PhUSE 2010

Paper TS04

PDF Multi-Level Bookmarks via SAS

Steve Griffiths, GlaxoSmithKline, Stockley Park, UK

ABSTRACT

Within the GlaxoSmithKline Oncology team we recently experienced an issue within our patient profile outputs relating to the generation of two-level PDF bookmarks to assist with clinical review; so that each data group listing had a unique bookmark subsetted within each subject. This presentation discusses the evolution of a method to create the desired two-level bookmarks within our patient profiles.

INTRODUCTION

Like most pharmaceutical companies, we at GSK are trying to make the most of the new techniques and methods for output generation. One of the advantages with SAS® ODS is its ability to sort out a lot of the actual page formatting and output directly to PDF. Whilst this automated approach generates a reasonably good styled body for the document, it's quite a different matter when it comes to the PDF bookmarks when using multiple PROC REPORT statements under SAS9.1.3. We were unhappy with the inability to generate multi-level bookmarks especially within our patient profile outputs; preferring to have expandable subject bookmarks with each data group listed as a subset.

Until PROC DOCUMENT is extended to deal with PROC REPORT or SAS9.2 is available on our reporting environment with its possible solutions, an alternative method was needed to achieve these bookmarks.

This paper will focus on the implementation of a solution for our patient profile output.

THE PROBLEM

Using SAS ODS, it is very simple to add title information into a top level bookmark to obtain something akin to that shown in figure 1 below, using an ODS PROCLABEL statement code similar to:

```
ods proclabel "Subjid =      202: Demography";
```

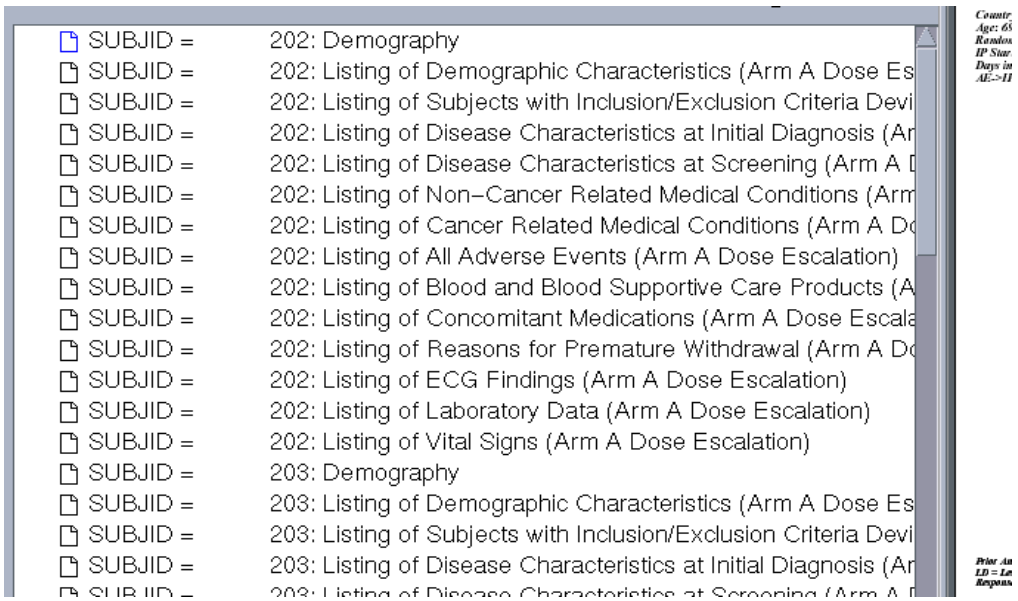

Fig. 1: Basic bookmark list as seen in Unix

prior to the call to (in this case) PROC REPORT. Of course the subject and titles can be placed within macro variables to obtain all the different bookmarks.

A mechanism exists within SAS to add a title to a second level bookmark using the CONTENTS= statement within the PROC REPORT command, but this is not perfect as demonstrated in the following code. Note that a ODS PROCLABEL needs to be executed twice.

```
ods listing close;
ods noptitle;
ods pdf file='dummy.pdf' pdftoc=2;
ods proclabel='Subject 1';

data temp1;
set dddata.sp_l_ns_l1;
if subjid=1;
run;

proc report data = temp1 split='~' headline nowd spacing=2 headskip center missing
ls=108 ps=43 contents='table1';
column SUBJID country invid invname centreid ptstudy2 ptsubjid  prtrtgrp prvdur;
define _all_ / display ;

… define statements removed for clarity …

run;

data temp1;
set dddata.demog_l3_nonstd;
if subjid=1;
run;

ods proclabel='Subject 1';

proc report data = temp1 split='~' headline nowd spacing=2 headskip center missing
ls=108 ps=43 contents='table3';
column invid subjid basecp  tpernum pertstdt pertendt days;
define _all_ / display ;

… define statements removed for clarity …

run;

ods pdf close;
```

Figure 2 shows the bookmarks generated by this sample code.

Fig. 2: Bookmarks generated by applying second level bookmarks directly.

Since we really require the collapsing bookmarks with each subject listing displayed under the main title bookmark as shown in figure 3 it is necessary to determine an alternative approach.

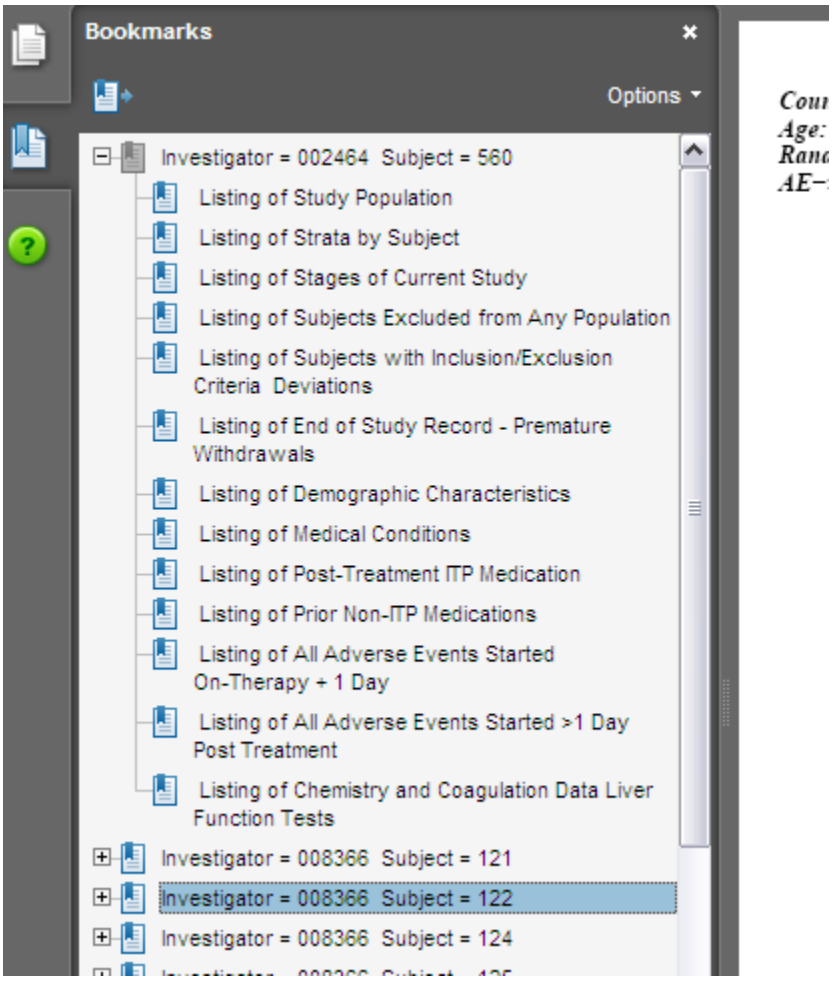

Fig. 3: Desired two-level bookmark.

Direct manipulation of the PDF file is not possible but the relevant information can be appended to a Postscript file in the form of language statements, which when processed using PS2PDF are converted to bookmarks. In order to achieve this a couple pieces of information are needed for each bookmark; namely the current page and for the parent bookmark the number of children.

The statements needed for parent and children take the following forms respectively:

[/Title (*Parent Title*)
/Count *number of children*
/Page *First Page*
/View [/XYZ *value value value*]
/OUT pdfmark

[/Title (*Child Title*)
/Page *First Page*
/View [/XYZ *value value value*]
/OUT pdfmark

The final pdfmark statement is used to represent PDF features.

FIRST APPROACH

Initial development was restricted to a specific number of reports so the number of children could be simply hardcoded. Therefore the issue was dealing with the page numbering. Since each component listing could be any length in page size it was necessary to determine how many pages were actually generated for each report. The first versions of the procedure used differing 'Page X of Y' style macros within PROC REPORT to determine the number of pages (and adding this information to a main page count variable) and were met with varying degrees of success. Ultimately it wasn't long before the

bookmark and associated page failed to correspond to each other due to differences between the ASCII version of the listing and the ODS formatted version.

SECOND APPROACH

During the development of the first approach it was discovered that the page information was present within the Postscript file, and so there must be a way to extract and utilize it within an ongoing page count.

Using a Unix pipe to pass a search string through an output file it's possible to obtain the number of pages produced by an individual report.

```
filename pages pipe "grep -i %%pages: outfile.PS";
data _null_;
infile pages lrecl=40 pad end=eof;
input text $20.;
if eof then call symput('page',trim(left(scan(text,2,' '))));
run;

%let _page_ = %eval(&_page_ + &page);
```

Using a global macro variable (&_page_ in this case) it's then possible to keep a record of the total page count which can be passed to the Postscript file.

It is necessary to append the complied bookmark information onto the finalized Postscript file, so another file is needed to keep track of the commands to be added. In order to reduce the amount of external files needed a SAS catalog source was used to accomplish this. This catalog was then 'included' and the commands executed to write directly to the Postscript file.

```
filename outfile CATALOG "work._outfile" mod;
%let _page_ = 1;
```

Contained within the loop for each subject…

```
/*
/ Create the SAS command which will create the parent bookmark.
/ Note for the first subject, &_PAGE_ will be 1.
/ Again working on a pre-defined list of reports - 14 in this case
/-----------------------------------------------------------------------------*/
data _null_;
file outfile(bookmark.source) linesize=10000;
mark="put ""[/Title (&g_subjid = &&subjid&j) /Count  14 /Page &_page_ /View [/XYZ 103
775 null] /OUT pdfmark"";";
put mark;
run;
```

Contained within the loop for each report…

```
/*
/ Create the SAS command which will create the child bookmark.
/ Note for the first subject/report, &_PAGE_ will be 1 since we have not performed
/ the search for %%page.
/-----------------------------------------------------------------------------*/
data _null_;
file outfile(bookmark.source) linesize=10000;
mark="put ""[/Title (&xdatagroup) /Page &_page_ /View [/XYZ 144.6 116.68  null] /OUT
pdfmark"";";
put mark;
run;
```

This particular sub-loop would then generate the report, perform the string search to determine the number of pages generated; update the rolling page count and end. Processing on the next report would start with the updated &_PAGE_ value passing into the next bookmark command.

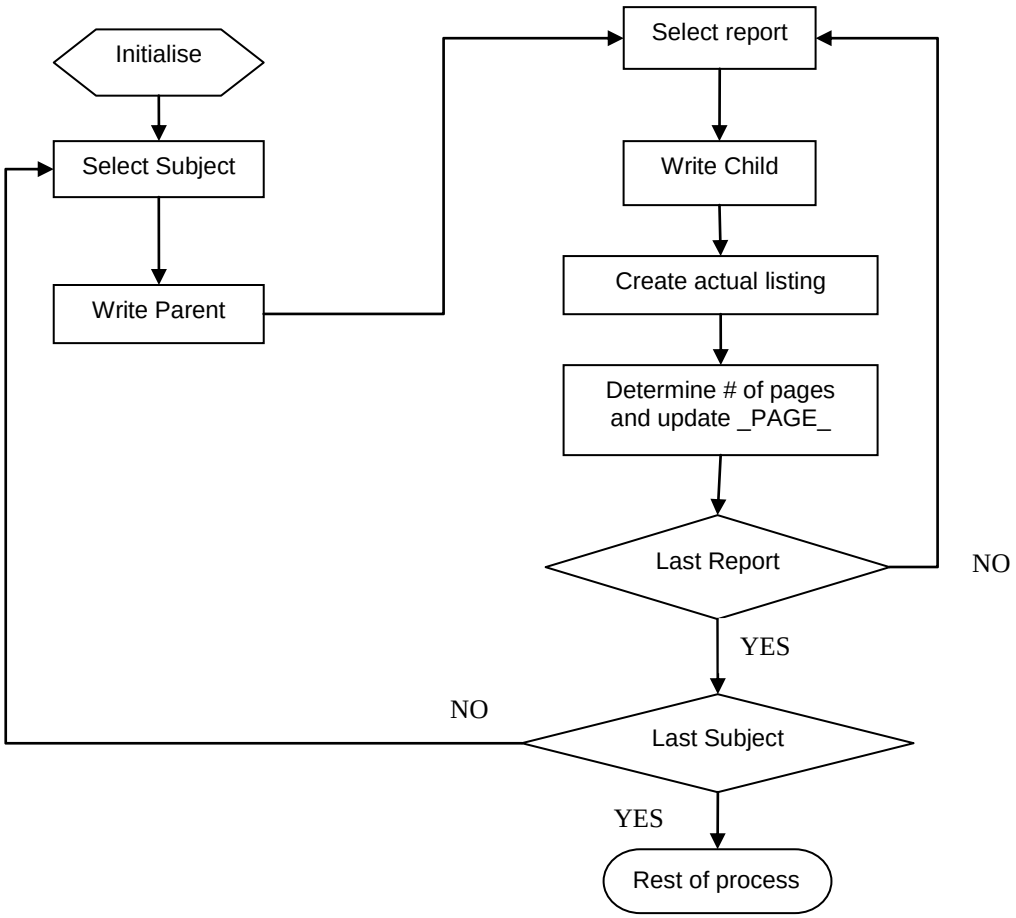

Fig. 4: Flow chart for process

The last remaining problem to solve was how to 'filter' off a copy of the current report to compute the pages whilst simultaneously keeping it in the main file. Luckily it is possible to open multiple file destinations in ODS PRINTER with the use of the ID= identifier demonstrated below:

As part of the initialisation:

```
ods listing close;
ods escapechar='\';
ods ps (id=main) file="outfile.ps" style=styles.rtf uniform;
```

Within the actual reporting loop:

```
ods ps (id=temp) file="tmp_rep.ps" style=styles.rtf startpage=yes uniform;
```

The report is then created in both output destinations. Immediately after close the **id=temp** output file:

```
ods ps (id=temp) close;
```

The next time the inner reporting loop executes, a new 'temp' ODS PRINTER destination is established and it is then possible to determine the pages for the newer report.

Whilst this method is successful only one summary output is produced per page, which did not meet the requirement of multiple outputs per page.

THIRD APPROACH

Due to the fact that each listing could flow over multiple pages or indeed only take up a few lines, it seems reasonable to allow SAS to actually deal with the formatting problem of how many outputs per page whilst applying the constraint of starting a new page for each subject. The method of 'filtering' off a copy of the current report would not work since it would never be known when page breaks were going to be introduced, so it's back to producing the Postscript file in its entirety first.

In the previous method, searching the individual Postscript files gave information on the total number of pages produced for each report, but searching the entire file gives you the page(s) on which any report can be found. It is therefore necessary to create a data set with the page number for the first occurrence of any given report for each subject, yet allowing multiple instances of that page number if several reports are present on the page in question – so additional variables are needed.

Through the use of header information supplied through the TITLE statement it is also possible to obtain subject information. The actual report title is harder to obtain since it is effectively free text and impossible to search for, unless prefixed by an unique character sequence – in this case a double hash (##).

Modifying the earlier Unix pipe to search for these multiple strings…

```
%let txt_string = egrep –i 'page:|subject:|##' outfile.PS;
filename pages pipe "&txt";
```

…and post processing the resultant dataset it is possible to obtain something akin to a table of contents for the entire report which in turn can be parsed into the necessary commands needed for the Postscript file.

```
 /*
/ Using this create a data set with the page numbers associated with each datagroup
/ for each subject.
/-------------------------------------------------------------------------------*/
data info(drop=dummy text start stop );
infile pages linesize=5000 lrecl=5000 truncover dlm=',';
length text $2500;
informat text $varying5000.;
input text ;
n=_n_;
length subjid 8 page 8 title dummy $200 centid $8;
retain subjid page &g_centid;

/* Obtain PAGE information                                                    */
if index(upcase(text),'%%PAGE:') then
   page = input(trim(left(scan(text,2,' %'))),best.);

/* Obtain SUBJECT and CENTRE information                                      */
/* Use the fact that text with Postscript file enclosed within ( ) to search for   */
/* desired values                                                             */
if indexw(upcase(text),'SUBJECT:') then do;
     dummy=scan(text,2,'()');
     subjid=input(trim(left(scan(dummy,-1,' ():'))),best.);
     centid=trim(left(scan(dummy,-3,' ')));
end;

/* Obtain TITLE information                                                   */
if index(upcase(text),'##') then do;
     start=index(upcase(text),'##');
     stop=index(upcase(text),')');
     title=substr(text,start+2, stop-(start+2));
end;

if title ne ' ' and index(title,'####') eq 0;
run;
```

```
/*
/ keep first occurrence (first page) of datagroup / title
/-------------------------------------------------------------------------------*/
proc sort data=info;
by centid subjid title page ;
run;

data info;
set info;
by centid subjid title page;
if first.title;
run;
```

One of the consequences of allow SAS to sort out the page formatting is that text wrapping can occur within the title (in order to fit the table width), and as such the complete title may not be easy to locate using the search. Therefore occasionally there can be a few reports which end up appearing to have the same title when truncated. This is particularly true producing the 'null report' when there is no data to present. Luckily this can be prevented with the inclusion of a STYLE to the DEFINE statement within PROC REPORT.

```
define none / display center style=[cellwidth=13cm];
```

However there may still be instances where the title, although unique is still too long for the table width, and the entire title is needed for the bookmark. To deal with this, a master bookmark data set containing subject and title information is complied during the reporting loop which is merged with the info data set.

It's now possible to determine the number of 'child' bookmarks, which should be the same for all subjects, and when used in conjunction with the merged bookmarked data set the set of commands for the Postscript file can be complied, and appended to the actual Postscript file.

```
/*
/ Create file of SAS commands to place PS bookmark code into PS file
/ Using a -ve number in the count will automatically collapse all the bookmarks.
/-------------------------------------------------------------------------------*/
filename outfile "bookmark.txt" mod;
filename outfile2 "outfile.PS"  mod;

data _null_;
set _temp_;
by centid &g_subjid page n;
nlistings=listings * -1;
file outfile;
if first.&g_subjid then do;
put "put""[/Title (Centre = "g_centid " Subject = " subjid ")/Count " nlistings "/Page
" page "/View [/XYZ 103 775 null] /OUT pdfmark"";";
end;
put "put""[/Title (" title ") /Page " page "/View [/XYZ 144.6 116.68  null] /OUT
pdfmark"";";
run;

/*
/ Run SAS generated code appending bookmarks to PS file
/-------------------------------------------------------------------------------*/
data _null_;
file outfile2;
%inc outfile;
run;
```

The file is then converted and housekeeping performed to remove the external files.

```
x ps2pdf "outfile.PS" "outfile.pdf";
```

```
x rm "outfile.PS";
x rm "bookmark.txt";
```

LIMITATIONS

When using this approach on a large subject population having a considerable number of listings, there have been a few occasions when the program has aborted due to memory issues. The easiest solution to this has been to create subject subgroups and create several patient profile listings.

CONCLUSION

This paper has shown that it is possible to create parent and multiple child bookmarks in PDF using SAS, and whilst there are solutions available which use Java to manipulate files, it was necessary to create a method which would worked within the GSK regulated reporting system.

Although this paper deals with the implementation of a solution for a specific problem within GSK Oncology, the fundamentals should be easily transferable to create multiple level bookmarks for other scenarios.

Hopefully when support for PROC REPORT is provided within PROC DOCUMENT this sort of file manipulation will not be needed.

REFERENCES

Chung, Chang Y. and Dunn, Toby (2005), "Page X of Y with Proc Report." Paper CC31. *Proceedings of the Pharmaceutical Industry SAS Users Group Conference 2005*
http://www.lexjansen.com/pharmasug/2005/coderscorner/cc31.pdf

Jansen, Lex. (2001), "Creating PDF Documents including Links, Bookmarks and a Table of Contents with the SAS® Software." Paper FDA05. *Proceedings of the Pharmaceutical Industry SAS Users Group Conference 2001*.
http://www.lexjansen.com/pharmasug/2001/proceed/fdacomp/fda05_jansen.pdf

Karunasundera, Tikiri. (2006), "Creating and Modifying PDF Bookmarks" *SAS Conference Proceedings: Western Users of SAS Software 2006*
http://www.lexjansen.com/wuss/2006/data_presentation_and_business_intelligence/DPR-Karunasundera.pdf

CONTACT INFORMATION

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

Steve Griffiths
GlaxoSmithKline
1-3 Iron Bridge Road
Stockley Park West
Uxbridge
Middlesex UB11 1BU
stephen.j.griffiths@gsk.com