Reading and Parsing Large PDFs with SSIS

Objective:
To give the ability to open, parse, and process a large Adobe PDF file destined for either CSV output or writing to a database.

The Thought Process:
I needed the ability to open a large PDF and read the lines from it. The PDF file I used for this application was 19,457 pages long.

C# and Adobe PDF files:
I used the iTextSharp library and references to open and read this PDF.

Power BI and this particular PDF:
I did try to open (or connect) this same PDF (BuffaloCity.pdf) with Microsoft Power BI and after twenty minutes of waiting, I receive the following error:

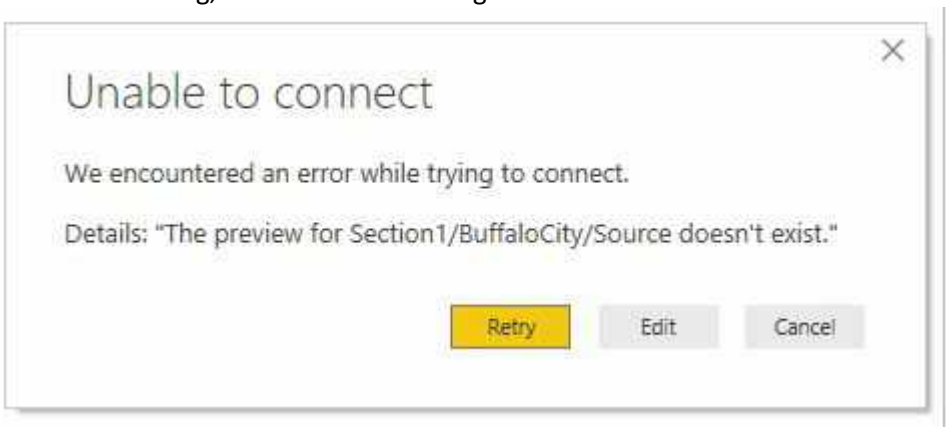


Figure 1. Power BI Error Message

How it works:
Currently, there is just a single script task employed in this package.

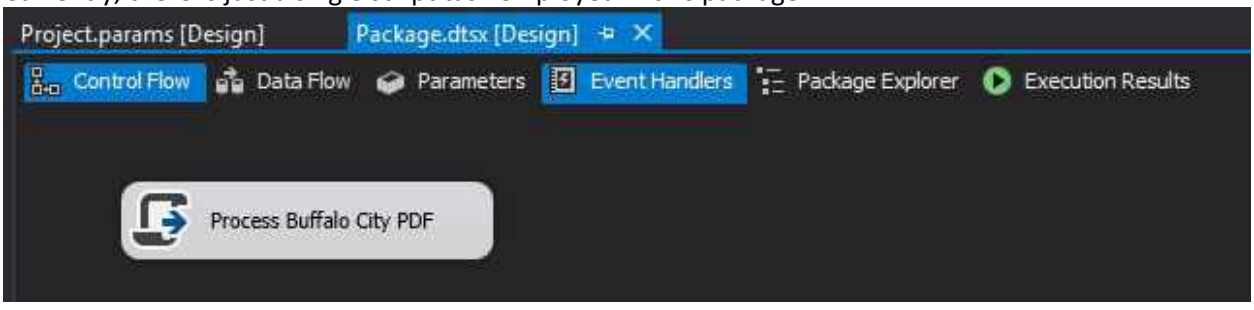


Figure 2. A single Script Task is used in this package

Project.params are used in this project. A single parameter of the PDF filename is utilized. The following shows the Project.params window.

Figure 3. The Project.params window

This parameter is passed onto the Script Task through the **ReadOnlyVariables** as a Project parameter.

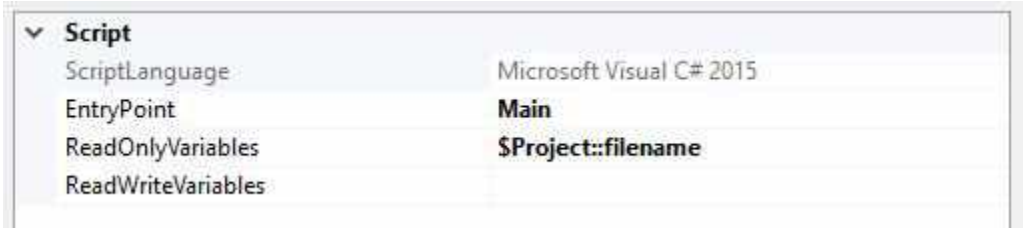


Figure 4. The ReadOnlyVariables in the Script Task Editor

Some quirks with the PDF are:

- The first page is completely blank.
- Close to the end of the PDF file there are two pages where not useable information is located. I had to write checking into the code to ignore these two pages.
- The last 22 pages of the PDF contain summaries and totals and are not of any value to the result.

The package does contain a timer to record the amount of time taken for the scrip to run.

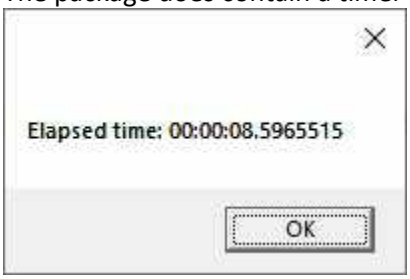


Figure 5. Program elapsed time

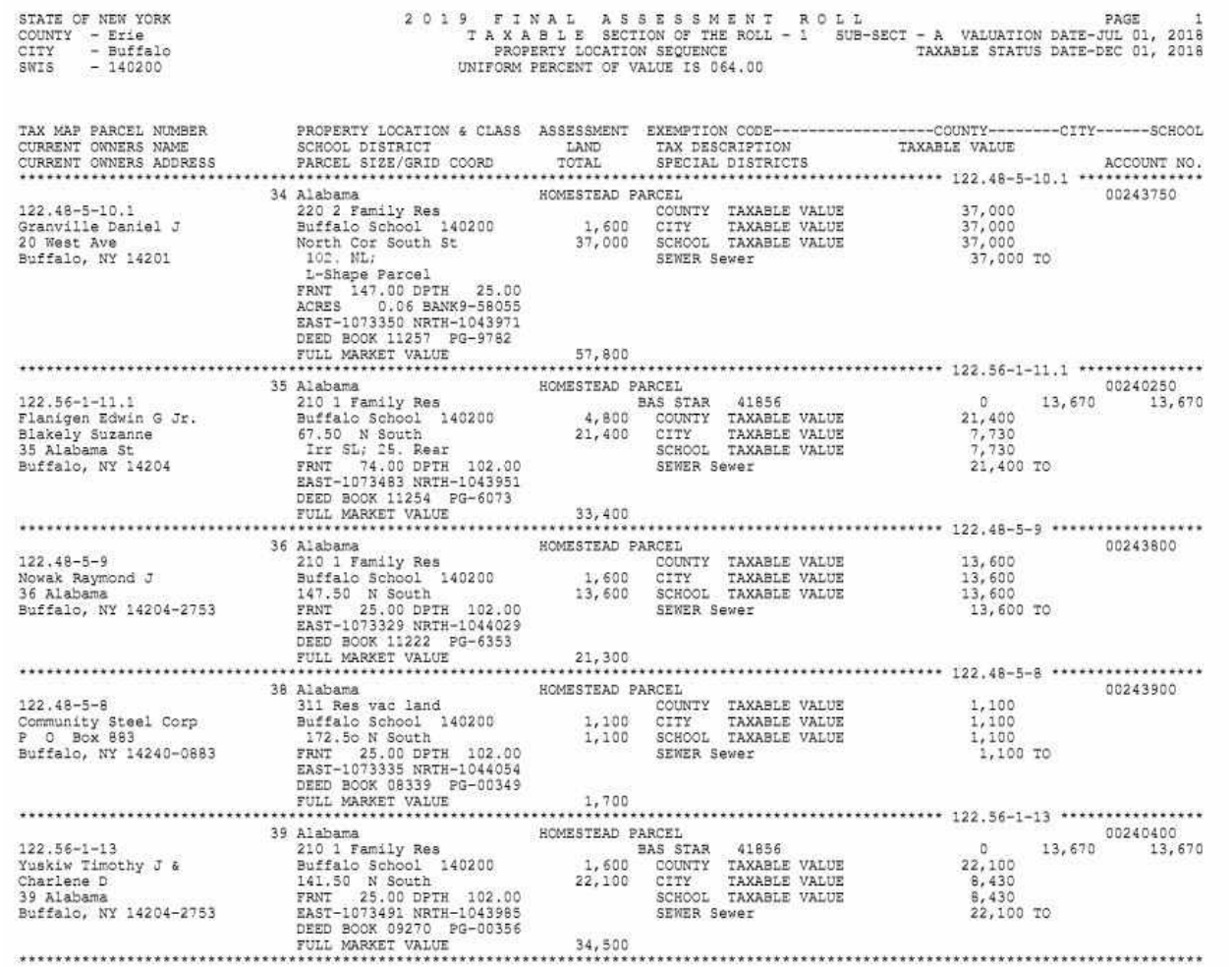
Currently, this Script Task does output a CSV file based on the input filename. Below is a portion of the CSV file.

1	Parcel	Homestead Parcel	Deed Book	Market Value
93837	133.26-7-6	11829800	10977 PG-5282	16400
93838	133.26-7-7.1	11829950	10919 PG-8448	2156300
93839	133.26-1-35	11165800	08253 PG-00339	351600
93840	133.34-2-22.112	11832765	11293 PG-2612	31300
93841	133.43-5-3	11169900	11137 PG-3856	7200
93842	133.43-6-3	11992000	08151 PG-00265	75000
93843	133.47-4-39	11866100	09577 PG-00246	9400
93844	133.47-4-38	11866200	08629 PG-00283	9400
93845	133.47-4-37	11866300	06602 PG-00367	266400
93846	133.79-1-75	11870800	10867 PG-4761	2200
93847	133.23-3-21	11879100	03979 PG-00540	706300
93848	123.14-4-25	11193700	03858 PG-00350	102300
93849	123.63-1-19	11882800	11102 PG-6043	402800
93850	133.34-2-22.12	11832770	11222 PG-4662	2127800
93851	133.67-3-1	11926900	01286 PG-00560	164200
93852	123.58-1-8	11201700	02933 PG-00300	12700
93853	123.11-1-5	11201800	08566 PG-00037	34800
93854	123.41-5-25	11208700	07252 PG-00231	9400
93855	123.11-1-6	11211350	02089 PG-00205	13000
93856	123.11-1-4	11211300	09182 PG-00470	13000
93857	123.11-1-3	11216400	08566 PG-00035	15600
93858	133.57-1-3	11957600	06097 PG-00293	203400
93859	123.59-1-7	11230400	06814 PG-00467	7800
93860	133.51-1-34	11963800	05773 PG-00061	0
93861	133.57-5-20	11970200	09797 PG-00403	800
93862	123.82-3-82	11241100	11104 PG-9301	6600
93863	123.82-3-49	11244500	06502 PG-00121	200
93864	123.82-4-26	11249000	11314 PG-5360	2200

BuffaloCity

Figure 6. CSV output file

The **BuffaloCity.pdf** file was not in a tabular format and a lot of processing had to be done to extract the data into useable form. Here is a screenshot of a page from the pdf file.



STATE OF NEW YORK 2019 FINAL ASSESSMENT ROLL PAGE 1
COUNTY - Erie TAXABLE SECTION OF THE ROLL - 1 SUB-SECT - A VALUATION DATE-JUL 01, 2018
CITY - Buffalo PROPERTY LOCATION SEQUENCE TAXABLE STATUS DATE-DEC 01, 2018
SWIS - 140200 UNIFORM PERCENT OF VALUE IS 064.00

TAX MAP PARCEL NUMBER / CURRENT OWNERS NAME / CURRENT OWNERS ADDRESS	PROPERTY LOCATION & CLASS / SCHOOL DISTRICT / PARCEL SIZE/GRID COORD	ASSESSMENT / LAND / TOTAL	EXEMPTION CODE / TAX DESCRIPTION / SPECIAL DISTRICTS	TAXABLE VALUE	COUNTY / CITY / SCHOOL
	34 Alabama	HOMESTEAD PARCEL			122.48-5-10.1 / ACCOUNT NO. 00243750
122.48-5-10.1 Granville Daniel J 20 West Ave Buffalo, NY 14201	220 2 Family Res Buffalo School 140200 North Cor South St 102. NL; L-Shape Parcel FRNT 147.00 DPTH 25.00 ACRES 0.06 BANK9-58055 EAST-1073350 NRTH-1043971 DEED BOOK 11257 PG-9782 FULL MARKET VALUE	1,600 37,000 57,800	COUNTY TAXABLE VALUE CITY TAXABLE VALUE SCHOOL TAXABLE VALUE SEWER Sewer	37,000 37,000 37,000 37,000 TO	
	35 Alabama	HOMESTEAD PARCEL			122.56-1-11.1 / ACCOUNT NO. 00240250
122.56-1-11.1 Flanigen Edwin G Jr. Blakely Suzanne 35 Alabama St Buffalo, NY 14204	210 1 Family Res Buffalo School 140200 67.50 N South Irr SL; 25. Rear FRNT 74.00 DPTH 102.00 EAST-1073483 NRTH-1043951 DEED BOOK 11254 PG-6073 FULL MARKET VALUE	4,800 21,400 33,400	BAS STAR 41856 COUNTY TAXABLE VALUE CITY TAXABLE VALUE SCHOOL TAXABLE VALUE SEWER Sewer	0 13,670 13,670 21,400 7,730 7,730 21,400 TO	
	36 Alabama	HOMESTEAD PARCEL			122.48-5-9 / ACCOUNT NO. 00243800
122.48-5-9 Nowak Raymond J 36 Alabama Buffalo, NY 14204-2753	210 1 Family Res Buffalo School 140200 147.50 N South FRNT 25.00 DPTH 102.00 EAST-1073329 NRTH-1044029 DEED BOOK 11222 PG-6353 FULL MARKET VALUE	1,600 13,600 21,300	COUNTY TAXABLE VALUE CITY TAXABLE VALUE SCHOOL TAXABLE VALUE SEWER Sewer	13,600 13,600 13,600 13,600 TO	
	38 Alabama	HOMESTEAD PARCEL			122.48-5-8 / ACCOUNT NO. 00243900
122.48-5-8 Community Steel Corp P O Box 883 Buffalo, NY 14240-0883	311 Res vac land Buffalo School 140200 172.5o N South FRNT 25.00 DPTH 102.00 EAST-1073335 NRTH-1044054 DEED BOOK 08339 PG-00349 FULL MARKET VALUE	1,100 1,100 1,700	COUNTY TAXABLE VALUE CITY TAXABLE VALUE SCHOOL TAXABLE VALUE SEWER Sewer	1,100 1,100 1,100 1,100 TO	
	39 Alabama	HOMESTEAD PARCEL			122.56-1-13 / ACCOUNT NO. 00240400
122.56-1-13 Yuskiw Timothy J & Charlene D 39 Alabama Buffalo, NY 14204-2753	210 1 Family Res Buffalo School 140200 141.50 N South FRNT 25.00 DPTH 102.00 EAST-1073491 NRTH-1043985 DEED BOOK 09270 PG-00356 FULL MARKET VALUE	1,600 22,100 34,500	BAS STAR 41856 COUNTY TAXABLE VALUE CITY TAXABLE VALUE SCHOOL TAXABLE VALUE SEWER Sewer	0 13,670 13,670 22,100 8,430 8,430 22,100 TO	

Figure 7. The BuffaloCity.pdf file

The script for this project is below:

```
#region Namespaces
using System;
using System.Data;
using System.IO;
using Microsoft.SqlServer.Dts.Runtime;
using System.Windows.Forms;
using iTextSharp.text.pdf;
using iTextSharp.text.pdf.parser;
using System.Text.RegularExpressions;
using System.Collections.Generic;
using System.Linq;
#endregion

namespace ST_fda72e7f5204486fba61471ad19fea7f
{
        [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
        public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
        {
    public static DateTime timeStart;

    public static DateTime getDateTime()
    {
       DateTime dt = DateTime.Now;
       return dt;
```

```
}

public static string getCSVFile(string f)
{
   string file = f.Substring(f.LastIndexOf('\\') + 1);
   string fo = file.Replace(".pdf", ".csv");
   return fo;
}

public static string getFolderFromPath(string f)
{
   string s = f.Substring(0, f.LastIndexOf('\\'));
   string folder = s.Substring(0, s.LastIndexOf('\\'));
   return folder;
}

public static string stripAmount(string amt)
{
   string a = string.Empty;
   a = amt.Replace("$", "");
   a = a.Replace(",", "");
   return a;
}

public void Main()
            {
   timeStart = getDateTime();
   string newPage = "\n";
   string newline = "\r\n";
   string fileIn = Dts.Variables["filename"].Value.ToString();
   if (File.Exists(fileIn))
   {
      // Data to Collect
      string Parcel = string.Empty;
      string HomesteadParcel = string.Empty;
      string MarketValue = string.Empty;
      string DeedBook = string.Empty;
      string FullMarketValue = string.Empty;
      // output array
      var outList = new List<string>();

      // regular variables
      string comma = ",";
      string modLine = string.Empty;
      int tempStringLength = 0;
      int breakline = 1;
      string carp = string.Empty;
      string regGoodData = string.Concat(Enumerable.Repeat("*", 30));
      string regBadData = string.Concat(Enumerable.Repeat("*", 130));

      // CSV Column Headings
      string columnHeadings = "Parcel,Homestead Parcel,Deed Book,Market Value";

      // booleans
      bool goodData = false;
      // use Headings in CSV?
      bool useHeadings = true;
      // flag to state when data is ready to write to array
      bool addToArray = false;
      // flag to exit loop when line denotes no more data to read
      bool endOfFile = false;
      // set to make sure the first line is parsed
      bool firstline = false;

      // Regular Expressions
      // Regular Expression to find Money figure
```

```
string regAmount = @"[,0-9]+";
// Regular Expression to find Homestead Parcel Number
string regHomesteadParcel = "HOMESTEAD PARCEL";
// Regular Expression to find Deed Book (whatever)
string regDeedBook = @"\d{3,}[ ]*PG[-]\d{3,}";
// Regular Expression to find FullMarketValue
string regFullMarketValue = "FULL MARKET VALUE";

// time to initialize array
if (useHeadings == true)
{
   outList.Add(columnHeadings + newline);
}

// Open PDF Reader
PdfReader reader = new PdfReader(fileIn);
string text = string.Empty;
for (int page = 1; page <= reader.NumberOfPages; page++)
{
   goodData = false;
   text = PdfTextExtractor.GetTextFromPage(reader, page) + newPage;

   modLine = text.Substring(0, text.Length);
   tempStringLength = modLine.Length;

   if (modLine.Contains(regBadData))
   {
      do
      {
         breakline = modLine.IndexOf(newPage);
         if (breakline <= 0)
         {
            goodData = false;
            break;
         }

         // get substring of full page
         carp = modLine.Substring(0, breakline);

         if (carp.Trim().StartsWith(regBadData))
         {
            goodData = false;
            break;
         }

         if (carp.Trim().StartsWith(regGoodData))
         {
            goodData = true;
            firstline = true;
         }

         if (goodData == true)
         {
            if (firstline == true)
            {
               string[] blobs = carp.Split(' ');
               Parcel = blobs[1];
               firstline = false;
            }
            Match match3 = Regex.Match(carp, regHomesteadParcel, RegexOptions.IgnoreCase);
            if (match3.Success)
            {
               int position = match3.Index + regHomesteadParcel.Length;
               var carp2 = carp.Substring(position, (carp.Length - position));
               HomesteadParcel = carp2.Trim();
            }
```

```
                        Match match4 = Regex.Match(carp, regDeedBook, RegexOptions.IgnoreCase);
                        if (match4.Success)
                        {
                           DeedBook = match4.Groups[0].Value.Trim();
                        }
                        Match match5 = Regex.Match(carp, regFullMarketValue, RegexOptions.IgnoreCase);
                        if (match5.Success)
                        {
                           int position = match5.Index + regFullMarketValue.Length;
                           var carp2 = carp.Substring(position, (carp.Length - position));
                           Match match5a = Regex.Match(carp2, regAmount, RegexOptions.IgnoreCase);
                           if (match5a.Success)
                           {
                              var carp3 = match5a.Groups[0].Value.Trim();
                              FullMarketValue = stripAmount(carp3);
                              addToArray = true;
                           }
                        }
                        if (addToArray == true)
                        {
                           addToArray = false;
                           outList.Add(Parcel + comma + HomesteadParcel + comma + DeedBook + comma +
FullMarketValue + newline);
                        }

                     }

                     tempStringLength = (modLine.Length - (breakline + 1));
                     modLine = modLine.Substring((breakline + 1), tempStringLength);

                  } while ((breakline != 0) || (endOfFile == true));
               }
            }
            // Close PDF Reader
            reader.Close();

            string fileOut = (string)Dts.Variables["filename"].Value;
            string filename = getCSVFile(fileOut);
            string foldername = getFolderFromPath(fileOut);

            //string FileFullPath = foldername + outputFolder + filename;
            string FileFullPath = foldername + "\\" + filename;
            StreamWriter sw = null;
            sw = new StreamWriter(FileFullPath, false);
            outList.ForEach(a => sw.Write(a));
            sw.Close();
         }
         else
         {
            string messageTitle = string.Empty;
            string messageBody = string.Empty;
            messageTitle = "File Not Found!";
            messageBody = fileIn + " was not found.";
            messageBody += newPage + newPage;
            messageBody += "Please check the path and/or file and try again.";
            MessageBox.Show(messageBody, messageTitle, MessageBoxButtons.OK, MessageBoxIcon.Error);
            Dts.TaskResult = (int)ScriptResults.Failure;
         }
         TimeSpan tSpan = DateTime.Now.Subtract(timeStart);
         MessageBox.Show("Elapsed time: " + tSpan.ToString());
         Dts.TaskResult = (int)ScriptResults.Success;
          }

      #region ScriptResults declaration
      enum ScriptResults
      {
```

```
        Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
        Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
    };
    #endregion

        }
}
```

Known Issues (Possibly):
This project currently only exports a CSV and does not write to a database. This could be added as an enhancement for future projects.

Possible future enhancements:

- Add the ability to insert into a database using a system or project variable.

Michael Keller
Accumatch
Reading and Parsing Large PDFs with SSIS
October 29, 2019 (Revised October 30, 2019)