PhUSE 2008

Paper PO05

Excel 2 SAS

Duong Tran, Independent Contractor, London, UK

ABSTRACT

There are numerous well known methods to transfer data from MS Excel® to SAS® such as using the Import Wizard, Proc Import, SAS/ACCESS, Dynamic Data Exchange (DDE) and the Excel Libname Engine etc, but there is another that uses the Extensible Markup Language (XML) Libname Engine that is less well known but I believe will be popular in future. This method is platform **independent**, **flexible, simple** and only need SAS Base license.

INTRODUCTION

This paper gives a brief introduction of XML, outlines steps necessary to import Excel into SAS via the XML Libname Engine **(XMLLE)** then introduces **%xcel2sas** (a SAS utility macro to convert XML document into SAS dataset) and emphasis its design novelty. That is the concept of using one of the Excel sheet to define meta data to simplify the generalisation of importing Excel into SAS.

WHAT IS XML AND HOW TO IMPORT ITINTO SAS?

XML is a simple, very flexible plain text format designed for exchanging data. So what does this mean?

The acronym XML is broadly used so it can be confusing. Note there is no mention of the word Language in the above definition. For the purpose of this paper **all** you need to know is that XML is just a plain text file with a certain structure (for other aspects of XML see www.w3.org). Let look some examples - what an XML document look like and how to import it into SAS by contrasting it to the more familiar Comma Delimited (CSV) format.

Example 1

Hello_World.csv (Comma delimited document)

```
Hello,World
I am a comma delimited, plain text document
```

Hello_World.xml (XML document)

```
<?xml version="1.0"?>
<Hello_World>
   <Record1>
     <Field1>Hello</Field1>
     <Field2>World</Field2>
   </Record1>
   <Record2>
     <Field1>I am an XML document</Field1>
     <Field2>also just plain text </Field2>
   </Record2>
</Hello_World>
```

Both of these documents are just plain text. The difference is their structure that described the data. The comma delimited document uses the comma "," whereas the XML document uses the angle brackets "<>" with designer made up tags. Note the XML document need to be well formed (i.e. the tags are hierarchy) as would the "," in the CSV document need to be in the intended place.

Importing a CSV and an XML document

We are 'all' familiar on how to import a CSV document via the SAS **infile** and **informat** statements but less so with an XML document. To import an XML document as a SAS dataset we use the XMLLE and a mapping file that describe the level of the structure and the attributes of the data. So in principal there is nothing different between the

two methods, just different in the implementation for the two structures. That is to read data you must provide the information on how to read it.

Example 2: Importing a CSV document

```
data hello_world;
%let _EFIERR_ = 0;
infile 'hello_world.csv' delimiter = ',' MISSOVER DSD lrecl=32767 firstobs=2;
   informat field1 $20.;
   informat field2 $20.;
   format   field1 $20.;
   format   field2 $20.;
input
  field1 $
  field2 $
  ;
  /* set ERROR detection macro variable */
  if _ERROR_ then call symputx('_EFIERR_',1);
run;

proc print;
run;
```

Example 3: Importing an XML document

```
/* hello_world.xml */
<hello_world>
  <Hworld>
    <Row>
      <Field1>Hello</Field1>
      <Field2>World</Field2>
    </Row>
    <Row>
      <Field1>I am an XML document</Field1>
      <Field2>also just plain text</Field2>
    </Row>
  </Hworld>
</hello_world>


/* hello_world.map */
<?xml version="1.0" ?>
<SXLEMAP version="1.2">
<TABLE name="Hworld">
<TABLE-PATH syntax="xpath">/hello_world/Hworld/Row</TABLE-PATH>
    <COLUMN name="Field1">
    <PATH>hello_world/Hworld/Row/Field1</PATH>
      <DATATYPE>String</DATATYPE>
      <TYPE>Character</TYPE>
      <LABEL>Variable Name</LABEL>
      <LENGTH>20</LENGTH>
    </COLUMN>
    <COLUMN name="Field2">
    <PATH>hello_world/Hworld/Row/Field2</PATH>
      <DATATYPE>String</DATATYPE>
      <TYPE>Character</TYPE>
      <LABEL>Variable Label</LABEL>
      <LENGTH>20</LENGTH>
    </COLUMN>
</TABLE>
</SXLEMAP>


filename xmldata 'hello_world.xml'; /* data file    */
filename xmlmap  'hello_world.map'; /* mapping file */

libname xmldata xml xmlmap=xmlmap;

data hello_world;
  set xmldata.hworld;
run;
proc print;
run;
```

So here are the results of the two printings:

Field1	Field2
Hello	World
I am a comma delimited	plain text document

Field1	Field2
Hello	World
I am an XML document	also just plain text

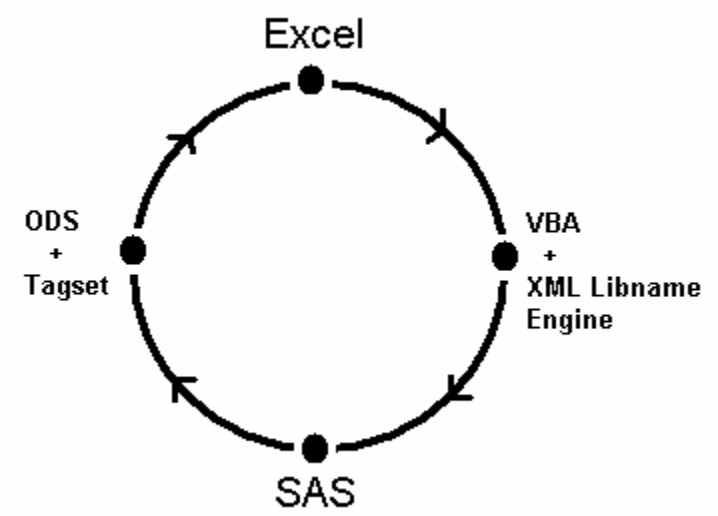
So what are the reasons why XML is more appealing? This outside the scope of this paper but you can search the net on this topic.

A NOVEL WAY TO IMPORT EXCEL INTO SAS

We have seen how to import an XML document into SAS dataset but what has this got to do with Excel? You may or may not realise but you can now save Excel file in XML format (see example in **Appendix 1**). So why not save the Excel file in XML then use the XMLLE as discussed above? You have guessed it - this isn't as easy as its sound because the Excel XML is in the form of an XML spreadsheet tag hierarchy which is why the SAS Institute (SI) has provided a macro **%xlxp2sas**[1] and an XMLmap file for this task. But this macro is free so what is the problem? Again, you have guessed it - there are limitations in **%xlxp2sas** which is probably why this macro has not gain popularity.

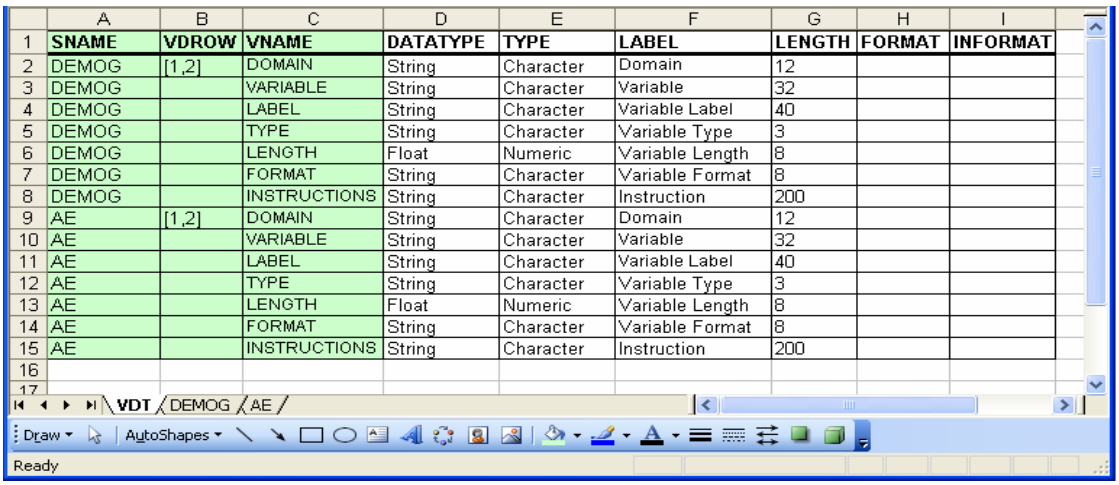
There is a **much** simpler, flexible and cheap solution in my opinion. We already have the XML tagsets for exporting to Excel, **what if we have something similar but in the reverse process**? That something in fact could be Visual Basic Application (VBA). There is no reason why we can't use VBA to write out the Excel data in XML (without the schemas) and its **.map** file. As far as importing an XML document into SAS concern - why would SAS want deal with the cosmetic stuff in the XML document? **Figure 1** shows a vision of exchanging data between SAS and Excel.

Figure 1



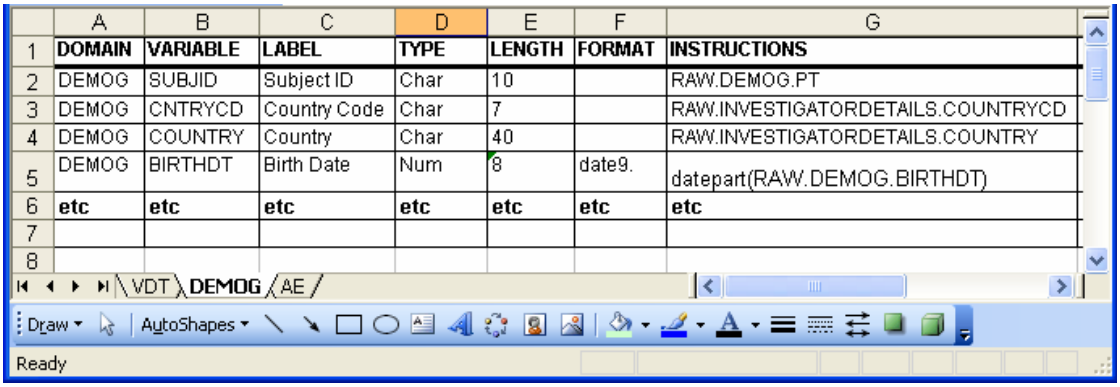
By the way the SI provides an XMLmap application to create the **.map** file but on this case one might as well design the information for the **.map** file on an additional sheet. This I believe will make the importing process more flexible and simpler. **Figure 2** shows what this might look like - an example of the structure of the Variable Definition Table sheet (VDT). This data will be written to the **.map** file and use to attributes variables when other sheets being read and **Figure 3** are the data that will be written to the **.xml** file.

Figure 2



	A	B	C	D	E	F	G	H	I
1	SNAME	VDROW	VNAME	DATATYPE	TYPE	LABEL	LENGTH	FORMAT	INFORMAT
2	DEMOG	[1,2]	DOMAIN	String	Character	Domain	12		
3	DEMOG		VARIABLE	String	Character	Variable	32		
4	DEMOG		LABEL	String	Character	Variable Label	40		
5	DEMOG		TYPE	String	Character	Variable Type	3		
6	DEMOG		LENGTH	Float	Numeric	Variable Length	8		
7	DEMOG		FORMAT	String	Character	Variable Format	8		
8	DEMOG		INSTRUCTIONS	String	Character	Instruction	200		
9	AE	[1,2]	DOMAIN	String	Character	Domain	12		
10	AE		VARIABLE	String	Character	Variable	32		
11	AE		LABEL	String	Character	Variable Label	40		
12	AE		TYPE	String	Character	Variable Type	3		
13	AE		LENGTH	Float	Numeric	Variable Length	8		
14	AE		FORMAT	String	Character	Variable Format	8		
15	AE		INSTRUCTIONS	String	Character	Instruction	200		
16									

Figure 3



	A	B	C	D	E	F	G
1	DOMAIN	VARIABLE	LABEL	TYPE	LENGTH	FORMAT	INSTRUCTIONS
2	DEMOG	SUBJID	Subject ID	Char	10		RAW.DEMOG.PT
3	DEMOG	CNTRYCD	Country Code	Char	7		RAW.INVESTIGATORDETAILS.COUNTRYCD
4	DEMOG	COUNTRY	Country	Char	40		RAW.INVESTIGATORDETAILS.COUNTRY
5	DEMOG	BIRTHDT	Birth Date	Num	8	date9.	datepart(RAW.DEMOG.BIRTHDT)
6	**etc**	**etc**	**etc**	**etc**	**etc**	**etc**	**etc**
7							
8							

Once we have the **.xml** and **.map** files then it is just a matter of feeding them into the **%xcel2sas** macro to convert the data into SAS dataset. **Note** by designed - this process of importing Excel data into SAS becomes very user friendly and flexible. Imagine you received some data in Excel – all you have to do is to add the VDT sheet and define what you want to read then just run the **%xcel2sas** macro. Note optionally you could have created the **.map** file by other means.

```
%macro xcel2sas (tabs    =       /* specify tables                    */
                ,odset   =       /* output dataset                    */
                ,xmldat  =       /* the xml data file                 */
                ,xmlmap  =       /* the map file for the xml data file */
                );
  :
  Extract of excel2sas

  libname xmlpath xml "&xmldat" xmlmap = "&xmlmap";

  /* read a table */
  %do n = 1 %to &ntabs;
    data %scan(&tabs, &n);
      length table $32;
      set xmlpath.%scan(&tabs, &n);
      table = "%scan(&tabs, &n)";
    run;
  %end;

  /* stack all the tables */
  data &odset;
```

```
    set %do n = 1 %to &ntabs;
           %scan(&tabs, &n)
        %end;
        ;
  run;
  :

%mend;

%xcel2sas (tabs    = demog ae
          ,odset  = all
          ,xmldat = ddt.xml
          ,xmlmap = vdt.map
          );
```

Note: if you have SAS/ACCESS licence then you could use Proc Import. The design would be much the same.

DISCUSSION

With the advent of the XMLLE to import Excel into SAS, other methods are **'less'** attractive in my opinion. As far as I am aware '**all**' issues highlighted in other published papers and documentations on this subject area are solved with this XMLLE method. Also a major drawback on some of these methods is data integrity on importing. The XMLLE method "should" not have this issue although its efficiency needs investigate further.

Future Excel version may have the option to save Excel file in XML format without the schemas (i.e. the cosmetic stuff in the XML document). In this case VBA might not need be involved.

CONCLUSION

This paper demonstrates the flexibility/adaptability of the XMLLE plus VBA method to import Excel data into SAS. The significant **advantages** of this method are that you **do not** need SAS/ACCESS license and it is **platform independent**.

The novelty introduction of a meta data sheet simplified the generalisation of the **%xcel2sas** macro and thus made it very user friendly.

REFERENCES

[1] Vincent DelGobbo. "Creating AND Importing Multi-Sheet Excel Workbooks the Easy Way with SAS®". SUGI 31, paper 115-31

[2] Helen Sun, Cindy Wong. "A Macro for Importing Multiple Excel Worksheets into SAS® Data Sets". SUGI 30, paper 040-30

[3] Qi, H. "A Practical Approach to Transferring Data from Microsoft Excel to SAS in Pharmaceutical Research". SUGI 29, paper 082-29

[4] Koen Vyverman. "Creating Custom Excel Workbooks from Base SAS® with Dynamic Data Exchange: A Complete Walkthrough". SUGI 27 paper 190-27

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Duong Tran
Phone: 07752326413
Email: trand000@aol.com
www.tranz.co.uk

APPENDIX

```
<?xml version="1.0"?>
<?mso-application progid="Excel.Sheet"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:o="urn:schemas-microsoft-com:office:office"
 xmlns:x="urn:schemas-microsoft-com:office:excel"
 xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:html="http://www.w3.org/TR/REC-html40">
 <DocumentProperties xmlns="urn:schemas-microsoft-com:office:office">
  <Author>DB Tran</Author>
  <LastAuthor>DB Tran</LastAuthor>
  <Created>2008-08-11T21:06:35Z</Created>
  <Company>Microsoft Corporation</Company>
  <Version>11.6360</Version>
 </DocumentProperties>
 <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
  <WindowHeight>9150</WindowHeight>
  <WindowWidth>14940</WindowWidth>
  <WindowTopX>360</WindowTopX>
  <WindowTopY>255</WindowTopY>
  <ProtectStructure>False</ProtectStructure>
  <ProtectWindows>False</ProtectWindows>
 </ExcelWorkbook>
 <Styles>
  <Style ss:ID="Default" ss:Name="Normal">
   <Alignment ss:Vertical="Bottom"/>
   <Borders/>
   <Font/>
   <Interior/>
   <NumberFormat/>
   <Protection/>
  </Style>
 </Styles>
 <Worksheet ss:Name="Sheet1">
  <Table ss:ExpandedColumnCount="2" ss:ExpandedRowCount="2" x:FullColumns="1"
   x:FullRows="1">
   <Column ss:Width="84"/>
   <Column ss:Width="64.5"/>
   <Row>
    <Cell><Data ss:Type="String">Hello</Data></Cell>
    <Cell><Data ss:Type="String">World</Data></Cell>
   </Row>
   <Row>
    <Cell><Data ss:Type="String">I am an Excel file </Data></Cell>
    <Cell><Data ss:Type="String">In XML format</Data></Cell>
   </Row>
  </Table>
  <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
   <Selected/>
   <Panes>
    <Pane>
     <Number>3</Number>
     <ActiveRow>5</ActiveRow>
     <ActiveCol>1</ActiveCol>
    </Pane>
   </Panes>
   <ProtectObjects>False</ProtectObjects>
   <ProtectScenarios>False</ProtectScenarios>
  </WorksheetOptions>
 </Worksheet>
  </Workbook>
```