

"The Microsoft® Word and SAS® System for Windows® Connection "

Orion V. Hanger, Nielsen Media Research

ABSTRACT

When using the SAS System for Windows, it is sometimes desirable to write SAS programs that interact with other Windows applications via DDE (Dynamic Data Exchange). This paper provides a concise discussion of the basics for making the connection between SAS and Word, including how to invoke Word, read from Word, write to Word and send WordBasic commands to Word via a SAS routine. With these capabilities, SAS code can be written that automates a Microsoft Word word processing task (such as generating form letters containing data primarily managed within the SAS environment).

Note: This paper and it's examples were developed using SAS release 6.08 and Microsoft Word 6.0 (the rules are a little different with prior releases of Word).

INVOKING MS-WORD

SAS code for starting another Windows application is very simple:

```
options noxwait noxsync;
x 'c:\winword\winword.exe' ;
```

The 'X' statement is common across all implementations of the SAS System and is used to issue operating system commands (such as invoking other applications). In this case, the Microsoft Word for Windows application is being started. Depending on how Word has been installed, the command path for this may be different on different platforms. The 'noxwait' and 'noxsync' option values are used so that the SAS System will continue to execute as the active application when Word is invoked.

READING FROM MS-WORD

Understanding 'reading from and writing to' Word from a SAS program requires the understanding and creation of 'bookmarks' in the Word document with which SAS is going to interact. In Word, a 'bookmark' is defined to identify a specific item or location in a document. Multiple bookmarks can be defined and any that are defined are saved with the document. To create a bookmark in Word, select the desired item or move the cursor to the

desired location, select EDIT and BOOKMARK, provide a name for the bookmark and select ADD. Once defined, a bookmark item/location can be selected/moved to via the EDIT menu (either GOTO or BOOKMARK). A bookmark will exist until it is deleted via EDIT and BOOKMARK (items within the bookmark will still exist) or until everything within the bookmark is deleted.

To read a bookmarked section in a Word document with a SAS routine, the bookmark name must be referenced in a FILENAME statement with DDE as the device type:

```
filename in1 dde
'winword|c:\sasforum\worddoc2.doc!test1';
data test1; infile in1 missover;
    length cvar1 $ 3 cvar2 $ 4 cvar3 $ 1 nvar1 8;
    input cvar1 cvar2 cvar3 nvar1;
run;
filename in1 clear;
```

In the above example, a bookmark named 'test1' in the 'c:\sasforum\worddoc2.doc' document that is currently open in the Microsoft Word application ('winword') is read by the SAS program and the created variables are output to a SAS dataset. In this case the bookmark area was a small Word table containing 4 columns/variables (3 character and one numeric). The type and length of the variables are predefined with a LENGTH statement and list input is used. By default, SAS assumes the logical record length to be 132. This will have to be explicitly defined if it is longer than the default.

WRITING TO MS-WORD

A bookmark must also be defined in Word to indicate an insertion point (where SAS will start inserting text/data) and this bookmark referenced in a FILENAME statement for SAS to write to a Word document:

```
filename in2 dde
'winword|c:\sasforum\worddoc2.doc!test2';
data _null_; set test1; file in2;
    put cvar1 cvar2 cvar3 nvar1;
run;
filename in2 clear;
```

In this example, 4 variables in dataset 'test1' are written to the Word document starting at the 'test2' bookmark. The value of each variable starts at the next tab mark in the document (this is not true if the 'notab' option is used on the FILENAME statement) and there are as many lines inserted into the document as there are observations in the input SAS dataset. Again, SAS assumes a logical record length of 132 if none is specified. When writing via DDE, the output is normally collected in the DDE 'buffer' and then transferred at the end of the data step. The text string 'DDE_FLUSH' can be sent to cause an immediate transfer of the contents of the DDE buffer.

SENDING COMMANDS TO MS-WORD

In addition to reading from and writing to a Word document, the SAS System can also manipulate/control the Word application by sending it WordBasic commands. In a nutshell, WordBasic is the Microsoft Word macro programming language. It is documented in the online help for Word and a separate manual can be obtained from Microsoft (or similar types of texts at computer book stores). The basic syntax of a WordBasic command includes a starting keyword and one or more parameters (optional or required, depending on the command).

As with reading and writing, a DDE link to Word must be established for this communication to occur. But instead of creating a fileref to a Word bookmark, a fileref is created to the Word 'system':

```
filename wrdcmds dde 'winword|system';
data _null_ file wrdcmds;
  put '[FileOpen .Name="c:\sasforum\fern2.doc"]';
run;
filename wrdcmds clear;
```

In the above example, the Word application command level is assigned the fileref 'wrdcmds'. The subsequent DATA _NULL_ step uses this fileref in a FILE statement and sends a command to Word (via PUT) to open the "c:\sasforum\fern2.doc" document. The command string (note that it is a quoted text string on the PUT statement) that is sent to Word must be within brackets and may contain its own quoted text (such as the value for the file name parameter on the command used in this example).

A FINAL EXAMPLE

The following code is an example of how the above capabilities can be used to print a form letter in Word for every observation in an 'address list' dataset:

```
options noxwait noxsync;
libname sasforum 'c:\sasforum';
```

```
filename wrdcmds dde 'winword|system';
x 'c:\winword\winword.exe';
data _null_ file wrdcmds;
  put '[FileOpen .Name="c:\tstform5.doc"]';
  stop;
run;
filename outname dde
  'winword|c:\tstform5.doc!name' notab;
filename outaddr dde
  'winword|c:\tstform5.doc!address' notab;
filename outctstz dde
  'winword|c:\tstform5.doc!ctystzip' notab;
filename outdate dde
  'winword|c:\tstform5.doc!date' notab;
data _null_ set sasforum.addrlist;
  date=today();
  file outname; put name '!DDE_FLUSH';
  file outaddr; put address '!DDE_FLUSH';
  file outctstz; put ctystzip '!DDE_FLUSH';
  file outdate; put date mdy8. '!DDE_FLUSH';
  file wrdcmds; put '[FilePrintDefault]';
run;
data _null_ file wrdcmds;
  put '[AppClose]'; stop;
run;
filename wrdcmds clear;
filename outname clear;
filename outaddr clear;
filename outctstz clear;
filename outdate clear;
```

In this example, the 'tstform5' document is a form letter with bookmarks defined for name, address, city/state/zip and date. The Word application is automatically started, values are written to the document dynamically (note the 'IDDE_FLUSH' string), the updated document is printed for each observation and, then, Word is closed.

CONCLUSION

In the Microsoft Windows environment, Dynamic Data Exchange (DDE) can be used to create interactivity between Windows applications. This paper presented the very basics of how to perform this interactivity with Microsoft Word from within a SAS program. Functions covered included starting Word, reading from Word, writing to Word and sending Word commands. With these 'tools', SAS programs can be created to automate Microsoft Word tasks such as generating form letters.

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Windows and Microsoft are registered trademarks of the Microsoft Corporation in the USA and other countries.