Importing Data from RTF Output into SAS® utilizing Microsoft® Access/Word in an intriguing, efficient way

Ajay Gupta, MPI Research, Mattawan, MI
Matthew Lesko, MPI Research, Mattawan, MI

ABSTRACT

In the pharmaceutical/CRO industries, SAS® programmers often create summary tables and listings in Rich Text Format (RTF) using SAS® ODS. Because of strict quality control requirements, these Microsoft word readable RTF documents are typically visually compared with the output generated on the same specifications from a second programmer or statistician. When large number (or size) of files are involved, this method is time consuming and prone to human error. However, SAS does not provide a built-in procedure to import data directly from a Microsoft word (RTF formatted) document into SAS, however use of LIBNAME/ PROC IMPORT reads data from Microsoft Access.

Inspired by a published solution, this paper presents another creative way of converting word (.rtf) documents into sas datasets utilitizing Microsoft Access. The generated SAS datasets can then further be compared programmatically and by using PROC COMPARE. A SAS macro, %Convert_Doc2_SAS, will be introduced to do the converting process from RTF into SAS datasets. This convenient and reliable solution will help SAS programmers/statisticians to have better control over the quality of reports and save significant time by greatly reducing manual based quality control methods.

INTRODUCTION

The Rich Text Format is a document file format developed by Microsoft in 1987 for cross-platform document interchange. Most word processors such as Microsoft word are able to read and write RTF documents.

Microsoft Access is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. It is a member of the Microsoft Office suite of applications and is included in the Professional and higher versions for Windows and also sold separately.

In the pharmaceutical/CRO industry, tables and listings are often created in word (.rtf) documents using the Output Delivery System (ODS) destination. For quality control purposes, RTF tables are visually compared with the output generated on the same specifications from a second programmer or statistician. This method is time consuming and prone to human error. In order to replace human visual check methods programmatically, the SAS developer needs to convert the word (.rtf) documents into sas datasets. However, there is no existing SAS procedure which will directly import a RTF document into SAS.

To overcome this challenge, Jay Zhou (2009) introduced a method to convert Microsoft word document into sas datasets using Microsoft Excel via Dynamic Data Exchange (DDE).

This paper introduces another useful (and probably simpler) method to import data from any Word-readable documents, specifically RTF, into SAS. The entire process involves SAS, Word, and Access which is automated by a SAS macro called %Convert_Doc2_SAS. It will import the entire contents while reserving the document layout and structures, including tables with multiple columns, into SAS data. Since variable formats are predefined in the Microsoft Access table, this will avoid any formatting changes occur to the data during the conversion process.

TECHNIQUE & MECHANISM

The general process to convert a RTF document into a SAS dataset is as follows (note that the first two items are a one-time setup, and the rest can be managed via SAS code):

1. Create a new master Microsoft Access Database file (which will act like a template) and define a table having required variables and formats.
2. Create a Auto execution macro in the Master Access Database template which will have the following functions:
    a) Select table
    b) Set all warnings to yes.
    c) Select all records from the table.
    d) Paste the records.
3. Copy the master Microsoft Access Database template to the production location.
4. Open the target file in Word and copy the content.

5. Open the Microsoft Access Database from production location. Due to the auto execution of the Access macro, the Access table will have all the records inserted automatically into it from the clipboard buffer.
6. Read the content into SAS using LIBNAME or PROC IMPORT procedure.
7. Delete the Microsoft Access Database from the production location.

To automate these steps, the DDE solution is leveraged to build the communication bridge between SAS and Word or Access. WordBasic commands (Microsoft Corporation, 1999) can be sent from SAS via DDE to enable SAS to take control of Word and Access.

CREATE MASTER ACCESS DATABASE AND DEFINE A TABLE:

Using the graphical user interface provided by Microsoft for Access, programmers can easily create a new database (e.g. db1) and define the table (e.g. Table1). The table attribute can be defined in the design view per your requirements. To help avoid unwanted formatting changes, all of the variables are defined as character. For more information on using the Access, please refer to the help tab in Access. As there is no coding involved and in order to have a better understanding of the Microsoft Access Database, the whole process is described using screenshots.

Below is the screenshot of the Master Access Database and table defined as Table1:

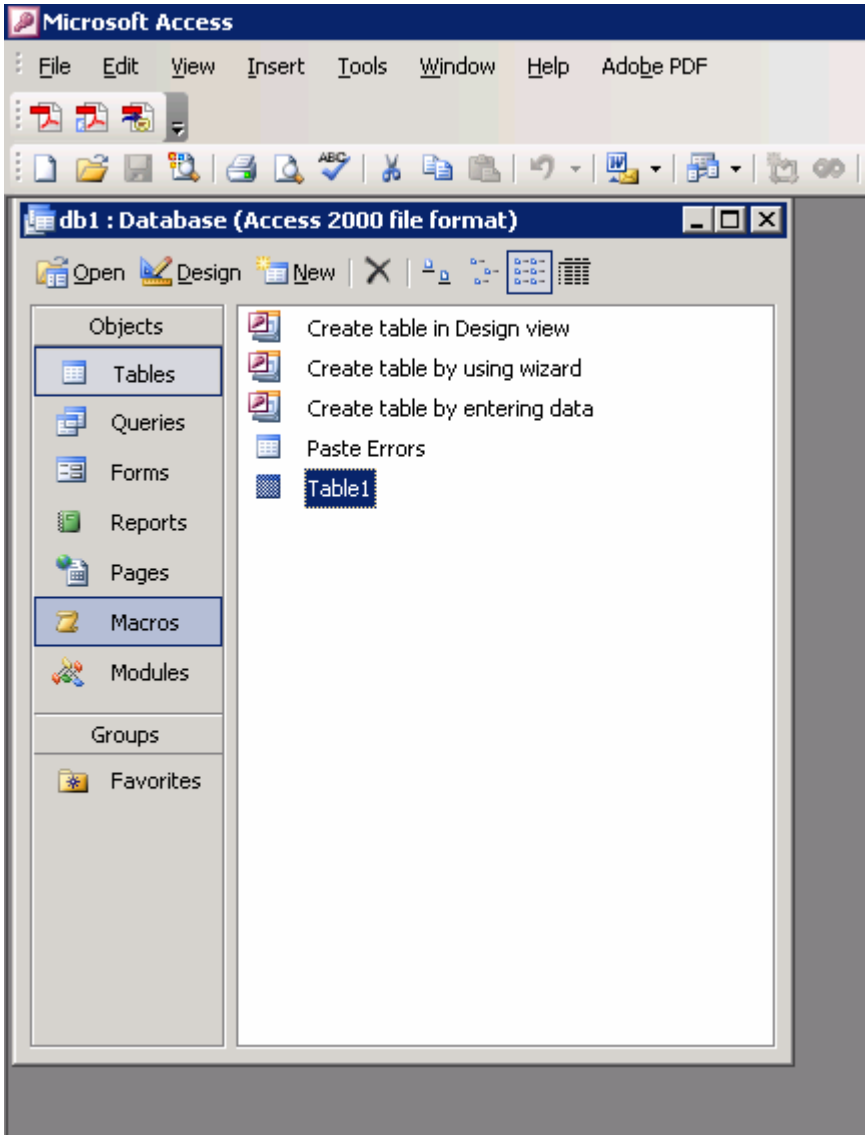

Figure 1. Master Access Database

Below is the screen shot of table1 in design view:

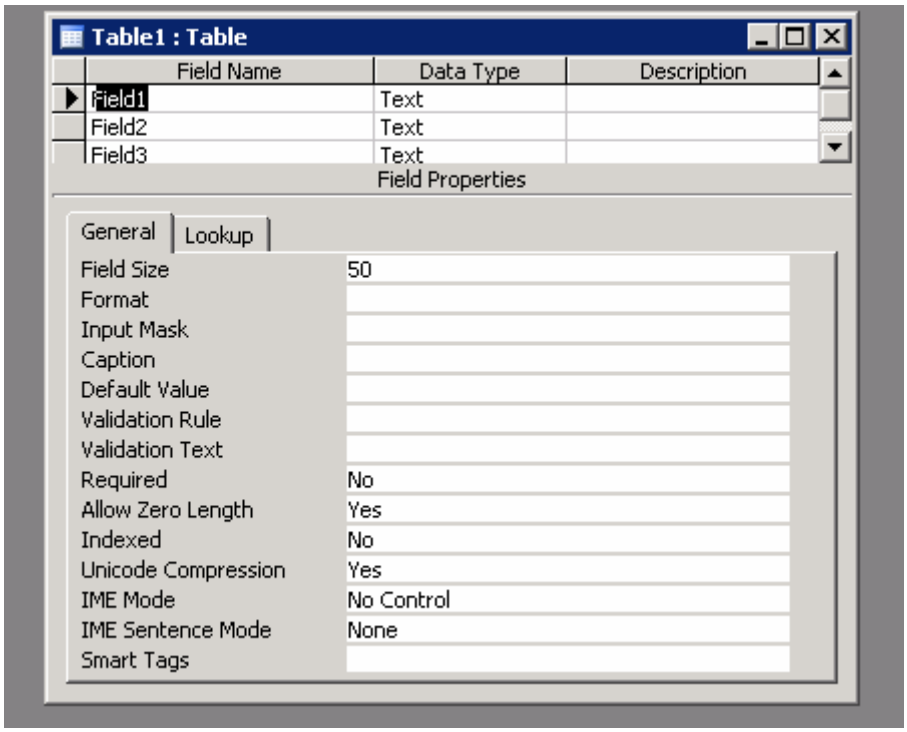

Figure 2. Design view of Table1

AUTO EXECUTION MACRO IN MASTER ACCESS DATABASE:
Auto execution macro will be executed at the startup of the Access Database. The new macro is created using the macros tab in the Access Database. Further, different commands are added using the design view.

Below is the screenshot of the Autoexec macro:

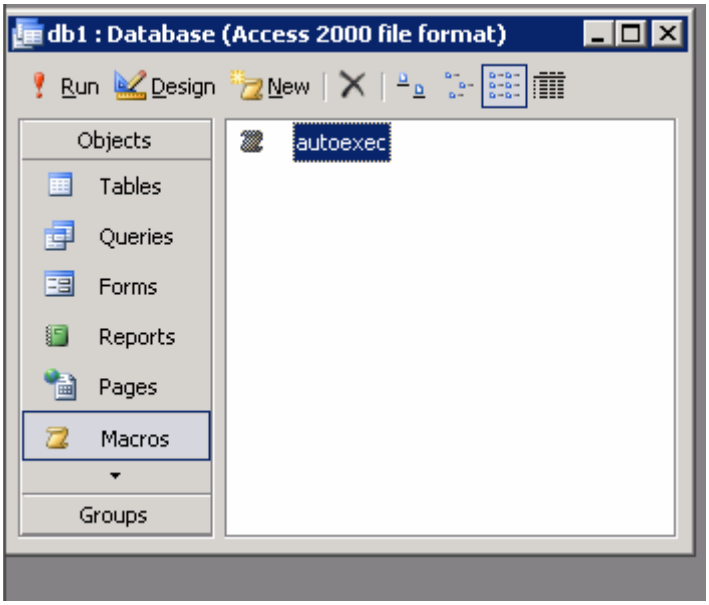

Figure 3. Autoexec macro in Master Access Database.

Below is the screenshot of Autoexec macro in the design view:

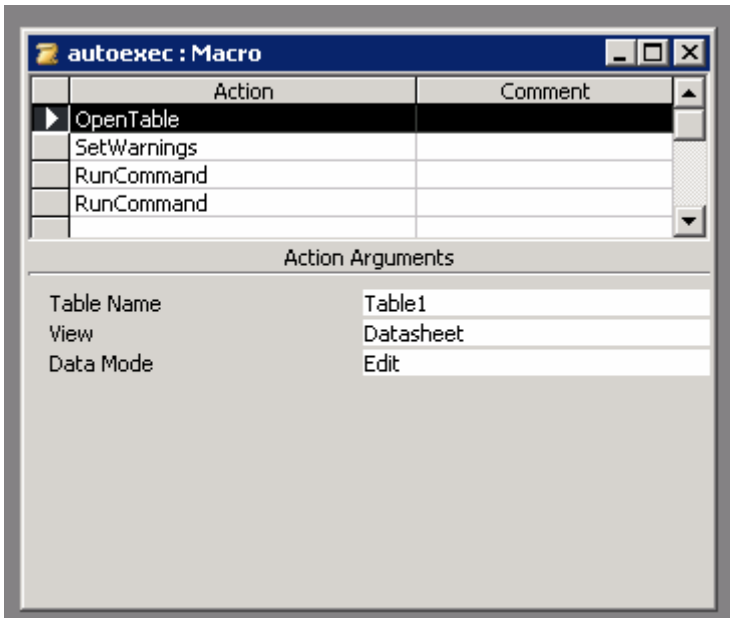

Figure 4. Design view of Autoexec macro

In the Autoexec macro the following values are assigned to the different series of actions:

1. For OpenTable insert 'Table1' in the Table Name tab.
2. For SetWarnings insert 'NO' in the Warnings On tab.
3. For the first RunCommand insert 'SelectAllRecords' in the Command tab.
4. For the second RunCommand insert 'Paste' in the Command tab.

COPY THE MASTER ACCESS DATABASE TEMPLATE TO THE PRODUCTION LOCATION

Prior to executing these statements, there are two system options needed: NOXWAIT and NOXSYNC. The NOXWAIT option specifies that the DOS command prompt window disappears without you having to type EXIT when the process is finished, while the NOXSYNC specifies that the process should execute asynchronously. That is, control is returned immediately to the SAS System and the command continues executing without interfering with your SAS session.

Since the Access table will have new data, due to execution of the autoexec macro in the Access database, we will always need a fresh copy of the Master Access Database Template.

The following command is used to copy the Master Access Database Template to the production location:

```
X copy "&location\Master\*.mdb" "&location";
```

Note, "&location" is your production location.

OPEN THE TARGET FILE IN WORD AND COPY THE CONTENT

In order for a client/server communication link to be established, both SAS and Word must be running. Therefore, for the first iteration, it is necessary to programmatically launch Word from a SAS session. There are several techniques available to launch Word from SAS. The simplest one is the following statement:

```
%let rc=%sysfunc(system(start winword));
```

The above command is dependent on the finishing of previous commands. Note, the SLEEP function can be used frequently in the SAS command/datastep that is dependent upon previous jobs finishing. This will help avoid errors from occurring due to delays in the execution of previous SAS commands/datasteps.

The syntax for the SLEEP function is given below:

```
data _null_;
   x=sleep(5);
run;
```

The above step will pause the SAS session for five seconds.

To communicate with Word from SAS, the following `FILENAME` statement is used to establish the linkage between SAS and Word via the DDE triplet:

```
filename word DDE 'Winword|System';
```

Another technique (Roper 2000) to launch Word from SAS is given below:

```
%MACRO OpenWord;

      Filename word DDE 'Winword|System';

      Data _NULL_;
            Length fid rc start stop time 8;
            fid = FOPEN('word','s');
            If (fid le 0) then do;
                  rc = system('start winword');
                  start = datetime();
                  stop = start + 10;
                  Do while (fid le 0);
                        fid = FOPEN( 'word', 's' );
                        time = DATETIME();
                        If (time ge stop) then fid = 1;
                  End;
            End;
            Rc=FCLOSE(fid);
      Run;

%MEND OpenWord;
%OpenWord;
```

Macro `%Openword` will start an instance of Word if none is running yet, and will keep trying for at most ten seconds to establish a DDE conversation via the system-triplet word. The filename statement will remain same as given above.

The next step is to open the target file “&in” by sending the `FileOpen` WordBasic command to Word with a `data _null_` step. Once the file is open in Word and the cursor makes it to the end of the document, select and copy the entire document by sending the `EditSelectAll` and `EditCopy commands as` shown in the following:

```
data _null_;
   file word;
      put '[FileOpen.Name = "' "&in" '"]';
      put "[EndOfDocument]";
      put "[EditSelectAll]";
      put "[EditCopy]";
      put '[FileClose]';
      put '[FileExit]';
run;
```

It is necessary to programmatically close the document with the `FileClose` command. Further, the FileExit command will exit Microsoft Word.

OPEN THE MICROSOFT ACCESS DATABASE FROM PRODUCTION LOCATION

The same approach as invoking Word is used to start a new Access session:

```
%let rc=%sysfunc(system(start msaccess &location\db1.mdb));
```

The above command will open the Master Access Database “db1.mdb” at production location.

Due to the Auto execution of macro in db1.mdb, the following events will happen:

- Table1 will be selected in edit mode

- All of the Access warnings will be set to 'NO', which has the same effect as pressing 'ENTER' whenever a warning or message box is displayed.
- All of the records in table1 are selected using the 'SelectAllRecords' command.
- All of the records from the word document are pasted in table1 using the 'Paste' command.

To communicate with Access from SAS, the following `FILENAME` statement is used to establish the linkage between SAS and Access via the DDE triplet:

```
filename access DDE 'Msaccess|System';
```

Similarly, the %OpenWord macro given above can be easily modified to launch Access from SAS.

The following command will save the data in Table1, and then close and exit the Access session.

```
data _null_;
   file access;
   put '[Save]';
   put '[Close]';
   put '[Quit]';
run;
```

After execution of the above datastep, the data will be stored in Table1.

READ THE CONTENT INTO SAS USING LIBNAME OR PROC IMPORT PROCEDUE

After the text or data is saved into an Access table, it is time to read it into SAS by either using the `LIBNAME` statement or `PROC IMPORT`. If the SAS/ACCESS license is available, the following libname can be used:

```
libname test access "&location\db1.mdb";
libname dat "&location";

data dat.&out.;
   set test.table1;
run;
```

In the above datastep, "&location" is the production location and "&out" is the output dataset created and saved in library "dat". This dataset can be further manipulated in SAS and compared with the dataset created by other programmers/statisticians using `PROC COMPARE`.

DELETE THE MICROSOFT ACCESS DATABASE FROM PRODUCTION LOCATION

To avoid any overwriting of the data, Access database file "db1.mdb" can be deleted as follows:

```
filename db "&location\db1.mdb";
%let rc=%sysfunc(fdelete(db));
filename db clear;
```

"&location" is the production location.

%Convert_Doc2_SAS

To facilitate and automate the above discussed steps from opening the Word file to reading the Access file into SAS, a SAS macro called %Convert_Doc2_SAS was developed for SAS v9.2 or above (see Appendix for details). The user can easily extend the macro to fit other SAS versions.

There are only three keyword parameters:

In: Define the path and the name of the input Word file, e.g., `C:\Demo\test.rtf.`
Out: Define the SAS dataset name, for e.g., raw.
Location: Production Location, for e.g., `C:\Demo.`

Below is the simple macro call to %Convert_Doc2_SAS.

```
%Convert_Doc2_SAS (in= C:\Demo\test.rtf, out=raw, location=C:\Demo)
```

CONCLUSION

%Convert_Doc2_SAS can convert the word readable RTF documents into SAS datasets without any manual pre-process. Since variable formats are predefined in the Microsoft Access table, this will avoid any formatting changes from occurring to the data during the conversion process. Hence, the integrity of keeping the original content (including special characters) and table structures is guaranteed. The only drawback of this method is that the process is partially based on the DDE technology which may not be supported by Microsoft Corporation in future MS Office releases.

REFERENCES

Zhou, Jay. 2009. Importing Data from Microsoft Word into SAS. Proceedings of the PharmaSUG 2009 Conference, Paper CC18.

Hagendoorn, Michael, Jonathan Squire, and Johnny Tai. 2006. Save Those Eyes: A Quality-Control Utility for Checking RTF Output Immediately and Accurately. Proceedings of the Thirty-first SAS Users Group International Conference, paper 066-31.

Lu, Zaizai, David Shen. 2005. Data Transfer from Microsoft Access to SAS Made Easy. Proceedings of the PharmaSUG 2005 Conference, Paper CC12.

Microsoft Corporation. 2000. Word 95 WordBasic Help File.
http://www.microsoft.com/downloads/details.aspx?familyid=1a24b2a7-31ae-4b7c-a377-45a8e2c70ab2&displaylang=en

Office 2003 Editions: Access VBA Language Reference.
http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=0447c5a0-5e58-4e69-b90e-c42ec7dbf887

Roper, C. A. 2000. Intelligently Launching Microsoft Excel from SAS, using SCL functions ported to Base SAS. Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference, paper 97.

Viergever William, Koen Vyverman. 2003. Fancy MS Word Reports Made Easy: Harnessing the Power of Dynamic Data Exchange. Proceedings of the Twenty-Eight Annual SAS Users Group International Conference, paper 16-28.

CONTACT INFORMATION

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

Ajay Gupta, M.S.
MPI Research Inc.
54943 North Main Street
Mattawan, MI 49071-9399
Phone: (269) 668-3336 Ext 1823
Email: Ajaykailasgupta@aol.com

Matthew Lesko, M.S.
MPI Research Inc.
54943 North Main Street
Mattawan, MI 49071-9399
Phone: (269) 668-3336 Ext 1737
Email: Matthew.Lesko@mpiresearch.com

APPENDIX

```
%macro Convert_Doc2_SAS(in=,out=,location=);
       options noxwait noxsync;
       X copy "&location\Master\*.mdb" "&location";
       %let rc=%sysfunc(system(start winword));

       data _null_;
              x=sleep(5);
       run;

       filename word DDE 'Winword|System';

       data _null_;
       file word;
              put '[FileOpen.Name = "' "&in" '"]';
              put "[EndOfDocument]";
              put "[EditSelectAll]";
              put "[EditCopy]";
              put '[FileClose]';
              put '[FileExit]';
       run;

       data _null_;
              x=sleep(2);
       run;

       %let rc=%sysfunc(system(start msaccess &location\db1.mdb));

       data _null_;
              x=sleep(5);
       run;

       filename access DDE 'Msaccess|System';

       data _null_;
              x=sleep(2);
       run;

       data _null_;
              file access;
              put '[Save]';
              put '[Close]';
              put '[Quit]';
       run;

       data _null_;
              x=sleep(5);
       run;

       libname test access "&location\db1.mdb";
       libname dat "&location";
       data dat.&out.;
              set test.table1;
       run;

       libname test clear;
       data _null_;
              x=sleep(2);
       run;

       filename db "&location\db1.mdb";
       %let rc=%sysfunc(fdelete(db));
       filename db clear;

%mend Convert_Doc2_SAS;
```