PharmaSUG2010 - Paper CC07

Updating SDTM Metadata Excel File (Define.xls) with SAS

Lin Yan, Merck & Co., Inc., Rahway, NJ

ABSTRACT

When preparing FDA submissions, one usually needs to create metadata Define.xls as working files in order to ensure that the structures and attributes of the submitted SAS datasets are in accordance with FDA endorsed SDTM. To do that, continuous updating the Define files may be required.. When updating a Define file, one can choose to modify the Excel file directly. However, this common practice can be cumbersome when some changes need to be applied to multiple Excel spreadsheets. For example, it may be necessary to add some identical comments to the baseline definition of the Lab and many other finding domains with subsequent updates to the comments. In this case, working on the Define file from SAS, most of time, can be more efficient. This paper illustrates a method of modifying Define Excel files using SAS programs step by step, which includes some technical tricks to ensure correct data transfer between Excel files and SAS data sets.

1 INTRODUCTION

In the process of preparing FDA submissions, a SAS programmer usually creates metadata as Excel files that describe the names, labels, lengths, controlled terms, roles, and derivation rules of variables of the submitted SAS datasets. This is part of the process that makes sure that the submitted SAS datasets are in accordance with SDTM. It is a common practice to directly make modification to the Excel files. This practice is feasible if minimal changes are needed and if the frequency of modification is low. However, this practice becomes too labor intensive, tedious and error prone when changes are applied to multiple Excel spreadsheets. For example, the author may have to add a field of comments for some variables in multiple spreadsheets each of which is for one domain, for example, adding comments for the Lab baseline flag LBBLFL in Lab domain LB, comments for Vital Sign baseline flag VSBLFL in VS domain, comments for clinical finding baseline flag CFBLFL in Clinical Findings CF domain, etc. and all of these comments are identical except the prefix of the variable name. Copying the comments from one domain to another is time-consuming and error-prone.

This paper presents a method that facilitates and centralizes the modifications using SAS programs. Using SAS programs to handle Excel files also has its pitfalls when converting data from Excel files to SAS datasets or from SAS datasets to Excel files. With care and attention to detail, these concerns can be easily overcome with various techniques. This paper will go into using SAS for this purpose and also presents some technical tricks to avoid data conversion problems.

2 Methodology

2.1 Procedure to modify the Define using SAS

The scenario to be considered for this paper is one in which you may have a Define.xls with more than 30 domains, each represented in its own worksheet, and we need to add comments for more than just one domain.

2.1.1 Convert Define Excel file to SAS

There is more than one method to convert an Excel file to SAS. The straight forward method is using the SAS procedure PROC IMPORT.

After getting the number of total domains (assign it as NDOMAIN) and the name of each domain, the following code can be used to import the Excel file to a SAS dataset. Assuming each domain's sequence number is I and the name is domain_&i, the following code can be applied.

```
Proc import datafile="C:\CDISC\eSub\Define\define.xls"
                    out= domain_&i
                    dbms=EXCEL replace;
                    sheet=&&domain&i;
                    getnames=no;
                    MIXED   =YES;
                    SCANTEXT=YES;
                    USEDATE =No;
                    SCANTIME=No;
 run;
data test.define;
   %do &I %to &ndomain.; set domain_&I; %end;
Run;
```

As SAS sets the columns as character or numeric by the first eight rows of data, we can use **mixed=yes** to input variables containing both character or numeric. Since all the variables in the Define do not contain a date or date time format, we can set **usedate=no and scantime =no.**

2.1.2 Modify The Define SAS Dataset

We add comments for the common variables across many of the findings and event domains such as –BLFL, --DY, --DT, –STDTC, and --ENDTC. To ensure consistency across the domains with the similar variables, we are able to achieve this with a simple data step and one dataset.

First pick those variables from related domains. Then we add those comments for intended variables. The code can be as follows.

```
        data define ;
              length var $5.;
                set test.define;
                var = substr(name, 3);
               if var='BLFL' then comments = "Calculation: "||strip(domain)||"BLFL=Y for
last non missing record on or before the first dose date (RFSTDTC).";
             else if var='DY' then comments = ….;
                run;
```

2.1.3. Transfer SAS Dataset Back to Define Excel File

After getting all the updates done in a SAS dataset, transfer the SAS dataset back to the define.xls. We can use SAS procedure PROC EXPORT or Dynamic Data Exchange. The first method has some shortcomings which will be discussed later in 2.2.1.

The second method uses Dynamic Data Exchange (DDE). Dynamic Data Exchange (DDE) is an MS Windows way to dynamically transfer data between Windows-based applications. Using this method, SAS can read or write data to other windows applications, for example Excel. Here are the steps.

First invoke Excel

```
options noxwait noxsync;
x '"C:\Program Files\Microsoft office\OFFICE11\EXCEL.EXE" ';
```

Secondly, set connection with Excel

```
FILENAME DDTEM DDE 'EXCEL|SYSTEM';
```

Under SAS data steps, open target Excel file which will be updated.

```
 DATA _NULL_;
            FILE DDTEM;
            PUT '[OPEN("C:\Define\Define.xls")]';
            RC = SLEEP(0.5);
RUN;
```

The SLEEP command holds the SAS system to ensure that the Excel file opened before SAS proceeds with the data updating in the steps that follow.

Define the area in the Excel file to tell SAS the place to insert the data. In the example, eleven columns in an Excel sheet are updated with the data in the SAS dataset Define.

```
FILENAME DDE_SAS DDE "EXCEL|&&dname&i.!R3C1:R80C7" NOTAB;

DATA _NULL_;
            FILE DDE_SAS;
            SET  Define ;
            PUT
                        domain $ "09"x
                        name  $ "09"x
                        label $ "09"x
                        type   $ "09"x
                        length $ "09"x
                        role  $ "09"x
                        codelist  $ "09"x
                        mandatory $  "09"x
                        origin  $ "09"x
                        seq  $ "09"x
                        comments $ "09"x;
 RUN;
```

2.1.4. Compare The Modified Define Data to Original Define Data

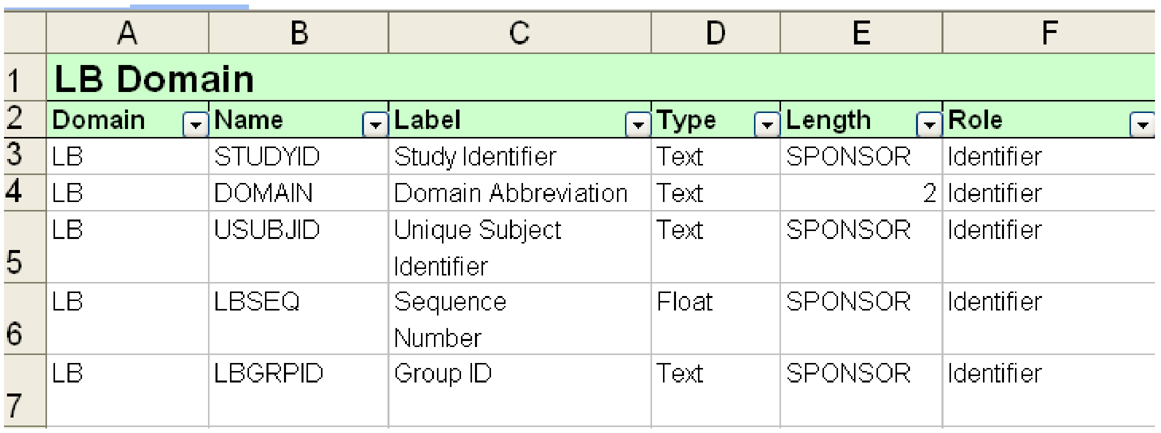
We need to ensure that there were no unintended changes to the original Define metadata from the time it was loaded to when it was put out as a new Excel file again. This can be achieved by again bringing in the new Define.xls file into SAS to use PROC COMPARE to compare the SAS datasets of the define files before and after updating.

```
 PROC COMPARE base=define_old compare = define_new out=diffs outnoequal outbase
outcomp ;
       id domain name;
       var label type format origin role codelist mandatory origin seq comments;
  RUN;
```

2.2. Avoid Inaccurate Data Conversion Between Excel And SAS

2.2.1 Prevent Overwriting Header Formats in Each Domain

You will want to retain the original sheet and column headers formats in each domain. The LB domain, for example, has the sheet and column headers in the first row and are formatted with a green background and some hidden functions such as an Excel list function on the columns.

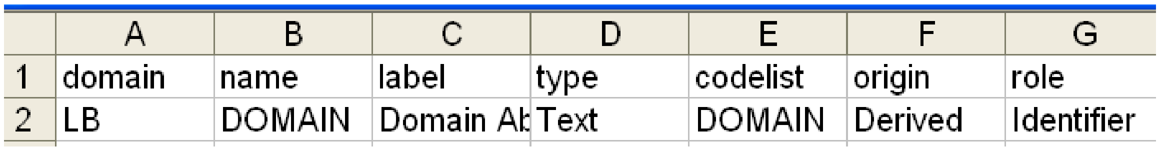


	A	B	C	D	E	F
1	LB Domain					
2	Domain	Name	Label	Type	Length	Role
3	LB	STUDYID	Study Identifier	Text	SPONSOR	Identifier
4	LB	DOMAIN	Domain Abbreviation	Text	2	Identifier
5	LB	USUBJID	Unique Subject Identifier	Text	SPONSOR	Identifier
6	LB	LBSEQ	Sequence Number	Float	SPONSOR	Identifier
7	LB	LBGRPID	Group ID	Text	SPONSOR	Identifier

If we use SAS PROC EXPORT (code below) to send the updated data to this sheet, the whole sheet will be overwritten.

```
 PROC EXPORT data=test.define.test
             outfile='c:\myfiles\class'
             dbms=EXCEL;
  RUN;
```

The features in the original sheet, such as color and list function will disappear. The output looks like the following.

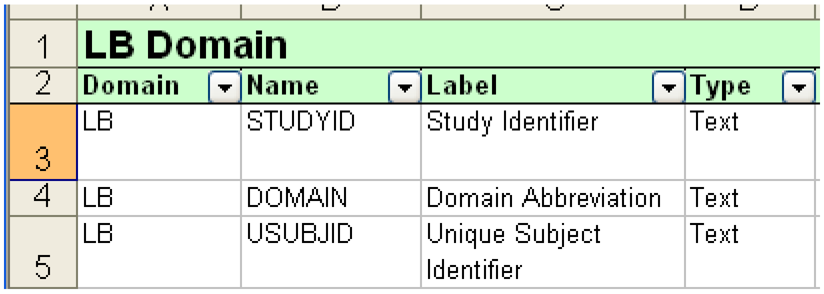


	A	B	C	D	E	F	G
1	domain	name	label	type	codelist	origin	role
2	LB	DOMAIN	Domain Ab	Text	DOMAIN	Derived	Identifier

However DDE can handle the issue by replacing only the cells you want to replace and not the whole sheet. The code is presented in 2.1.3.

2.2.2. Inaccurate Info in The Cells

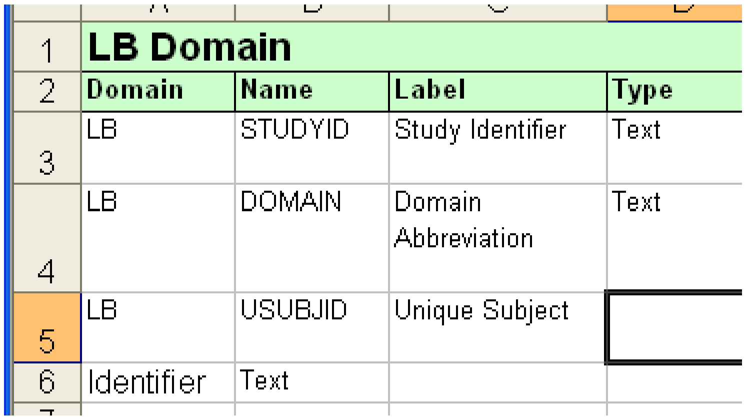
Sometimes in the Excel file there will be fields entered by Alt+Enter keys which will put the data in more than one line within a cell. For example, in the LB domain, the label of variable USUBJID is separated into 2 rows thus making "Unique Subject" and "Identifier" show on 2 rows.



1	LB Domain			
2	Domain	Name	Label	Type
3	LB	STUDYID	Study Identifier	Text
4	LB	DOMAIN	Domain Abbreviation	Text
5	LB	USUBJID	Unique Subject Identifier	Text

After we read the record into SAS using PROC IMPORT, the field value becomes "Unique SubjectIdentifier" with no space between "Unique Subject" and "Identifier".

Using DDE in SAS, after transferring data from SAS to Excel the word "Identifier" will be out of the same cell and switched to the next line as shown below.



1	LB Domain			
2	Domain	Name	Label	Type
3	LB	STUDYID	Study Identifier	Text
4	LB	DOMAIN	Domain Abbreviation	Text
5	LB	USUBJID	Unique Subject	
6	Identifier	Text		

This issue can be resolved by using a SAS function to translate the linefeed into a space. After transferring back from SAS to excel, the words will be in the desired place. The code is displayed below.

```
    Label=TRANSLATE(Label," ",'0A'x); *here Label is the variable whose value
needs the translation;
```

CONCLUSIONS

It is possible to update the metadata excel file in SAS. This paper showed the author's preferred way to convert an Excel file to a SAS dataset using PROC IMPORT, then how to modify the contents of a SAS dataset in SAS and transfer SAS dataset back to an Excel file. When transferring SAS data back to Excel, we can use DDE in SAS to avoid changes in the cells not requiring an update. By translating a linefeed to a space using the SAS function TRANSLATE, we can prevent a record with embedded linefeed from being split into different cells.

ACKNOWLEDGEMENT

The author thanks John Troxell for his technical advice in preparing this paper and thanks John Bowen for his careful review in this paper. Also the author thanks the Department of Scientific Programming, Merck Co., & Inc for its support and opportunities.

REFERENCES

Bessler, LeRoy, SAS®-with-Excel Application Development: Tools and Techniques, proceedings of the 2005 Pharmaceutical Users Software Exchange

SAS® Institute (1999) Software: Reference, Version 9,
Copyright 1999 by SAS® Institute Inc., Cary, NC, USA

CONTACT INFORMATION

Your comments and questions are valuable and appreciated. Authors can be reached at

Lin Yan
Merck & Co., Inc.
Rahway, NJ 07065 U.S.A.
Email: lin_yan2@merck.com