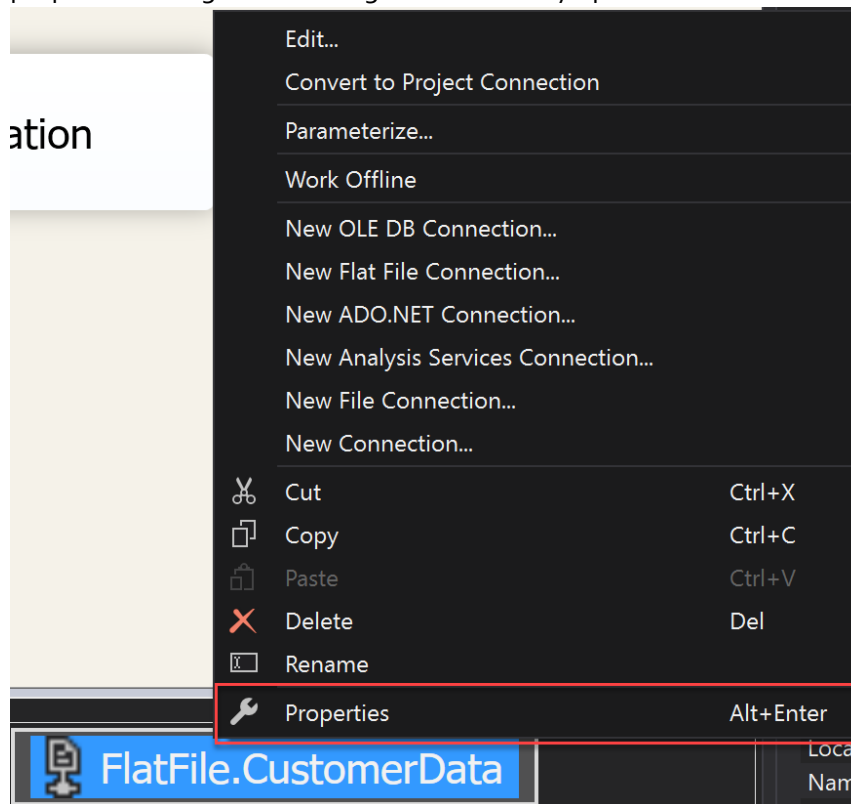


The image shows the 'Flat File Connection Manager Editor' dialog box. The 'Connection manager name' is 'FlatFile.CustomerData'. The 'Description' field is empty. On the left, there is a tree view with 'General' selected. The main area contains fields for file properties: 'File name' is 'C:\Temp\CustomerData.txt', 'Locale' is 'English (United States)', 'Code page' is '1252 (ANSI - Latin I)', 'Format' is 'Delimited', 'Text qualifier' is '<none>', 'Header row delimiter' is '(CR)(LF)', and 'Header rows to skip' is '0'. There is an unchecked checkbox for 'Column names in the first data row'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

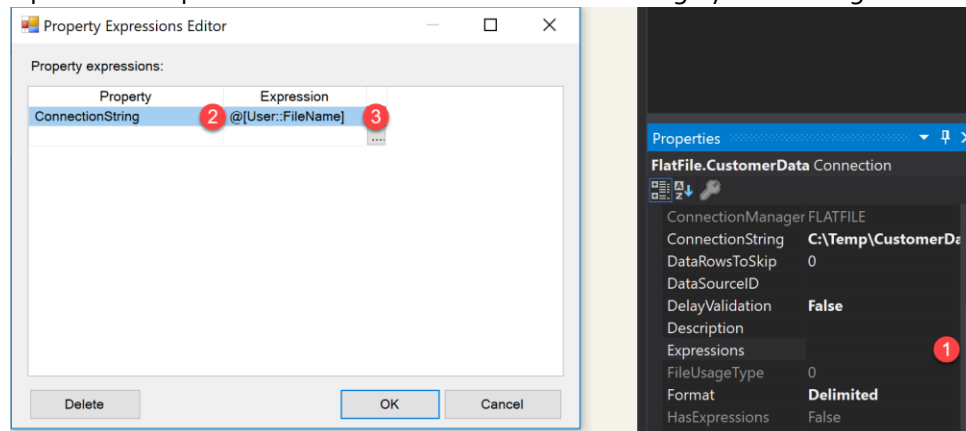
35. Before closing the Flat File Destination Editor, click on Mappings to confirm column mapping and remove the following columns by setting them ignore as in the image below.

Input Column	Destination Column
MiddleName	MiddleName
LastName	LastName
Suffix	Suffix
<ignore>	EmailPromotion
<ignore>	AdditionalContactInfo
<ignore>	Demographics
<ignore>	rowguid
<ignore>	ModifiedDate

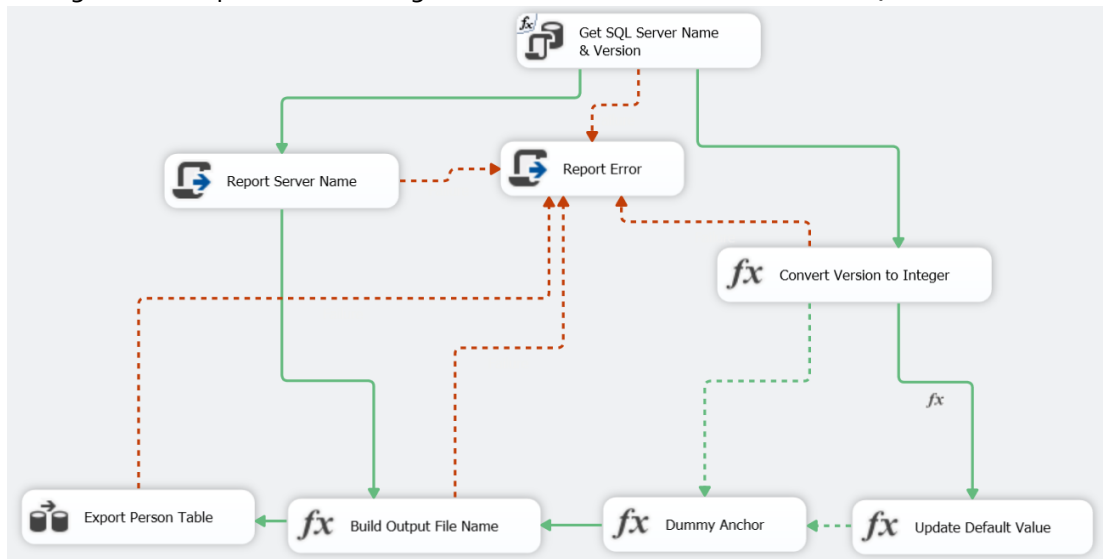
36. Right click on the FlatFile.CustomerData connection and select properties. This will open properties dialog in bottom right if not already open..



37. Update the expression information for connection string by connecting it to FileName variable.



38. Change the multiple constraint logical condition to OR for Red Lines. So, it looks like below.



39. Execute the package. The package should fail!

40. Review the "Execution Results" tab to try to understand the error.

41. The workflow is expecting two values to be returned from the first task "Get SQL Server Name & Version". However, the original statement captured one value in step #6.

42. Update the project parameter value to "SELECT @@ServerName AS ServerName, SERVERPROPERTY('ProductMajorVersion') AS SQLVersion".

43. Adjust the result set mapping to ServerName and SQLVersion.

General	Result Name	Variable Name
Parameter Mapping	0	User::ServerName
Result Set	1	User::SQLVersion
Expressions		

44. Execute package again to see the execution path.