Paper D09-2009

Mail Merge using SAS

Michael Stout, DePuy Orthopaedics, Inc, a Johnson & Johnson Company
Warsaw, IN

ABSTRACT

Have you ever wanted to add a formatted letter to the front of your SAS output? Using the SAS® ODS® facility is an excellent method for combining letters with SAS® output. Creating letters using a word document and then collating them for small to medium mailings can be tedious and time consuming. SAS® ODS®, PROC SQL®, PROC IMPORT®, inline styles and Microsoft Excel® come to the rescue and make it fairly easy to create standardized letters on the fly. This paper shows one method that may be used to dynamically generate letters while saving time and improving the quality and presentation of the reports being prepared for distribution.

BACKGROUND

The contracts specialist needed a way to streamline the process of making compensation payments to surgeons. During some initial meetings we discovered that cover letters were manually created for each payment. This manual process was time consuming and letters were not always formatted the same way. Data Management was then tasked with improving the process. This led to the development of what I call **MAC(ro) Letter**.

MACRO

The `%Print_Certification_Letter` macro creates a standardized letter that can be bundled with a set of SAS tables, listings and charts.

SYNTAX: `%Print_Certification_Letter(OwnRTF=Y, inv_num=);`

This procedure has two named parameters: OwnRTF and INV_NUM. OwnRTF instructs the program to create a new RTF output destination. The default value for OwnRTF is 'Y' which opens a new RTF output destination. A value of 'N' instructs the program to use a RTF output destination that is already open. The OwnRTF parameter must be set to 'N' if you want to combine a letter with other ODS output.

Inv_Num is used to subset the data. The investigator number is the key required to read data used within the body of the letter. A generic subset parameter (i.e. &where or &subset) could be used if you prefer.

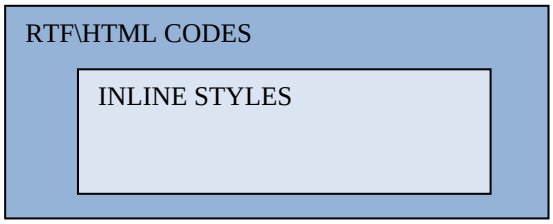
This macro inserts text and attributes into a large text string using inline styles\formats and data stored in macro variables to produce standardized letters that may be attached to other ODS output.

INLINE STYLES

Inline Styles are special codes used to format text by changing the style attributes for text following the inline style. These codes are used to assign attributes such as font color, size and style. The ODS escape character is used to instruct SAS to start processing style attributes. The escape sequence for v9.2 is '(*ESC*)', which can be overridden on the ODS options statement. The default escape character for v8 is '03'x. SAS recommends using a @ or ^. Changing the escape character, makes it easier to use inline formatting functions, since the code is easier to read. The carot "^" is used as the escape character in this paper.

```
ods escapechar '^';     /* Over-ride escape character to carot "^"   */
```

Inline formatting may be used to justify, underline, over-line, strikethrough and change font size and color for titles, footnotes and text. Carriage returns, subscripts, superscripts, page numbers and special characters can be inserted into text strings. In addition, RTF and HTML code can be inserted into text strings to format the text. All of the options and more are available to enhance ODS output.



Below is a list of sample in-line codes and RTF\HTML codes that can be used to enhance the appearance of the letter.

INLINE FUNCTIONS	V9	V8
New Line	^n or ^{newline n}	^n
Superscript	^{super x}	Same as v9
Subscript	^{sub x}	Not available in v8
Style	^{style [color=blue] <text> }	^S={foreground=blue} <text> ^S={}
Unicode	^{Unicode nnnn}	Not available in v8
Page X of Y (RTF only)	^{page of}	Same as v9
This Page (RTF & HTML)	^{this page}	Same as v9
Last Page (RTF & HTML)	^{last page}	Same as v9
RTF \HTML CODES	V9	V8
Bold	^\b <txt> ^\b0	Same as v9
Italic	^\i <txt> ^\i0	Same as v9
Underline	^\ul <txt> ^\ul0	Same as v9
Strikethrough	^\strike <txt> ^\strike0	Same as v9
Subscript	^\sub <txt> ^\sub0	Same as v9
Superscript	^\super <txt> ^\super0	Same as v9

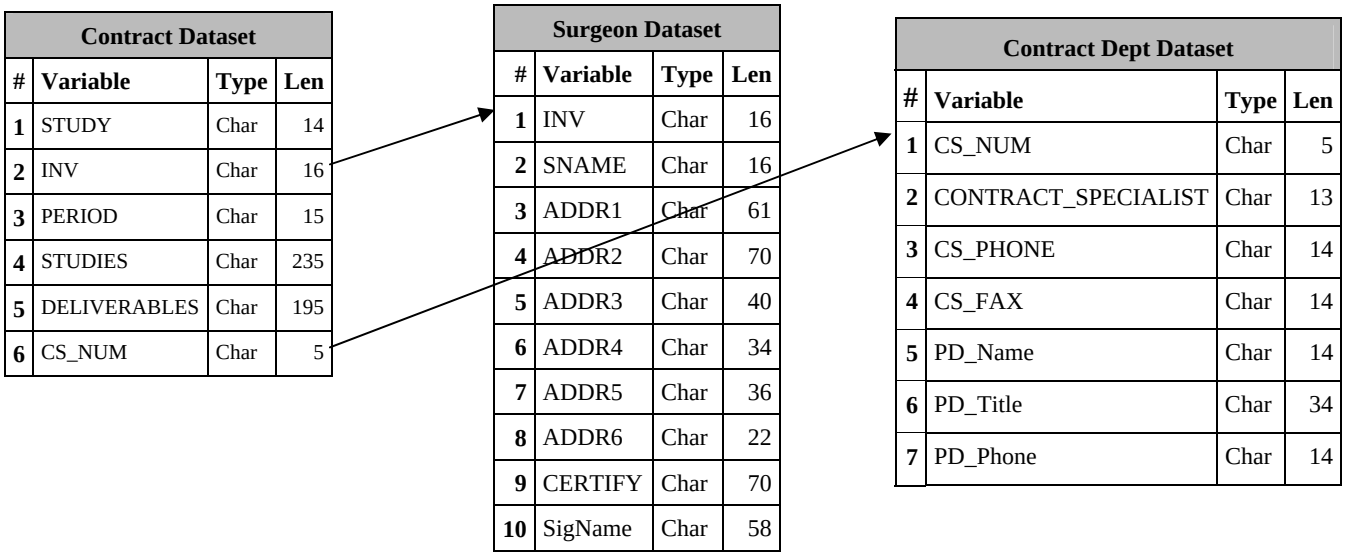
DATASETS

An Excel® file is used to store data needed to create the letters. To facilitate data entry and reduce maintenance, data has been normalized. This eliminates the need to store the same data values multiple times. For the Mail Merge application the data is actually stored on three tables: Contract, Surgeon and Contract Department (see Figure 1). These tables are read and then converted into SAS datasets.

Contract data is stored on the first sheet and contains information specific to each contract. This dataset contains one record for each surgeon\site and study. A surgeon may participate in multiple studies. In this situation, the same surgeon would have a different record for each study. The second table contains information specific to the surgeon such as name and address information. The investigator number is used to link the contract and surgeon data. Data specific to the contracts department are stored on the last sheet and contains the contract specialists name, phone number and fax. The contract specialist number is used to link the contract and contract department data.

Text to be inserted into the body of the letter can come from any valid SAS input destination. For convenience we chose to use an Excel spreadsheet to store the data needed to produce the letters.

Figure 1



Contract Dataset

#	Variable	Type	Len
1	STUDY	Char	14
2	INV	Char	16
3	PERIOD	Char	15
4	STUDIES	Char	235
5	DELIVERABLES	Char	195
6	CS_NUM	Char	5

Surgeon Dataset

#	Variable	Type	Len
1	INV	Char	16
2	SNAME	Char	16
3	ADDR1	Char	61
4	ADDR2	Char	70
5	ADDR3	Char	40
6	ADDR4	Char	34
7	ADDR5	Char	36
8	ADDR6	Char	22
9	CERTIFY	Char	70
10	SigName	Char	58

Contract Dept Dataset

#	Variable	Type	Len
1	CS_NUM	Char	5
2	CONTRACT_SPECIALIST	Char	13
3	CS_PHONE	Char	14
4	CS_FAX	Char	14
5	PD_Name	Char	14
6	PD_Title	Char	34
7	PD_Phone	Char	14

Your data needs may be simple and only require one table or more complex and need to read data from many tables. How the data is stored is not important. The data could be stored in SAS datasets, flat files, comma delimited files or any format that SAS can read. Your decision may be based on specific requirements such as time, skills, maintenance, audit trail of changes, and people resources. A good rule of thumb is to keep it simple.

For the mail merge application, a decision was made to use Excel to enter and maintain the data. Since the amount of data is relatively small there was little need for physically linking the files and keeping track of changes. If this is a requirement for your company, you might consider using a database to collect and store this information.

MACRO VARIABLES

For this application, PROC IMPORT® is used to read data values from the Excel and PROC SQL® is then used to populate macro variables. The macro variables are the key to understanding how values get inserted into the letter. This involves a two step process: reading the values and assigning the data to macro variables. The following piece of code shows how to read data from a spreadsheet using PROC IMPORT®. The import procedure reads the address data from the certification letters Excel spreadsheet specified on the DATAFILE option and creates a SAS dataset named StudyEvals. The GETNAMES= parameter instructs SAS to generate SAS variable names from the first row in the dataset. RANGE is used to determine which cells of data to import. For this application, the entire sheet of data is read.

```
/* READ ADDRESS DATA AND LOAD INTO SAS MACRO VARIABLES */
PROC IMPORT OUT=StudyEvals
DATAFILE="L:\Warsaw\Orthopaedics\Clinical_Affairs\Biostats_and_DM_General\SAS\SystemTables\Certification_Letters.xl
s" DBMS=EXCEL2000 REPLACE;
 RANGE="Address$";
 GETNAMES=YES;
 RUN;
```

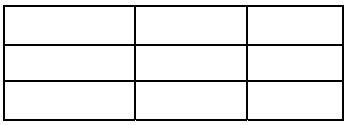
PROC SQL® is then used to assign address data to macro variables. The INTO option makes it easy to map the data values to macro variables. This is done by placing a colon ":" before the variable name. This tells SAS to load the value into a macro variable. These values can then be referenced in SAS programs by proceeding the variable with an ampersand "&". In this example the value of SNAME is assigned to macro variable &SNAME, SigName to &SigName, Addr1 to &Addr1 and so on. Make sure the number of variables on the SELECT statement match the variables listed on the INTO option. The WHERE clause is used to subset the data and read one record from the Address file. The investigator number is passed to the macro as a named parameter.

```
proc sql noprint;
     select  SNAME, SigName, Addr1, Addr2, Addr3, Addr4, Addr5, Addr6, Certify
       into :SName, :SigName, :Addr1, :Addr2, :Addr3, :Addr4, :Addr5, :Addr6, :Certify
       from StudyEvals
  where inv="&inv_num";
quit;
run;
```

The same process is used to create macro variables from the Contract and Contract Department tables.

ODS WITH A ONE CELL TABLE

If you have ever worked with the ODS RTF output destination, you will notice it generates tables to display data. The size of each cell is based on the length of the data displayed. Most reports have many rows and columns of data. For this application, only ONE cell is needed. In fact, the cell is so big that it covers the entire page of output. The length of the text field that will contain the letter is 6,000 bytes long. The length of the fields may vary based on the size of letter being produced.



Many Rows and Columns

1 Row and 1 Column

USING THE MACRO: TEMPLATE (LETTER)

By now, you have a basic understanding of inline codes and macro variables. You now have everything you need to create your own letter. I will now show you how to create a letter in RTF format. Here is the basic format of a template for producing a letter.

```
data CertLetter;
   length txt $6000.;
   txt =" <body of letter>";
  output;
  <add more text with implicit output if needed>
run;
```

Use a word processor to create a letter. You can use the word processor to lay out the letter and the use spell checker before creating the Letter template. You may want to insert place holders in the body of the letter where data is to be inserted such as address lines. I have had end-users print a letter and then highlight values to be replaced. Since most letters are small, I generally type them into the template myself. If an electronic version is available, text can be cut and pasted into the template.

ADDING PRINT ATTRIBUTES

Once the basic text is entered into the template, begin inserting &macro variables and carriage returns (^n) into the text. The RTF output destination can wrap text that does not fit into a cell by using the CELLWIDTH option on a define statement of PROC REPORT. Use the carriage return to force text to appear on a new line or to add blank lines to the body of the letter. The %BQUOTE and %TRIM functions are used to prevent the value &Sname from being evaluated by the macro facility and to remove trailing spaces from the string. Please note that all of the macro variable references will get resolved by the macro facility and the text strings will be inserted in the body of the letter. In the following code, a formatted date, address and surgeon name will be inserted into the letter.

```
txt =
"&fDATE
^n
^n
^n&Addr1
^n&Addr2
^n&Addr3
^n&Addr4
^n&Addr5
^n&Addr6
^n
^nDear %trim(%BQUOTE(&Sname)):
^n
^nThank you for your participation in DePuy Orthopaedics Clinical Research Activities. ";
```

Create a SAS dataset that contains one (or more) records and one field named TXT. The TXT field holds the entire letter in character format. It must be big enough to hold all of the text, inline formats and RTF code. Macro functions are used to format values stored in the macro variables. This is necessary, because the dataset may contain special characters or have spaces at the end of the string that need trimmed.

TAB CHARACTERS

This Mail merge application was written using SAS 8.2. The coding could be simplified by using UNICODES to insert tab characters. This would eliminate the need to concatenate strings together since the UNICODES can be embedded with the text string (UNICODE 0009) is the tab character. Tab characters are used to line up values on the letter.

```
"^nPeriod:       " ||  "09"x ||"&Period                 ====>        "^nPeriod: ^{Unicode 0009}&Period
^nStudy(s):     " || "09"x ||"%left(&Studies)" ||       ====>        ^nStudy(s): ^{Unicode 0009}%left(&Studies)
"^nDeliverables:" || "09"x ||"&Deliverables            ====>        ^nDeliverables: ^{Unicode 0009}&Deliverables"
```

STYLES

Take advantage of changing the fonts used in the letter. It is easy to change the font attributes using a styles function. The following code changes the font size to 3.5 and makes the text bold.

"^S={font_weight=bold font_size=3.5} Investigator Certification Statement ^S={}

Macro variables can also be used to store style attributes. This is helpful when the specific attributes are used multiple times. It serves as a shortcut and makes the coding easier to read and maintain. The following code is used to set the color of the text to RED and display characters 'XX'. Any value in red indicates that the contract specialist must review/enter a value in the letter before the final copy can be printed and sent to the surgeon.

```
%let FRD =  ^S={foreground=Red}" || "XX" || "^S={};
```

FORMAT DATE

The date that appears on the letter may be formatted. The WORDDATE format was used to format the system date that appears on the letter.

```
 CALL SYMPUT("FDATE",LEFT(PUT("&SYSDATE"D,WORDDATE.)));
```

SIGNATURE LINE

The signature line presented a host of issues. There was a requirement to have a place on the form for the surgeon to date and sign the form. The font selected for printing the letter was a proportional font. This caused the label for the Date to vary based on the size of the surgeon's name. To resolve the issue, a non-proportional font is used to print the signature line.

```
_____                                    _____
Name of Surgeon, M.D.                    Date
```

The number of spaces between the name and date were also calculated and stored in a macro variable named &SIGLINE. This macro variable is used in the letter to set and print the signature line.

```
data _NULL_;
 SigLine = trim("&SigName") || repeat(' ',50-1-length(trim("&SigName"))) || 'Date';
 CALL SYMPUT("SigLine",SigLine);
run;
```

Courier is a fixed font which means every character uses the same amount of space on the line. The code listed below sets the fond, makes the text bold and sets the size of the font to three. This code is used to print the signature line on the letter.

```
^n^S={font=('Courier') font_weight=bold font_size=3.0}_____
_____
^n&SigLine^S={}
```

PRINT LETTER

The Last step in the process is the creation of the letter in RTF format. PROC REPORT® is used to generate the letter. An image can be added to spruce up the output. We use it to add our company logo. Style attributes have also been added to the DEFINE option. The JUST option left justifies the text. ASIS tells the viewer to display the text as it appears. CELLWIDTH is used to tell how large to make the cell that contains the letter. I chose a width of seven because it fits nicely on a 9x11 sheet of paper. This can be adjusted based on your specific needs. The last option, font, is used to set the font used on the letter and set the size of the font displayed. The attributes set on the define statement affect the txt fields. These values are overridden by using inline formats, such as style in the body of the letter.

```
/*************************************************************************
 Print Certification Letter
*************************************************************************/
```

```
proc report data=CertLetter nowd style={bordercolor=white preimage='X:\clinical\LeedsID\logo_005.jpg'  };
  columns txt;
  define txt / "  style={ just=left bordercolor=white  asis=ON background=DMWHITE cellwidth=7 in font=('Times New
Roman') font_size=3  };
 run;
```

OTHER THINGS TO CONSIDER

Name RTF files with meaningful names. We use the surgeon name as part of the filename to make it easier for the clinical specialist to find and view the letters. Since the letters are produced on a quarterly basis the output is saved to a new dated folder each quarter (i.e. 2009Q3) and broke out by study.

..\Compensation\<study>\<YYYYQn>

Consider keeping older versions of a letter just in case the letter needs to be reproduced.

When making changes to the RTF file you may have to reset the line space between the rows of text. Select the Page Layout tab in Microsoft Word and change the spacing before and after the text to 0 pt..

CONCLUSION

The distribution of output using letters is an excellent method of preparing reports for distribution to internal and external customers. The letters can be used to summarize the information and make the overall packet of letters, tables, listing, graphs and charts look professional.

REFERENCES

SAS® Institute Inc. (2000). *SAS® Language Reference: Dictionary, Version 8.* Cary, NC:SAS® Institute Inc.
SAS® Institute Inc. (1999). *SAS® Procedure Guide, Version 8.* Cary, NC:SAS® Institute Inc.
SAS® Institute Inc. (2009). SAS® Website.

ACKNOWLEDGMENTS

I would like to thank Brian Knepple, SAS Programmer, for helping to write and test the SAS code used to create letters. I would also like to thank Mary Ann Sadenwater, SAS Programmer for reviewing the final paper.

CONTACT INFORMATION

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

Michael Stout
Team Lead, SAS Programming
DePuy Orthopaedics, Inc.
PO Box 988
700 Orthopaedic Drive
Warsaw, Indiana 46581-0988
Work Phone: (574) 372 7379
Fax: (574) 371 4950
E-mail: Mstout2@its.jnj.com

SAS CODE

```
%macro Print_Certification_Letter(OwnRTF=Y,
                                  inv_num=);

%let FRD =  ^S={foreground=Red}" || "XX" || "^S={};
%let FBL =  ^S={foreground=Blue}" || "XX" || "^S={};


/*****************************************************************
Read data and load values into macro variables
*****************************************************************/
PROC IMPORT OUT=StudyEvals
DATAFILE="L:\Warsaw\Orthopaedics\Clinical_Affairs\Biostats_and_DM_General\SAS\
SystemTables\Certificaton_Letters.xls" DBMS=EXCEL2000 REPLACE;
   RANGE="Contract$";
   GETNAMES=YES;
   RUN;

proc sql noprint;
  select Period, Studies, Deliverables, CS_NUM
         into :Period, :Studies, :Deliverables, :CS_Key
  from StudyEvals
  where inv="&inv_num" and Upcase(Study) = Upcase("&Studycode");
quit;
run;

proc IMPORT OUT=StudyEvals
DATAFILE="L:\Warsaw\Orthopaedics\Clinical_Affairs\Biostats_and_DM_General\SAS\
SystemTables\Certificaton_Letters.xls" DBMS=EXCEL2000 REPLACE;
   RANGE="Address$";
   GETNAMES=YES;
   RUN;

proc sql noprint;
  select  SNAME, SigName, Addr1, Addr2, Addr3, Addr4, Addr5, Addr6, Certify
         into :SName, :SigName, :Addr1, :Addr2, :Addr3, :Addr4, :Addr5,
:Addr6, :Certify
         from StudyEvals
  where inv="&inv_num";
quit;
run;

PROC IMPORT OUT=StudyEvals
DATAFILE="L:\Warsaw\Orthopaedics\Clinical_Affairs\Biostats_and_DM_General\SAS\
SystemTables\Certificaton_Letters.xls" DBMS=EXCEL2000 REPLACE;
   RANGE="Specialist$";
   GETNAMES=YES;
   RUN;

proc sql noprint;
  select CS_NUM, Contract_Specialist, CS_Phone, CS_Fax, PD_Name, PD_Title,
PD_Phone
         into :CS_Num, :Contract_Specialist, :CS_Phone, :CS_Fax, :PD_Name,
:PD_Title, :PD_Phone
  from StudyEvals
```

```
  where CS_NUM = "&CS_Key";
quit;
run;


ods noptitle escapechar'^';
options orientation=portrait;
ods rtf;
ods rtf  style=styles.rptsty;

%if OwnRTF = Y %then %do;
  ods rtf file='&OutputDir.\Certification Letter &inv_num..rtf' style=rptsty;
%end;
title;

data _NULL_;
  CALL SYMPUT("FDATE",LEFT(PUT("&SYSDATE"D,WORDDATE.)));
  SigLine = trim("&SigName") || repeat(' ',50-1-length(trim("&SigName"))) ||
'Date';
  CALL SYMPUT("SigLine",SigLine);
run;


/*****************************************************************
Insert values into form letter
*****************************************************************/

data CertLetter;
length txt $6000.;
txt =
"&fDATE
^n
^n
^n&Addr1
^n&Addr2
^n&Addr3
^n&Addr4
^n&Addr5
^n&Addr6
^n
^nDear %trim(%BQUOTE(&Sname)):
^n
^nThank you for your participation in DePuy Orthopaedics Clinical Research
Activities.
^nThe details of your recent activities are:
^n
^nPeriod:       " ||  "09"x ||"&Period
^nStudy(s):     " || "09"x ||"&STUDIES" ||
"^nDeliverables:" || "09"x ||"&Deliverables
^n
^n
^n
^nThe certification statement below must be completed and returned to
&Contract_Specialist in the Clinical Research Department.  Without this
statement, payment cannot be issued.  Please feel free to contact
&Contract_Specialist at
^n&CS_PHONE if you have any questions or need additional assistance.
^n
^n_____
_____
```

```
^n
^n
^n
^n ^S={font_weight=bold font_size=3.5}" || "Investigator Certification
Statement ^S={}
^n
^n
I certify that %TRIM(%BQUOTE(&CERTIFY)), provided clinical data as identified
above according to the terms of the clinical research agreement with DePuy.
^n
^n
^n
^n";
output;

txt =
"^n^S={font=('Courier') background=blue font_weight=bold
font_size=3.0}_____________________________________
_____________________
^n&SigLine^S={}
";
output;
run;



/*************************************************************************
 Print Certification Letter
*************************************************************************/
proc report data=CertLetter nowd style={bordercolor=white
preimage='X:\clinical\LeedsID\logo_005.jpg'
  };
 /* preimage */
  columns txt;
  define txt / ''  style={ just=left bordercolor=white  asis=ON
background=DMWHITE cellwidth=7 in font=('Times New Roman') font_size=3  };

  compute after /
    style={ just=left bordercolor=white  asis=ON background=DMWHITE
cellwidth=7 in font=('Arial') font_size=2.5  };
  endcomp;

run;

%if OwnRTF = Y %then %do;
  ods rtf close;
%end;

options orientation=landscape;
ods rtf;
ods rtf  style=styles.rptsty;
%mend;
```