Reading folder and subfolders for PDFs with SSIS

Objective:
To give the ability to open, parse, and process Adobe PDF files in one folder and all the subfolders destined for either CSV output or writing to a database.

The Thought Process:
I wanted to program the ability to read a folder and subfolders for PDF files and process them. The PDF files which are similar are contained in separate folders. The script task that processes the pdf is based on the folder it is in. For example, all PDFs in an Alabama folder is processed through the Alabama script task. This way, several PDF files can be processed using the same code because the files are so similar.

C# and Adobe PDF files:
I used the same iTextSharp library for this script.

How it works:
Besides the timer, the whole process is contained in an **ForEach Loop**. The script creates a list of folders and PDF files and processes each PDF file sequentienally (alphabetically). The **Start Timer** and **Stop Timer** have been added to give feedback on how long the process took.

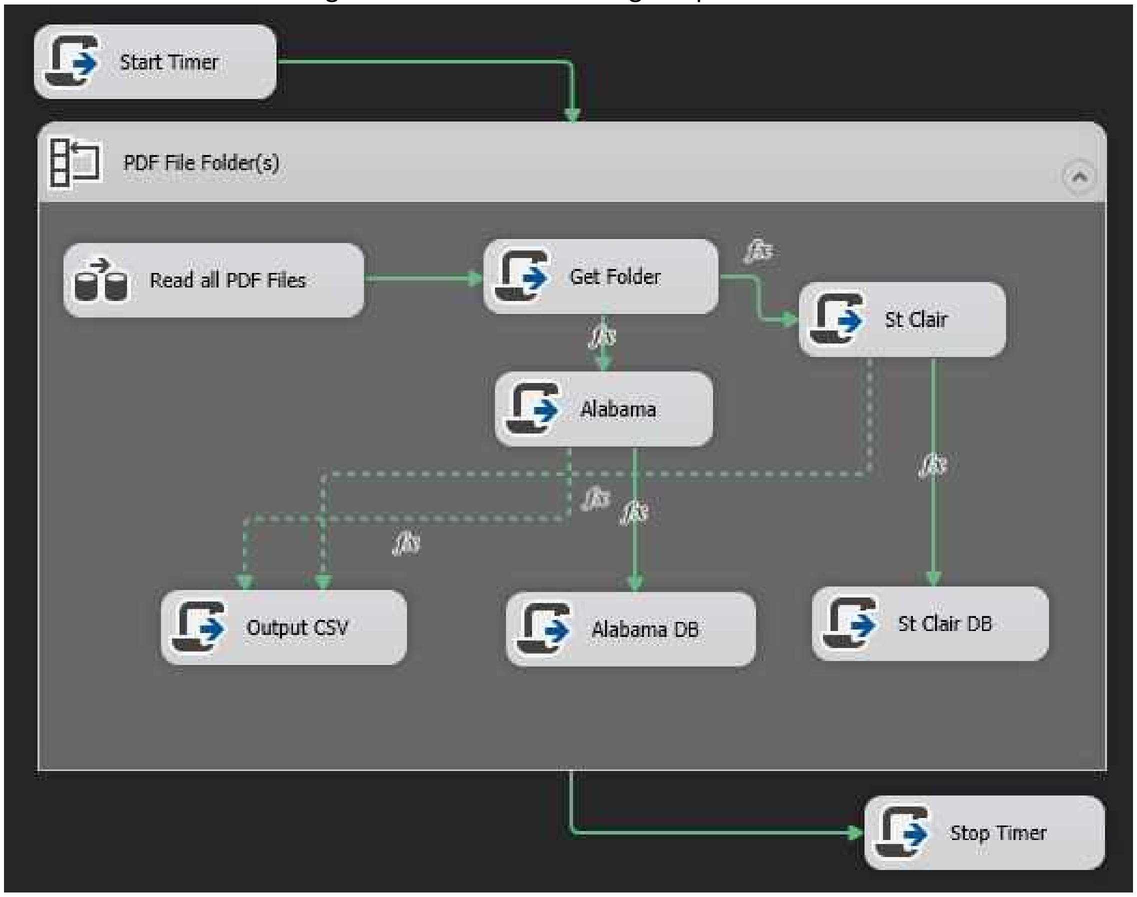
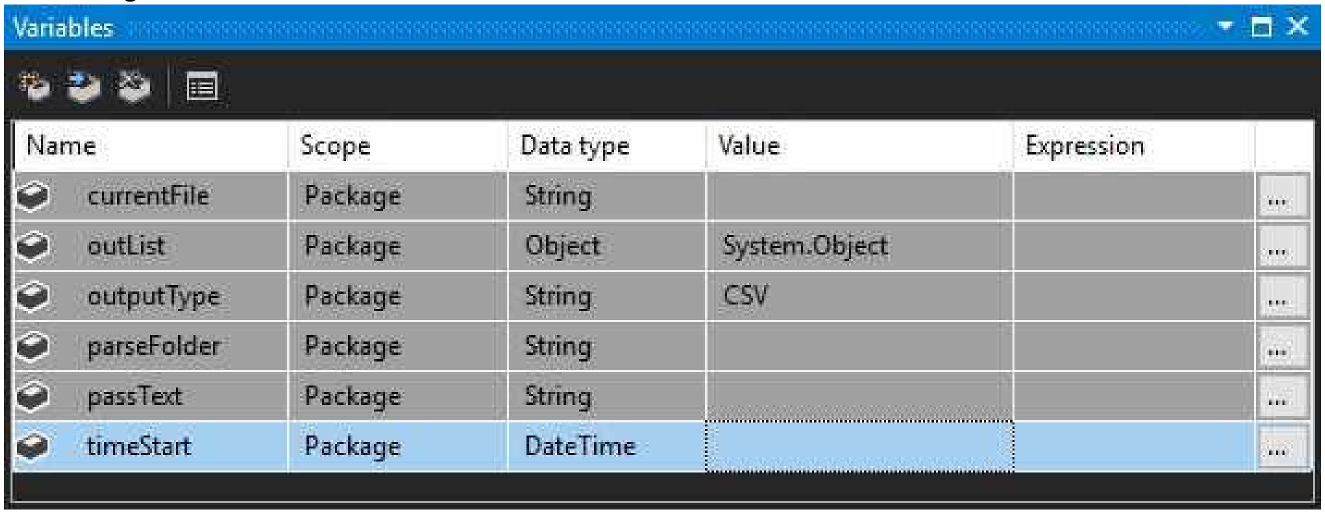


Figure 1. The main project

The Package variables are shown below.



Variables

Name	Scope	Data type	Value	Expression
currentFile	Package	String		
outList	Package	Object	System.Object	
outputType	Package	String	CSV	
parseFolder	Package	String		
passText	Package	String		
timeStart	Package	DateTime		

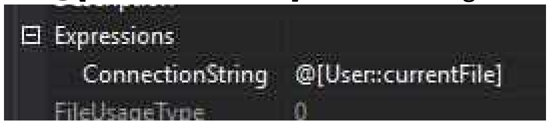
Figure 2. The Package variables

The variables are described as follows:

- **currentFile** – This is the filename of the file that the script is currently processing.
- **outList** – The array of data that is compiled to be processed later. This must be declared as an object because it is a list.
- **outputType** – Switch for the determining if the output should be a CSV file or processed through SQL. Options are **CSV** and **SQL**.
- **parseFolder** – The folder name where the PDFs were found. This helps to determine which script task to run.
- **passText** – The raw PDF file data that is passed to another script to process.
- **timeStart** – Keeps the date and time of when the script starts running.

The **Start Timer** is executed first to set the timeStart variable.

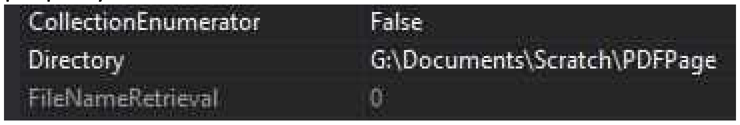
The PDF Files connection manager contains the **ConnectionString** Expression. This expression is set to the **@[User::currentFile]** variable string.



⊟ Expressions	
ConnectionString	@[User::currentFile]
FileUsageType	0

Figure 3. The PDF Files ConnectionString Expression

The folder which the **ForEach Loop Container** starts its search for PDFs is declared in the **Directory** property.



CollectionEnumerator	False
Directory	G:\Documents\Scratch\PDFPage
FileNameRetrieval	0

Figure 4. The directory property of the ForEach Loop Container

The first item in the **ForEach** Loop Container is a **Data Flow Task**. This task requires a **Connection Manager** to be created. I called this Connection Manager: **PDF Files**.

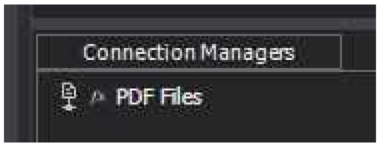
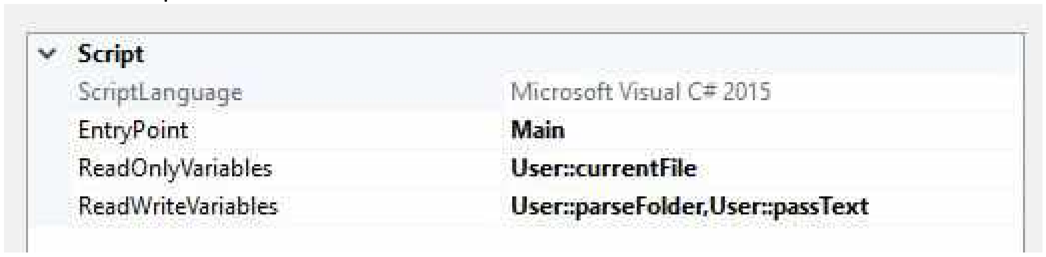


Figure 5. PDF Files Connection Manager

The current file is sent to the next task: Get Folder. The following are the variables that are passed between the previous task and next task.



Script	
ScriptLanguage	Microsoft Visual C# 2015
EntryPoint	**Main**
ReadOnlyVariables	**User::currentFile**
ReadWriteVariables	**User::parseFolder,User::passText**

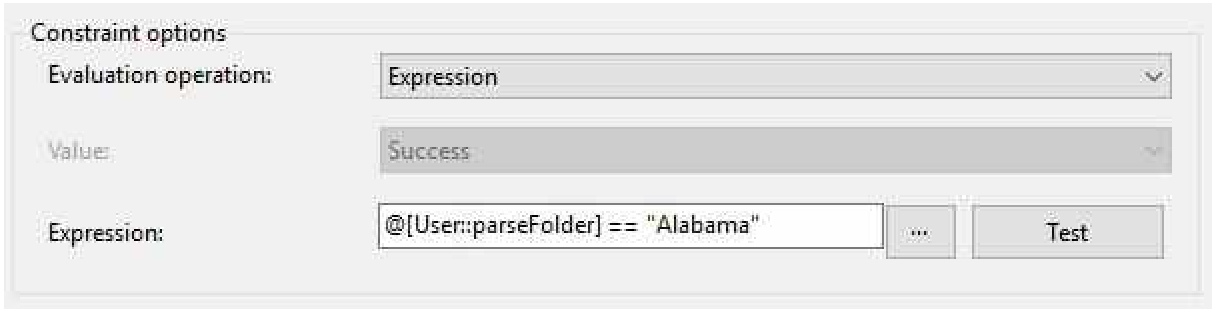
Figure 6. The Get Folder variables

This script takes the name of the PDF, opens it, reads the lines into a text string, and then reads what folder the PDF file was found in. The folder name determines which script task is run next. This script receives the text string read from the PDF file.

This project only contains two scripts but could contain more based on the folders and pdf files contained in them. The Expression in the Constraint options are shown below:

- @[User::parseFolder] == "Alabama"
- @[User::parseFolder] == "StClair"

We will use Alabama only from this point on. The folder name is named **Alabama** and contains several PDF files related to Alabama PDFs.



Constraint options

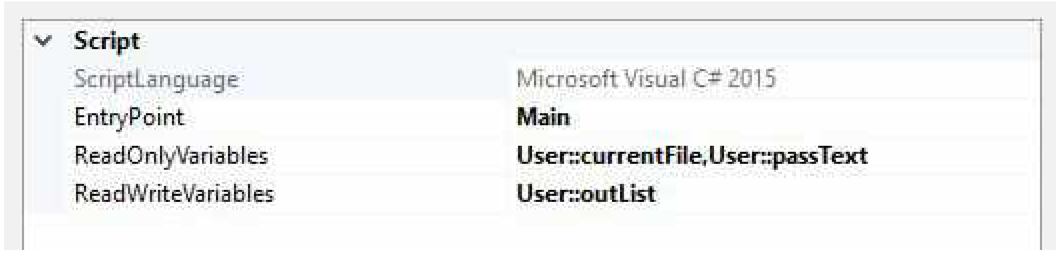
Evaluation operation: Expression

Value: Success

Expression: @[User::parseFolder] == "Alabama"

Figure 7. The Alabama folder has been read

The following variables are read and written by the "Alabama" Script Task.

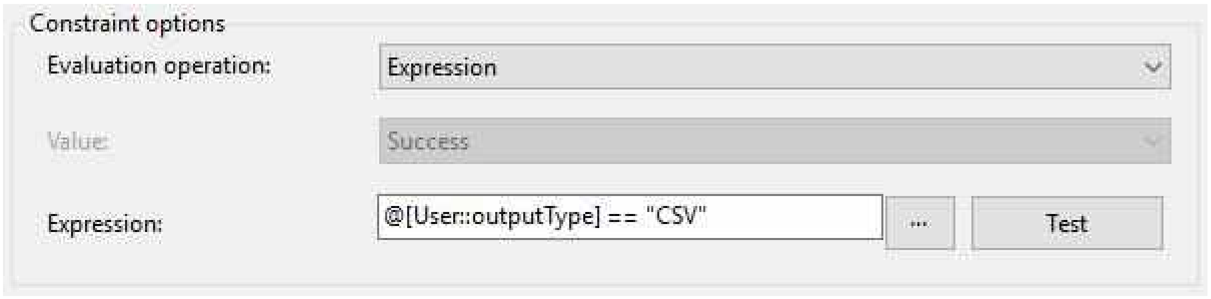


Script	
ScriptLanguage	Microsoft Visual C# 2015
EntryPoint	**Main**
ReadOnlyVariables	**User::currentFile,User::passText**
ReadWriteVariables	**User::outList**

Figure 8. The Alabama variables

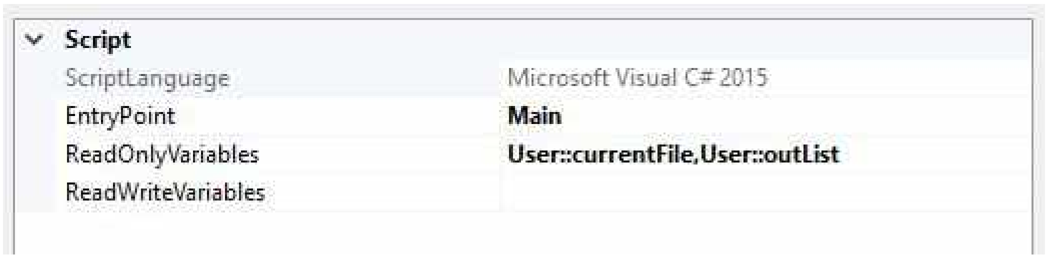
This Script Task processes the PDF text strings and adds them to the outList variable in CSV format.

An expression is added to the connection string to either **Output CSV** or **Alabama DB**. This is shown set below:


Constraint options

Evaluation operation:	Expression
Value:	Success
Expression:	@[User::outputType] == "CSV"

Figure 9. The Output CSV Script Task will be executed if this is set to "CSV"


Script

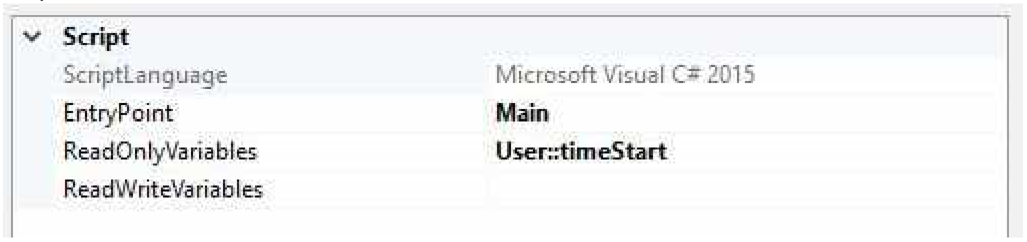
ScriptLanguage	Microsoft Visual C# 2015
EntryPoint	**Main**
ReadOnlyVariables	**User::currentFile,User::outList**
ReadWriteVariables	

Figure 10. The Alabama Output CSV Script Task

The **currentFile** is passed to determine what the name of the CSV file will be. The PDF extension is replaced with CSV for creating a unique filename. The **outList** contains the CSV data.

The folders will be read alphabetically and when all the folders have been read and processed the **ForEach Loop Container** will then run the **Stop Timer** Script Task.

The Stop Timer Script Task requires the **timeStart** variable to be passed to perform a calculation on the elapsed time.


Script

ScriptLanguage	Microsoft Visual C# 2015
EntryPoint	**Main**
ReadOnlyVariables	**User::timeStart**
ReadWriteVariables	

Figure 11. The Stop Timer Script Task variable

A MessageBox should now show the elapsed time.

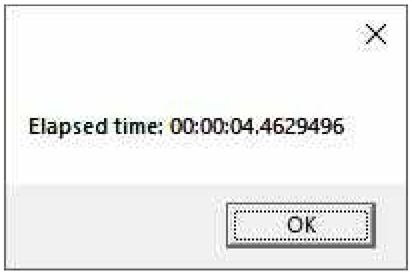

Elapsed time: 00:00:04.4629496

Figure 12. The MessageBox showing elapsed time

Known Issues (Possibly):
The output folder is “hard coded” in the Output CSV Script Task. This could easily be converted to a Package variable for future projects.

Possible future enhancements:

- Change the “hard coded” output folder name into a Package variable.

Michael Keller
Accumatch
Reading folder and subfolders for PDFs with SSIS
October 29, 2019 (Revised October 30, 2019)