PharmaSUG 2021 - Paper AD-036

Annotating CRFs More Efficiently

Noory Kim, Bhagyashree Shivakumar, Statistics & Data Corporation

ABSTRACT

Annotating a CRF entirely by hand can be tedious and time-consuming. Also, the result can be fragile: A change in the CRF, such as the addition or deletion of pages, can require a lot of rework (as it is not possible to drag and drop annotations between pages of a PDF file). How can we streamline our process and make its output more robust? This paper presents a way to update the metadata of CRF annotations iteratively, in which SAS® and SAS XML Mapper process and transfer the metadata between Excel and PDF files. The metadata can be adjusted either by resizing boxes within a PDF file or by replacing values in an Excel spreadsheet. This approach assumes users have access to SAS XML Mapper and a version of Adobe Acrobat that can import and export XFDF files. Knowledge of a programming language other than SAS is not required.

INTRODUCTION

The annotated case report form (aCRF) is a required component of the clinical data submission package sent to the FDA (Food and Drug Administration). It is a PDF (Portable Document Format) document bundled with the SDTM (Study Data Tabulation Model) data sets in the package.

Manually annotating a CRF can be time-consuming within a PDF editor (such as Adobe Acrobat) due to the inability to drag and drop an annotation between different pages. Also, if the CRF is updated, then the only manual option is to copy and paste between the file with the older version and the file with the newer version.

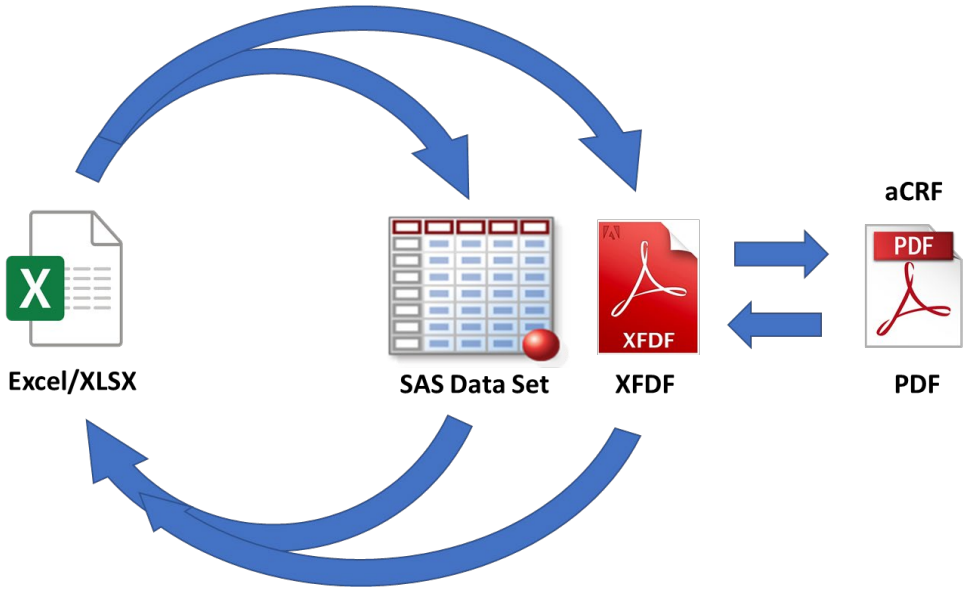


Figure 1: An Overview of Our Process. Our process stores annotations in an Excel file and converts them into an XFDF file. The XFDF file can then be imported into and exported from the CRF. In the PDF the annotations can be repositioned and resized by hand. The adjustments are merged back with a SAS data set containing a copy of the annotation data. The updated data is then exported back into Excel.

By extracting annotations out of the PDF file, we can store and update them within an Excel spreadsheet, where some tasks can be done more easily than in a PDF editor. Table 1 gives a rough comparison of the time needed to carry out tasks related to annotating a CRF.

While some of the ratings in Table 1 may be debatable, the takeaway message remains the same: Our process can save time spent on almost any task done in bulk. Moving annotations within the same page can still be done more easily within a PDF editor.

Task	Time Needed	
	PDF Editor Only	Our Process
Adding an annotation	⏳⏳	⏳⏳
Modifying the text of an annotation	⏳	⏳
Moving an annotation within the same page	⏳	⏳⏳
Moving an annotation to a different page	⏳⏳	⏳⏳
Resizing an annotation	⏳	⏳⏳
Adding annotations **in bulk**	⏳⏳⏳	⏳⏳
Modifying the text of annotations **in bulk**	⏳⏳⏳	⏳⏳
Moving annotations **in bulk** within the same page	⏳	⏳⏳
Moving annotations to different pages **in bulk**	⏳⏳⏳	⏳
Resizing annotations **in bulk**	⏳⏳⏳	⏳⏳
Transferring annotations from one draft of the CRF to the next **in bulk**	⏳⏳⏳	⏳⏳

Table 1: A Rough Comparison of Time. Key: one green sandglass (⏳) = Quick; two orange sandglasses (⏳⏳) = Moderate; three red sandglasses (⏳⏳⏳) = Slow. Our process saves the time needed for carrying out most tasks done in bulk. An exception is the task of moving annotations in bulk within the same page.

AN OUTLINE OF OUR PROCESS

The steps of the process are as follows:

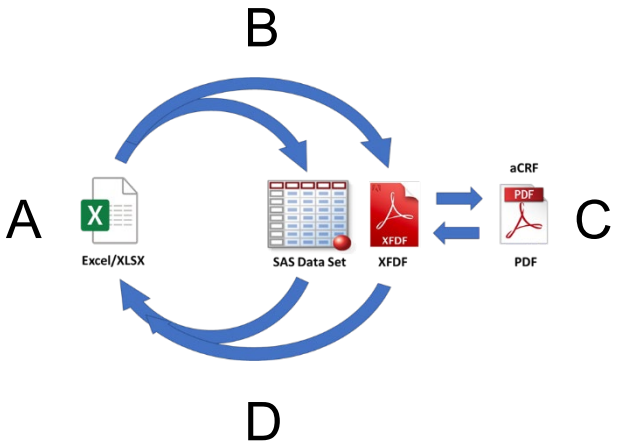
A. Add/modify annotations within Excel.

B. Transfer annotations from Excel (XSLX) to XFDF.

   1. Import the spreadsheet into SAS.
   2. Use SAS to output the annotations
      a. as an XFDF file, and
      b. as a SAS data set.

C. Transfer annotations from XFDF to and from a PDF file.

   1. Import the XFDF file into PDF.
   2. Adjust annotations within the PDF.
   3. Export XFDF out of PDF.



D. Transfer annotations from XFDF to Excel (XLSX).
    1. Use SAS XML Mapper to extract data out of an XFDF file and transfer them to SAS data sets.
    2. Merge SAS data sets. If you already have a SAS data set generated from Step B, merge with that data set to update coordinate values.
    3. Convert the SAS data set to XLSX.
E. Repeat steps A to D as needed.

The aCRF draft is finished after the completion of step C1 or C2.

If you are starting from scratch with a blank CRF, then begin with step A. Otherwise, if you are starting with an existing aCRF, then begin with step C3.

ASSUMPTIONS

SOFTWARE USED

This is the complete list of software used in our implementation:

- Excel
- SAS
- SAS XML Mapper
- PDF editor (Adobe Acrobat)

The “medium of exchange” we can use to transfer annotations between the PDF and Excel files is an XFDF (XML Forms Data Format) file. This file stores the PDF annotations in the form of an XML (Extensible Markup Language) data structure.

There is another medium of exchange, FDF (Forms Data Format), but it is less useful for our purposes here. For annotating CRFs, we discourage using FDF for the following reasons: (1) It is harder for humans to read and interpret, and (2) it cannot be parsed by SAS XML Mapper.

Since the XFDF file conforms to the XML standard, we can use SAS XML Mapper to extract the annotation data contained in it, without the need to use another programming language.

CRF

It is assumed that all pages of the CRF to be annotated will have Portrait orientation.

STEPS A AND B: FROM EXCEL (XLSX) TO XFDF

STARTING POINT: GATHERING USER INPUTS IN EXCEL

Our macro %XLSX_TO_XFDF inputs an Excel file with the data listed in Table 2.

Column name	Explanation/Purpose	Values mandatory?	Values to assign	Associated XFDF tag or option
PAGENUM	To specify the page number where the box will go.	Yes	PDF page number.	page=
DOMAIN	To indicate the SDTM domain	Yes	The domain name abbreviation (usually two characters long). ex. “DM”	
DOMAINSEQ	To specify the order of coloring boxes belonging to the same domain	Yes, for title boxes	1, 2, 3, …	
TITLEBOX	To indicate those annotations which state the domain name with its abbreviation (and go at the top of the CRF page).	Yes, for title boxes	Assign Y for title boxes. Null otherwise.	
ANNOTATION	To specify the text to appear in the box	Yes	Annotation text	<p dir=”ltr”>
FONTSIZE	To specify a font size	No	A number for font size (pt)	
X1	To specify the X coordinate of the left side of the box.	No		
Y1	To specify the Y coordinate of the lower side of the box.	No		
X2	To specify the X coordinate of the right side of the box.	No		
BOXLENGTH	To specify the length of the box. (Equivalent to X2 minus X1.)	No		
COORD	To specify where the X and Y coordinates of the corners of the box. This determines both the size of the box and its placement on the page.	No**	A set of PDF page coordinates separated by commas. The order is as follows: X1, Y1, X2, Y2.	rect=

Table 2: Starting Point: User Input Collected in a Spreadsheet. A list of variables included in the Excel spreadsheet containing the annotation data. Our macro inputs these values from Excel and outputs the XFDF file using the DATA step shown in Figure 2.

END POINT: OUTPUTTING AN XFDF FILE USING SAS

The structure of annotations in an XFDF file is covered in Appendix 1: Parsing and Simplifying XFDF Files. Knowing the structure of the XFDF allows us to plan how to output such files using SAS. We can use SAS to output the file with a series of PUT statements using single quotation marks (to avoid interfering with the double quotation marks use to specify attributes).

Figure 2 shows the final DATA step in our macro %XLSX_TO_XFDF for outputting the annotations as an XFDF file. This step has been "macrotized" to replace specific values with variable names; these are shown as *italicized and in gray*. The FILE statement outputs the file with the XFDF file extension in the filename.

```
data _null_;
  file "output.xfdf";
  set input_dataset end=end;

  if _n_ = 1 then do;
    put '<?xml version="1.0" encoding="UTF-8"?>';
    put '<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">';
    put '<annots>';
  end;

    put '<freetext ';
    put '  page    ="' xpagec    '"';
    put '  rect    ="' coord     '"';
    put '  subject ="' domain    '"';
    put '  color   ="' backcolor '"';
    put '  name    ="' id        '"';
    put '>';

    put '<contents-richtext>';
    put '<body xmlns="http://www.w3.org/1999/xhtml" ';
    put '  xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/" ';
    put '  xfa:APIVersion="Acrobat:11.0.0" ';
    put '  xfa:spec="2.0.2" ';

    put '  style="text-align:left;           ';
    put '         font-weight:bold;          ';
    put '         font-family:Arial;         ';
    put '         font-stretch:normal;       ';
    put '         font-style:italic;         ';
    put '         font-size:' fontsizec 'pt; ';
  if titlebox = 'Y' then do;
    put '         color:#000000;             ';
  end;
  else do;
    put '         color:#FF0000;             ';
  end;
    put '        "                           ';

    put '><p dir="ltr">annotation</p';
    put '></body>';

    put '</contents-richtext>';
    put '</freetext>';

  if end then do;
    put '</annots>';
    put '</xfdf>';
  end;

run;
```

Figure 2: Generating an XFDF File with the Annotations. The SAS DATA step in this example outputs an XFDF file named *output.xfdf* (in the FILE statement). Values for variables *italicized in gray* will come from an input data set (here referred to as *input_dataset* in the SET statement).

Note that this DATA step has a few IF-ELSE statements to output some of these rows conditionally, including choosing the color of the annotation text.

For the DATA step we have just seen in Figure 2, we need to assign values for the variables shown in Table 3. Note that these will all need to be character-type variables, as they will be directly inserted into PUT statements.

Variable	Associated XFDF element or option	Explanation/Purpose	Values to assign
XPAGEC	`page=`	To specify the page number where the box will go.	XFDF page number*, as a character value. (*XFDF page 0 corresponds to PDF page 1.)
COORD	`rect=`	To specify where the X and Y coordinates of the corners of the box. This determines the length and height of the box and its position on the page.	A set of PDF page coordinates separated by commas. The order is as follows: X1, Y1, X2, Y2.
DOMAIN	`subject=`	To indicate the domain the annotation is associated with.	SDTM.DOMAIN (or SDTM. RDOMAIN for annotations associated with SUPP—data sets)
BACKCOLOR	`color=`	To specify the background color of the annotation box.	RGB value
ID	`name=`	To assign a unique identifier to enable merging of revised COORD data later.	Integers as character values: ‘1’, ‘2’, etc.
ANNOTATION	`<body>`	To specify the annotation text	The annotation text
FONTSIZEC	`<body>`	To specify the font size of the annotation text	The font size of the annotation text, as a character value

Table 3: Input Variables Needed for the DATA step in Figure 2. These variables all need to have character values, as they are to be inserted into PUT statements.

ADDITIONAL OUTPUT: A COPY OF THE ANNOTATIONS IN A SAS DATA SET

The SAS data set will have the same information that we have in our Excel file. With this data we can carry forward our metadata into subsequent cycles, as we shall see later in Figure 7.

FROM STARTING POINT TO END POINT: GENERATING AND APPROXMATING VALUES BASED ON USER INPUTS

Table 3 shows the set of variables we need for the DATA step that outputs the XFDF file that was shown in Figure 2.

Now we need to figure out how to take the data saved in the spreadsheet (as listed in Table 2) and generate values for the DATA step that was shown in Figure 2. We can simplify the user input needed by leveraging the correspondences between the input variables to auto-generate values for other variables.

Domain and Annotation Text

A user starting an aCRF from scratch will need to enter these values. Otherwise, these can be extracted from an existing version of the aCRF.

Page Number

The user will need to input PAGENUM, the page number of the PDF where each annotation will go. XFDF files, however, start the page count with zero. Our macro subtracts one to get the XFDF page count and stores it as a character value:

```
xpagec = strip(put(pagenum - 1, 8.));
```

Font Size

There are two types of annotations: (1) "title" annotations which announce the domain at the top of the page (e.g. "DM=Demographics"), and (2) otherwise "regular" annotations. We can assign a default font size to each type. The title annotations have a larger font, and we assign it an 18-point font, whereas we assign a 12-point font to the regular annotations. To distinguish between the two types, we added the variable TITLEBOX to indicate which annotations are title annotations.

We also allow these defaults to be overridden for special cases such as when a smaller font is needed to fit an annotation where space is tight.

For our macro, a default value is assigned when this value is null, so the user only needs to specify the font size when a non-default value is needed.

User input in spreadsheet		
TITLEBOX	ANNOTATION	FONTSIZE
Y	DM=Demographics	
	SUBJID	
	SUPPDM.QVAL when QNAM=RACEOTH	10

Values assigned by macro
FONTSIZEC
18
12
10

Figure 3: Assigning a Default Font Size. We can assign a default font size (shown in bold type), based simply on whether the annotation is a "title" for the SDTM domain or is just a regular annotation. The user can opt to override the default value by specifying a font size in Excel.

Background Color

There will be a one-to-one correspondence between DOMAIN and BACKCOLOR on any given page of the aCRF. This enables us to assign values of BACKCOLOR based on DOMAIN; this spares the user from having to enter these values into the spreadsheet.

User input in spreadsheet				
PAGENUM	DOMAIN	DOMAINSEQ	TITLEBOX	ANNOTATION
1	DM	1	Y	DM=Demographics
1	DM			SUBJID
1	SV	2	Y	SV=Subject Visits
1	SV			VISIT
1				NOT SUBMITTED

Values assigned by macro	
DOMAINSEQ	BACKCOLOR
1	**#BFFFFF** (blue)
1	**#BFFFFF**
2	**#FFFFAA** (yellow)
2	**#FFFFAA**
.	**#FFFFFF** (white)

Figure 4: Assigning a Text Box Background Color Based on DOMAIN. We can associate a different background color (predefined in our macro) for each SDTM domain listed on a page. The order in which background colors are assigned (also predefined in our macro), follows the values of DOMAINSEQ. Domain-less annotations are also assigned a color (white). Note that DOMAINSEQ is only required for title annotations; our macro merges this value with the other annotations for the same domain.

Background colors are typically assigned in a certain order (e.g. cyan blue as the first color), so we will also need to specify the sequence by which domains will be assigned colors. We add the variable DOMAINSEQ to specify this order. Then our macro assigns color accordingly.

```
if      domainseq = 1 then backcolor = '#BFFFFF';    * blue;
else if domainseq = 2 then backcolor = '#FFFFAA';    * yellow;
...
```

We also assign a color to domain-less annotations, such as “NOT SUBMITTED”.

Since the color values are defined in our macro, no direct user input is needed.

Each domain on a page will have one and only one title box, so it is sufficient to have DOMAINSEQ specified for just title boxes. Our macro can merge these unique values to the regular annotations. This further reduces the amount of user input needed.

X- and Y-Coordinates

The COORD values for the RECT= attribute each consist of four numbers separated by commas. Assigning these values would be particularly tedious if entirely by hand, but we can make at least a few simplifications.

1. When these numbers are exported out of an existing aCRF, they come with six decimal places. We can round these numbers to the nearest integer, which is sufficient precision for our purposes.
2. Instead of specifying these coordinates of new annotations, we can elect to (1) approximate their box dimensions and (2) stack them near the center of the page. Then later, in the PDF editor, the user can resize these annotations and drag-and-drop them to where they are needed.

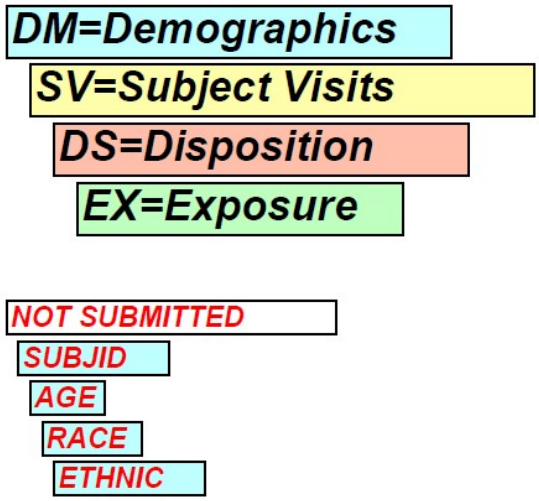


Figure 5: Stacking Annotations without Coordinate Values. Annotations without pre-specified values for COORD or any of its components (X1, Y1, BOXLENGTH) can be placed in a stack toward the center of the PDF page. The lengths of these boxes are approximated from the font size and the length of the annotation text. Later, in the PDF editor, the user can resize these annotations and drag-and-drop them to where they need to go.

User input in spreadsheet					
ANNOTATION	X1	Y1	X2	BOXLENGTH	COORD
DM=Demographics					10,815,190,838
SV=Subject Visits	200	815	380		
DS=Disposition	400	815		170	
EX=Exposure					

Values assigned by macro
COORD
10,815,190,838
200,815,380,**838**
400,815,**570**,**838**
[entirely auto-generated]

Figure 6: Options for Inputting Coordinate Values. We give the user several options for inputting x- and y-coordinate values (COORD) in Excel. The options are shown in order of decreasing difficulty from the standpoint of the user. The first option is most practical after COORD values have been exported out of a PDF file already having annotations. The middle two options allow the user to define the box length either directly (BOXLENGTH) or indirectly (X2 minus X1). The option requiring the least effort upfront is the option to input no coordinates at all (as for EX in this example), in which case the annotation will be part of a stack as seen in Figure 5.

We can set up our macro to give the user multiple options for inputting values for COORD. Figure 6 shows examples of how COORD values might be constructed based on user input. In these examples,

- DM already has COORD specified, so no modification is needed. Manually entering complete COORD values would be very tedious, but these values can be exported out of the PDF file of an existing aCRF and carried forward into subsequent iterations of our process.
- SV has both X-coordinates and Y1, the lower Y-coordinate. Our macro adds a height based on font size: Y2 = Y1 + height based on font size.
- DS has a value for BOXLENGTH, which is added to X1 to get X2.
- EX has no coordinate information, so an autogenerated value is given. The box length is approximated from the length of the annotation text. This annotation will be placed toward the center of the PDF page; the user can then resize or move the annotation as needed in the PDF editor.

We need to prioritize these options in case user input is ambivalent. One way to prioritize these options is as follows:

1. Use COORD if available.
2. Otherwise, use X1 and X2 if available.
3. Otherwise, use X1 and BOXLENGTH if available.
4. Otherwise, add to the stack in the staging area.

Approximating Box Height and Length

For cases without complete COORD values, Table 4 gives more details on how we might approximate the box dimensions based on existing data.

Box dimension	Approximation

Box height	Based on FONTSIZE, if available.
	Otherwise, use default values that depend on whether TITLEBOX is 'Y' or null.
	Code example:
	`select(fontsize);`
	`  when(18) boxheight = 23;`
	`  when(12) boxheight = 15;`
	`  when(10) boxheight = 13;`
	`  otherwise boxheight = fontsize * 1.25;`
	`end;`

Box length	Based on (X1 and X2) or (X1 and BOXLENGTH), if available.
	Otherwise, use value based on the font size and the number of characters in the annotation text.
	Code example:
	`if nmiss(x1, x2) = 0 then boxlength = x2 - x1;`
	`  * overrides user defined boxlength if X2 is also specified;`
	`if boxlength < .z then do;`
	`  if titlebox = 'Y' then boxlength = (fontsize * 0.7) * length(annotation);`
	`  else boxlength = (fontsize * 0.9) * length(annotation);`
	`end;`

Table 4: Approximating Text Box Height and Length. To reduce the need for the user to input COORD values, we can approximate box dimensions based on FONTSIZE and the length of the annotation text (i.e. the number of characters in the annotation text).

Stacking Annotations without Any Coordinate Data

Here is an example of how we can enumerate both title annotations and regular annotations, in preparation of stacking them toward the middle of the page (as illustrated in Figure 5):

```
…
by xpage domainseq domain;

if first.xpage then do;
      big_j = .;
      small_j = .;
end;

if titlebox = 'Y' then do;
      big_j+1;
      k = big_j;
end;
else do;
      small_j+1;
      k = small_j;
end;
…
```

Afterwards, we can auto-generate values for X1 and Y1. Then X2 and Y2 can be derived using box height and box length.

```
…
if titlebox = 'Y' then do;
      if x1 < .z then  x1 = 190 + 10 * k;
      if y1 < .z then  y1 = 725 - 25 * k;

      y2 = y1 + boxheight;
end;
else do;
      if x1 < .z then  x1 = 195 +  5 * k;
      if y1 < .z then  y1 = 600 - 17 * k;

      y2 = y1 + boxheight;
end;

if x2 < .z then x2 = x1 + boxlength;
…
```

The numbers in gray can be tweaked to adjust to a different page size or (landscape) orientation.

For more details on PDF x- and y-coordinates, refer to pages 3-4 of Black (2015).

STEP C: FROM XFDF TO PDF AND BACK TO XFDF

IMPORTING XFDF ANNOTATIONS INTO A PDF FILE

See Appendix 2: Importing XFDF Annotations into a PDF File.

ADJUSTING PDF ANNOTATIONS

Here we make edits within a PDF editor. This is where all the work would be done if annotating a CRF entirely by hand. This is the easiest place to move annotations manually within each page. Any annotations stacked in the middle (as illustrated in Figure 5) can be distributed at this time.

EXPORTING PDF ANNOTATIONS INTO AN XFDF FILE

See Appendix 3: Exporting PDF Annotations into an XFDF File.

STEP D: FROM XFDF TO EXCEL (XSLX)

STARTING POINT: CONVERTING AN XFDF FILE INTO SAS DATA SETS

We now have an XFDF file exported from the PDF. We need to convert these into SAS data sets. To do this, we can use SAS XML Mapper. Instructions on using SAS XML Mapper are given in Appendix 4: Using SAS XML Mapper to Transfer Annotations from an XFDF File to SAS Data Sets.

END POINT: OUTPUTTING AN EXCEL FILE WITH THE ANNOTATIONS

We want to end up with an Excel spreadsheet with the variables listed in Table 2. After prepping the values needed for those variables, we then use PROC EXPORT to transfer them to an Excel file.

FROM STARTING POINT TO END POINT: MERGING SAS DATA SETS GENERATED BY SAS XML MAPPER

We now have an XFDF file exported from the PDF. We then need to merge datasets. The merges needed depend on whether you started the process with Step A (with an Excel file) or Step C (with only an existing aCRF).

If the annotations already exist in a spreadsheet

if you already have annotations in an Excel spreadsheet, and did not change anything besides box sizes and positions in the PDF editor, then you only need to merge the new coordinates with the SAS data set generated during Step B. The new coordinate (COORD) values are in the *Freetext* data set generated by SAS XML Mapper. The new coordinates reflect any changes the user made to the positions or sizes of annotations in the PDF editor. The numbers in these COORD values will have six decimal places; we can simplify them by rounding these numbers to integers.

To update our annotation metadata with these new COORD values, we can merge by values of ID, one of the variables in Table 3. This merge is illustrated in Figure 7.

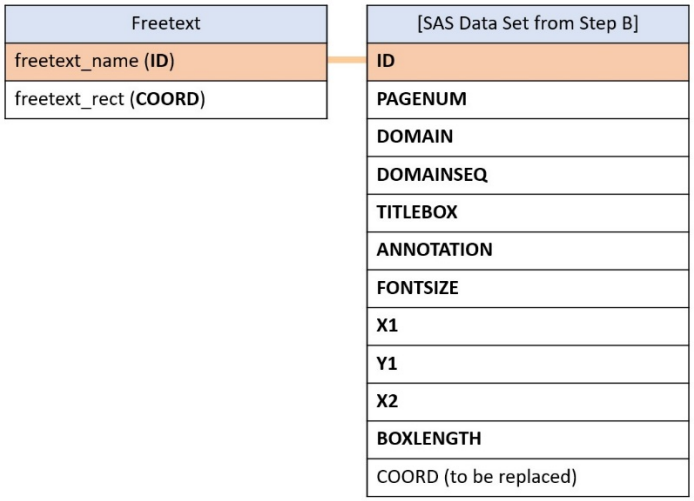


Figure 7: Updating Coordinate Values of Annotations Already in a SAS Data Set. The data set names are in the blue cells at the top of each table. Variables linking the data sets are shaded in the same color and linked with lines. The Freetext data set is generated by SAS XML Mapper, while the data set on the right was generated during the transfer of annotations from Excel to XFDF (Step B).

If the annotations do not yet exist in a spreadsheet

If you are starting from an existing aCRF and do not yet have your annotations in an Excel file, then it will take several merges to combine the data into one data set with variables to match the Excel data structure outlined in Table 2. Figure 8 shows the data sets generated by SAS XML Mapper and the links between them.

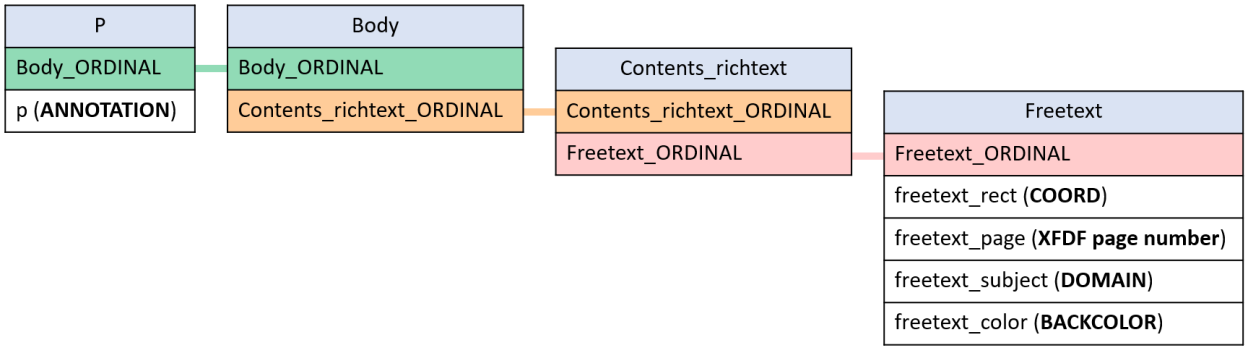


Figure 8: Merging Data Sets Generated by SAS XML Mapper. The data set names are in the blue cells at the top of each table. Record identifier variables that data sets have in common are shaded in the same color and linked with lines; these variables can be used to merge the data sets The variables to be transferred to an Excel file are in the white cells; the data associated with these variables are in parentheses.

After merging, we make some modifications to prepare the data for export into a spreadsheet. These modifications include (1) converting the XFDF page count into PDF page numbers by adding 1, and (2) rounding the coordinate numbers in COORD to the nearest integer.

CONCLUSION

We present here a method for reducing the effort it takes to annotate an aCRF by providing a way to recycle and refine our annotations. The savings in time is especially great when revising an aCRF, as this method enables us to transfer the annotations in batch rather than having to cut and paste them manually between different pages in PDF files.

Also, we have sought to decrease the burden of user input by assigning default or approximated values based on already existing values. At the same time, we still give the user flexibility to override these default or auto-generated values as needed.

REFERENCES

Adobe Systems Inc. *XML Forms Data Format Specification*. Version 3.0, August 2009.

Black, Steven. (2015) "Have SAS Annotate your Blank CRF for you! Plus dynamically add color and style to your annotations." PharmaSUG 2015, paper AD05. www.pharmasug.org/proceedings/2015/AD/PharmaSUG-2015-AD05.pdf

CDISC (2014) *Define-XML v2.0.* Release Package 2014-04-24. www.cdisc.org/standards/foundational/define-xml/define-xml-v2-0

Gu, Yating. (2019) "Auto-Annotate Case Report Forms with SAS® and SAS XML Mapper." PharmaSUG 2019, paper AP-125. www.pharmasug.org/proceedings/2019/AP/PharmaSUG-2019-AP-125.pdf

ISO 19444-1:2016(en). "Document management — XML Forms Data Format — Part 1: Use of ISO 32000-2 (XFDF 3.0)." www.iso.org/obp/ui/#iso:std:iso:19444:-1:ed-1:v1:en

Pyrnokokis, Ilias. (2015) "Developing annotated CRF: SAS, Excel and patience as your friends." PhUSE 2015, paper PP29. www.lexjansen.com/phuse/2015/pp/PP29.pdf

SAS XML Mapper. Available at https://support.sas.com/downloads/browse.htm?cat=12.

ACKNOWLEDGMENTS

Thanks to Leslie Schneider for help with generating and editing figures. Thanks to Danielle Rupper, Kevin Uchimara, and Leslie Schneider for help with proofreading drafts of the paper. Thanks also to Lingjiao Qi and Bharath Donthi for additional comments. All errors remain the responsibility of the authors.

CONTACT INFORMATION

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

Noory Kim
nkim@sdcclinical.com

Bhagyashree Shivakumar
bshivakumar@sdcclinical.com

APPENDIX 1: PARSING AND SIMPLIFYING XFDF FILES

To learn how to generate an XFDF file with the annotations (as was done in Figure 2), it first helps to understand (1) how an XFDF file is structured and (2) how annotation positions are coded in XFDF.

Overall Structure

The overall structure of the XFDF file is shown in Figure 9.

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
<annots>

<freetext … >
       [Annotation 1]
</freetext>


<freetext … >
       [Annotation 2]
</freetext>

</annots>
</xfdf>
```

Figure 9: Overall Structure of an XML/XFDF File. The actual text is shown in a `monospace font`, whereas the locations of the annotations are indicated by placeholders in a sans-serif font.

Each annotation resides within a FREETEXT element, where *element* is defined as the "sandwich" between its opening tag (`<freetext …>`) and its closing tag (`</freetext>`). All FREETEXT elements are nested inside the ANNOTS element.

One Annotation

```
><freetext color="#BFFFFF" creationdate="D:00000000000000Z" flags="print"
date="D:20091112084313-05'00'" name="a60dc033-5534-4833-9ae9-8ea63131213a"
page="2" fringe="0.500150,0.500150,0.500150,0.500150"
rect="21.880200,746.255000,212.123000,771.740000" subject="IE"
><contents-richtext
><body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
xfa:APIVersion="Acrobat:7.0.8" xfa:spec="2.0.2" style="font-size:18.0pt;font-
weight:bold;font-style:italic;font-family:Arial"
><p dir="ltr"
>DM=Demographics</p
></body
></contents-richtext
><defaultappearance
>0 0 0 rg /Arial,BoldItalic 18 Tf</defaultappearance
><defaultstyle
>font: italic bold Arial 18.0pt; text-align:left; color:#000000 </defaultstyle
></freetext
><freetext …
```

Figure 10: One XFDF Annotation. Text for the first FREETEXT element in the XFDF file exported from the aCRF of the CDISC define-xml v2.0 package. The grayed-out portions are not part of the section for the first FREETEXT element. The annotation text is shown here in bold type.

To see how the annotations are structured, we can export the annotations from the aCRF example in the CDISC define-xml v.2.0 package. The lines containing the first annotation in the file are shown in Figure 10. Note that the line breaks do not necessarily coincide with where a FREETEXT element starts and ends. The characters grayed out here are not part of the first FREETEXT element, but rather part of the preceding or subsequent element.

One Annotation, Reformatted

To make the annotation easier for a human to decipher, we can reformat this to have (1) indents and (2) line breaks occurring between the tag boundaries (“><”) instead of before the end tag boundary (“>”). The result is shown in Figure 11.

```
<freetext color="#BFFFFF" creationdate="D:00000000000000Z" flags="print"
date="D:20091112084313-05'00'" name="a60dc033-5534-4833-9ae9-8ea63131213a"
page="2" fringe="0.500150,0.500150,0.500150,0.500150"
rect="21.880200,746.255000,212.123000,771.740000" subject="IE">
       <contents-richtext>
              <body xmlns="http://www.w3.org/1999/xhtml"
              xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
              xfa:APIVersion="Acrobat:7.0.8" xfa:spec="2.0.2" style="font-
              size:18.0pt;font-weight:bold;font-style:italic;font-family:Arial"
              ><p dir="ltr">DM=Demographics</p
              ></body>
       </contents-richtext>
       <defaultappearance>
              0 0 0 rg /Arial,BoldItalic 18 Tf
       </defaultappearance>
       <defaultstyle>
              font: italic bold Arial 18.0pt; text-align:left; color:#000000
       </defaultstyle>
</freetext>
```

Figure 11: One XFDF Annotation Reformatted. Text from Figure 10 reformatted with line breaks and tabs to make it easier for a human to read. The grayed-out closing carets are left in the following line to prevent line breaks from appearing in the text of the annotation itself.

Note: Within the BODY element, the closing brackets of tags are on the same line as the opening bracket of any tag immediately following. We avoid introducing line breaks between the tags here; otherwise, the line breaks would appear as part of the annotation text.

```
<body …
><p dir="ltr">DM=Demographics</p
></body>
```

One Annotation, Simplified

```
<freetext color="#BFFFFF" name="1" page="2" rect="21,746,212,771" subject="DM">
       <contents-richtext>
              <body xmlns="http://www.w3.org/1999/xhtml"
              xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
              xfa:APIVersion="Acrobat:7.0.8" xfa:spec="2.0.2" style="font-
              size:18.0pt;font-weight:bold;font-style:italic;font-family:Arial"
              ><p dir="ltr">DM=Demographics</p>
              ></body>
       </contents-richtext>
</freetext>
```

Figure 12: One XFDF Annotation Simplified. The text from Figure 11 is simplified for our purposes. Simplifications include the removal of attributes and sections not needed. NAME has been assigned a shorter value, the numbers in RECT have been rounded to the nearest integer, and SUBJECT has been assigned the SDTM domain corresponding to this annotation.

We will not need all these elements or attributes for our purposes, so we can remove or modify some of them to simplify further. The simplifications made here include the following:

1. Remove the following optional attributes: CREATIONDATE, FLAGS, DATE, FRINGE. (For the FREETEXT element, only the PAGE and RECT attributes are required. See XFDF Specification, page 50.)
2. Round down the numbers in the RECT attribute to the nearest integer. (These numbers determine the position and dimensions of the text box containing the annotation. We won't be needing a greater degree of precision than that.)
3. Shorten the NAME attribute. (Later on, we will replace these values with ID numbers that are unique to each annotation. This will enable us to update RECT values.)
4. Update the SUBJECT attribute to match the SDTM domain. (This is to follow the convention in the aCRF example in the define-xml v.2.0 package.)
5. Remove the DEFALUTAPPEARANCE and DEFAULTSTYLE elements. (We will be defining font options within the STYLE attribute of the BODY element instead.)

APPENDIX 2: IMPORTING XFDF ANNOTATIONS INTO A PDF FILE

To import XFDF annotations into a PDF, go to the Comments sidebar menu, and select *Import Data File*.

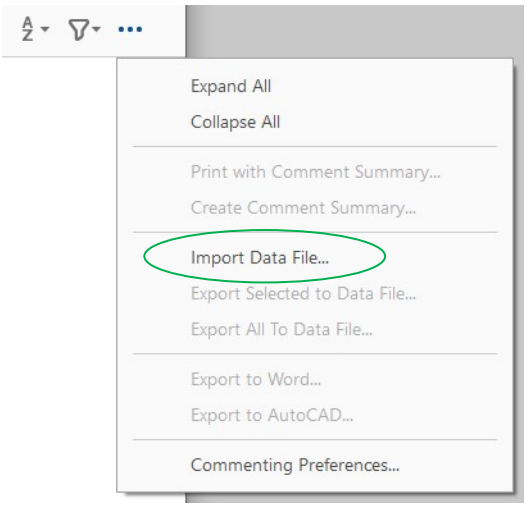


Figure 13: Importing XFDF Annotations into a PDF File.

APPENDIX 3: EXPORTING PDF ANNOTATIONS INTO AN XFDF FILE

To import XFDF annotations into a PDF, go to the Comments sidebar menu, and select *Export All to Data File*.

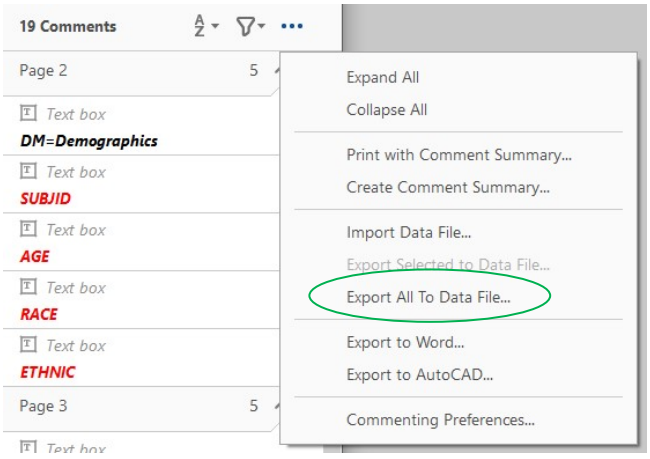


Figure 14: Exporting PDF Annotations.

In the popup box that appears next, change *Save as type* from “Acrobat FDF Files (*.fdf)” to “Acrobat XFDF Files (*.xfdf)”.

Figure 15: Exporting PDF Annotations as an XFDF File.

APPENDIX 4: USING SAS XML MAPPER TO TRANSFER ANNOTATIONS FROM AN XFDF FILE TO SAS DATA SETS

Since an XFDF file is an XML file, we can extract data in the XFDF file using an XML parser such as SAS XML Mapper, which transfers the data to a SAS data set.

Refer to page 3 of Gu (2019) for a screenshot of the user interface to SAS XML Mapper.

The steps of using SAS XML Mapper (version 9.4) are as follows:

1. Open the XFDF you exported out of the PDF file:

   From the menu bar, select *File*, then *Open XML*.

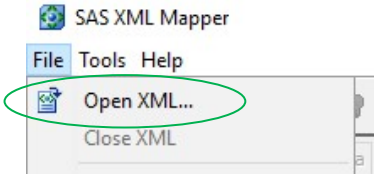


Figure 16: Opening Your XFDF File in SAS XML Mapper, Part 1.

   To be able to view XFDF files, you need to change *Files of type* from "XML file (*.xml)" to "All files".

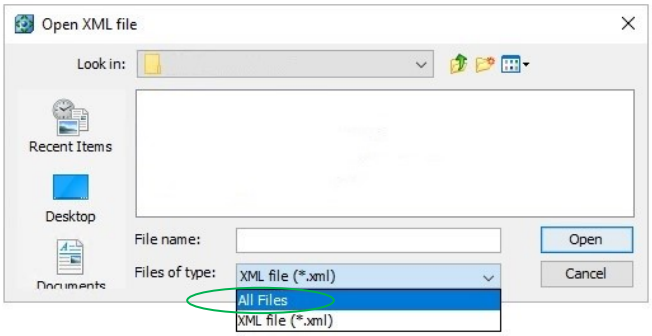


Figure 17: Opening Your XFDF File in SAS XML Mapper, Part 2

2. Transfer the XML data to a SAS XML map: From the menu bar, select *Tools*, then *AutoMAP using XML*.

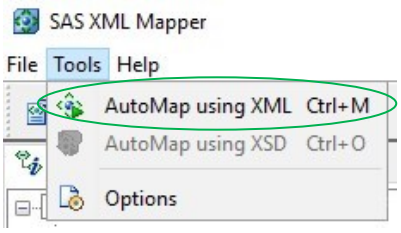


Figure 18: Generating an XMLMap

3. Save the SAS XML map and the associated SAS program:

   Select *File*, then *Save XMLMap As…*

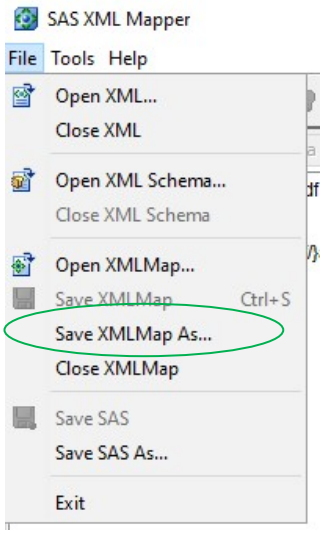


Figure 19: Saving the XMLMap

   Select *File*, then *Save SAS As…*

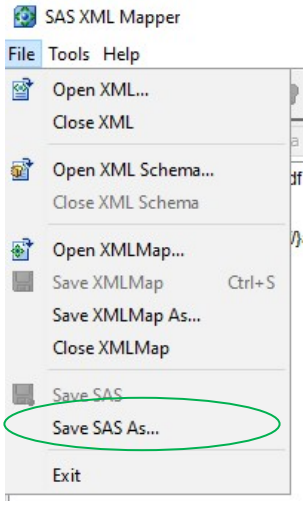


Figure 20: Saving the SAS Program that Extracts Data from the XMLMap

4. To extract the data from the XFDF file into SAS data sets, run the SAS code saved in the previous step. The data is separated into different data sets such as those shown in Figure 8. The data sets each correspond to an XML element, and data sets for nested XML elements share a common ID variable that can be used to merge them.