Paper PO05-2011

Creating a Clean Multi-Tabbed SDTM Dataset Specification Spreadsheet

Dongju Liu, Advanced Clinical, Deerfield, IL
William Conover, Advanced Clinical, Deerfield, IL

Abstract

This paper describes an efficient approach of creating SDTM data set specifications for SAS data sets used in clinical trials. First, a standard SDTM data set specification template is created according to the SDTM Implementation Guide (SDTM IG) V3.1.2 and is then saved as a multi-tabbed EXCEL workbook. Data set specifications for individual clinical trials are drafted based on this standard template. After review and revision, a SAS program is run to remove any unneeded columns and rows, and to create a final 'clean' version of the specifications. The Dynamic Data Exchange and EXCEL macros are the main tools used in the SAS program.

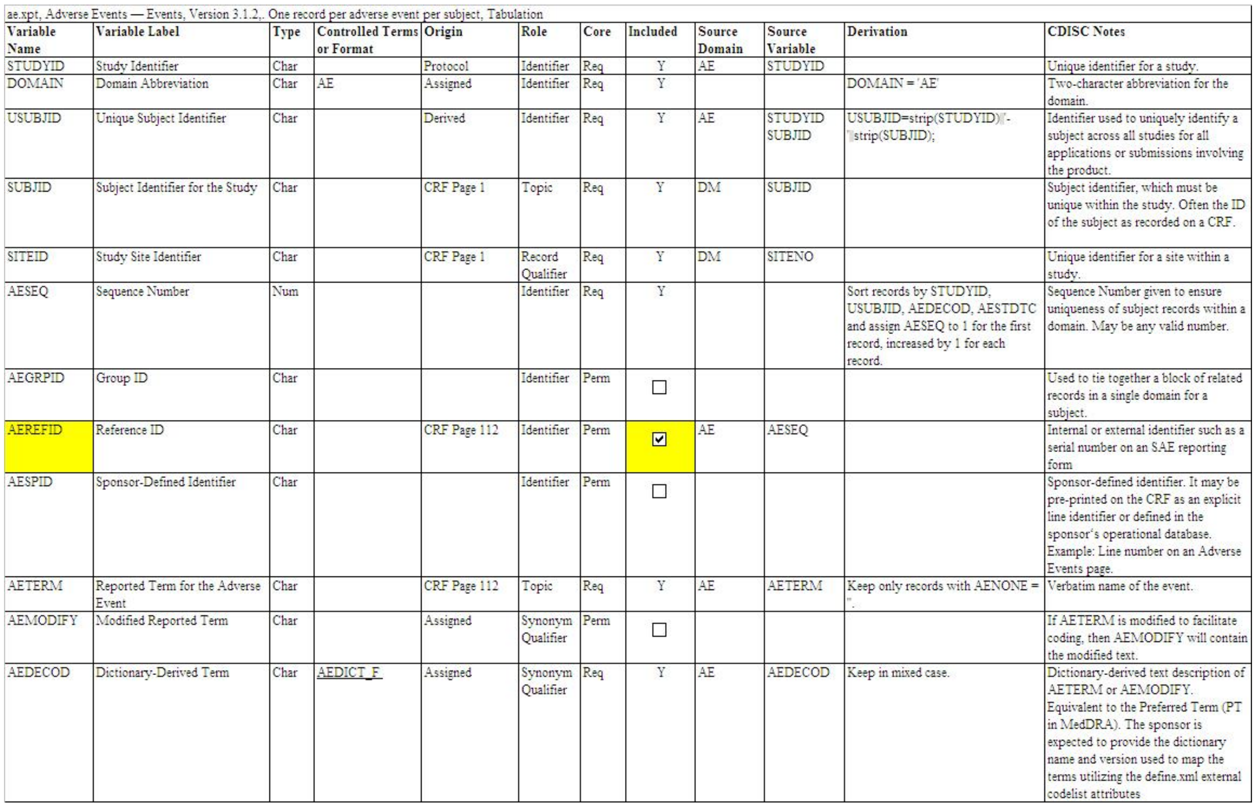
Introduction

The CDISC Study Data Tabulation Model (SDTM) defines a standard structure for SAS data sets that are to be submitted to regulatory authorities. The first step of implementing CDISC SDTM standards is to develop a mapping specification document. More than 30 standard domains are defined in SDTM IG V3.1.2. All collected data for a clinical trial needs to be mapped into these domains, or new domains created following the same appropriate structures. Typically, not all of the 30 plus domains are needed. Thus, the final specifications only contain the domains required.

A multi-tabbed EXCEL spreadsheet is utilized to create a data set specifications template, displaying one domain (i.e. SAS data set) per sheet, one row for each distinct variable, and spreadsheet columns mimicking the SDTM IG V3.1.2. This template includes all standard domains and all variables in each domain. Applying this template to a particular clinical trial typically involves removing variables/rows and data sets/sheets not required. After final review, a SAS program named FINALSPEC_SHEET will be run to eliminate the unneeded columns and rows. Using SAS to remove these items, rather than doing it manually, not only reduces the risk of making mistakes but saves development time. PC SAS 9.1.3 and Microsoft Office Excel 2003 are used for this application.

Creating the Specification Template

The specification template is created as an EXCEL workbook. Each sheet of the template includes multiple columns that hold metadata attributes and describe the data mapping required. The *CDISC Notes* column is copied from SDTM IG V3.1.2, and is helpful for frequently referenced items in the SDTM IG. The *Included* column is the key column of our application. It has a value of 'Y' for required and expected variables, and contains a checkbox for each permissible variable. When drafting the data set specifications, only the needed variables should be checked. Figure 1 shows part of AE domain specifications for a clinical trial. From the figure, it can be seen that the permissible variable AEREFID is checked in the *Included* column, while the other permissible variables are not checked.

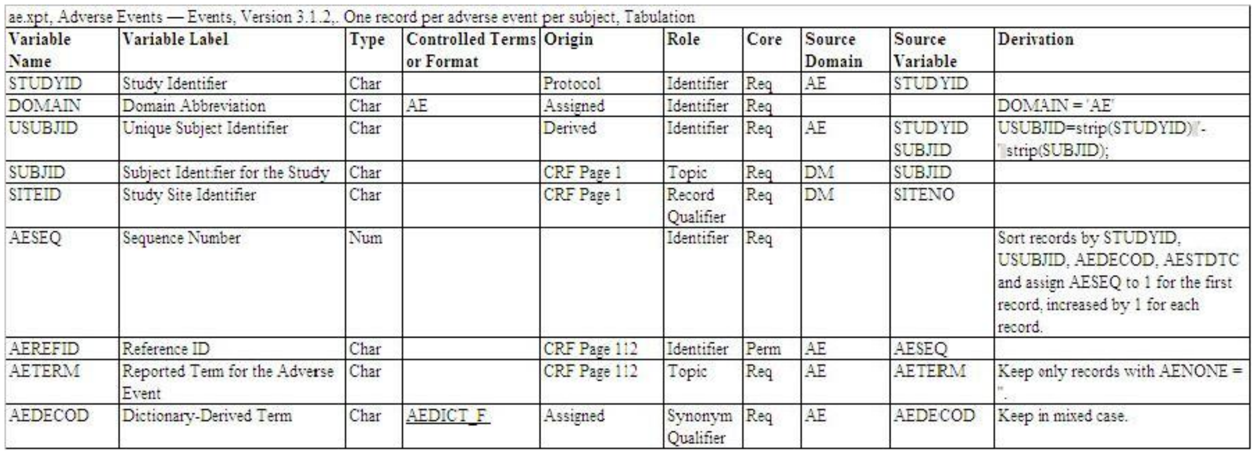


ae.xpt, Adverse Events — Events, Version 3.1.2,. One record per adverse event per subject, Tabulation

Variable Name	Variable Label	Type	Controlled Terms or Format	Origin	Role	Core	Included	Source Domain	Source Variable	Derivation	CDISC Notes
STUDYID	Study Identifier	Char		Protocol	Identifier	Req	Y	AE	STUDYID		Unique identifier for a study.
DOMAIN	Domain Abbreviation	Char	AE	Assigned	Identifier	Req	Y			DOMAIN = 'AE'	Two-character abbreviation for the domain.
USUBJID	Unique Subject Identifier	Char		Derived	Identifier	Req	Y	AE	STUDYID SUBJID	USUBJID=strip(STUDYID)||'-'||strip(SUBJID);	Identifier used to uniquely identify a subject across all studies for all applications or submissions involving the product.
SUBJID	Subject Identifier for the Study	Char		CRF Page 1	Topic	Req	Y	DM	SUBJID		Subject identifier, which must be unique within the study. Often the ID of the subject as recorded on a CRF.
SITEID	Study Site Identifier	Char		CRF Page 1	Record Qualifier	Req	Y	DM	SITENO		Unique identifier for a site within a study.
AESEQ	Sequence Number	Num			Identifier	Req	Y			Sort records by STUDYID, USUBJID, AEDECOD, AESTDTC and assign AESEQ to 1 for the first record, increased by 1 for each record.	Sequence Number given to ensure uniqueness of subject records within a domain. May be any valid number.
AEGRPID	Group ID	Char			Identifier	Perm	☐				Used to tie together a block of related records in a single domain for a subject.
AEREFID	Reference ID	Char		CRF Page 112	Identifier	Perm	☑	AE	AESEQ		Internal or external identifier such as a serial number on an SAE reporting form
AESPID	Sponsor-Defined Identifier	Char			Identifier	Perm	☐				Sponsor-defined identifier. It may be pre-printed on the CRF as an explicit line identifier or defined in the sponsor's operational database. Example: Line number on an Adverse Events page.
AETERM	Reported Term for the Adverse Event	Char		CRF Page 112	Topic	Req	Y	AE	AETERM	Keep only records with AENONE = ''.	Verbatim name of the event.
AEMODIFY	Modified Reported Term	Char		Assigned	Synonym Qualifier	Perm	☐				If AETERM is modified to facilitate coding, then AEMODIFY will contain the modified text.
AEDECOD	Dictionary-Derived Term	Char	AEDICT_F	Assigned	Synonym Qualifier	Req	Y	AE	AEDECOD	Keep in mixed case.	Dictionary-derived text description of AETERM or AEMODIFY. Equivalent to the Preferred Term (PT in MedDRA). The sponsor is expected to provide the dictionary name and version used to map the terms utilizing the define.xml external codelist attributes

Figure 1. Draft Data Set Specification Sheet

Figure 2 shows the same sheet after processing by the FINALSPEC_SHEET program.



ae.xpt, Adverse Events — Events, Version 3.1.2,. One record per adverse event per subject, Tabulation

Variable Name	Variable Label	Type	Controlled Terms or Format	Origin	Role	Core	Source Domain	Source Variable	Derivation
STUDYID	Study Identifier	Char		Protocol	Identifier	Req	AE	STUDYID	
DOMAIN	Domain Abbreviation	Char	AE	Assigned	Identifier	Req			DOMAIN = 'AE'
USUBJID	Unique Subject Identifier	Char		Derived	Identifier	Req	AE	STUDYID SUBJID	USUBJID=strip(STUDYID)||'-'||strip(SUBJID);
SUBJID	Subject Identifier for the Study	Char		CRF Page 1	Topic	Req	DM	SUBJID	
SITEID	Study Site Identifier	Char		CRF Page 1	Record Qualifier	Req	DM	SITENO	
AESEQ	Sequence Number	Num			Identifier	Req			Sort records by STUDYID, USUBJID, AEDECOD, AESTDTC and assign AESEQ to 1 for the first record, increased by 1 for each record.
AEREFID	Reference ID	Char		CRF Page 112	Identifier	Perm	AE	AESEQ	
AETERM	Reported Term for the Adverse Event	Char		CRF Page 112	Topic	Req	AE	AETERM	Keep only records with AENONE = ''.
AEDECOD	Dictionary-Derived Term	Char	AEDICT_F	Assigned	Synonym Qualifier	Req	AE	AEDECOD	Keep in mixed case.

Figure 2. Final Data Set Specification Sheet

Linking Checkbox Values

For SAS to know if a variable should be included in the specification sheet, the checkbox needs to be linked to the cell it references (in the same row as the variable name). To link a checkbox to a cell follow these steps:

- Click and drag the checkbox to the cell that needs to be linked
- Right click on the checkbox
- From the pop-up menu that appears, select 'Format Control…'
- Choose the 'Control' tab, and put cursor on the text box next to 'Cell link'
- Click in the spreadsheet cell you want to link, then click OK
- Now the checkbox is linked to the variable in the same row as the linked cell

Removing Variables and Columns

To finalize the data set specifications for a particular clinical trial, unneeded variables and columns should be removed. The SAS program FINALSPEC_SHEET completes this task by using the Dynamic Data Exchange (DDE) to execute EXCEL macros. These methods have been discussed in many other publications and so the details of using the DDE will not be covered here. However, it is noteworthy to mention that this approach is useful for users who only have BASE SAS. Also noteworthy to this method is that the SAS code is short and clear since the major task is done by the EXCEL macros. The only change required in the SAS program below is the file name. A common practice is to record the required EXCEL macros in a separate spreadsheet and reference as needed.

The SAS program FINALSPEC_SHEET is below.

```
****************** NAME OF SDTM DATASET SPECS ***********************;
%let specs = ABC_123_SDTM_Specs;
*********************************************************************;

options noxwait noxsync;

********* SPECIFYING THE LOCATION OF THE EXCEL EXECUTABLE ***********;
x '"C:\Program Files\Microsoft Office\OFFICE11\EXCEL.exe"';
*********************************************************************;

*-------------------------*
  Program Path (&progpth)
*-------------------------*;
libname here '';
data _null_;
   call sympút('progpth',pathname('here'));
run;
*-------------------------*;

data _null_;
   x=sleep(2); * wait for EXCEL to start *;
run;

filename DDEcmds dde "excel|system";

* open the workbook that contains the macro(s) *;
data _null_;
   file DDEcmds;
   put %unquote(%str(%'[open("&progpth.SpecMac.xls")]%'));
   x=sleep(2); * Wait for file to open *;
   * open the workbook to be processed with the macro(s) *;
   put %unquote(%str(%'[open("&progpth.&specs..xls")]%'));
   x=sleep(2); * Wait for file to open *;
   put '[run("SpecMac.xls!WorksheetLoop",FALSE)]';
   put '[error(false)]';

   * save the result with a new XLS file name *;
   put '[error(false)]';
   put %unquote(%str(%'[save.as("&progpth.&Specs.(Clean).xls")]%'));
   x=sleep(1); * wait for file to be saved *;
   * Close that WorkBook *;
   put '[close(TRUE)]';
   x=sleep(1); * wait for file to close *;
   * Exit *;
   put '[error(false)]';
   put '[quit()]';
run;
```

At the beginning of the code, the data set specification file name is specified, in this example 'ABC_123_SDTM_Specs', and is the only place that needs updating per application.
The statement:

```
   put %unquote(%str(%'[open("&progpth.&specs..xls")]%'));
```

instructs EXCEL to open the desired workbook. The program name and path:

```
 “C:\Program Files\Microsoft Office\OFFICE11\EXCEL.exe”
```

…is the location of the EXCEL executable file. Depending on where Microsoft Excel has been installed on the current PC, this may need to be changed accordingly. SPECMAC is the EXCEL file name with the saved EXCEL macros. Please note that the “processed” multi-tabbed SDTM data set specification spreadsheet will be renamed with ‘(Clean)’ appended to the original file name and will be saved in the same directory. Reference *SAS® and Excel, A Winning Combination, Part 2: Dynamic Data Exchange (DDE), a Popular Solution around the World* listed in the references for a better understanding of the DDE code.

The EXCEL Macros

Two EXCEL macros are used to delete the unneeded variables and columns, and are displayed below. The first macro is WORKSHEETLOOP, and is used to read the multiple sheets in the workbook. In the loop, a second macro REMRNC, is called to delete unused Variables (rows with Checkbox unchecked), the columns ‘Included’, ‘CDISC Notes’ and the Checkboxes.

Keep in mind that the below macros are not SAS code. They can be copied and pasted into an EXCEL workbook macro library.

```
Sub WorksheetLoop()
   Dim WS_Count As Integer
   Dim I As Integer

   ' Set WS_Count equal to the number of worksheets in the active workbook.
   WS_Count = ActiveWorkbook.Worksheets.Count

   ' Begin the loop.
   For I = 1 To WS_Count
      Sheets(I).Activate
      remrnc
   Next I
End Sub

Sub remrnc()
   Dim intRow
   Dim intLastRow
   Dim cb As CheckBox

   ' Assign a big number enough number for the last row.
   intLastRow = ActiveWorkbook.ActiveSheet.Range("H655").End(xlUp).Row

   ' Delete CDISC Notes Column.
   Columns("L").Delete

   ' Loop to delete rows with checkbox value equals to False (not checked).
   For intRow = 3 To intLastRow  Step 1
      ActiveWorkbook.ActiveSheet.Rows(intRow).Select
      If Cells(intRow, "H").Value = False Then
         Rows(intRow).Delete
      End If
   Next intRow

   ' Delete Included Column.
   Columns("H").Delete

   ' Delete Checkboxes.
   For Each cb In ActiveSheet.CheckBoxes
      cb.Delete
   Next

   ActiveWorkbook.ActiveSheet.Range("A3").Select
End Sub
```

Running FINALSPEC_SHEET

To finalize specifications by removing unneeded rows and columns, simply place the FINALSPEC_SHEET SAS program in the same directory as the spreadsheet with the EXCEL macros (SPECMAC.xls), and the data set specification spreadsheet. Run FINALSPEC_SHEET in batch mode with SAS 9.1.3 or higher. Once it starts running, a window will open and the user will need to click the ‘*Enable Macros*’ button. The processed/clean multi-tabbed SDTM data set specification spreadsheet will be saved in the same directory with ‘(Clean)’ appended to the original file name.

Conclusion

Creating a SDTM data set specification standard template as a multi-tabbed spreadsheet is a time saving tool. The template has additional reference columns to improve efficiency of creating specifications that are later removed prior to sending the specifications to the SAS programmers. Unneeded rows and columns from the spreadsheet are removed via the program FINALSPEC_SHEET. In addition, creating new domain specifications is as easy as copying and pasting a new worksheet. With all these time saving tools available for the Base SAS user, creating SDTM specifications has become much less burdensome.

References

LeRoy Bessler, ‘SAS® and Excel, A Winning Combination, Part 2: Dynamic Data Exchange (DDE), a Popular Solution around the World’, *MWSUG 2010 Conference Proceedings*.

Acknowledgements

Thanks to the Biostatistics team for all the support. Special thanks to Angela Roster for her help in editing this paper and designing the poster.

Contact Information

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

Dongju Liu
Advanced Clinical
10 Parkway North, Suite 350,
Deerfield, IL 60015
Phone: 847-267-1176
Fax: 847-267-1432
E-mail: dliu@advancedclinical.com
Web: http://www.advancedclinical.com/