PharmaSUG 2022 - Paper AP-110

A SAS Macro to Support the Supplemental Qualifier Section in cSDRG

Jamuna Purma, Merck & Co., Inc., Rahway, NJ, USA
Bindya Vaswani, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

The programing group is responsible for completing the Clinical Study Data Reviewer's Guide (cSDRG). The cSDRG should include a high-level overview of the SDTM data submitted for the clinical study. The PHUSE template used to generate the cSDRG contains a supplemental qualifier section 3.4. Traditionally, this section was filled in by manually checking each dataset tab in the analysis dataset (ADaM) specification document, to identify the supplemental domain variables included in the statistical analysis.

This paper introduces a SAS macro that identifies supplemental qualifier variables used in supporting analyses. It creates a table with required columns that can be easily transferred into the cSDRG section 3.4. Additionally, this macro flags any supplemental qualifier variables specified in the ADaM define document used in generating analysis datasets, but not in SDTM domains, to ensure the consistency between cSDRG section 3.4 and ADaM programming specification.

INTRODUCTION

The Clinical Study Data Reviewer's Guide (cSDRG) is a key component of the data tabulation package, created during the submission process. The PHUSE template provides step-by-step instructions for creating the cSDRG document structure and the author needs to only focus on the content. PHUSE working group has created a highly standardized document with comprehensive instructions for this purpose.

This paper provides an efficient way of populating section 3.4 in the cSDRG document. The section 3.4 describes supplemental qualifier elements that support the analysis datasets. The paper illustrates how to build a program that can be used to generate required details for this 3.4 section in cSDRG. The macro program reads the ADaM define document and generates an output with a list of supplemental qualifiers. It also verifies the supplemental qualifier variables referenced in the define document against the actual SDTM domains.

A lot of us in the pharmaceutical industry use the Pinnacle 21 (P21) tool for preparing the submission package for regulatory submission. This tool validates data quality and compliance with CDISC standards and FDA submission readiness. Using the P21 enterprise software, we can create P21 excel spreadsheet version of the ADaM define document. Some may prefer creating ADaM define.xml without using P21 tool.

CREATE METADATA DATASETS FROM THE ADAM DEFINE DOCUMENT

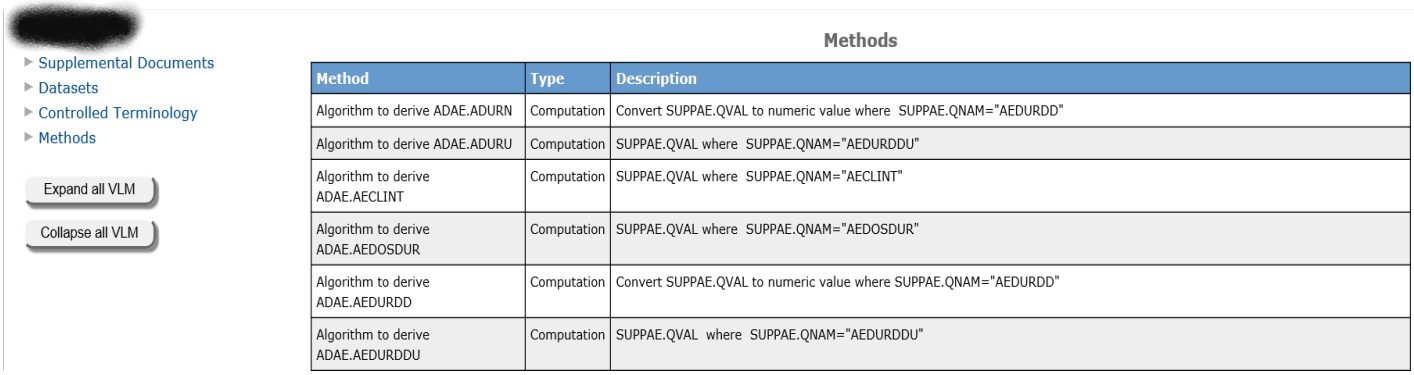
To begin with, we need to create interim datasets, using one of the two options below.

OPTION 1: USING SAS CODE WITH XML VERSION OF DEFINE

If the XML version of the ADaM define document is created without using P21 tool, use the following SAS code to read define.xml and create interim datasets. Please determine the correct dataset name that holds "METHODS" section information ("Description" column) as shown below in screenshot (**Figure 1**).

The SAS code below demonstrates how to import an XML document using the AUTOMAP option to automatically generate an XMLMap file in SAS. By specifying the "AUTOMAP=replace" in the LIBNAME statement, SAS analyzes the structure of the specified XML document and generates XMLMap syntax, which describes how to interpret the XML markup into a SAS dataset, variables (columns), and observations (rows). The "AUTOMAP=replace" is supported by the XMLV2 engine only.

The dataset created from "METHODS" section in **Figure 1** will be used further in Macro section 1.



Method	Type	Description
Algorithm to derive ADAE.ADURN	Computation	Convert SUPPAE.QVAL to numeric value where SUPPAE.QNAM="AEDURDD"
Algorithm to derive ADAE.ADURU	Computation	SUPPAE.QVAL where SUPPAE.QNAM="AEDURDDU"
Algorithm to derive ADAE.AECLINT	Computation	SUPPAE.QVAL where SUPPAE.QNAM="AECLINT"
Algorithm to derive ADAE.AEDOSDUR	Computation	SUPPAE.QVAL where SUPPAE.QNAM="AEDOSDUR"
Algorithm to derive ADAE.AEDURDD	Computation	Convert SUPPAE.QVAL to numeric value where SUPPAE.QNAM="AEDURDD"
Algorithm to derive ADAE.AEDURDDU	Computation	SUPPAE.QVAL where SUPPAE.QNAM="AEDURDDU"

Figure 1. Screenshot of XML Version of Define Document "METHODS" Section

Use FILENAME statements to define the XML file and Map file location, and LIBNAME statement to map XML markup to individual datasets, using XMLV2 engine.

```
filename define 'c:\Documents\define.xml';
filename map 'c:\Documents\definegenerate.map';
libname define xmlv2 automap=replace xmlmap=map;
proc copy in=define out=work;
run;
```

Identify METHODS section with the description column ("translatedtext" column created from XMLMAP option in above step found in this example). Output as METHODS dataset.

```
proc print data=work.description3;run;

data methods(rename=(translatedtext=description));
   Set work.description3;
run;
```

OPTION 2: USING EXCEL VERSION OF P21 DEFINE

The code below reads excel version of P21 define document and creates a dataset for each tab. It excludes duplicate tabs, if any. We use the SAS CALL EXECUTE to perform this process. The CALL EXECUTE routine accepts a single argument that is a character string or character expression. The character expression is usually a concatenation of strings containing SAS code elements to be executed after they have been resolved. CALL EXECUTE dynamically builds SAS code during DATA step iterations and it's an effective SAS code generator like a SAS macro loop.

The dataset created from "METHODS" tab in **Figure 2** will be used further in Macro section 1.

	A	B	C	D	E
1	ID	Name	Type	Description	Expression Context
2	ADAE.ADURN	Algorithm to derive ADAE.ADU	Computation	Convert SUPPAE.QVAL to numeric value where SUPPAE.QNAM="AEDURDD"	
3	ADAE.ADURU	Algorithm to derive ADAE.ADU	Computation	SUPPAE.QVAL where SUPPAE.QNAM="AEDURDDU"	
4	ADAE.AECLINT	Algorithm to derive ADAE.AEC	Computation	SUPPAE.QVAL where SUPPAE.QNAM="AECLINT"	
5	ADAE.AEDOSDUR	Algorithm to derive ADAE.AED	Computation	SUPPAE.QVAL where SUPPAE.QNAM="AEDOSDUR"	
6	ADAE.AEDURDD	Algorithm to derive ADAE.AED	Computation	Convert SUPPAE.QVAL to numeric value where SUPPAE.QNAM="AEDURDD"	

Figure 2. Screenshot of P21 Define Version of Excel File "METHODS" Tab

Assign P21 Excel input file path and name, using LIBNAME statement. Convert the individual sheets from excel into datasets, removing any duplicates.

```
options validvarname=any;
libname p21spec XLSX "c:\Documents\p21spec.xlsx";
 proc contents data = p21spec._all_
               out = myds noprint;
  run;
  proc sort data = myds
            out = myds1(keep = memname) nodupkey;
      by memname;
  run;
```

Then execute the following data _NULL_ step to read work datasets one by one in alphabetical order, if any dataset name contains a space, it will be removed. The METHODS dataset resulting from this will be used further in Macro section 1.

```
data _null_;
    set myds1;
    call execute("data "
                || compress(memname)
                || ";  set p21spec.'"
                || strip(memname)
                || "'n; run;") ;
run;
```

MACRO SECTION 1: SEARCH FOR WORDS SUPP AND QNAM IN METHODS DATASET

With either of the two options above, we have created an interim dataset METHODS. Proceed with searching the key words "QNAM" and "SUPP", originating from METHODS tab from the define document. It could be little tricky to remove non-applicable text from the "description" variable, and only keep the required QNAM related text. For example, if an ADaM variable refers to multiple SUPPXX domains, it must extract multiple occurrences of SUPPXX and corresponding QNAM. Code below can be used to handle this scenario.

```
data method2;
  length dsname $15 var1 $900;
   set methods;
   s='QNAM';
   y=0;
   n=count(upcase(description),'QNAM');
   if index(upcase(description),"QNAM")>0 then
        do;
             /*Search keyword QNAM from right to left */
        do i=1 to n until(y=0);
             p1=find(description,s,y+1);
              if findw(upcase(description),"QNAM",) >0 or
                  findw(upcase(description),"QNAM IN")=0 then
               do;
                   var1=substr(description,p1,55);
                    suppdom=substr(description,p1-8,100);
               end;
                   if var1 eq ' ' and
```

```
                    findw(upcase(description),"QNAM IN")>0 then
                  do;
                    pos1=find(description,'(');
                    pos2=find(description,')');
                    if pos1 ne ' ' then
                      var1=substr(description,pos1-8,pos2);
                  end;
          /* Repeat this step if there are multiple SUPPXX domains */
                     dsre=prxparse('/SUPP\w\w/');
                     dspos=prxmatch(dsre,description);
                     dsname=substr(description,dspos,6);
                output;
         /* Use similar do loop for handling left to right search
           to ensure all multiple occurrences of SUPXX/QNAM are
          included */
              end;
        end;
   keep description var1 dsname n;
run;
```

MACRO SECTION 2: CREATE DATASET WITH CLEAN LIST OF SUPPXX AND QNAM NAMES

The following code removes the line feeds and any special characters, keeps the SUPPXX name (created in the output dataset above), extracts corresponding variable QNAM, then assigns those to new variables DSNAME and QNAM (screenshot shown in **Figure 3**).

Sometimes the input file may have line feeds('0A'x), either remove the line feeds or replace with a special character using TRANWRD function to facilitate special character handling.

Example: replace line feed with ~:

```
var1=tranwrd(description,'0A'x,'~');
```

```
data qnam;
   length _qnam _qnam1 _qnam2 _qnam3 _qnam4 $500;
   set method2;
        if findw(upcase(var1),"QNAM")>0 or findw(upcase(var1),"QNAM
        IN")=0 then do;
         if findw(upcase(var1),"QNAM IN")=0 then
                do;
                 _qnam=strip(scan(var1,2,'= ".,>0,.;'));
                end;
            end;
        if findw(upcase(var1),"QNAM IN")>0 then
            do;
               _qnam=scan(var1,2,'()');
            end;
        _qnam3=compress(_qnam,"""';:)(");
        _qnam4=scan(_qnam3,1,'~');
        qnam_tmp=strip(_qnam4);
        keep dsname qnam_tmp;
   run;
```

The code above gives us the QNAM list separated by commas and needs to be re-arranged vertically so that it can be compared with SDTM QNAM list later.

Example: parse comma separated values in a loop to restructure the horizontal values to vertical.

```
do i=1 by 1 until (qnam=' ');
     qnam = strip(scan(qnam_tmp, i, ','));
     if qnam > ' ' then output;
end;
```

	dsname	qnam
1	SUPPAE	ACN1
2	SUPPAE	ACN2
3	SUPPAE	ACN3
4	SUPPAE	ACN4
5	SUPPAE	AECLINT
6	SUPPAE	AEDURDD
7	SUPPAE	AEDURDDU
8	SUPPAE	AEEPRELI
9	SUPPAE	ELEMENT
10	SUPPAE	REL1
11	SUPPAE	REL2
12	SUPPAE	REL3
13	SUPPAE	REL4
14	SUPPAE	SM1
15	SUPPAE	SM2
16	SUPPAE	SM3
17	SUPPAE	SM4
18	SUPPAE	SPDYRLEP
19	SUPPAE	STDYRLEP
20	SUPPCM	CMDECOD1

Figure 3. Screenshot of Output List of SUPPXX Domain Names with Corresponding QNAM Variable Names

MACRO SECTION 3: VERIFY SUPPXX/QNAM LIST FROM DEFINE WITH CORRESPONDING SDTM SUPPXX DOMAINS

Once we have the list of SUPPXX domain names with corresponding QNAM variable names from define document, we proceed with the following code, which will compare that list with SDTM domains to determine whether each of those QNAM variables exist in corresponding SUPPXX domain. If there are any mismatches, it will be output into an excel file (screenshot shown in **Figure 4**). This code also retrieves corresponding QLABEL variable and adds it to the final dataset.

Example: create a count macro variable to assign number of datasets.

```
proc sql noprint;
     select count(distinct dsname) into:n_suppds
     from qnam
quit;
```

Example: loop through all the SDTM datasets, and corresponding SUPPXX list of datasets from Macro section2.

```
libname sdtmlib 'c:\documents\sdtmfolder';

%let libsdtm=sdtmlib;

%do i=1 %to &n_suppds;

%put &&supp_dsname_&i.;
```

```
    proc sql;
     create table work.SUPPXX_&i. as
        select distinct b.dsname, a.qnam, a.qlabel
           from &libsdtm..&&supp_dsname_&i. as a left join qnam as b
         on a.qnam  = b.qnam
            where b.dsname="&&supp_dsname_&i."
         order by dsname;
    quit;
  %end;
```

Create an excel file with list of QNAM values found in define document but not in SUPPXX domains.

```
    proc export data=qnam_data
            outfile="c:\documents\qnam0not0data.xlsx"
           dbms=xlsx replace;
           sheet="qnam_not_data";
    run;
```

	A	B	C	D	E	F	G
1	DSNAME	QNAM					
2	SUPPCM	WCLSCDxx					
3	SUPPCM	WCLSxx					
4	SUPPDM	RACE3					
5	SUPPMH	MHLLT					
6							
7							
8							
9							
10							
11							
12							

Figure 4. Screenshot of Output Excel File with Any Mismatches.

MACRO SECTION 4: CREATE OUTPUT RTF FILE WITH LIST OF SUPPXX, QNAM AND QLABEL

The following code creates an RTF file with a list of SUPPXX domain with QNAM and QLABEL. The output list is sorted in alphabetical order, with each SUPPXX name appearing on a separate page, so that it can be easily inserted into cSDRG section 3.4 (example screenshot of RTF output file in **Figure 5**). Use PROC TEMPLATE to generate the RTF file, with customized page layout/font to match the PHUSE cSDRG template.

```
   data rtf0file;
     set qnam;
        attrib dsname label="Domain"
        qnam   label="QNAM"
     qlabel label="Description";
   run;

proc template;
  define style Styles.Custom;
```

```
 parent = Styles.RTF;
 replace fonts /
 'TitleFont' = ("Times New Roman",13pt,Italic)/*Titles statements*/
 'TitleFont2' = ("Times New Roman",12pt, Italic) /* Subtitle */
 'StrongFont' = ("Times New Roman",10pt)
 'EmphasisFont' = ("Times New Roman",10pt)
 'headingEmphasisFont' = ("Times New Roman",11pt)
 'headingFont' = ("Times New Roman",11pt,Bold)
 'docFont' = ("Times New Roman",11pt) /* Data in table cells */
 'footFont' = ("Times New Roman",13pt) /* Footnotes statements */
 'FixedEmphasisFont' = ("Courier",9pt,Italic)
 'FixedStrongFont' = ("Courier",9pt)
 'FixedHeadingFont' = ("Courier",9pt)
 'BatchFixedFont' = ("Courier",6.7pt)
 'FixedFont' = ("Courier",9pt);
 replace Body from Document /
  bottommargin = 1in
  topmargin = 1in
  rightmargin = 1in
  leftmargin = 1in;
  replace Table from Output /
  frame = box
  rules = all /* internal borders: none, all, cols, rows, groups */
  cellpadding = 0pt
  outputwidth=100%
  cellspacing = 0pt /*the space between table cells */
  borderwidth = .75pt /* the width of the borders */;
 style SystemFooter from SystemFooter /
  font = fonts("footFont");
end;
run;

proc sort data=rtf0file;
 by dsname;
run;

%let rtf_file='c:\documents\suppxx0qnam0list.rtf';
options orientation=portrait nodate;
ods rtf file="&rtf_file" style=Custom ;

 proc print data=rtf0file noobs label;
  by dsname;
  pageby dsname;
 run;

ods rtf close;
```

Domain=SUPPAE

QNAM	Description
ACN1	Action Taken with SM1
ACN2	Action Taken with SM2
AECLINT	Clinical Interest
AEDURDD	Duration of Adverse Event Diff of Dates
AEDURDDU	Duration Units
AEEPRELI	Epi/Pandemic Related Indicator
ELEMENT	Description of Element
REL1	Causality SM1
REL2	Causality SM2
SM1	Study Medication 1
SM2	Study Medication 2
SPDYRLEP	Stop Day Rel to Epoch
STDYRLEP	Start Day Rel to Epoch

Figure 5. Screenshot of RTF ODS Output of SUPPAE as an Example.

INSERT RTF TABLE INTO CSDRG SECTION 3.4

Finally, the output generated from the macro program (one table for each SUPPXX domain) can be copied directly into cSDRG section 3.4 (screenshot shown in **Figure 6**).

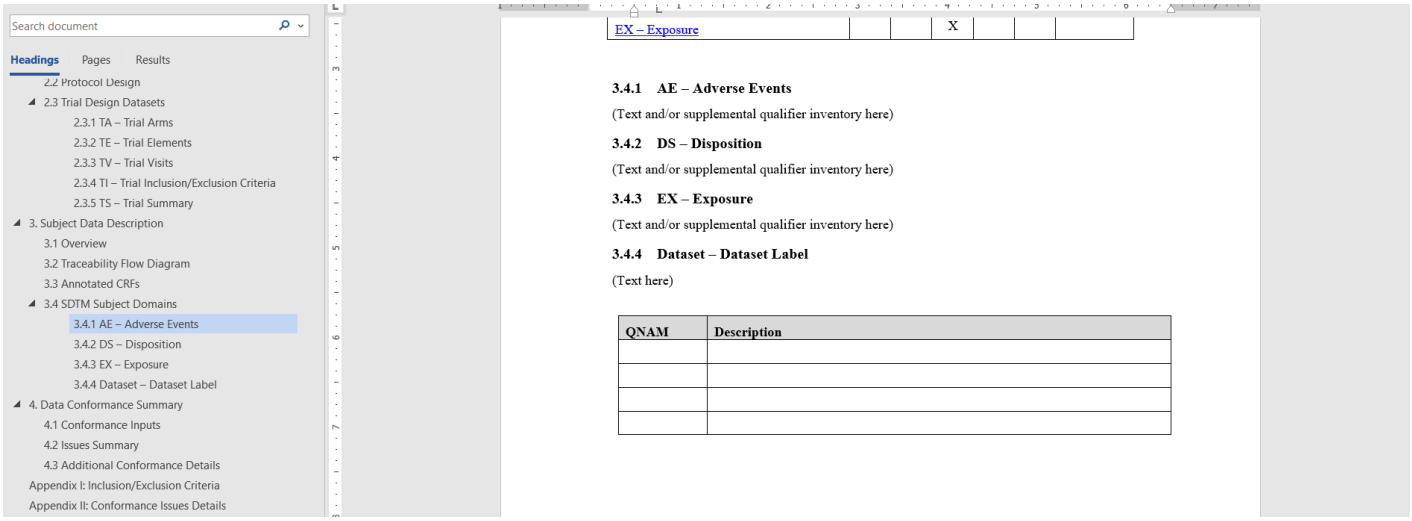


Figure 6. Screenshot of PHUSE cSDRG Template Section 3.4

CONCLUSION

This paper illustrates automating the process of identifying the supplemental qualifier and corresponding variables from define document, verifying those against SDTM domains, and using the output generated by the macro program, to directly update the cSDRG section 3.4. This effort significantly improves quality and efficiency, when creating the submission package.

REFERENCES

https://support.sas.com/en/documentation.html

PHUSE template can be found at

https://advance.phuse.global/display/WEL/Clinical+Study+Data+Reviewer%27s+Guide+%28cSDRG%29+Package

Pinnacle 21 tool website

https://www.pinnacle21.com/

Lauren, Haworth. 2004. " SAS with Style: Creating your own ODS Style Template for RTF Output". Proceedings of PharmaSUG 20004, paper HW04.

Jeff, Xia. 2021. "A SAS Macro to Generate Slim Version of Define for RTOR". Proceedings of PharmaSUG 2021, paper AP-053.

ACKNOWLEDGMENTS

The authors would like to thank Jeff Xia for his great suggestions and valuable input to this paper.

CONTACT INFORMATION

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

Jamuna Purma
Merck & Co., Inc.,
Email: jamuna.purma@merck.com
Web: www.merck.com

Bindya Vaswani
Merck & Co., Inc.,
Email: bindya.vaswani@merck.com
Web: www.merck.com