PharmaSUG 2019 – Paper AD-121

The Application of SAS-Excel Handshake in DDT

Maggie Ci Jiang, Teva Pharmaceuticals, Fraser, PA

ABSTRACT

To execute the SAS code embedded in excel is always challenging due to the fact that the handshake takes place between the two different SAS and Excel software utilities. In clinical trial, when we do CDSIC data conversion, for example, instead of writing separate SAS programs to convert the Define Definition Table (DDT) to SAS datasets, we would like to have the SAS code written in excel so we can compare side by side the SAS code mapping to the rules and definitions recorded in excel file which serves as the DDT. This approach is particularly convenient for users in the clinical programming development as it allows the code mapping to be more visible; and to provide an easier way for the reviewers to trace the consistency and accuracy for the conversion between the rules and the real-time SAS code. This paper is to present the challenge of this SAS-Excel handshake with a complete real-world CDISC SDTM data conversion example. The SAS-Excel handshake individual examples in this paper include simple SAS code, the SAS built-in functions and the external customized macros. The benefits and the drawbacks of utilizing such SAS-excel handshake in DDT are also going to be discussed as well.

INTRODUCTION

Per FDA Conformance Guide, for the data standards in clinical trials, the sponsor is required to submit the study reports with the data in CDISC standards. This CDISC standard is now more robotic in development such as RAW -> SDTM -> ADaM, which opens up the opportunity to automate the data conversion in an efficient process.

However, the complication is obvious when we do the programming conversion from raw dataset to CDISC SDTM dataset or from SDTM dataset to ADaM dataset. Currently the common process of such conversions is to write separate SAS programs to complete the conversion. With this type of conversion process, the users need to review the multiple documents like the rules in DDT and the dozens of individual SAS programs etc. in order to conduct the quality control, which is truly a complicated reviewing process; and error prone if any of the documents is not synchronized. More, it’s likely to require more resources due to the reason that one person writes the DDT document and the other writes the programs.

In the CDISC data conversion, the Define Definition Table (DDT) plays an important role. This DDT is usually written in excel in a certain consistent layout. If we could take this advantage in our programming effort in the data conversion, it can efficiently save our work in many ways such as maintaining and updating the developed programs.

The application of SAS-Excel handshake in DDT intends to develop the data conversion in one excel document so to have all in one instead of having additional dozens of separate individual programs. This paper presents the practical approach by going from simple to complex steps with a real-world conversion example from RAW dataset to SDTM dataset.

WHAT IS DDT?

DDT is the Data Definition Table that is the actual metadata prepared in excel before the data conversion takes place. Each tab contains the actual domain information that provides the details and traceability of where the variable comes from; what value is expected after the data conversion and how the derivation is applied to the variable and so on.

There are in general a set of domains created for a clinical study such as Demographics, Adverse Events, Laboratory, Medical History, Vital Signs etc.. Throughout this paper, the Demographics domain is used as the example to show how the conversion is achieved with SAS-Excel handshake approach.

There are some pre-work required before the conversion happens. Here's how the Demographics domain looks like in the excel layout after the completion of mapping the raw data values into the SDTM variables.

Example Table 1: Definition Table of Demographics

SEQ Order (L)	SDTM Variable (M)	SDTM Label (N)	Conversion Type (O)	Length (P)	Controlled Terms or Format (Q)	Origin (R)	Core (S)	Role (T)	Conversion Derivation Details (U)	Conversion Comments (V)
1	STUDYID	Study Identifier	CHAR	100		Protocol	Req	Identifier	=DM.project	Set to Constant
2	DOMAIN	Domain Abbreviation	CHAR	2		Assigned	Req	Identifier	="DM"	Set to Constant
3	USUBJID	Unique Subject Identifier	CHAR	30		Derived	Req	Identifier	Concatenate of stuydID' || '_' || DM.subject	Set to Constant
4	AGE	Age	NUM	8		CRF	Exp	Record Qualifier	= DM.age	
5	AGEU	Age Units	CHAR	10	(AGEU)	CRF	Exp	Variable Qualifier	= DM.ageu	
6	SEX	Sex	CHAR	10	(SEX)	CRF	Req	Record Qualifier	= DM.sex	
7	RACE	Race	CHAR	50	(RACE)	CRF	Exp	Record Qualifier	= DM.RACE	
10	ARM	Description of Planned Arm	CHAR	100	*	Assigned	Req	Record Qualifier	= "DUMMY A" or "DUMMY B" OR = DM.ARM	Use the word " DUMMY A" or "DUMMY B" while blinded; Otherwise =DM.ARMCD
11	ARMCD	Planned Arm Code	CHAR	20	*	Assigned	Req	Record Qualifier	= "A" or "B" OR = DM.ARMCD	Use the word A or B from DUMMY A or DUMMY B while blinded; Otherwise =DM.ARMCD
15	RFSTDTC	Subject Reference Start Date/Time	CHAR	20	ISO 8601	Derived	Exp	Record Qualifier	= minimum (EX.exstdtc) and minimum (EX.exsttm)	Minimum of EXSTDTC (date) when a subject gets the study drug and Minimum of EXSTTM (time) when a subject gets the study drug
16	RFENDTC	Subject Reference End Date/Time	CHAR	20	ISO 8601	Derived	Exp	Record Qualifier	= maximum (EX.exstdtc) or maximum (EX.exendtc) or maximum (DS.DSSTDTC)	Maximum of EXSTDTC or maximum of EXENDTC or Study completion date whichever is greater

This is a simplified table from the original version as the focus is to only use the needed information to express the purpose of doing the SAS-excel handshake.

In this Example Table 1, the listed DM variables have been defined and ready for programming into the SAS dataset. Traditionally programmers write a separate program to create this DM dataset. There's nothing wrong doing that way. However, there may be a optimized way of implementing it. Below sections will explore the possibility of complete the dataset creation by filling the code in excel and to be executed by calling the program driver.

THE TYPES OF APPOACHES IN UNTILIZING THE SAS CODE IN EXCEL

Three major common types in SAS-Excel coding handshake are observed in practice. The first type is to write the SAS code straight in excel cell, examples like DATA step, PROC procedures and simple statement [Table 1 SEQ 1-7]. The second type is to apply the SAS built-in functions like SUBSTR, SCAN and MAX [Table 1 SEQ 10-11]. The third type involves the customized functions or macros which are often required in data process [Table 1 SEQ 15-16].

SIMPLE STRAIGHT CODE IN EXCEL

In general, almost all data step or PROC can be executed from excel directly.

Example Table 2: Straight Assignment Demonstration

SEQ Order (L)	SDTM Variable (M)	SDTM Label (N)	Conversion Type (O)	Length (P)		Conversion Derivation Details (U)	Conversion Comments (V)	SAS Code (W)

1	STUDYID	Study Identifier	CHAR	100		=DM.project	Set to Constant	**data** dm_1(rename=(AGEn=AGE AGEUn=AGEU SEXn=SEX RACEn=RACE)); set dm(drop=studyid);
								length STUDYID $100 DOMAIN $2 USUBJID $30 AGEn $8 AGEUn SEXn $10 RACEn $50 ARM $100 ARMCD $20;
								
								STUDYID=%str(project);

2	DOMAIN	Domain Abbreviation	CHAR	2		="DM"	Set to Constant	DOMAIN=%bquote('DM');
3	USUBJID	Unique Subject Identifier	CHAR	30		Concatenate of stuydID' || '_' || DM.subject	Set to Constant	USUBJID=%str('ABC_1234'||%str('_')||%str(subject));
4	AGE	Age	NUM	8		= DM.age		AGEn=AGE;
5	AGEU	Age Units	CHAR	10		= DM.ageu		AGEUn=AGEU;
6	SEX	Sex	CHAR	10		= DM.sex		SEXn=SEX;
7	RACE	Race	CHAR	50		= DM.RACE		RACEn=RACE;

In Example Table 2, SAS code has been implemented into the cell for each variable in Column "W" "SAS Code" column. As now it's straight for users to compare the SAS code in Column W to Conversion Derivation Details in Column U. In this Table 2 example, the SAS code is very straight and simple assignment.

SAS BUILT-IN FUNCTIONS IN EXCEL

In continuation of the previous example DM_1, SAS built-in functions have been implemented for the variables ARM and ARMCD derivation in SEQ 10 and 11.

Example Table 3: SAS Built-In Functions Demonstration

SEQ Order (L)	SDTM Variable (M)	SDTM Label (N)	Conversion Type (O)	Length (P)		Conversion Derivation Details (U)	Conversion Comments (V)	SAS Code (W)

10	ARM	Description of Planned Arm	CHAR	100		= "DUMMY A" or "DUMMY B"
						
						OR
						
						= DM.ARM
							Use the word " DUMMY A" or "DUMMY B" while blinded;
							
							Otherwise =DM.ARMCD
								length REMAINDER 8;
								REMAINDER=mod(_n_,2);
								if REMAINDER=0 then ARM=%str('DUMMY A');
								else ARM=%str('DUMMY B');

11	ARMCD	Planned Arm Code	CHAR	20		= "A" or "B"
						
						OR
						
						= DM.ARMCD
							Use the word A or B from DUMMY A or DUMMY B while blinded;
							
							Otherwise =DM.ARMCD
								ARMCD=substr(ARM,7,1);
								drop AGE AGEU SEX RACE;
								keep DOMAIN USUBJID SUBJECT AGEn AGEUn SEXn RACEn ARM ARMCD;
								RUN;
								proc sort data=dm_1; by subject;**run;**

With this Example Table 3, SAS built-in functions *MOD* and *SUBSTR* have been applied into the derivation process, which shows the coding statement is as convenient as it's been done as a separate individual SAS program.

In row SEQ 11 as you can see, RUN has been used to close up the first part DM_1 of the DM conversion. It's upon the programmer's decision to design the number of steps and procedure that are needed.

CUSTOMIZED FUCTIONS OR MACROS IN EXCEL

Now let's look at the Example Table 1 again. In the row SEQ 15, there's the need to process the date and time conversion, from character format to numeric format, or to convert the source date and time to the desired format.

The conversion of date and time is observed to be complicated which may take a page to implement the derivation handling different forms of data presentations from the source such as full date and time, partial date and time, or missing date or time etc., and it's also observed that more domains across the study require this type of derivation as well.

It's efficient to have this customized function written as a standalone macro so it can be called any time and whenever it's needed. In this presentation, the macro %CharToDTM has been created externally and is called from the excel cell row SEQ 15 column W.

Example Table 4: External Macro Demonstration

SEQ Order (L)	**SDTM Variable (M)**	**SDTM Label (N)**	**Conversion Type (O)**	**Length (P)**		**Conversion Derivation Details (U)**	**Conversion Comments (V)**	**SAS Code (W)**

15	RFSTDTC	Subject Reference Start Date/Time	CHAR	20		= minimum (EX.exstdtc) and minimum (EX.exsttm)	Minimum of EXSTDTC (date) when a subject gets the study drug and Minimum of EXSTTM (time) when a subject gets the study drug	**data** ex_1;
								set ex; where EXSTDAT>''; format dttm is8601dt. ;
								%CharToDTM(charDT=EXSTDAT,charTM=EXSTTIM, sasDTTM=dttm, charDTTM=dttmf); length RFSTDTC RFENDTC $20; RFSTDTC=''; RFENDTC='';
								run;
								proc sort data=ex_1; by subject dttm; where dt>.; **run;**
								data dm_2; set ex_1; by subject dttm; if first.subject; RFSTDTC=put(dttm,is8601dt.); if length(EXSTTIM)<=5 then RFSTDTC=dttmf; keep subject RFSTDTC; **run;**

16	RFENDTC	Subject Reference End Date/Time	CHAR	20		= maximum (EX.exstdtc) or maximum (EX.exendtc) or maximum (DS.DSSTDTC)	Maximum of EXSTDTC or maximum of EXENDTC or Study completion date whichever is greater	**data** dm_3; set ex_1; by subject dttm; if last.subject; RFENDTC=put(dttm,is8601dt.); if length(EXSTTIM)<=5 then RFENDTC=dttmf; keep subject RFENDTC; **run;**
								
								Data DM; merge dm_1 dm_2 dm_3; by subject; **run;**

The row SEQ 15 Column W has three steps to complete the RFSTDTC derivation process. Per the rules in column U, it's required to follow the defined steps to achieve the result. As you can see it's definitely okay to include more than one step in one cell. As the result, DM_2 has been created as part of the DM process.

Note here, the row SEQ 16 has actually used the result from row SEQ 15, the converted result of the date and time. RFENDTC has retrieved the subject's last date and time as its value from EX_1.

And for this simplified example, in row SEQ 16 column W, final DM is added by merging DM_1, DM_2 and DM_3 datasets. Now it's ready to be executed from the customized program driver call.

THE DRIVER TO EXECUTE THE CODE IN EXCEL

A separate independent program is developed to run the SAS code that has been filled in the excel cell Column W.

The suggested steps below intends to run the SAS code from excel cells automatically so the executable program is created separately. Once it's been set, it can be reused easily. Here are the steps:

1. Define the library location of the excel and external files:

```
libname in1 "C:\temp\mapping.xlsx";

filename test C:\temp\&dIN..sas";
```

2. Determine what tab or domain is to be implemented:

```
%let dIN=%str('DM');
```

3. Retrieve the SAS code from each cell in Column W, and write into a temporary file:.

```
data _dIN
   set in1."&dIN$"n;
     if _N_>1;
run;
libname In1 clear;

/*save each cell code as a line of string into &ln1- &ln12 in this example*/
   data _null_;
       set _dIN;
       where W20^=' ';
       call symput('ln'||(left(_n_)),left(W20) );
   run;

/*find how many cells that have code, in this example, max=12*/
   proc sql noprint;
       select distinct count(W20) into: pgmmax
       from _dIN;
   quit;

/*write each line of string into an external file as a complete program*/

   Filename test "c:\temp\&dIN..sas";
   data _null_;
       length rline $30000;
       file test;
       %do i = 1 %to &pgmmax;
           rline=%str("&&ln&i"); put %str(rline);
       %end;
   run;
```

4. Final run the extracted SAS code as a whole SAS program saved previously:

```
%inc "C:\temp\&dIN..sas";
data _null_;
     put "call execute('run;');";
run ;
```

Appendix Example gives the complete code that provides the view of how the executable code looks like after being retrieved from the excel cells. For the benefit of improving the programming performance, the steps from 1-4 can be enhanced into a customized macro so it can manipulate multiple tabs or domains efficiently if you prefer.

CONCLUSION

This paper has presented the advantages of utilizing the SAS-Excel handshake in developing data conversion specifically by following the DDT per CDISC standard. As it's been observed that it's feasible to practice the application of SAS-Excel handshake in DDT in Clinical SDTM Data development. The same approach is applicable to the analysis ADaM Data as well. The overall benefits for the programming development are its easy quality control, synchronizing review and convenient updates. It also allows the possibility of data conversion to be processed in an automation manner. However, programmers need be alert when utilizing the SAS functions as not all built-in functions are completely compatible with the excel software and need be winded around.

REFERENCE

[1] ADaMIG Version 1.1 < CDISC Analysis Data Model Team >, 2016-02-12

[2] FDA Techinical Conformance Guide v3.3, 8.3 Study Data Traceability

APPENDIX

Example: The executable code after being retrieved from the excel cells:

```
data dm_1(rename=(AGEn=AGE  AGEUn=AGEU SEXn=SEX  RACEn=RACE));  set dm(drop=studyid);
  length STUDYID $100 DOMAIN $2 USUBJID $30 AGEn $8 AGEUn SEXn $10 RACEn $50
         ARM $100  ARMCD $20;
  STUDYID=project;
  DOMAIN='DM';
  USUBJID='ABC_1234'||'_'||subject;
  AGEn=AGE;
  AGEUn=AGEU;
  SEXn=SEX;
  RACEn=RACE;
  length REMAINDER 8;
  REMAINDER=mod(_n_,2);
  if REMAINDER=0 then ARM='DUMMY A';
  else ARM='DUMMY B';
  ARMCD=substr(ARM,7,1);
  drop AGE  AGEU SEX  RACE;
  keep  DOMAIN USUBJID SUBJECT  INVNAM AGEn  AGEUn SEXn  RACEn ARM ARMCD;
RUN;
proc sort data=dm_1; by subject;run;
data ex_1;
  set ex; where EXSTDAT>''; format dttm is8601dt. ;
  if length(EXSTDAT)=10 then EXSTDAT='0'||EXSTDAT; dt=.;tm=.;dttm=.;
dy=upcase(substr(EXSTDAT,1,2)); mth=put(upcase(substr(EXSTDAT,4,3)),$mths.);
yr=substr(EXSTDAT,8,4);
  if dy ^='UN' & mth^='999' & mth^='' & yr>. then do; dt= mdy( mth, dy, yr );
dtc=put(dt,is8601da.); end;
  else if dy='UN' & mth^='999' & mth^='' & yr>. then do; dtc=strip(yr)||'-
'||strip(mth)||'-'||'UN'; end;
  else if dy='UN' & (mth='999' | mth='') & yr>. then do; dtc=strip(yr)||'-'||'UNK'||'-
'||'UN'; end; if length(EXSTTIM)<5 then EXSTTIM='0'||EXSTTIM; hh=substr(EXSTTIM,1,2);
min=substr(EXSTTIM,4,2);
  if length(EXSTTIM)>5 then sec=substr(EXSTTIM,7,2); else sec='00';
  if length(EXSTTIM)<=5 then tm=input(EXSTTIM,??time5.);
  else if length(EXSTTIM)=8 then tm=input(EXSTTIM,??time8.);
  else tm=.;
  dttm = dhms( mdy( mth, dy, yr ), hh, min, sec );
  format dt date9.; tm=tm; format tm time8.;
  format dttm is8601dt.;
  if sec^='00' & length(EXSTTIM)>5 & dttm>. then dttmf=put(dttm,is8601dt.);
  else if length(EXSTTIM)>3 & length(EXSTTIM)<=5 & dt>. then
    dttmf=put(mdy( mth, dy, yr ),yymmdd10.)||'T'||strip(hh)||':'||strip(min);
  else if length(EXSTTIM)<=3 & dt>. then dttmf=put(mdy( mth, dy, yr ),yymmdd10.);
  else dttmf=''; length RFSTDTC RFENDTC $20; RFSTDTC=''; RFENDTC='';
```

```
run;
proc sort data=ex_1; by subject dttm; where dt>.; run;
data dm_2;
  set ex_1;  by subject dttm; if first.subject; RFSTDTC=put(dttm,is8601dt.);
  if length(EXSTTIM)<=5 then RFSTDTC=dttmf;  keep subject RFSTDTC;
run;
data dm_3; set ex_1;  by subject dttm;
  if last.subject; RFENDTC=put(dttm,is8601dt.);
  if length(EXSTTIM)<=5 then RFENDTC=dttmf;
  keep subject RFENDTC;
run;
Data DM;
  merge dm_1 dm_2 dm_3; by subject;
run;
```