PharmaSUG 2015 - Paper AD05

Have SAS Annotate your Blank CRF for you! Plus dynamically add color and style to your annotations.

Steven Black, Agility-Clinical Inc., Carlsbad, CA

ABSTRACT

Creating the BlankCRF.pdf annotations by hand is an arduous process and can take many hours of precious time. Once again SAS comes to the rescue! Using the SDTM specifications data we can have SAS create all of the annotations needed and place these on the appropriate page within the BlankCRF.pdf. In addition we can dynamically color and adjust the font size of these annotations.

This process uses SAS, Adobe Acrobat's FDF language, and Adobe Reader to complete the process.

In this paper I will go through each of the steps needed and provide detailed explanations into exactly how to accomplish each task.

INTRODUCTION

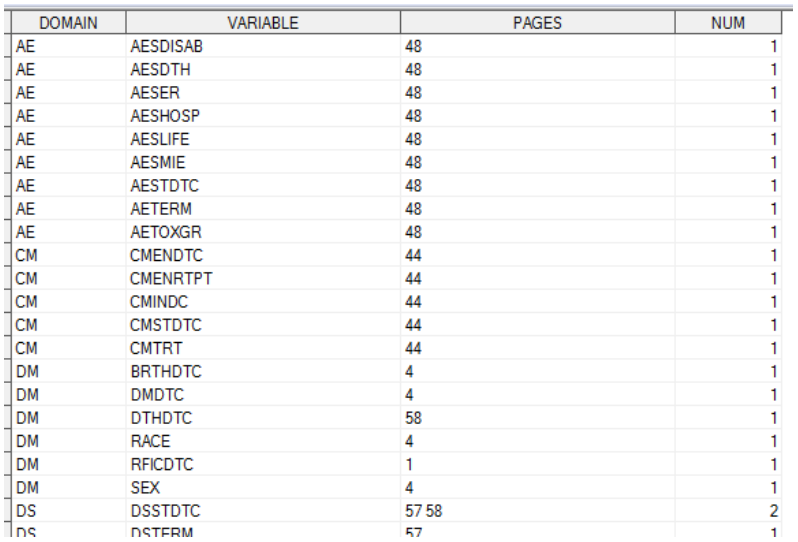
There have been a few papers already published on this topic, I thought that providing some additional details on how I accomplished this task may be of use to others. This paper will separate the tasks into 7 steps. Although I do not provide the full code used, I do provide enough per each step that replicating the work should not be too onerous.

STEP ONE: A SEMI-SETUP SDTM SPECIFICATION FOR EACH DOMAIN NEEDED

From my experience most companies use MS Excel to hold their SDTM specifications and it is generally assumed that each of the various SDTM domains are held in their own respective tab within the Excel workbook.
For each domain I only need to use the columns: Variable Name, and Origin. In the excel workbook I use the tabs: table of contents (TOC), Value level data and Supplemental (SUPP) data (when applicable), and any domains that are non-trial specific (DM, MH, etc.). I'm only interested in those variables that have a specific CRF page assigned to them in the Origins column. Essentially even a very simple SDTM specification document will work for this process.

STEP TWO: IMPORTING THE SDTM DATA AND GETTING THINGS LINED UP.

I first import the table of contents (TOC) sheet to allow for a more dynamic approach to know which domains have been used within the spreadsheet. Once all data from all the specified domains has been imported, I set all the data into one dataset and have a simple dataset the looks like this:

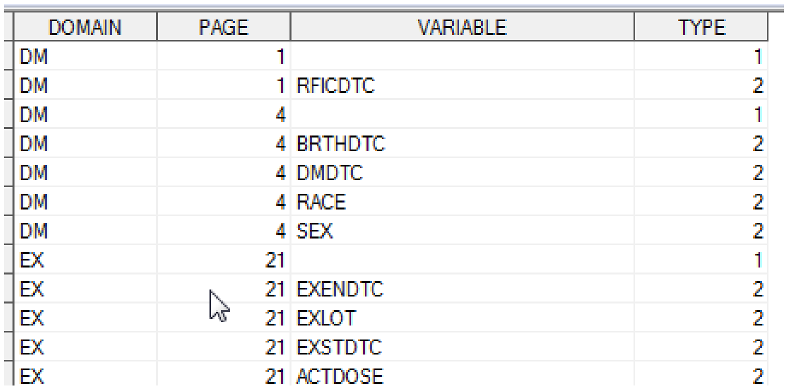


DOMAIN	VARIABLE	PAGES	NUM
AE	AESDISAB	48	1
AE	AESDTH	48	1
AE	AESER	48	1
AE	AESHOSP	48	1
AE	AESLIFE	48	1
AE	AESMIE	48	1
AE	AESTDTC	48	1
AE	AETERM	48	1
AE	AETOXGR	48	1
CM	CMENDTC	44	1
CM	CMENRTPT	44	1
CM	CMINDC	44	1
CM	CMSTDTC	44	1
CM	CMTRT	44	1
DM	BRTHDTC	4	1
DM	DMDTC	4	1
DM	DTHDTC	58	1
DM	RACE	4	1
DM	RFICDTC	1	1
DM	SEX	4	1
DS	DSSTDTC	57 58	2
DS	DSTERM	57	1

This dataset contains the variables: Domain, Variable, Pages, and Num. The variable Pages equals the page numbers noted in the origins column after having removed all other character text. The variable Num is the number of pages noted per each variable - created using the COUNTC function counting the spaces and adding 1. Data is limited to only those with a page number noted.

I then create a record per each page noted so that in the example above the variable DSSTDTC will have two records one for page 57 and one for page 58. I then created a new variable called page with its respective page number (code provided in appendix).

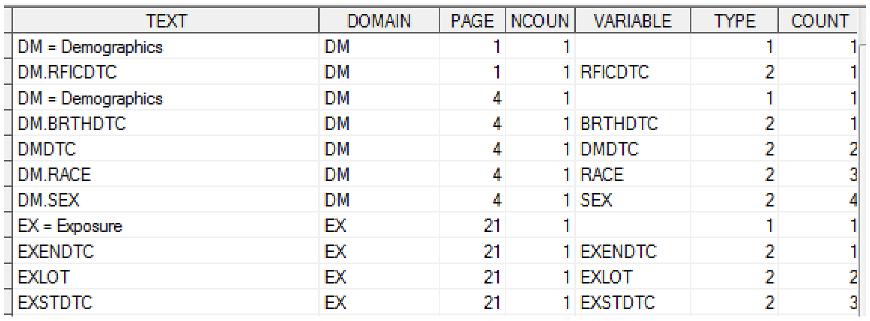
I then create a new dataset which contains (1) the distinct domains captured per each page and (2) the variable data per page.



DOMAIN	PAGE	VARIABLE	TYPE
DM	1		1
DM	1	RFICDTC	2
DM	4		1
DM	4	BRTHDTC	2
DM	4	DMDTC	2
DM	4	RACE	2
DM	4	SEX	2
EX	21		1
EX	21	EXENDTC	2
EX	21	EXLOT	2
EX	21	EXSTDTC	2
EX	21	ACTDOSE	2

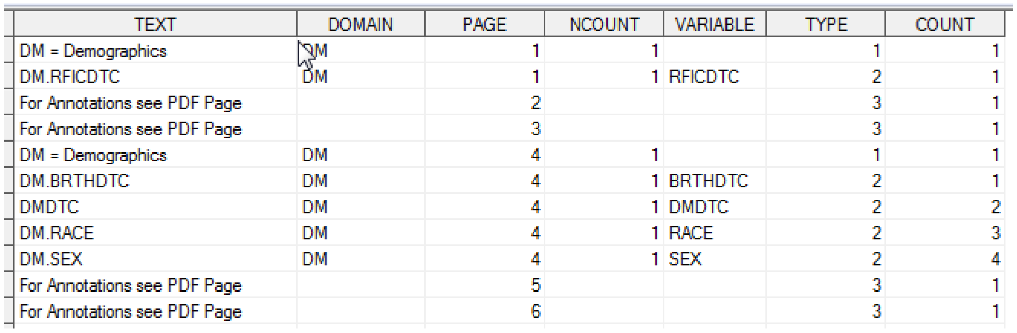
In this new data I have also created a variable type which marks if the row is a domain header or variable data. I also drop any unneeded variables.

Next I count the number of variables per page and the number of domains per page, I also create the text to be used in the annotations. For those with type 1, I use the description found in the TOC document per each domain. For those with type 2, I adjust the values from the DM domain to contain the text 'DM.' if the value of the variable does not already have the DM prefix (i.e. RACE becomes DM.RACE). Otherwise I create the text as the variable data (code provided in appendix).



TEXT	DOMAIN	PAGE	NCOUN	VARIABLE	TYPE	COUNT
DM = Demographics	DM	1	1		1	1
DM.RFICDTC	DM	1	1	RFICDTC	2	1
DM = Demographics	DM	4	1		1	1
DM.BRTHDTC	DM	4	1	BRTHDTC	2	1
DMDTC	DM	4	1	DMDTC	2	2
DM.RACE	DM	4	1	RACE	2	3
DM.SEX	DM	4	1	SEX	2	4
EX = Exposure	EX	21	1		1	1
EXENDTC	EX	21	1	EXENDTC	2	1
EXLOT	EX	21	1	EXLOT	2	2
EXSTDTC	EX	21	1	EXSTDTC	2	3

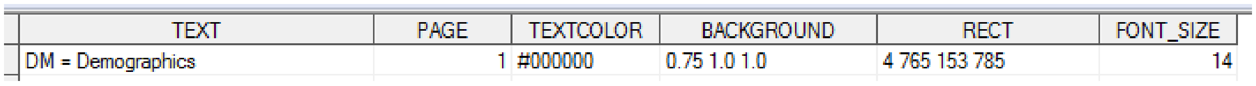
Lastly, I know in my final output I'll need an annotation per each page in the BlankCRF.pdf so I create a dummy dataset with page variable equal to 1 to 150. I then merge this onto the other data. I then reset the variable TYPE to equal 3 and set these blank text rows to 'For Annotations see PDF Page '. The exact reference page would be hand entered later once annotations have been placed on the page. The max number (150) can be greater than the number of pages in the BlankCRF.pdf. Annotations that do not match a specific page on the BlankCRF.pdf will be ignored, so having more annotations than pages is fine. I've set this max number to be a macro parameter for easy modification.



TEXT	DOMAIN	PAGE	NCOUNT	VARIABLE	TYPE	COUNT
DM = Demographics	DM	1	1		1	1
DM.RFICDTC	DM	1	1	RFICDTC	2	1
For Annotations see PDF Page		2			3	1
For Annotations see PDF Page		3			3	1
DM = Demographics	DM	4	1		1	1
DM.BRTHDTC	DM	4	1	BRTHDTC	2	1
DMDTC	DM	4	1	DMDTC	2	2
DM.RACE	DM	4	1	RACE	2	3
DM.SEX	DM	4	1	SEX	2	4
For Annotations see PDF Page		5			3	1
For Annotations see PDF Page		6			3	1

STEP THREE: COLORS AND RECTANGLES.

It is easier to understand creating an annotation while looking at a single complete annotation. So in this single example we have several variables: Text, Page Number, Text Font Color, Background Color, Rectangle Size, and Font Size.



TEXT	PAGE	TEXTCOLOR	BACKGROUND	RECT	FONT_SIZE
DM = Demographics	1	#000000	0.75 1.0 1.0	4 765 153 785	14

Text/Font Color

I use two text colors in creating the annotations, #000000 (black) for the domain headers and #FF0000 (red) for the variable names. Font colors use Hex Coloring schemes.

Background Color

Background colors are setup as Latex colors which can be hard to interpret but a handy reference for both the Hex and Latex color assignment is http://latexcolor.com/. In our example we use (0.75 1.0 1.0) which is a light blue color. I use around 10 different colors so I can color each of the domains a different color. Here is quick list of my color choices.

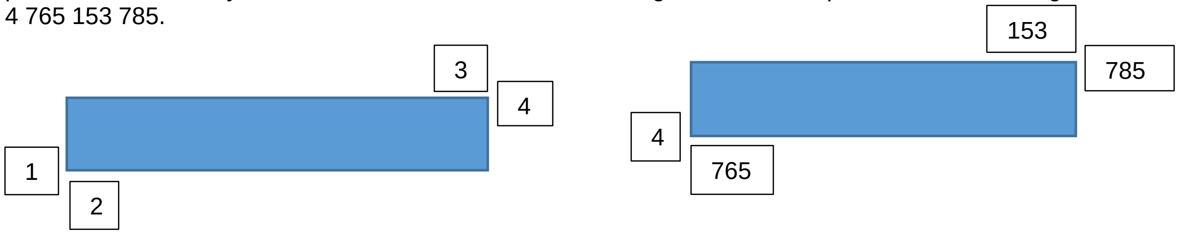
1.0 1.0 0.0 - yellow
1.0 1.0 0.66 - light yellow
0.66 0.75 1.0 - purple
1.0 0.75 0.0 - orange
0.0 0.75 1.0 - blue
0.75 1.0 0.75 - light green
0.75 1.0 1.0 - light blue
1.0 0.66 0.75 - pink
1.0 0.75 0.66 - light orange

Font Size

I use a larger font size (14) for the domain headers and a smaller size (11) for variable labels. I use a macro variable to dynamically set the base font size and increase the font for the headers.

Rectangles

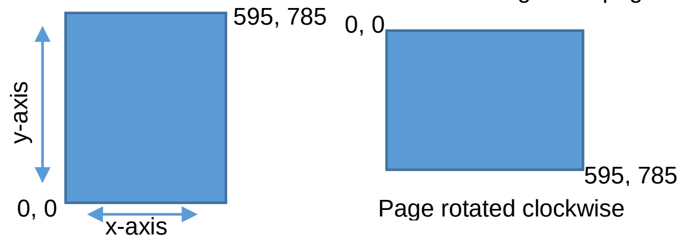
Creating the rectangle is the most complicated piece of code. The rectangle is the size and location of annotation box which holds the annotation text. There are four numbers associated with the rectangle, each contains the position of the x and y axis of two of the corners of the rectangle. In this example we have a rectangle coordinates of 4 765 153 785.



(1 - lower x-axis position, 2 - lower y-axis position, 3 - upper x-axis position, 4 - upper y-axis position)

How the coordinates fit on a pdf page.

A pdf document begins in the lower left hand size with the position 0, 0 while in portrait orientation, when rotated 90° the orientation starting point also rotates. I have not needed to take this rotation into consideration but may need to in the future with an additional macro parameters noting those pages which are set to landscape position.



The max x-axis position is 595 and max y-axis position can vary between 843 and 785. There is a little trial and error regarding top placement, so I built this in as a macro parameter. 300 is essentially the middle of the x-axis position in document page. A rectangle height of 20 is generally large enough to hold Arial font size 12-14 and a height of 16 fits the 9-11 sized Arial font.

The length of the rectangle will depend upon the size of the text and font type used. This can be a bit tricky but I've created a simple formula using font size and text length to create a constant variable to multiply against the text length using Arial font type.

```
font_adjust=&font_size-10;
txt_l=length(strip(text));
if txt_l <6 then constant=10+font_adjust;
else if txt_l <10 then constant=9+font_adjust;
else if txt_l <23 then constant=8+font_adjust;
else if txt_l <36 then constant=7+font_adjust;
else constant=6+font_adjust;
```

Then for the length of the rectangle I use txt_l*constant.

So for my rectangle I want to start just over from the left side of the page and at the top. So I set lower x-axis position to 4, then for my lower y-axis position I set to top page limit (785) - 20 = (765), my upper x-axis position is set to the text length (17)* constant (9) = (153), finally my upper y-axis position is set to the top page limit (785).

I the dataset I have two types of data domain headers and variable data. I have it set up so that the domain header move from left to right from the top of the page and then down. The variable data is located after the domain header data from top to bottom. This can be accomplished using the retain statement hold the previous values and then beginning the next rectangle a few places over from the last (code shown in appendix).

STEP FOUR: LEARNING FDF CODE

Adobe Acrobat has essentially created it's own language called fdf. Luckily with just a bit of help, learning the ins and outs of fdf is not too difficult. It helps that, if needed, we can take a pdf annotation and export it and see what it looks like in fdf code.

For creating annotations, fdf code requires that we provide an opening wrapper code similar to html code, second we create a place holder for each annotation we are going to be creating. Third we create each annotation. Fourth we provide a closer wrapper to the file.

Creating the code:

Within SAS we create the code as a text string within the dataset outputting the code as we go along.

```
*** create the code ***;
data _coding;length code $1000;

*** begin code ***;
code='%FDF-1.2'; output;
code="1 0 obj<</FDF<</Annots["; output;
```

In this step we create reference numbers per ach of the annotations, these number at set to the number of annotations +1, so out first annotation will be 2. For subsequent annotations we would repeat the process. We then end this section.

```
code="2 0 R"; output;
code="]>>/Type/Catalog>>endobj"; output;
```

Now we create the annotation using the reference number we created above, placing in our background color, text, font type, size, justification, and color, page number and rectangle area. We then close the annotation.

```
 *** loop through number of annotations to be created ***;
 code="2 0 obj<<"; output;                     * open object 2 *;
 code="/C[0.75 1.0 1.0]"; output;              * background color *;
 code="/Contents(DM = Demographics)"; output;  * text of annotation *;
 code="/DS(font: italic bold Arial,sans-serif 14.0pt; text-align:left;
color:#000000)"; output;                       * font type size alignment and color *;
 code="/F 4/Page 0"; output;                   * page number *;
 code="/Rect[4 765 153 785]"; output;          * rectangle area *;
 code="/Subj(Text Box)/Subtype/FreeText/Type/Annot>>endobj"; output; * set annotation
type and close object 2 *;
```

Finally we close the file.

```
*** end code ***;
code="trailer"; output;
code="<</Root 1 0 R>>"; output;
code='%%EOF'; output;

run;
```

In this example we created one annotation. In order to create all the needed annotations I loop through each record using a macro and macrotize the data using PROC SQL (code shown in the appendix). As a note the page number is set to one page lower than noted on the SDTM specifications. This is due to an auto-correction that is made when importing the data into the BlankCRF.pdf.

STEP FIVE: CREATE THE FDF CODE FILE

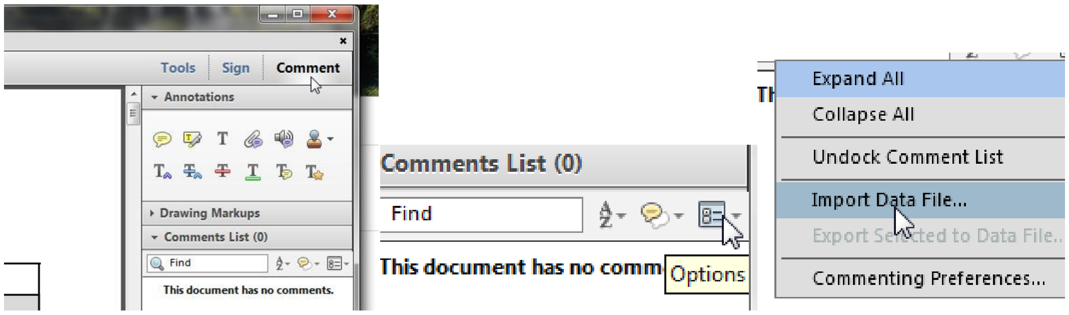
To create the final code to be used for the annotations I use a FILE statement removing any blank rows.

```
data _null_ ;
set _coding (keep=code where=(code ne ''));
FILE  "output_file_location.fdf";
PUT code;
run ;
```

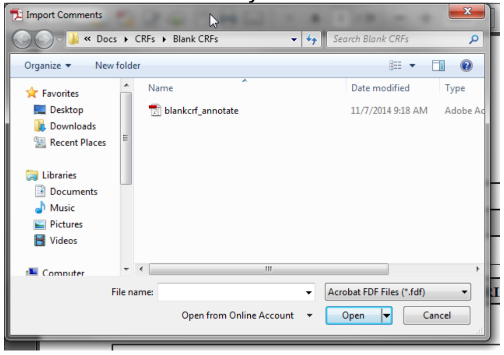
I've created the whole process in the macro with a limited number of parameters these include: name and location of STDM specifications file, name and location of FDF code file, font size wanted, adjustable area of top page, and max number of pages in the Blank CRFs (default set to 150).

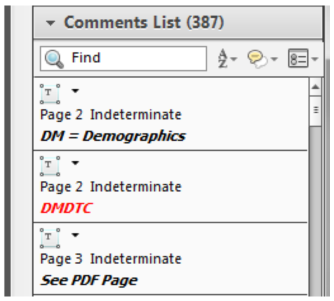
STEP SIX: IMPORT THE FDF FILE INTO ADOBE READER AND BE AMAZED.

Open the Blank CRFs document in Adobe Reader (Version 11 was used for this paper). Click the 'Comment' option, this will create the comment tool sidebar. Once this is available click the small box icon for Options. Select "Import Data File…" option.



Then browse to where you have saved the FDF file. Select the file and click open.





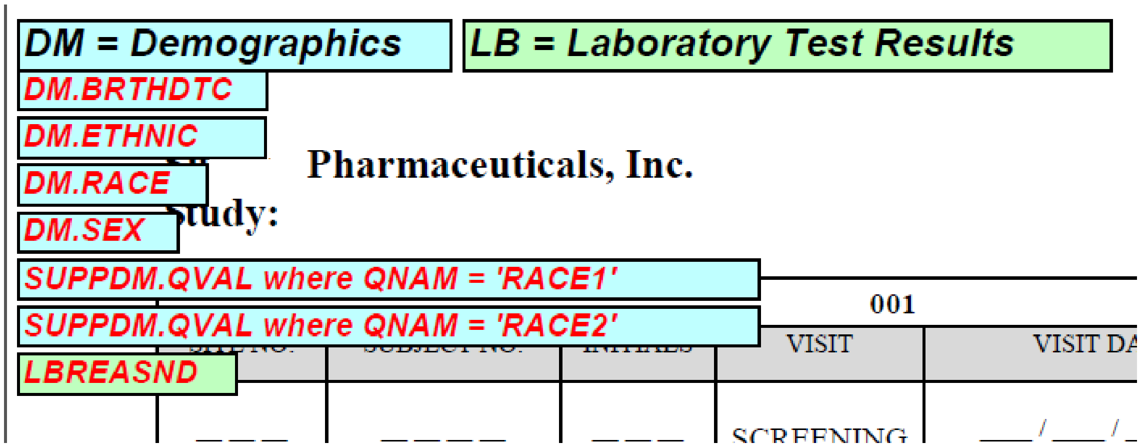
Page 2 Indeterminate
DM = Demographics

Page 2 Indeterminate
DMDTC

Page 3 Indeterminate
See PDF Page

This will then import all the annotations into the BlankCRF.pdf document and place them on the respective page. The annotations will also appear in the comments box.

On those pages with annotations the headers will appear first followed by the variables per each header. The Coloring of the domain will occur in the order specified in the color format.



Once done save the pdf with a new name. Time will still need to be taken to move the variable list near the respective variable(s) in the pdf, but using SAS to create the annotations and the ease of importing them saves all the time it would have taken to add each annotation to the page and ensures consistency with all annotations created.

Additional tinkering may need to occur to get everything perfect, such as modifying the top page variable and the spacing of the rectangles but with this information and code from this paper it should be very manageable. If needed simply undo the import of the annotations and re-import with the updated file once the modifications are completed.

SUMMARY

Using SAS to create the annotations for the BlankCRFs.pdf can dramatically decrease the time it take to complete the BlankCRFs. In this paper I illustrated how to set up the SDTM data, create the annotation text and code, and finally how to import these annotations in the BlankCRF.pdf document. I also provide some of the more granular information regarding coloring and sizing the annotations. I hope that the paper is found useful for other who want to further use SAS to accomplish great things. A basic macro template is provided in the appendix.

REFERENCES

Spruck, Dirk & Kawohl, Monika 2004 “Using SAS to Speed up Annotating Case Report Forms in PDF Format” Proceedings of the 2004 Pharmaceutical SAS Users Group. Cary, NC: SAS Institute Inc. www.lexjansen.com/pharmasug/2004/coderscorner/cc02.pdf

CONTACT INFORMATION

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

Name: Steve Black
Enterprise: Agility-Clinical Inc.
Address: 6005 Hidden Valley Rd.
City, State ZIP: Carlsbad CA, 92011
Work Phone: 760-658-5919
E-mail: steven.c.black@gmail.com

APPENDIX

Included is a basic macro template but the importing of SDTM data removed, as the SDTM specifications document can vary wildly between companies, but the resulting data is easy to standardize to fit the needs of the macro.

```
*** begin macro call with parameters (default set for font size, top of the page and
the max page, if missing) ***;
%macro pdf_annotate (stdm_file,output_file,font_size=11,top_page=800,max_page=150);


*** IMPORTING OF DATA NOT INCLUDED IN THIS CODE ***;
*** each domain dataset has a _ prefix. ***;
*** variables needed pages, domain, and variable ***;

*** The domain and description format is created from the table of contents data ***;
proc sql noprint;
select Dataset, Description into: domain1 -:domain999,:descrip1- :descrip999
from _toc_data;
%let numdom=&sqlobs;
quit;

*** create format for domains based off of TOC data ***;
proc format;
value $descs
%do x = 1 %to &numdom;
"&&domain&x"="&&descrip&x"
%end;
;

*** create formats for background colors (see above for translation) ***;
value colors
1='0.75 1.0 1.0'
2='1.0 1.0 0.66'
3='1.0 0.75 0.66'
4='1.0 0.66 0.75'
5='0.66 0.75 1.0'
6='1.0 1.0 0.0'
7='0.0 0.75 1.0'
8='1.0 0.75 0.0'
9='0.75 1.0 0.75'
;

run;

*** set in all datasets (limit data to where domain ne missing) ***;
data _allin;
set _:;
if domain ne '';

*** create pages and num (the number of different pages mentioned in the document)
***;
pages=strip(compbl(tranwrd(pages,',',' ')));
num=countc(trim(pages),' ')+1;
run;

*** macrotize max number of pages ***;
proc sql noprint;
select max(num) into: maxpages
from _allin;
quit;

*** set in _allin data the number of time per &maxpages using a limiting statement to
only select the data needed ***;
```

```
data _body;
set %do x =1 %to &maxpages;
_allin (in=x&x where=(num>=&x))
%end;
;

*** reset num to the set in occurance number ***;
%do x = 1 %to &maxpages;
if x&x then num=&x;
%end;

*** create the true page number per annotation ***;
page=input(scan(pages,num,' '),best.);

run;

*** sort body data removing duplicate annotations ***;
proc sort data=_body nodupkey;
by page domain variable;
run;

*** create header data per page and domain ***;
proc sort data=_body out=_headers (keep=domain page) nodupkey;
by page domain;
run;

*** bring in header and body data into one dataset ***;
data _whole;
set _headers (in=a) _body (in=b);
if a then type=1;
if b then type=2;

drop num pages;
run;

*** sort by page and type ***;
proc sort data=_whole;
by page type;
run;

*** create count of data per page and type ***;
data _whole_count;
set _whole;
count + 1;
by page type;
if first.type then count = 1;
run;

*** re-sort data for merging ***;
proc sort data=_whole_count;
by page domain;
run;

*** needed to ensure coloring of header and body annotation is the same ***;
*** merging the header data back onto the whole data and creating a new count variable
***;
*** this count variable becomes the coloring scheme ***;
data _prep; length text $300;
merge _whole_count (keep=domain page count type
where=(old_type=1) rename=(type=old_type count=ncount))
_whole_count;
by page domain;
```

```
*** modify the text presented in the annotation based on type ***;
*** fix DM variables to include DM. if missing ***;
if type=1 then text=strip(compbl(domain||' = '||put(domain,$descs.)));
else if type=2 then do;
       if upcase(domain)='DM' and strip(substr(upcase(variable),1,2)) not ='DM' then
text='DM.'||strip(variable);
       else text=strip(variable);
end;

drop old_type;
run;

*** re-sort by page and type ***;
proc sort data=_prep;
by page type;
where page ne .;
run;

data _template;
do page = 2 to &max_page;
output;end;
run;

data _prep2;
merge _prep (in=a) _template (in=b);
by page;
if b and not a then do;
type=3;
text='For Annotations see PDF Page ';
count=1;
end;

run;

*** data to create the boxes for annotations ***;
data _rectangles;
set _prep2;length lrectx lrecty urectx urecty $5 textcolor $10 background $15 rect
$50;
by page;

*** hold values of these variables ***;
retain h_x_adjust h_y_adjust b_x_adjust b_y_adjust;

*** adjust font size to be between 9-12  ***;
%if 8< &font_size >12 %then %let font_size=11;;

*** using the font size and length of text determine a constant value to create the
length of the box ***;
font_adjust=&font_size-10;
txt_l=length(strip(text));
if txt_l <6 then constant=10+font_adjust;
else if txt_l <10 then constant=9+font_adjust;
else if txt_l <23 then constant=8+font_adjust;
else if txt_l <36 then constant=7+font_adjust;
else constant=6+font_adjust;

*** create boxes for header annotations ***;
if type=1 then do;

font_size=&font_size+3;

       if count=1 then do;
              h_x_adjust=0;
```

```
             h_y_adjust=&top_page;
      end;

*** if the box is about to reach the end of page then drop and move down ***;
      if h_x_adjust > 500 then do;
             h_x_adjust=0;
             h_y_adjust=h_y_adjust-23;
             lrectx= put(h_x_adjust+4,5.0);
             lrecty= put(h_y_adjust-20,5.0);
             urectx= put(h_x_adjust+(txt_l*constant),5.0);
             urecty= put(h_y_adjust,5.0);
             h_x_adjust=input(urectx,best.);
      end;
      else do;
             lrectx= put(h_x_adjust+4,5.0);
             lrecty= put(h_y_adjust-20,5.0);
             urectx= put(h_x_adjust+(txt_l*constant),5.0);
             urecty= put(h_y_adjust,5.0);
             h_x_adjust=input(urectx,best.);
      end;

*** create rect, textcolor and background color variables ***;
      rect=strip(compbl(lrectx||lrecty||urectx||urecty));
      textcolor='#000000';
      background=put(count,colors.);
end;

*** create boxes for body annotations ***;
if type=2 then do;
font_size=&font_size;
      if count=1 then do;
             b_y_adjust=h_y_adjust-19;
             b_x_adjust=0;
      end;

*** if the annotations reach the bottom of the page then moves over to the top of page
***;
      if b_y_adjust < 20 then do;
             b_x_adjust=b_x_adjust+120;
             b_y_adjust=h_y_adjust-19;
             lrectx= put(b_x_adjust+4,5.0);
             lrecty= put(b_y_adjust-16,5.0);
             urectx= put(b_x_adjust+(txt_l*constant),5.0);
             urecty= put(b_y_adjust,5.0);
             b_y_adjust=input(lrecty,best.)-2;
      end;
      else do;
             lrectx= put(b_x_adjust+4,5.0);
             lrecty= put(b_y_adjust-16,5.0);
             urectx= put(b_x_adjust+(txt_l*constant),5.0);
             urecty= put(b_y_adjust,5.0);
             b_y_adjust=input(lrecty,best.)-2;
      end;

*** create rect, textcolor and background color variables ***;
      rect=strip(compbl(lrectx||lrecty||urectx||urecty));
      textcolor='#FF0000';
      background=put(ncount,colors.);
end;
if type=3 then do;
font_size=&font_size+3;
             h_x_adjust=246;
             h_y_adjust=&top_page;
             lrectx= put(h_x_adjust+4,5.0);
             lrecty= put(h_y_adjust-16,5.0);
             urectx= put(h_x_adjust+(txt_l*constant+20),5.0);
             urecty= put(h_y_adjust,5.0);
       rect=strip(compbl(lrectx||lrecty||urectx||urecty));
       textcolor='#FF0000';
       background=put(count,colors.);
end;

run;

*** mactotize all data to use in the coding process ***;
proc sql noprint;
select count(*) into: max
from _rectangles;
%let max=&max;
select text, page, rect, textcolor, background, font_size into:
txt1 -: txt&max,: pg1-: pg&max,: rect1 -: rect&max,: txtclr1 -: txtclr&max,: bkgnd1 -:
bkgnd&max,: fontsz1 -: fontsz&max
from _rectangles;
quit;

*** create the code ***;
data _coding;length code $1000;

*** begin code ***;
code='%FDF-1.2'; output;
code="1 0 obj<</FDF<</Annots["; output;

*** loop through number of annotations to be created ***;
%do x= 1 %to &max;
       code="%eval(&x+1) 0 R"; output;
%end;

code="]>>/Type/Catalog>>endobj"; output;

*** loop through number of annotations to be created ***;
%do x = 1 %to &max;
       code="%eval(&x+1) 0 obj<<"; output;
       code="/C[%trim(&&bkgnd&x)]"; output;
       code="/Contents(%str(&&txt&x))"; output;
       code="/DS(font: italic bold Arial,sans-serif &&fontsz&x...0pt; text-align:left;
color:%trim(&&txtclr&x) )"; output;
       code="/F 4/Page %trim(%eval(&&pg&x-1))"; output;
       code="/Rect[%trim(&&rect&x)]"; output;
       code="/Subj(Text Box)/Subtype/FreeText/Type/Annot>>endobj"; output;
%end;

*** end code ***;
code="trailer"; output;
code="<</Root 1 0 R>>"; output;
code='%%EOF'; output;

run;

*** outputs the final code as an fdf document ***;
data _null_ ;
set _coding (keep=code where=(code ne ''));
FILE  "&output_file..fdf";
PUT code;
run ;

*** ends pdf_annotate macro ***;
```

```
%mend;

%pdf_annotate (STDM_FILE =C:\STUDY_SDTM.xlsx, output_file = C:\blankcrf_annotate,
FONT_SIZE = 12,TOP_PAGE=785,MAX_PAGE=82);
```