Print this Page **SQL-Server-Performance.com**

Importing Poorly Formatted Text Files using SSIS

By : Dinesh Asanka

Introduction

Handling text files from SQL Server Integration Service (SSIS) is not new and I am not going to discuss about importing well formatted text files. There are many occasions when database developers must import text files which are not properly formatted. I will discuss three methods of doing this. For these situations we are going to use Script Component data flow task.

Knowledge-wise you need to have basic understanding of SSIS packages. If you have created SSIS package to import traditional text files that will be more than enough. We are going to do some .NET scripting, you need to have some understanding of .NET coding.

Resource-wise you need to have SQL Server Business Intelligence Development Studio installed in your computer.

Case 1: Importing Text files with different row delimiters in different rows

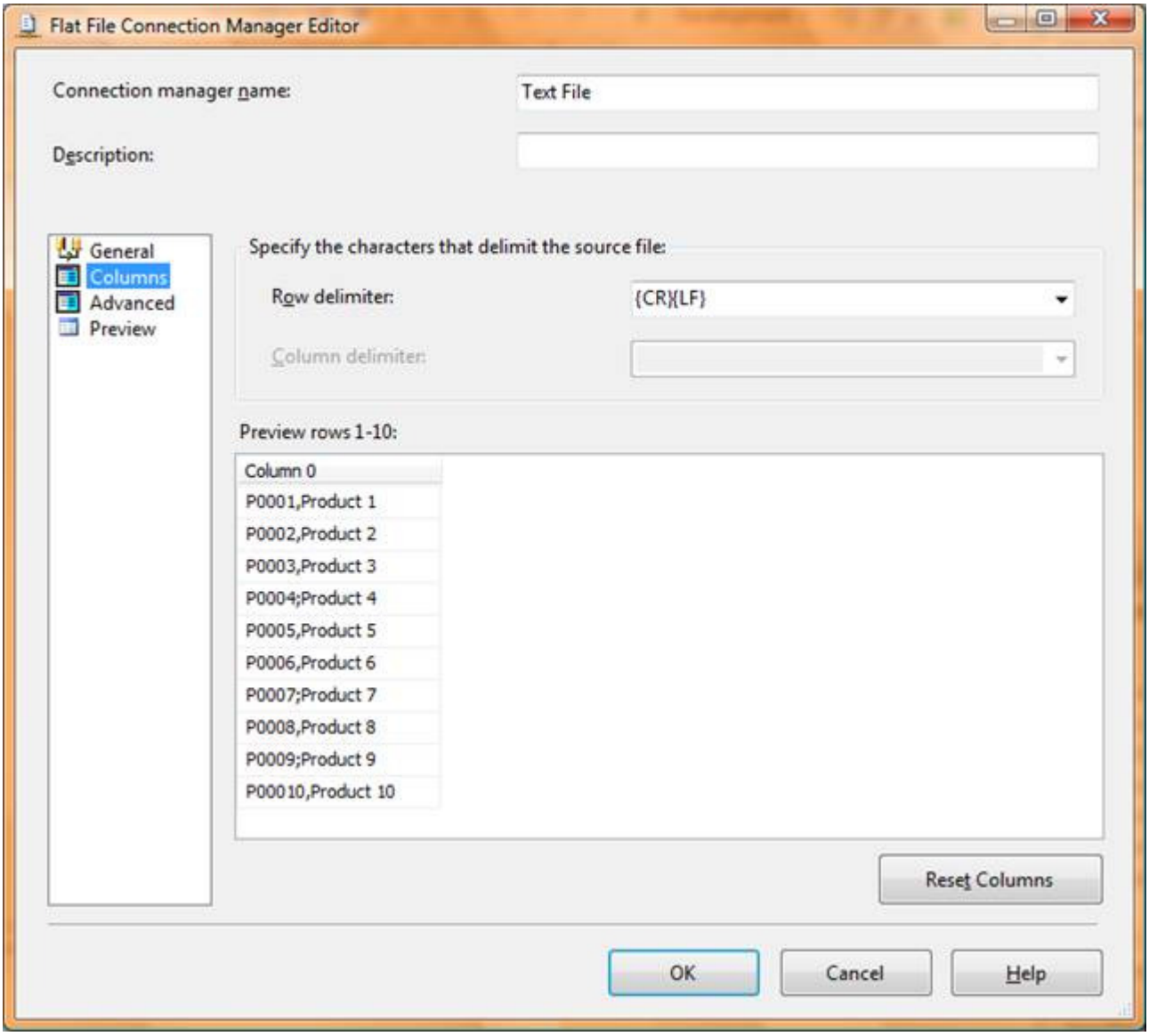
I am sure that you have imported text files. The text files normally have several columns which are separated by either comma (,), semi colon (;) or colon (:) etc. However, what would happen when there are different column delimiters. For example what if you have data in a text file like the following.

P0001,Product 1
P0002,Product 2
P0003,Product 3
P0004;Product 4
P0005,Product 5
P0006,Product 6
P0007;Product 7
P0008,Product 8
P0009;Product 9
P00010,Product 10

You can see that above two columns are separated either by comma or semi-colon. Let us see how can we create a package to import this data. I will explain how to write the first package in detail.

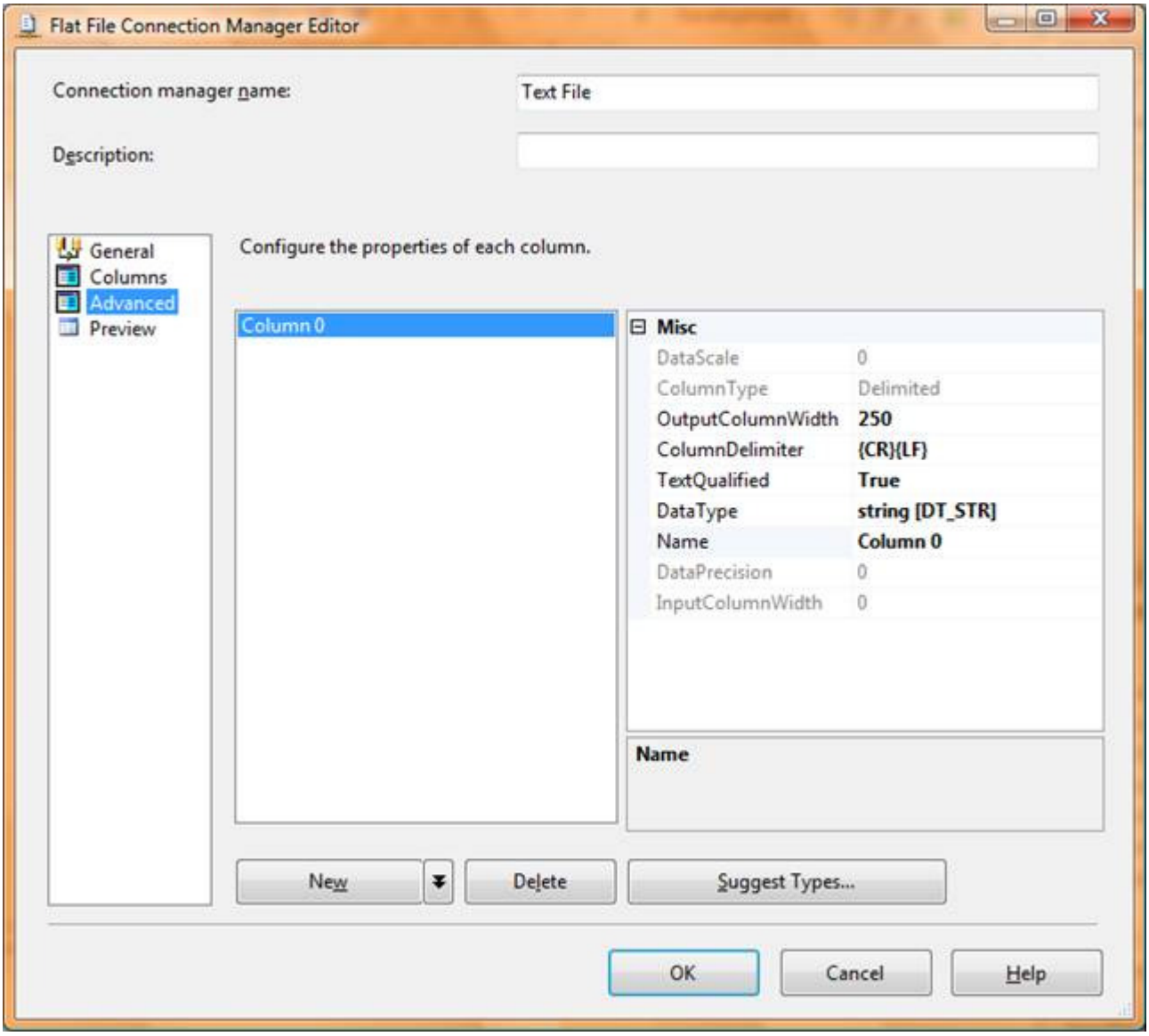
After creating a package project and new package in the project, create a connection manager named Text File. In the General section of the connection manager, you need to give the path for the text file.

Next you need to select columns options. Select row delimiter as {CR}{LF}.



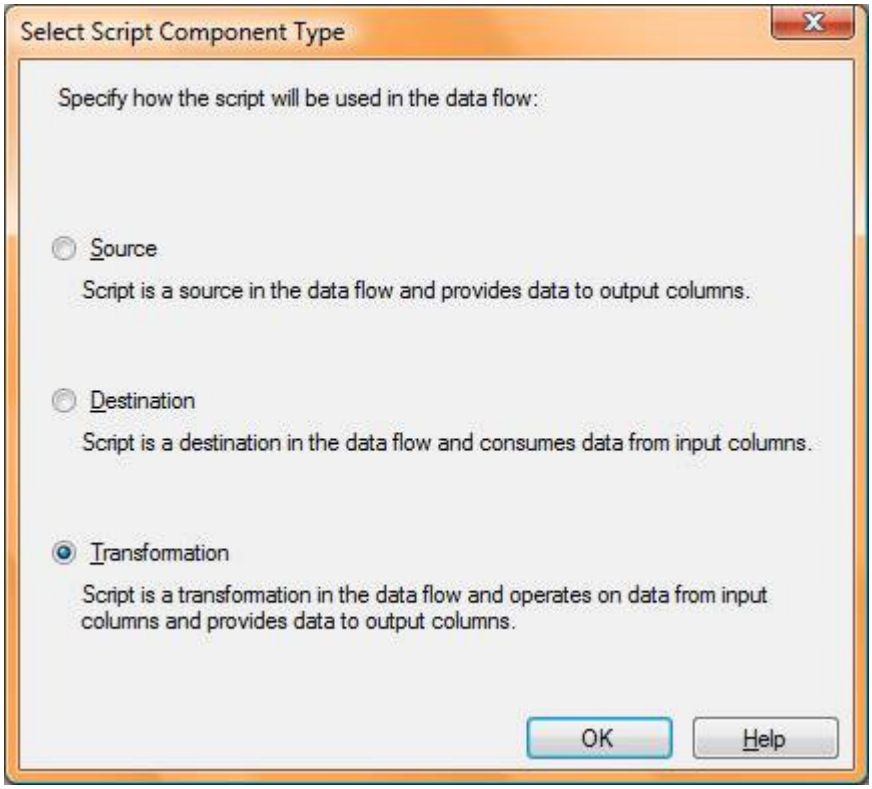
After selecting the row delimiter, you need to set the column delimiter to something other than {CR}{LF}. Otherwise it won't allow you to confirm this dialogue. However, this does not really matter as this option will be disabled later. After doing this you can see that entire record of the text file as shown in one column.

Next select advanced section. In that you have to give enough length to OutputColumnWidth. In this case it is 250.



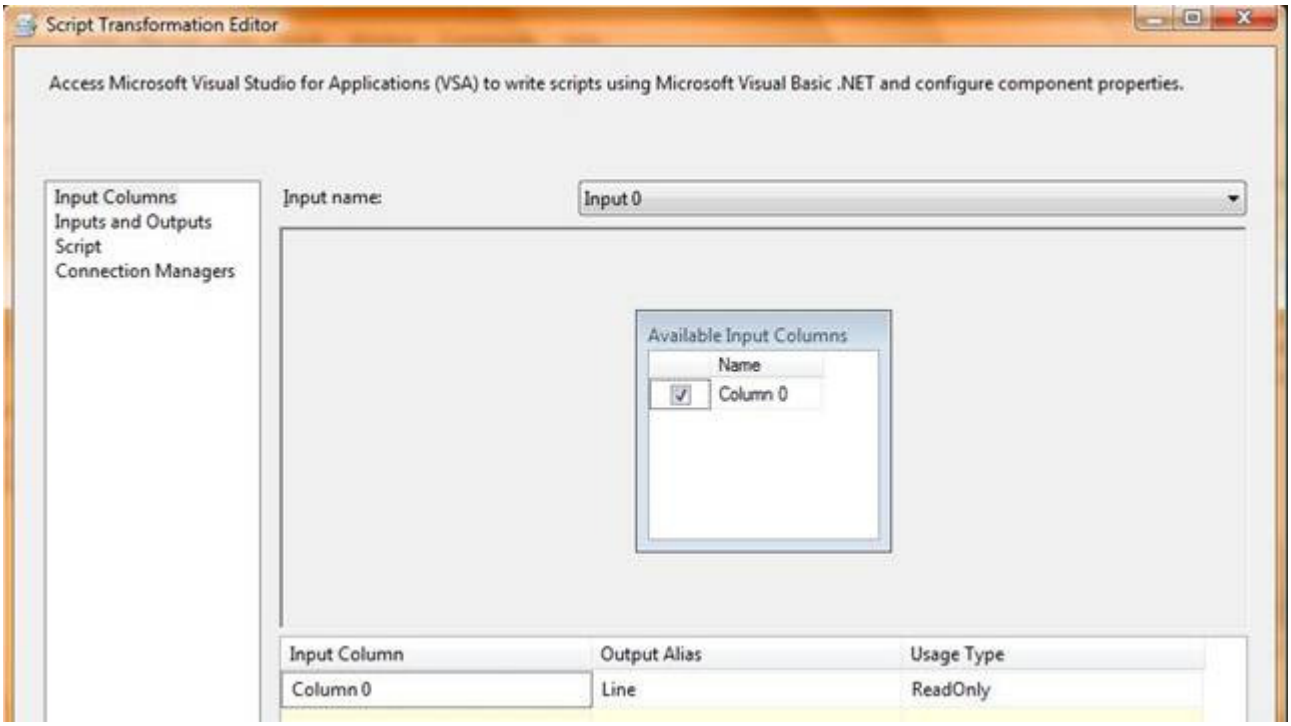
Then drag and drop a data flow task to control flow. Inside the added data flow, drag and drop Flat File Source. As you have only one text file connection manager by default - Text File will be attached to this Flat File Source. If not you have to assign correct text file connection from the connection option of the flat file source.

Next is adding most import component – the script component. Just drag and drop a script task to the data flow area. Just after dragging the script component you will get the following screen.

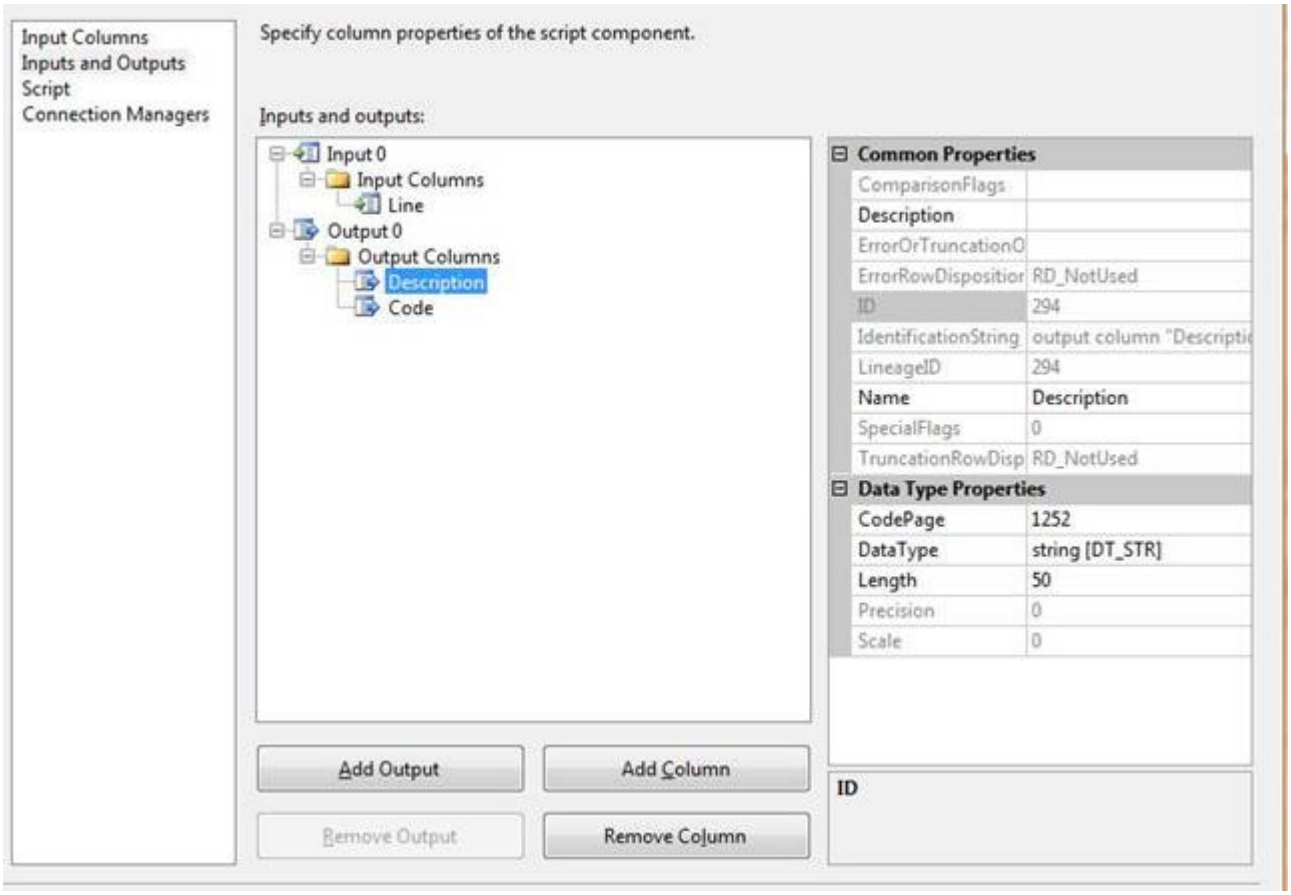


It is obvious that we are going to do a Transformation which is the last option. I will discuss about other two options in a separate article.

Next task is to configure the Script component. We have three options to configure - Input Columns, Input and Outputs and Script. If you select Input option you see the following screen. In this screen you can leave the Output Alias as it is Column 0 by default. But for the completeness I have changed this to Line.



Next option is , Inputs and Outputs. Here we have to define the input and output column properties.



As you can see on the input columns tree node, there is only one element which is Line. Line was defined in the Input Columns.

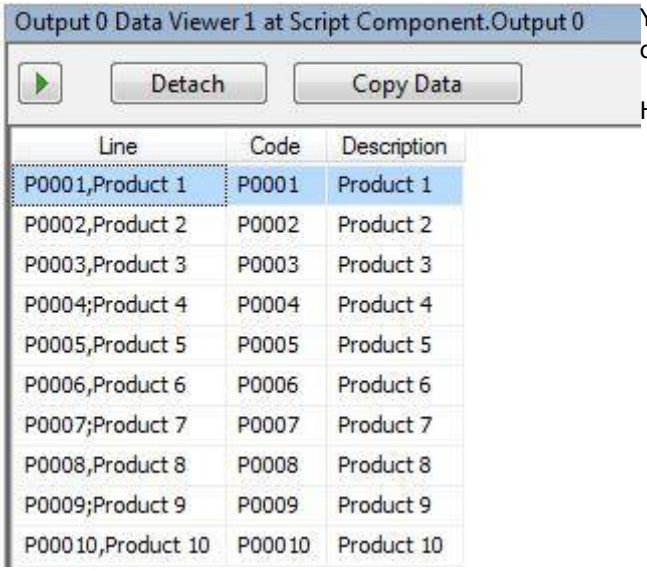
Next select Output Columns and add two columns , code and description by clicking the Add Column button. Allocate the correct data type and length for each column. In this case I have selected string [DT_STR] as data type and 50 as the data length.

An important configuration of the script component is Script option. Select the Script option and click the Design Script button. You will be taken to the Microsoft Visual Studio to add necessary .NET code.

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)

Dim strRow As String
Dim strColSeperator As String
Dim rowValues As String()
strRow = Row.Line.ToString()
If strRow.Contains(",") Then
strColSeperator = (",")
ElseIf strRow.Contains(";") Then
strColSeperator = ";"
End If

rowValues = Row.Line.Split(CChar(strColSeperator))
Row.Code = rowValues.GetValue(0).ToString()
Row.Description = rowValues.GetValue(1).ToString()

 End Sub
```

Above is the only code you need to add. From the *Contains* function we will identify the column separator for the row. Then using split function and passing the correct column delimiter we can separate the two columns.

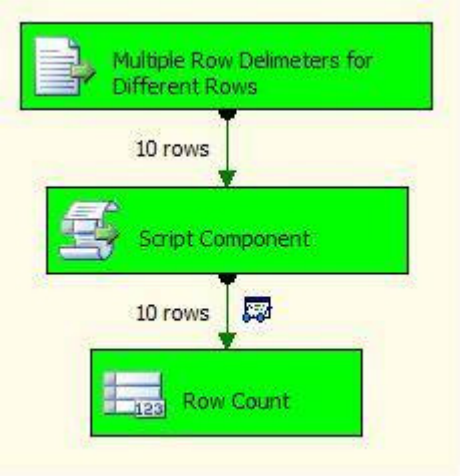
For demonstration purposes I have added a data viewer. Above is the output of the data viewer.



Line	Code	Description
P0001,Product 1	P0001	Product 1
P0002,Product 2	P0002	Product 2
P0003,Product 3	P0003	Product 3
P0004;Product 4	P0004	Product 4
P0005,Product 5	P0005	Product 5
P0006,Product 6	P0006	Product 6
P0007;Product 7	P0007	Product 7
P0008,Product 8	P0008	Product 8
P0009;Product 9	P0009	Product 9
P00010,Product 10	P00010	Product 10

You can see that data was separated despite containing different column delimiters.

Here is what the final package looks like:



Case 2: different Column Delimiters in Same Row

In Case 1 we discussed text files which have different column separators in different columns. What if you have different column delimiters in same row. For example, consider the following data set in a text file.

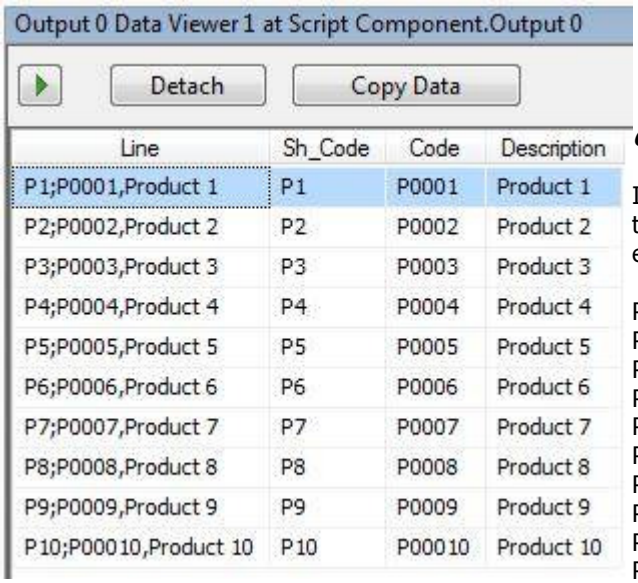
P1;P0001,Product 1
P2;P0002,Product 2
P3;P0003,Product 3
P4;P0004,Product 4
P5;P0005,Product 5
P6;P0006,Product 6
P7;P0007,Product 7
P8;P0008,Product 8
P9;P0009,Product 9
P10;P00010,Product 10

The first two columns are separated by semi colon while the other two columns are separated by a comma. There is a similar situation with Case 1 – the only difference is the script component.

You will need to add a another column to the output columns, namely Sh_code.

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)

Dim strRow As String
Dim strColSeperator As String
Dim rowValues1 As String()
Dim rowValues2 As String()

rowValues1 = Row.Line.Split(CChar(";"))
Row.ShCode = rowValues1.GetValue(0).ToString()
rowValues2 = rowValues1.GetValue(1).ToString().Split(CChar(","))
Row.Code = rowValues2.GetValue(0).ToString()
Row.Description = rowValues2.GetValue(1).ToString()

End Sub
```

This time we have used two split functions to seperate the data. The final output will appear as below:


Output 0 Data Viewer 1 at Script Component.Output 0

Detach | Copy Data

Line	Sh_Code	Code	Description
P1;P0001,Product 1	P1	P0001	Product 1
P2;P0002,Product 2	P2	P0002	Product 2
P3;P0003,Product 3	P3	P0003	Product 3
P4;P0004,Product 4	P4	P0004	Product 4
P5;P0005,Product 5	P5	P0005	Product 5
P6;P0006,Product 6	P6	P0006	Product 6
P7;P0007,Product 7	P7	P0007	Product 7
P8;P0008,Product 8	P8	P0008	Product 8
P9;P0009,Product 9	P9	P0009	Product 9
P10;P00010,Product 10	P10	P00010	Product 10

Case 3: Variable Columns

In the previous cases we have the same number of columns throughout. What if the number of columns also varied? For example, consider the following text file:

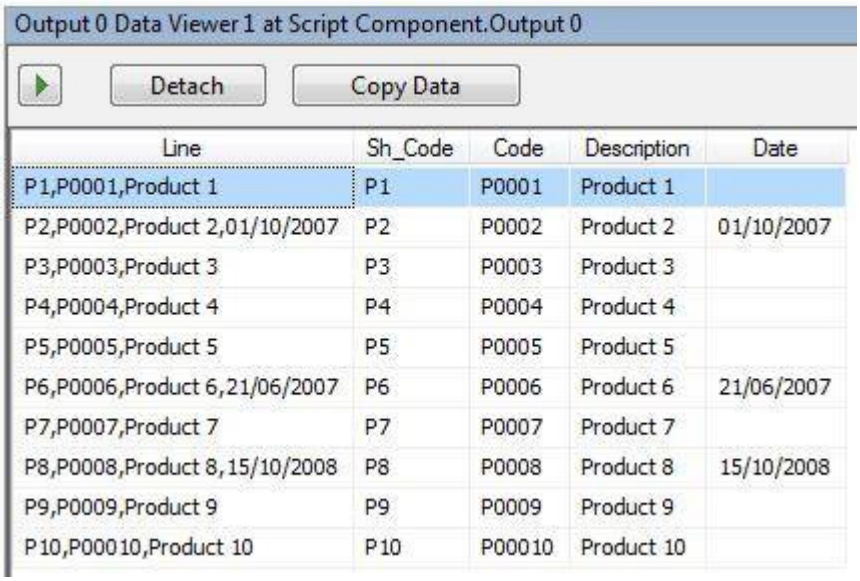
P1,P0001,Product 1
P2,P0002,Product 2,01/10/2007
P3,P0003,Product 3
P4,P0004,Product 4
P5,P0005,Product 5
P6,P0006,Product 6,21/06/2007
P7,P0007,Product 7
P8,P0008,Product 8,15/10/2008
P9,P0009,Product 9
P10,P00010,Product 10

You can see that the date is available for only few columns. You will not be able to use traditional text file handling with SSIS for above case.

Again, the difference with the previous two cases is the script component. To start with you will need to add new output column named Date. Then we need to add some .NET code for the script component:

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)

Dim strRow As String
Dim strColSeperator As String
Dim rowValues As String()

rowValues = Row.Line.Split(CChar(","))
Row.ShCode = rowValues.GetValue(0).ToString()
Row.Code = rowValues.GetValue(1).ToString()
Row.Description = rowValues.GetValue(2).ToString()

If rowValues.GetUpperBound(0) = 3 Then
Row.Date = rowValues.GetValue(3).ToString()
End If

End Sub

End Class
```

In this case aftr splitting, the script checks the number of elements the array has. If it 3 then you have a date column in the data row. Again we will check the data viewer output:


Output 0 Data Viewer 1 at Script Component.Output 0

Detach | Copy Data

Line	Sh_Code	Code	Description	Date
P1,P0001,Product 1	P1	P0001	Product 1	
P2,P0002,Product 2,01/10/2007	P2	P0002	Product 2	01/10/2007
P3,P0003,Product 3	P3	P0003	Product 3	
P4,P0004,Product 4	P4	P0004	Product 4	
P5,P0005,Product 5	P5	P0005	Product 5	
P6,P0006,Product 6,21/06/2007	P6	P0006	Product 6	21/06/2007
P7,P0007,Product 7	P7	P0007	Product 7	
P8,P0008,Product 8,15/10/2008	P8	P0008	Product 8	15/10/2008
P9,P0009,Product 9	P9	P0009	Product 9	
P10,P00010,Product 10	P10	P00010	Product 10	

Sample

For your reference I have added sample SSIS package for you along with the text files I have used. Download here

Case	Package	Text File
Case 1	MultipleRowDelDifferentRows.dtsx	MultipleRowDelDifferentRows.txt
Case 2	MultipleRowDelSameRows.dtsx	MultipleRowDelSameRows.txt
Case 3	MultipleNumberofColums.dtsx	MultipleNumberofColums.txt

You may need to change all the connection manager file paths to suit to your file path.

Conclusion

I am sure you have come across with many situations with text files. What are those situations and what were alternatives you implemented? I would love to hear your feedback at dineshasanka@gmail.com

---

© 2000 - 2007 vDerivatives Limited All Rights Reserved.