Using SAS® to Manage and Report Long Text Fields in a Clinical DBMS

Na Li, Pharmacyclics, Sunnyvale, CA

ABSTRACT

Using SAS® to manage and report data containing long text fields in a clinical Database Management System (DBMS) can be a daunting task. A long text field can be a memo field, comment field, or narrative field, ranging from a couple of characters to over thousands of lines with several paragraphs. Within the text field, there might be special embedded symbols such as carriage returns, line feeds, tabs, and page breaks. In order to report such text fields with a meaningful layout and to be able to programmatically select certain sections of the text field, it is essential to understand the retrieval process of the text field and to manage these special embedded symbols. This paper shows a step-by-step flow of the entire data retrieval, management, and reporting process.

INTRODUCTION

Clinical study information is usually entered on Case Report Forms (CRFs). In the CRFs, there are comment fields and memo fields. In addition to CRFs, the Serious Adverse Events are entered into MedWatch forms, which include the narrative describing the events in detail. The information tends to be less structured free style long text containing embedded special characters. Using SAS to search the part of the data by key words or to retrieve a certain section of data in these fields is difficult. Reporting this information along with the other clinical information in summary format for review by the safety committee is even more challenging.

This paper covers some methods of handling this type of programming challenge, starting at data retrieval from a DBMS external database into SAS, getting valuable information from the long text fields, and reporting such information in a nice layout along with other essential information. In this paper, the SAS System version 8 is used under the Windows operation system.

READING THE LONG TEXT DATA INTO SAS

Although many methods can be used to import data from an external data source, using an SQL Pass-Through query to retrieve data from a clinical database is a typical approach. SAS/ACCESS to ODBC® (Open Database Connectivity) may be considered the most flexible technique in handling large databases allowing conditional access to the data, and follows the standard Proc SQL format. The SAS/ACCESS Interface to ODBC needs to be licenses and installed for this method. This method can be used to access many data sources that provide an ODBC driver, including ORACLE, SQL server, and MS ACCESS database. By default, only the first 1024 characters will be retrieved from text fields. Information will be truncated if text fields have more than 1024 characters. To overcome this truncation, the DBMAX_TEXT= option overwrites the default in the Pass-Through connection parameters or the libname statement. The value can be 1 to 32767. However a large value of this option can dramatically slow down execution in SAS.

The following code is an example of the usage of PROC SQL to access an MS ACCESS database with the data source named “databs1”.

```sas
proc SQL;
  connect to ODBC(DSN=databs1 UID=testUser PWD=XXXX dbmax_text=12000);
  create table work.test as select * from
    CONNECTION TO ODBC(select * from table1);
  disconnect from odbc;
```
Except for the SAS/ACCESS interface to ODBC, the SAS/ACCESS SQL Pass-Through facility can connect to a DBMS such as DB2 and ORACLE and to send statements directly to the DBMS for execution. The following Pass-Through Facility example sends a query to an ORACLE database for processing:

```
proc SQL;
   connect to ORACLE(user=xxx password=xxx path='path1' dbmax_text=12000);
   create table tbl1 as select * from CONNECTION TO ORACLE(select * from ortbl1);
   disconnect from ORACLE;
quit;
```

This example uses the Pass-Through CONNECT statement to establish a connection with an ORACLE database with the specified values for the USER=, PASSWORD=, and PATH= arguments. The CONNECTION TO component in the FROM clause of the SELECT statement allows data to be retrieved from the database. The DBMS-specific statement that is sent to ORACLE is enclosed in the inner-most parentheses. The DISCONNECT statement terminates the connection to ORACLE. You can also use SAS/ACCESS software to work directly with DBMS tables by using the SAS/ACCESS LIBNAME statement. The command is:

```
LIBNAME datal ORACLE UID=xxx PWD=XXX PATH=path1 DBMAX_TEXT=12000;
```

**HANDLING SPECIAL EMBEDDED CHARACTERS**

Special embedded characters such as carriage returns, line feeds, tabs, page breaks, and so forth, are a function of the operating system. Usually, SAS reads one line of data until a carriage return and line feed character combination (the hexadecimal value '0D0A'x) are encountered or until just a line feed character (the hexadecimal value '0A'x) is encountered. Different systems use different characters. For example, in Windows the line feed is followed by carriage return with the hexadecimal '0D0A', but in UNIX the line feed is expressed as hexadecimal '0A'. In general, the hexadecimal '20' represents space, '0C' represents a new page, '09' represents a horizontal tab, and '0B' represents a vertical tab. The web site WWW.ASCIITABLE.COM provides the information of the special characters in ASCII form.

In order to pave the road for managing and reporting data with the special embedded characters, finding and removing these special characters is critical. The following code shows how to remove some typical embedded characters or remove all of the embedded characters. Before removing the embedded characters such as line feed and carriage return that affect the text layout, verify these characters are important since some other special characters (e.g., female symbol, copyright sign, gamma, beta) might need to be kept. The code below also identifies these embedded characters.

```
data meddesct; set meddesct;
   *** following code strips off some typical special characters;
   test2=compress(wordvb, '0D'x);  ** remove the carriage return;
   test2=compress(test2, '0A'x);   ** remove the line feed;
   *** following method is used to remove all the special characters
   *** which are not in the list by replacing the characters with space;
   *** verify each byte in string until no more non-printables are found;
   do until (test=0);
      ** for SAS 9.0 and above, use NOPRINT;
      /* test=notprint(string); */
      ** for other lower SAS version, use VERIFY;
      test=verify(upcase(wordvb), ' ABCDEFGHIJKLMNOPQRSTUVWXYZ,.1234567890[]');
      if test>0 then do;
```
specchar=substr(wordvb,test,1); *** SPECCHAR carries the special char;

** if a non-printable is found, replace it with a space;
sub
str(wordvb, test,1)=' '; end;
end;

*** use put statement to write the special characters into log;
format specchar $hex.;
put specchar=;
run;

Understanding the embedded special characters can be helpful in text selection. The program below shows the code to parse the first paragraph of the text field into a series of 1 to 130 variables (line1 to line130), depending on the length of the first paragraph, with each line stored in one variable holding 95 characters. The paragraph is separated by the hexadecimal character '0D'x in the following example.

data fstpar; set dataone;
array lines{*} $95 line1-line130;
label nlines = "Number of Lines of Description";
format nlines 3.;
j=1;
_work_=event; **event is the variable has long text;
do until(_work_=" ");
  _p_=index(_work_, '0D'x);
  ** find location of space character closest to, but not exceeding the 96th position of the remaining long text **;
  do _b_=95 by -1 until(substr(_work_,_b_+1)=: " ");
  if 1 < _p_ <= 96 then do;
    lines[j]=substr(_work_,1,_p_-1);  ** define Linel only **;
    _work_=' ';  ***only get the first paragraph;
    end;
  else do;
    lines[j]=substr(_work_,1,_b_);  *** define Linel to Line130 **;
    _work_=left(substr(_work_,_b_+2));  *** re-define remaining long text **;
    end;
  _j=_j+1;
  if _j>130 then put "warning, long paragraph needs more than 130 variables";
end;
nlines=_j-1;
run;

Sometimes certain sections of information associated with key words might be needed. For example, we need to consolidate the MedDRA mapped Adverse Event terms from the MedWatch form (a screen shot of the MedWatch form is shown in Figure 1) with the mapped terms from the CRFs. The information in the MedWatch form is stored as a long text field under the clinical DBMS and the information in the CRFs is stored as a SAS data set. The mapped terms in the long text field, after the key words: MedDRA version 7.0, need to be retrieved as a SAS variable first before the consolidation of the two systems takes place. The code below shows how to select certain pieces of information by using key words. The Adverse Event verbatim terms are the words before each bracket, which are retrieved as the values of the variable WORDPT. The MedDRA mapped terms are the words inside the paired brackets, which are retrieved as the values of the variable WORDSOC. Once the information is collected into SAS
variables, the information can be reported or compared with other information in a SAS data set.

Figure 1

Subject ID: 119008 / AD Date of Event: 01 Jan 2006 SAE #: 120-S-100

Event: Generalized rash

This is a report of a hospitalization due to a generalized rash in a 20 year old woman. The subject was hospitalized 11 days after the last study treatment due to a generalized rash. Ceftriaxone was started and the subject was discharged the next day. The event of generalized rash is possibly related to drug.

Subject 119008 has a relevant past medical history of adenocarcinoma of the lung (02 Jun 2004) with lymph nodes, adrenal, and brain metastases (17 Nov 2005).

This report is considered to fulfill the criteria of serious, related, and unexpected with respect to study treatment. It will be reported in an expedited fashion.

COSTART:
Generalized rash [SKIN RASH]

MedDRA version 7.0:
Rash generalized [Skin and subcutaneous tissue disorders],
Myocardial infarction [Cardiac disorders],
Pulmonary oedema [Respiratory thoracic and mediastinal disorders]

data meddesct; set meddesct;
length meddra $600; *** variable to keep the information after a key word;
*** EVENT field has the long text values;
*** only get the text after the key word MedDRA version 7.0:
medind = index(event, 'MedDRA version 7.0');
if medind>0 then meddra=substr(event, medind+20);
n=length(meddra)-length(compress(meddra, '',')))+1; *** get the counts of the terms;
stopend=n+1; *** the stop part, since i=i+1, use this point for the last one;
length word wordpt wordsoc $100;
label wordpt='AE verbatim Term'
    wordsoc='MedDRA Term';
i=1;  

do until (i=stopend);
    word=scan(meddra, i, ',');
    if word=' ' then leave;
    exc=index(word, '[');
    *** for the ones with soc terms in the bracket;
    if exc>0 then do;
        wordpt=trim(left(substr(word,1,exc-1) ));
        wordsoc=compress(trim(left(substr(word, exc+1) )); ']' );
    end;
    output;
    i=i+1;
end;
run;

**METHOD ON REPORTING THE TEXT FIELD**

After the information has been selected and all the unwanted embedded special characters are removed, the data is ready for a report. You can try to use ods html or ods rtf (e.g., ods rtf body=c:\temp\ae.rtf) to write out the result but neither retains the layout of the input data without additional programming codes. Proc REPORT does retain the layout but needs more code to create the header part when specific patient information is needed as a part of the report. Using a DATA _null_ step is a good method but you need to parse the input data into multiple lines with controlled line size first. Here is the example of using DATA _null_ to get a report into an RTF (WORD) format.

```sas
filename outfile 'C:\Protocols\apr2005\Output\l_fstpar.out';

proc printto print=outfile new; run;

data _null_; set medwatch; by ptid fu_no;
    file print ;
    array line {*} line1-line20;
    if first.fu_no then
        put @1 'ptid=' ptid @16 'SAE=' saenoc @32 'Followup# ' fu_no / @1 134*');
    do i=1 to nlines;
        put   @23 line{i};
    end;
    put //;
    if last.ptid then link foot;
    return;

foot:
    put @c1 134*');
    return;
run;
Proc printto; run;

*** the out2rtf macro is provided by David Ward, downloaded from internet, address: http://support.sas.com/rnd/base/topics/templateFAQ/out2rtf.sas;

%out2rtf(in=c:\temp\l_fstpar.out,
    out=c:\temp,ps=53,bold=1,company=XYZ Data Analysis,author=XYZ);
```
The output shown in Figure 2 combines the first paragraph of each of the two MedWatch form reports, where the first one is the initial narrative report and the other is the follow up of the previous reports. The report provides a brief safety summary of the Serious Adverse Event for review by the Safety Committee.

CONCLUSION
When searching for special information by key words, the special embedded symbols, such as line feeds, carriage return, etc., may first need to be removed in order to successfully retrieve the information. In some cases, however, it may be important to retain special symbols, i.e., in order to preserve the original formatting of information. An understanding of special embedded characters and retrieval methods can help in both cases. Retrieving valuable information in long text fields from an external database can help to monitor, analyze, and review the clinical trials. Using SAS to report the long text fields is very efficient to coordinate with other valuable information.

REFERENCES

ACKNOWLEDGMENTS
The author is grateful for the suggestions of Kathy Boussina, Eugene Yeh, and Paul Sherman. The author also would like to honor God for His inspiration, encouragement and support in writing this paper.

CONTACT INFORMATION
If you have any comments and questions regarding this paper, please contact:

Na Li
Pharmacycics, Inc.
995 East Arques Avenue
Sunnyvale, CA 94089
Office: (408) 990-7293
E-mail: nli@pcyc.com