Extracting CRF Annotations to Support Define.xml Generation and Review

Mike Tan, Reata Pharmaceuticals, Irving, TX, USA

Richard Addy, Reata Pharmaceuticals, Irving, TX, USA

Deidre Kreifels, Reata Pharmaceuticals, Irving, TX, USA

ABSTRACT:

The annotated Case Report Form (aCRF.pdf) is a crucial component in a data submission package. The variable and value annotations on the aCRF are linked to PDF page references in the define.xml, allowing reviewers to quickly associate submission data with the collection forms. For that to work, aCRF annotations and define.xml origins must be consistent. Aligning CRF page references in the define.xml with the aCRF annotation, and keeping them aligned, is an arduous task if handled manually. In this poster, we will share a programmatic approach that extracts CRF annotations (including pdf page numbers) into a SAS dataset. Previous literatures suggested exporting CRF annotations in FDF format. However, we found that exporting as XFDF format is more user-friendly because of its XML-like nature. We will share a CRF annotation schema that allows for efficient extraction by SAS programs. Furthermore, we will share a program design that uses global macro variables to turn on and off extraction features based on the need of a specific study. Finally, we will show how these metadata extracted from annotated CRF can be incorporated in programming specifications, define.xml generation and used to support define.xml review.

INTRODUCTION:

When the programming team receives clinical data, a blank Case Report Form (CRF) is usually attached. This CRF shows where raw datasets and variables were captured from. Then, statistical programmers would manually add annotations at each location where data was recorded and mapped to SDTM datasets. Since there are a variety of CRF designs, the annotation process will always involve manual review. These annotations will need to be verified against the SDTM define.xml prior to data submission. We have developed an automatic process to extract all annotations from the aCRF.pdf that can be used to generate or compare to the final define.xml.

WORKFLOW:

In figures 1 and 2 below, we show that extracting aCRF annotations into SAS dataset can be useful to creating SDTM specifications, generating define.xml, and validating define.xml page references. We will briefly describe our workflow in this section. The first step is to export all annotations from an aCRF into an xfdf file. Next, we will explain how to use an xml mapper to pull relevant information from the xfdf file. Then, we will provide a sample SAS code to import content from xfdf into SAS using the mapper. Finally, we will share our annotation guidelines that allow SAS to decipher the context of an annotation.

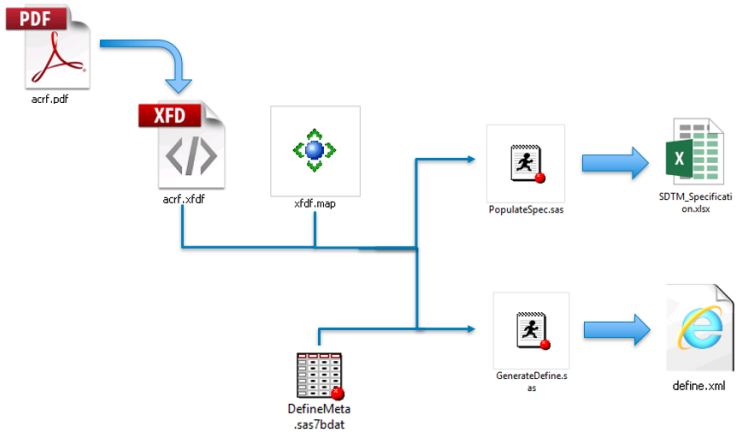


Figure 1. Workflow for populating programming specification and define.xml generation

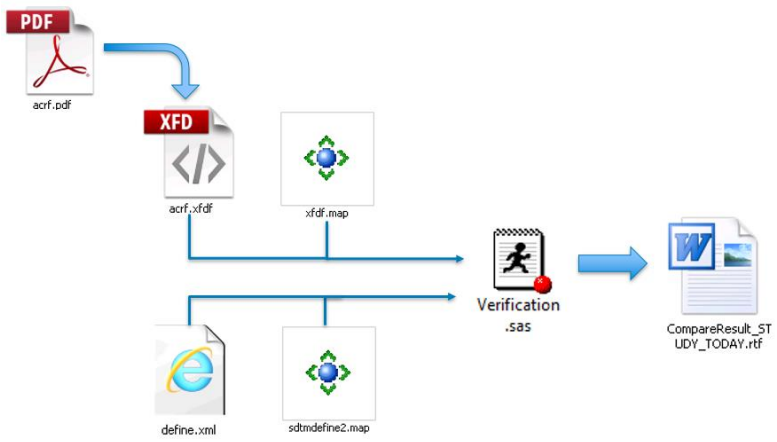


Figure 2. Workflow for validating define.xml against aCRF.

EXPORTING ANNOTATIONS TO XFDF:

The first step in this process is to export all annotations from a CRF to a separate file. Traditionally, it was exported to fdf format. We decided to export to xfdf format for reasons given in the below sections. Figure 3 is a series of screen shots that demonstrated how to export annotations to an xfdf file. This capability is available on both Adobe Acrobat Reader and Pro. The name and output directory do not have to be the same as the aCRF.pdf.

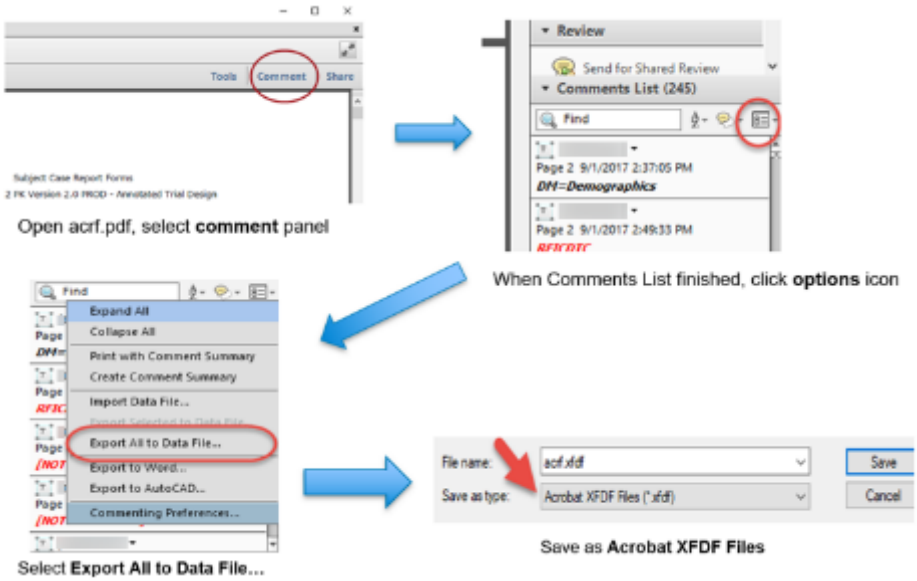


Figure 3. Steps to extract annotations to xfdf file.

WHAT IS AN XFDF FILE:

When users open a pdf file, they can insert data as comments directly onto the pdf. These comments can be extracted as plain text to fdf or xfdf formats. These exported files store all information about user added comments, including text, font size, color, background color, author, x-y coordinates, created date and time. We will focus on the xfdf format due to its XML like nature. As shown in figure 4 below, one annotation is wrapped inside a *freetext* element, and all *freetext* elements are wrapper inside the *annots* element. We can also see that the *page* element is an attribute of the *freetext* element, while the demographics label is a *p* element or paragraph. Occasionally, the annotation text would be divided into two *p* elements or two

span elements inside a paragraph. Therefore, the mapper file must also account for these conditions.

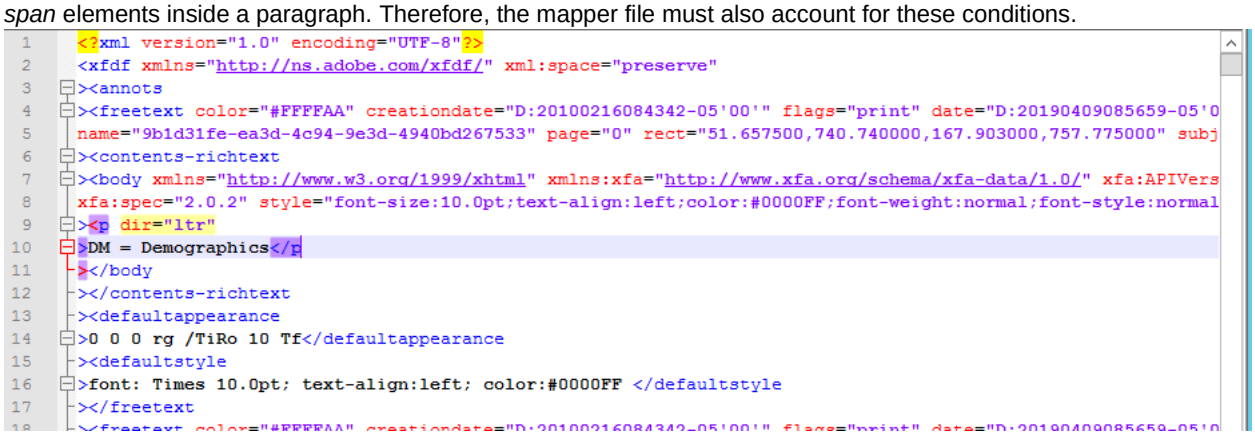


Figure 4. Part of an xfdf file opened in notepad++.

FDF VS. XFDF:

One reason a user might export annotations is to transfer the same annotations to another pdf. If a user exports annotation from one pdf and imports it on another pdf, all annotations will be placed exactly as in the original pdf. These fdf and xfdf files act as a medium for this transfer. To Adobe Acrobat, these two formats provide the same function. However, they are vastly different if we want to extract information using any other software. Figure 5 is a screen shot of the same annotations exported to fdf format. At first glance, the text colors look drastically different when both file formats were opened with Notepad++. This is because xfdf format is xml-like, and xml format is recognized by Notepad++. Meanwhile, fdf format is not recognizable by most text editing tools. Furthermore, information is placed together in lines of text that require complex regular expression to pull content from. Thus, we found xfdf format to a better option than fdf format.

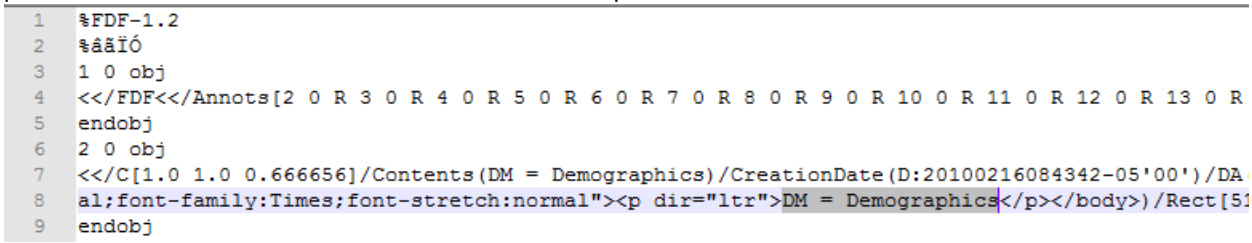


Figure 5. Part of a fdf file opened in notepad++.

WHAT IS A MAPPER:

An xml mapper (.map) file is used by SAS to extract data from xml formatted files by selecting specific content to import. For example, let's extract the page number and background color of all annotations. It is worth noting that page number starts at 0 and background colors were converted to Hex-color values. In figure 6, we displayed a mapper that creates a SAS dataset named "ANNOTS" with two columns, "rawPage" and "boxColor". The "rawPage" column has a label, stores integer type, and the values will come from the *page* tag of each *freetext* element in the *xfdf/annots* element of the xfdf file. For the full version of the mapper we used, see Appendix A. The simplest way to create a mapper file is to manually type one up using any text editing tool and save it as a .map file. Another way to create one is through the SAS XML Mapper tool. For detail instructions, please refer to the SAS XML Mapper website.

```
<?xml version="1.0" encoding="UTF-8"?>
<SXLEMAP name="xfdfmap" version="1.2">
    <TABLE name="Annots">
        <TABLE-DESCRIPTION>CRFAnnotationExtraction</TABLE-DESCRIPTION>
        <TABLE-PATH syntax="XPath">/xfdf/annots/freetext/contents-richtext/body/p</TABLE-PATH>

        <COLUMN name="rawPage" retain="YES">
            <PATH syntax="XPath">/xfdf/annots/freetext/@page</PATH>
            <TYPE>numeric</TYPE>
            <DATATYPE>integer</DATATYPE>
            <LABEL>Raw page number beginning at zero</LABEL>
        </COLUMN>

        <COLUMN name="boxColor" retain="YES">
            <PATH syntax="XPath">/xfdf/annots/freetext/@color</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>64</LENGTH>
            <LABEL>Color of annotation for grouping domains</LABEL>
        </COLUMN>
    </TABLE>
</SXLEMAP>
```

Figure 6. Part of a mapper file.

IMPORTING XFDF INTO SAS:

The code sample in figure 7 would read all content from the acrf.xfdf file into a working dataset in sas. This working dataset will be named ANNOTS and have formats specified by the mapper file.

```
* mapping files for extracting XML file formats ;
filename xfdfmap "C:\Users\mike.tan\Desktop\xfdf.map";

* ACRF annotation extracted XFDF file ;
filename crf_xfdf "C:\Users\mike.tan\Desktop\acrf.xfdf";

* create libname and load XML files into work directory ;
libname crf_xfdf xml xmlmap=xfdfmap access=READONLY;    * read CRF annotation ;

proc copy in=crf_xfdf out=work;
run;
```

Figure 7. SAS code to import xfdf file.

ANNOTATION SCHEMA:

The annotation schema is a set of rules to follow when annotating aCRF so a SAS program can correctly extract the necessary information from each annotation. We developed a list of conditions that allows for programmatically decipher annotations.

The first step is to remove extra text from annotations by following these examples:

Date of First Dose Administration (dd MON yyyy) EXSTDTC (date part)

Time of First Dose Administration (HH:nn) EXSTDTC (time part)

Figure 8. Option to remove text inside parenthesis: in this example, the "(date part)" is not useful to extract EXSTDTC variable.

if Yes then VSPOS = SITTING

Figure 9. Option to remove "If ... then" text: in this example, the "if Yes then" part is not useful to extract VSPOS variable.

We wrapped these removal options individually by a SAS macro function so we can easily turn it “on” or “off”. This design increases our flexibility to handle variety of aCRF, especially when working with multiple vendors.

The second step is to assign annotations to categories so they can be processed correctly. We will divide them into 8 categories here:

Patient Initials [NOT SUBMITTED]

Figure 10. Cat 1 - not submitted: check for brackets.

Figure 11. Cat 2 - note: check if annotation starts with "Note".

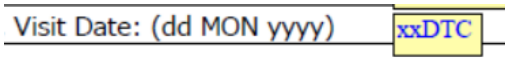


Figure 12. Cat 3 - xxDTC: check for "xxDTC".

Figure 13. Cat 4 - see pages: check if annotation starts with "See PDF Page".

Value level annotation came in the form of "`<DOMAIN.VARIABLE> where <CODEVAR> = <CODEVALUE>`" as shown in figures 14 and 15. We first check for the "where" keyword. If found, "`DOMAIN.VARIABLE`" is checked for a period. If found, value on left is the domain or subdomain, while content on right is variable that the value metadata is assigned to. Then, "`CODEVAR = CODEVALUE`" is checked for equal sign. If found, content on left is code variable to check, and content on right is the code value to check with.

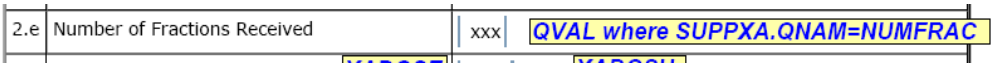


Figure 14. Cat 5, case 1 - supplemental domain value level annotation.

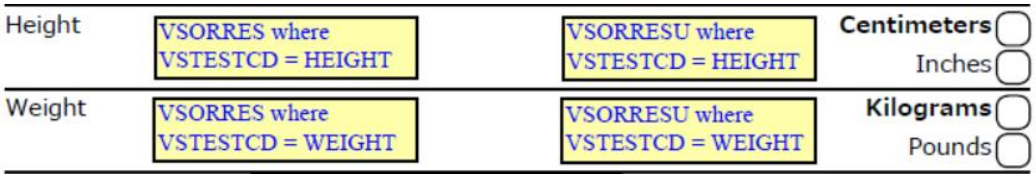


Figure 15. Cat 5, case 2 - value level annotation.

Figure 16. Cat 6, case 1 - domain label: if "=" found and value on left is a domain by checking against a pre-defined list.

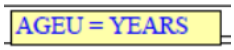


Figure 17. case 2 - variable: if value on left is not a domain then this is a variable annotation that belong to Cat 7.

Figure 18. Cat 7 - variable: if annotation text is less than or equal to 8 characters.

If space is limited on the CRF, multiple variables can be annotated together using the "/" delimiter. Then, the program would need to handle this condition.

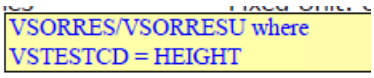


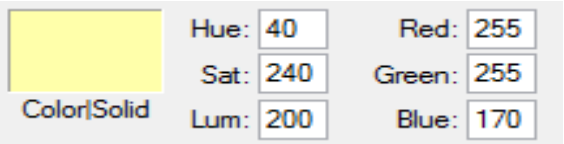
Figure 19. Multiple variables annotated together, separated by "/".

Lastly, if an annotation does not belong to any of the categories listed above, then it will be placed in a "Not Recognized" group that needs manual review.

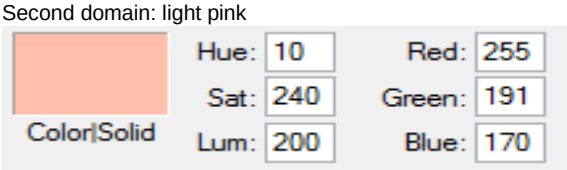
COLOR SCHEMA:

The color schema sets the standard for what colors we used as all annotation backgrounds per page. This allows us to group different annotations to their corresponding domain on the same pages. The first domain of every page, or not submitted pages, would use the first domain color. While each subsequent domain on the same page would follow this list below. It is important to keep colors consistent across studies because the xfdf file saved the colors in hex-color values. When importing into SAS, we have to match hex-color values to group annotations into their respective domains. See reference site 5 for corresponding hex-color values that will be exported to xfdf file.

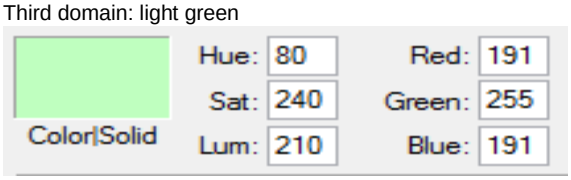
First domain: light yellow



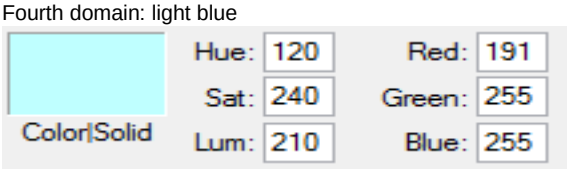
Second domain: light pink



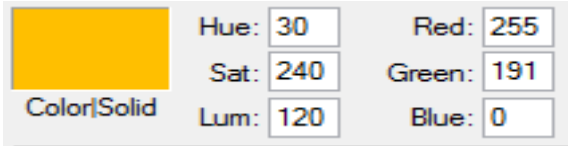
Third domain: light green



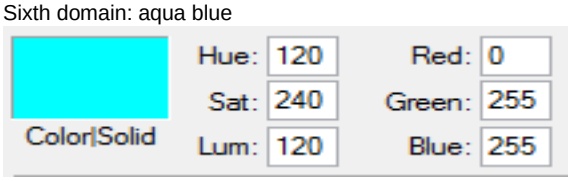
Fourth domain: light blue



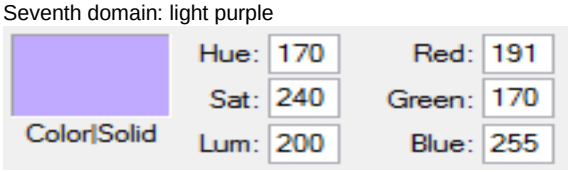
Fifth domain: orange



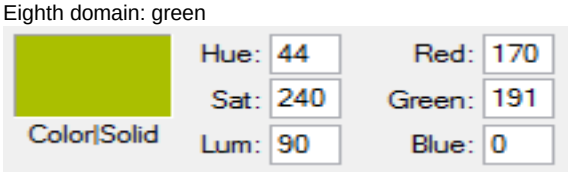
Sixth domain: aqua blue



Seventh domain: light purple



Eighth domain: green



Figures 20-27. List domain colors to annotate different domains on the same page by.

CONCLUSION:

In this poster paper, we demonstrated a step-by-step instruction on how to streamline CRF annotation into SAS. We first gave an overview of our approach, then dived into the details of each step. Based on experience working with other vendors, we followed a flexible design to accommodate variety of studies. If implemented thoroughly, it can reduce statistical programmers and reviewer's hours of work validating the aCRF.pdf is correctly matching the define.xml references.

REFERENCES:

1. Guideline for submission ready aCRF, https://www.phusewiki.org/docs/Frankfut%20Connect%202018/SA/PO/Papers/PP12-pp02-19541.pdf
2. QC of SDTM and aCRF using SAS, https://www.phusewiki.org/docs/Conference%202016%20AD%20Papers/AD03.pdf
3. Have SAS Annotate your Blank CRF for you! Plus dynamically add color and style to your annotations, https://www.pharmasug.org/proceedings/2015/AD/PharmaSUG-2015-AD05.pdf
4. Using SAS XML Mapper to Generate and Update an XMLMap, https://documentation.sas.com/?docsetId=engxml&docsetTarget=n0mxvt7afwoqron1pioilw0vzxq1.htm&docsetVersion=9.4&locale=en
5. RGB to Hex color conversion, https://www.rapidtables.com/convert/color/rgb-to-hex.html

CONTACT INFORMATION:

Mike Tan
Statistical Programmer
Reata Pharmaceuticals, Irving, TX

mike.tan@reatapharma.com

Richard Addy
Senior Statistical Programmer
Reata Pharmaceuticals, Irving, TX
richard.addy@reatapharma.com

Deidre Kreifels
Manager of Statistical Programming
Reata Pharmaceuticals, Irving, TX
deidre.kreifels@reatapharma.com

APPENDIX A: full mapper file to extract relevant content from an xfdf file.

```
<?xml version="1.0" encoding="UTF-8"?>
<SXLEMAP name="xfdfmap" version="1.2">
    <TABLE name="Annots">
          <TABLE-DESCRIPTION>CRFAnnotationExtraction</TABLE-DESCRIPTION>
          <TABLE-PATH syntax="XPath">/xfdf/annots/freetext/contents-richtext/body/p</TABLE-
PATH>

          <COLUMN name="rawPage" retain="YES">
                <PATH syntax="XPath">/xfdf/annots/freetext/@page</PATH>
                <TYPE>numeric</TYPE>
                <DATATYPE>integer</DATATYPE>
                <LABEL>Raw page number beginning at zero</LABEL>
          </COLUMN>

          <COLUMN name="boxColor" retain="YES">
                <PATH syntax="XPath">/xfdf/annots/freetext/@color</PATH>
                <TYPE>character</TYPE>
                <DATATYPE>string</DATATYPE>
                <LENGTH>64</LENGTH>
                <LABEL>Color of annotation for grouping domains</LABEL>
          </COLUMN>

          <COLUMN name="rect" retain="YES">
                <PATH syntax="XPath">/xfdf/annots/freetext/@rect</PATH>
                <TYPE>character</TYPE>
                <DATATYPE>string</DATATYPE>
```

```
            <LENGTH>64</LENGTH>

            <LABEL>Left, bottom, right, top coords from lower-left corner</LABEL>

      </COLUMN>


      <COLUMN name="style" retain="YES">

            <PATH syntax="XPath">/xfdf/annots/freetext/contents-
richtext/body/@style</PATH>

            <TYPE>character</TYPE>

            <DATATYPE>string</DATATYPE>

            <LENGTH>300</LENGTH>

            <LABEL>Group of attributes</LABEL>

      </COLUMN>


      <COLUMN name="textFragment">

            <PATH syntax="XPath">/xfdf/annots/freetext/contents-
richtext/body/p[1]</PATH>

            <TYPE>character</TYPE>

            <DATATYPE>string</DATATYPE>

            <LENGTH>32767</LENGTH>

            <LABEL>Text fragments in the annotation</LABEL>

      </COLUMN>


      <COLUMN name="textFragment2">

            <PATH syntax="XPath">/xfdf/annots/freetext/contents-
richtext/body/p[2]</PATH>

            <TYPE>character</TYPE>

            <DATATYPE>string</DATATYPE>

            <LENGTH>32767</LENGTH>

            <LABEL>Text fragments in the annotation</LABEL>

      </COLUMN>


      <COLUMN name="textSpan">

            <PATH syntax="XPath">/xfdf/annots/freetext/contents-
richtext/body/p/span[1]</PATH>

            <TYPE>character</TYPE>

            <DATATYPE>string</DATATYPE>

            <LENGTH>32767</LENGTH>
```

```
            <LABEL>Text fragments in the annotation</LABEL>
        </COLUMN>

        <COLUMN name="textSpan2">
            <PATH syntax="XPath">/xfdf/annots/freetext/contents-
richtext/body/p/span[2]</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>32767</LENGTH>
            <LABEL>Text fragments in the annotation</LABEL>
        </COLUMN>

        <COLUMN name="spanStyle">
            <PATH syntax="XPath">/xfdf/annots/freetext/contents-
richtext/body/p/span/@style</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>300</LENGTH>
            <LABEL>Attributes in Span</LABEL>
        </COLUMN>
    </TABLE>
</SXLEMAP>
```