PharmaSUG 2019 - Paper DS-024

Adding Table of Contents (TOC) for the Annotated CRF Automatically

Yang Zhou, dMed Biopharmaceutical Co., Ltd.

ABSTRACT

According to metadata submission guideline, annotated CRFs (aCRF) are required in the eCTD package and should be with bookmarks in 2 ways: one by timepoints, and the other by CRF topics. Bookmarks can often be inserted with hyperlinks to facilitate the online review. For an offline review, however, they appear unfriendly. In order to bring more convenience, we can generate Table of Contents (TOC) at the beginning of aCRFs for the offline review.

Differentiating from manual work, this paper will introduce how to add a TOC automatically in the aCRF file. C# and iTextSharp library are used in this automation process, and it is based on the PDF bookmarks.

INTRODUCTION

According to section4.2 (Bookmarking CRFs/eCRFs) and section4.3 (Table of Contents for the Annotated CRF) in Study Data Tabulation Model Metadata Submission Guidelines (SDTM-MSG) , Annotated CRFs included in the eCTD should be bookmarked 2 ways and a printable TOC may be included at the beginning of the annotated CRF. Additionally, the TOC should be hyperlinked to the appropriate page.

Usually we may use Adobe Acrobat Pro or Other PDF editor to add bookmarks to TOC one by one in aCRF. Repeated work not only takes a long time, but also has a quite high error rate. Different from complicated manual operations, iTextSharp library in C# has good performance in programming PDF documents (Figure 1).

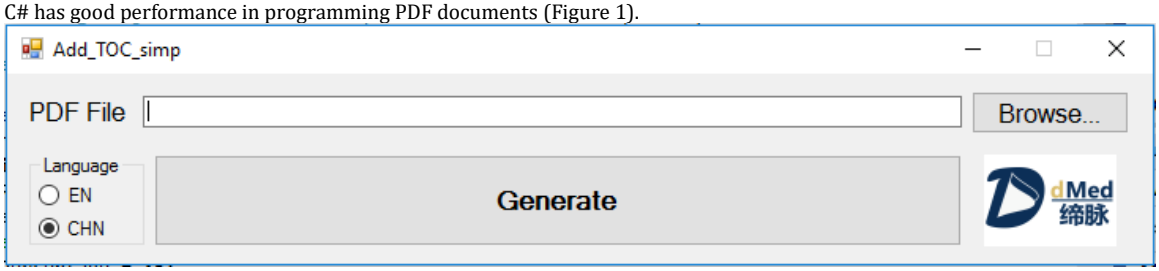


Figure 1

Four steps are needed to achieve the purpose. First, read bookmarks from aCRF. Second, loop through bookmarks and adding bookmarks into PdfTable. Third, insert blank pages at the beginning of the aCRF. Fourth, add PdfTable into blank pages, PdfTable is a table that can be put at an absolute position but can also be added to the document as the class Table. (Figure 2)

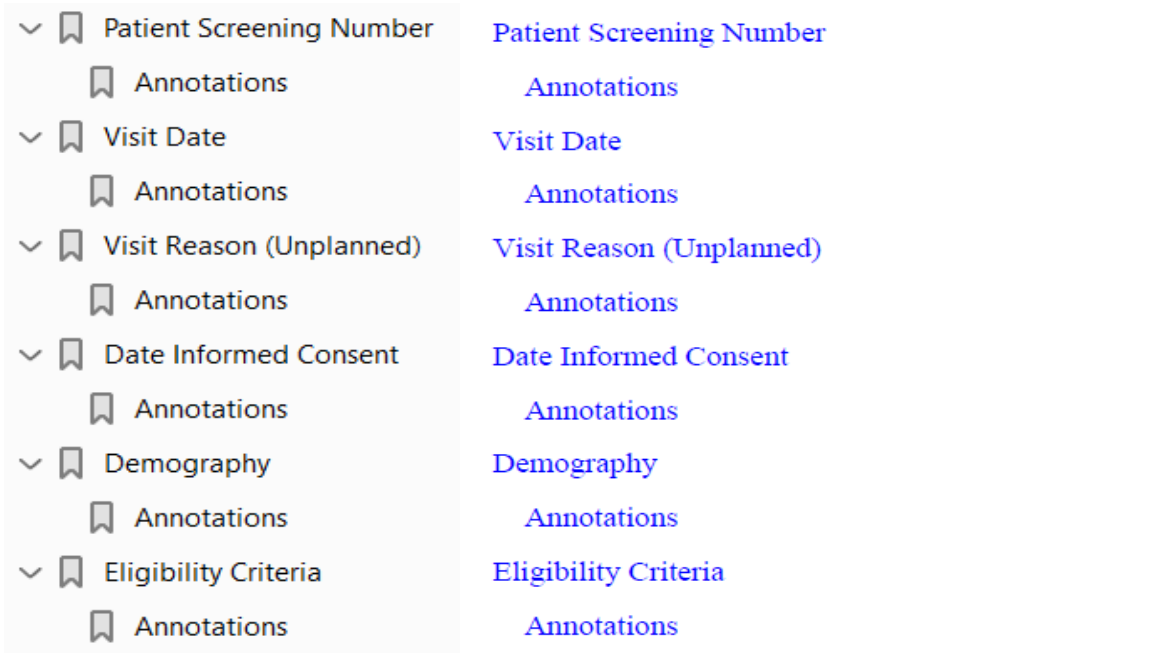
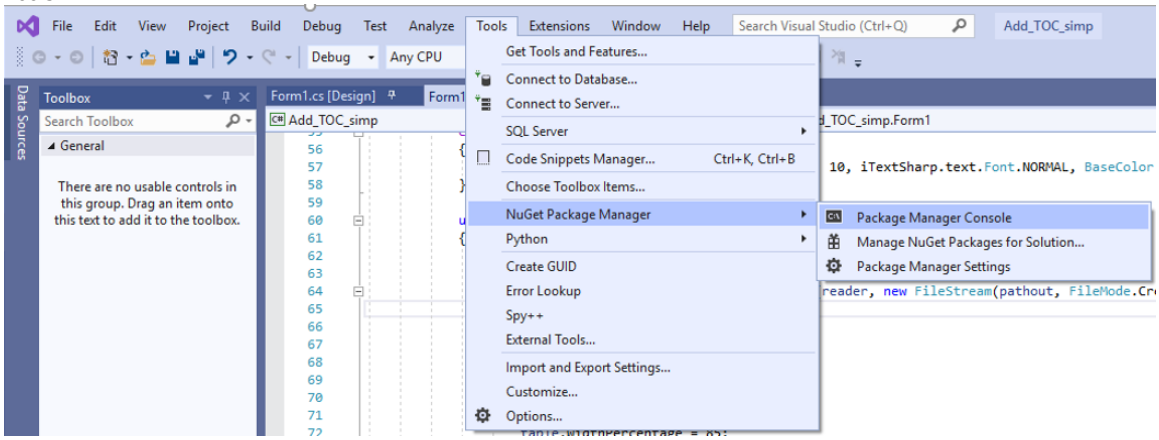


Figure 2 Bookmarks and Table of Contents

PREPARATION

The C# involved in this paper is run under Visual Studio 2019 Community. There are usually two ways to install iTextSharp library. One is installed by searching iTextSharp in Manage NuGet Packages for Solution in NuGet Package Manger in Tools, and the other is installed by inputting Install-Package iTextSharp in Package Manger Console in NuGet Package Manger in Tools.



After installation of iTextSharp, include below codes in the header of your app:

```
using iTextSharp;
using iTextSharp.text;
using iTextSharp.text.pdf;
```

Fonts included in iTextSharp library could not recognize Chinese characters. If your aCRF contains Chinese characters, fonts which could recognize Chinese characters need to be included in your computer. For example, the author installed simsun font to identify Chinese characters and TIMES_ROMAN font to identify other characters.

```
string simsunpath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\simsun.ttf";
BaseFont simsunfont = BaseFont.CreateFont(simsunpath, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
iTextSharp.text.Font link = new iTextSharp.text.Font(simsunfont, 10, iTextSharp.text.Font.NORMAL, BaseColor.BLUE);
if (radioButton_EN.Checked)
{
    link = new iTextSharp.text.Font(iTextSharp.text.Font.FontFamily.TIMES_ROMAN, 10, iTextSharp.text.Font.NORMAL, BaseColor.BLUE);
}
else if (radioButton_CHN.Checked)
{
    link = new iTextSharp.text.Font(simsunfont, 10, iTextSharp.text.Font.NORMAL, BaseColor.BLUE);
}
```

READING BOOKMARKS FROM ACRF

iTextSharp library could read bookmarks from aCRF file directly. Using PdfReader to read and parse a PDF document and GetBookmark method gets a List with the bookmarks.

```
using (PdfReader reader = new PdfReader(new FileStream(pathin, FileMode.Open)))
{
    IList<Dictionary<String, Object>> outlines = SimpleBookmark.GetBookmark(reader);
```

LOOPING THROUGH BOOKMARKS AND ADDING BOOKMARKS INTO PDFTABLE

As can be seen in the Figure 2, the bookmarks in aCRF are mostly tree structures. Each bookmark has a hyperlink corresponding to the pdf page.

Creating a PdfStamper based on PdfReader and Creating a PdfTable with one column. HeaderRows sets the number of the top rows that constitute the header. SkipFirstHeader tells you if the first header needs to be skipped. SplitRows set the rows that whether fit in the page will be split. Complete indicates if the PdfPTable is complete once added to the document. TotalWidth

sets the full width of the table. WidthPercentage sets the percentage width of the table from the absolute column width. (Figure 4)
Getting the title and page from bookmarks by looping through bookmarks. Constructing a chunk of text with title and a certain Font, then creating a GoTo action to an internal page get from bookmarks, setting the GoTo action for the chunk. Creating a new PdfCell, adding an iText element (the chunk constructed) to the cell and adding the cell element to table. (Figure 4)

```
using (PdfReader reader = new PdfReader(new FileStream(pathin, FileMode.Open)))
{
    IList<Dictionary<String, Object>> outlines = SimpleBookmark.GetBookmark(reader);

    using (PdfStamper stamper = new PdfStamper(reader, new FileStream(pathout, FileMode.Create)))
    {
        PdfPTable table = new PdfPTable(1);
        table.HeaderRows = 1;
        table.SkipFirstHeader = false;
        table.SplitRows = false;
        table.Complete = false;
        table.TotalWidth = 500;
        table.WidthPercentage = 85;

        string tabletitle, pageno = null;

        foreach (Dictionary<String, Object> sub0 in outlines)
        {
            tabletitle = sub0["Title"].ToString();
            pageno = sub0["Page"].ToString().Split()[0];

            PdfPCell cell = new PdfPCell();
            Chunk chunk = new Chunk(tabletitle, link);
            int pageno1 = int.Parse(pageno);
            PdfAction act = PdfAction.GotoLocalPage(pageno1, new PdfDestination(0), stamper.Writer);
            chunk.SetAction(act);
            cell.AddElement(chunk);
            cell.BorderWidth = 0;
            table.AddCell(cell);

            addcell(sub0, table, link, stamper, blank1, blank2);
        }

        table.Complete = true;
```

Figure 4

Since we do not know how many sub-bookmarks there are a bookmark contains and how many sub-sub-bookmarks there are a sub-bookmark contains, it is necessary to keep looping through bookmarks. We control the indentation by adding the blanks before the title. Whether a bookmark contains Kids key determines if it has sub-bookmarks. (Figure 5)

```
private void addcell(Dictionary<String, Object> sub_outline, PdfPTable table, iTextSharp.text.Font link, PdfStamper tmp, string blank1, string blank2)
{
    blank1 = blank1 + blank2;
    if (sub_outline.ContainsKey("Kids"))
    {
        IList<Dictionary<string, object>> child_outline = (IList<Dictionary<string, object>>)sub_outline["Kids"];
        foreach (Dictionary<String, Object> sub1_outline in child_outline)
        {
            string tabletitle = blank1 + sub1_outline["Title"].ToString();
            string pageno = sub1_outline["Page"].ToString().Split()[0];

            PdfPCell cell = new PdfPCell();
            Chunk chunk = new Chunk(tabletitle, link);
            int pageno_int = int.Parse(pageno);
            PdfAction act = PdfAction.GotoLocalPage(pageno_int, new PdfDestination(0), tmp.Writer);
            chunk.SetAction(act);
            cell.AddElement(chunk);
            cell.BorderWidth = 0;
            table.AddCell(cell);

            addcell(sub1_outline, table, link, tmp, blank1, blank2);
        }
    }
}
```

Figure 5

INSERTING BLANK PAGES AT THE BEGINNING OF THE ACRF

A blank page of A4 paper layout could be inserted about 35 bookmarks. According to the number of rows in the table, how many blank pages are needed could be easily calculated. InsertPage(int pageNumber, Rectangle mediabox) function can insert a blank page at the specified location. All the pages above and including pageNumber will be shifted up. If pageNumber is bigger than the total number of pages the new page will be the last one. pageNumber indicates the page number position where the new page will be inserted. mediabox indicates the size of the new page. GetPageSizeWithRotation gets the page size, taking rotation into account.

```
int tablenum = 0;
if (table.Rows.Count % 35 == 0)
{
    tablenum = table.Rows.Count / 35;
}
else
{
    tablenum = (table.Rows.Count - (table.Rows.Count % 35)) / 35 + 1;
}
for (int i = 0; i < tablenum; i++)
{
    stamper.InsertPage(1, reader.GetPageSizeWithRotation(1));
}
```

ADDING PDFTABLE INTO BLANK PAGES

Finally, we need to add the completed PdfTable to the blank pages. PdfContentByte is an object containing the user positioned text and graphic contents of a page. It knows how to apply the proper font encoding. GetOverContent gets a PdfContentByte to write over the page of the original document. WriteSelectedRows (int rowStart, rowEnd, float xPos, float yPos, PdfContentByte[] canvases) writes the selected rows to the document. rowStart indicates the first row to be written, zero index. rowEnd indicates the last row to be written + 1. If it is -1 all the rows to the end are written. xPos indicates the x write coordinate. yPos indicates the y write coordinate. canvases indicate an array of 4 PdfContentByte.

```
PdfContentByte canvas;

for (int i = 1; i <= tablenum; i++)
{
    canvas = stamper.GetOverContent(i);
    table.WriteSelectedRows(0 + (i-1)*35 , 35 + (i - 1) * 35, 80, 750, canvas);
}
```

CONCLUSION

This paper provides an approach to automatically add table of contents for the annotated CRF. With the help of this app, the TOC will be consistent with the bookmarks. Compared with the manual process, it improves the efficiency and reduces error.

It is not the only approach. However, it provides an idea which replaces manual operations through a series of methods. You can use other software to implement this process using similar approach in this paper.

REFERENCE

Study Data Tabulation Model Metadata Submission Guidelines (SDTM-MSG)
https://www.cdisc.org/standards/foundational/sdtmig/metadata-submission-guideline-msg-sdtmig

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:
Name: Yang Zhou
E-mail: yang.zhou@dmedglobal.com

SOURCE CODE

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Drawing;
6.  using System.IO;
7.  using System.Linq;
8.  using System.Text;
9.  using System.Threading.Tasks;
10. using System.Windows.Forms;
11. using iTextSharp;
12. using iTextSharp.text;
13. using iTextSharp.text.pdf;
14.
15. namespace Add_TOC_simp
16. {
17.     public partial class Form1 : Form
18.     {
19.         public Form1()
20.         {
21.             InitializeComponent();
22.         }
23.
24.         private void Button1_Click(object sender, EventArgs e)
25.         {
26.             OpenFileDialog fileDialog = new OpenFileDialog();
27.             fileDialog.Multiselect = true;
28.             fileDialog.Title = "请选择文件";
29.             fileDialog.Filter = "PDF(*.pdf*)|*.pdf*";
30.             if (fileDialog.ShowDialog() == DialogResult.OK)
31.             {
32.                 textBox1.Text = fileDialog.FileName;
33.             }
34.         }
35.
36.         private void Button2_Click(object sender, EventArgs e)
37.         {
38.             string pathin = textBox1.Text;
39.             string pathout = Path.GetDirectoryName(textBox1.Text) + "\\" + Path.
    GetFileNameWithoutExtension(textBox1.Text) + "_withTOC.pdf";
40.             addtoc(pathin, pathout);
41.         }
42.
43.         private void addtoc(string pathin, string pathout)
44.         {
45.             string blank1 = "";
46.             string blank2 = "    ";
47.
48.             string simsunpath = Environment.GetEnvironmentVariable("SystemRoot")
     + "\\fonts\\simsun.ttf";
49.             BaseFont simsunfont = BaseFont.CreateFont(simsunpath, BaseFont.IDENT
    ITY_H, BaseFont.EMBEDDED);
50.             iTextSharp.text.Font link = new iTextSharp.text.Font(simsunfont, 10,
     iTextSharp.text.Font.NORMAL, BaseColor.BLUE);
51.             if (radioButton_EN.Checked)
52.             {
53.                 link = new iTextSharp.text.Font(iTextSharp.text.Font.FontFamily.
    TIMES_ROMAN, 10, iTextSharp.text.Font.NORMAL, BaseColor.BLUE);
```

```
            }
            else if (radioButton_CHN.Checked)
            {
                link = new iTextSharp.text.Font(simsunfont, 10, iTextSharp.text.Font.NORMAL, BaseColor.BLUE);
            }

            using (PdfReader reader = new PdfReader(new FileStream(pathin, FileMode.Open)))
            {
                IList<Dictionary<String, Object>> outlines = SimpleBookmark.GetBookmark(reader);

                using (PdfStamper stamper = new PdfStamper(reader, new FileStream(pathout, FileMode.Create)))
                {
                    PdfPTable table = new PdfPTable(1);
                    table.HeaderRows = 1;
                    table.SkipFirstHeader = false;
                    table.SplitRows = false;
                    table.Complete = false;
                    table.TotalWidth = 500;
                    table.WidthPercentage = 85;

                    string tabletitle, pageno = null;

                    foreach (Dictionary<String, Object> sub0 in outlines)
                    {
                        tabletitle = sub0["Title"].ToString();
                        pageno = sub0["Page"].ToString().Split()[0];

                        PdfPCell cell = new PdfPCell();
                        Chunk chunk = new Chunk(tabletitle, link);
                        int pageno1 = int.Parse(pageno);
                        PdfAction act = PdfAction.GotoLocalPage(pageno1, new PdfDestination(0), stamper.Writer);
                        chunk.SetAction(act);
                        cell.AddElement(chunk);
                        cell.BorderWidth = 0;
                        table.AddCell(cell);

                        addcell(sub0, table, link, stamper, blank1, blank2);
                    }

                    table.Complete = true;

                    int tablenum = 0;
                    if (table.Rows.Count % 35 == 0)
                    {
                        tablenum = table.Rows.Count / 35;
                    }
                            else
                            {
                                tablenum = (table.Rows.Count - (table.Rows.Count % 35)) / 35 + 1;
                            }
                            for (int i = 0; i < tablenum; i++)
                            {
                                stamper.InsertPage(1, reader.GetPageSizeWithRotation(1));
                            }
```

```
108.
109.                             PdfContentByte canvas;
110.
111.                             for (int i = 1; i <= tablenum; i++)
112.                             {
113.                                 canvas = stamper.GetOverContent(i);
114.                                 table.WriteSelectedRows(0 + (i-
    1)*35 , 35 + (i - 1) * 35, 80, 750, canvas);
115.                             }
116.                         }
117.                     }
118.                 }
119.
120.                 private void addcell(Dictionary<String, Object> sub_outline, Pdf
    PTable table, iTextSharp.text.Font link, PdfStamper tmp, string blank1, string b
    lank2)
121.                 {
122.                     blank1 = blank1 + blank2;
123.                     if (sub_outline.ContainsKey("Kids"))
124.                     {
125.                         IList<Dictionary<string, object>> child_outline = (IList
    <Dictionary<string, object>>)sub_outline["Kids"];
126.                         foreach (Dictionary<String, Object> sub1_outline in chil
    d_outline)
127.                         {
128.                             string tabletitle = blank1 + sub1_outline["Title"].T
    oString();
129.                             string pageno = sub1_outline["Page"].ToString().Spli
    t()[0];
130.
131.                             PdfPCell cell = new PdfPCell();
132.                             Chunk chunk = new Chunk(tabletitle, link);
133.                             int pageno_int = int.Parse(pageno);
134.                             PdfAction act = PdfAction.GotoLocalPage(pageno_int,
    new PdfDestination(0), tmp.Writer);
135.                             chunk.SetAction(act);
136.                             cell.AddElement(chunk);
137.                             cell.BorderWidth = 0;
138.                             table.AddCell(cell);
139.
140.                             addcell(sub1_outline, table, link, tmp, blank1, blan
    k2);
141.                         }
142.                     }
143.                 }
144.             }
145.         }
```