PhUSE EU Connect 2018

Paper AD06

Assembling PDF Reports with PROC Documents

Sven Prasse, Entimo AG, Berlin, Germany

ABSTRACT

SAS® provides PROC Documents to build PDF documents out of various tables, listings and figures. By replaying pre-generated and validated outputs a single PDF document can be created without recalculating the statistics behind the outputs. This PDF can simply be enriched with overall page numbers or a TOC with hyperlinks and bookmarks. By using general templates the header and footer are printed over all different parts of the report to fit your company style guide. This presentation outlines the usage of PROC Documents in combination with the SAS document stored in a simple report program. By using metadata and the ODS Document storage the program generates a final report in one run. The metadata and the SAS document stores are the interface between the statistical programs and the reporting macro. This procedure enables you to properly divide output generation from report generation.

INTRODUCTION

During a feasibility analysis together with the product development department at Entimo, my colleagues and I investigated the possibilities to produce PDF reports with the SAS technology. In one of our investigated use cases we had to run different programs within entimICE® as a programming environment and store their outputs for a second process. This publishing process should produce a PDF file containing all the output contents. We decided to use ODS DOCUMENT to store the outputs for the publishing.

This was the first time that I had the possibility to use the ODS DOCUMENT functionality. The most charming feature of this type of ODS stream is the idea to separate the output calculation from the publishing of the document itself. I often see within the R&D of the pharmaceutical industry that programs run each time again to calculate the same output and link it into different documents or even worse the validation and programming for such outputs have to be processed for every new document that has to be produced. This give me the idea to use ODS DOCUMENT to calculate the outputs and validate all calculation programs in order to publish this specific output in many documents without rerunning, revalidating or reprogramming the calculation program.

The ODS system with the different formats is a great help for the programmers. Without any specific knowledge, it allows easily to produce a wide variety of formats. On the other hand, SAS disables features of specific formats in order to simplify the usage of the SAS functions. For example, the features of the build in TOC for the PDF are very poor. I show in the paper how it is possible to calculate the TOC without the help from the standard ODS feature and enhance the TOC with links into the final PDF.

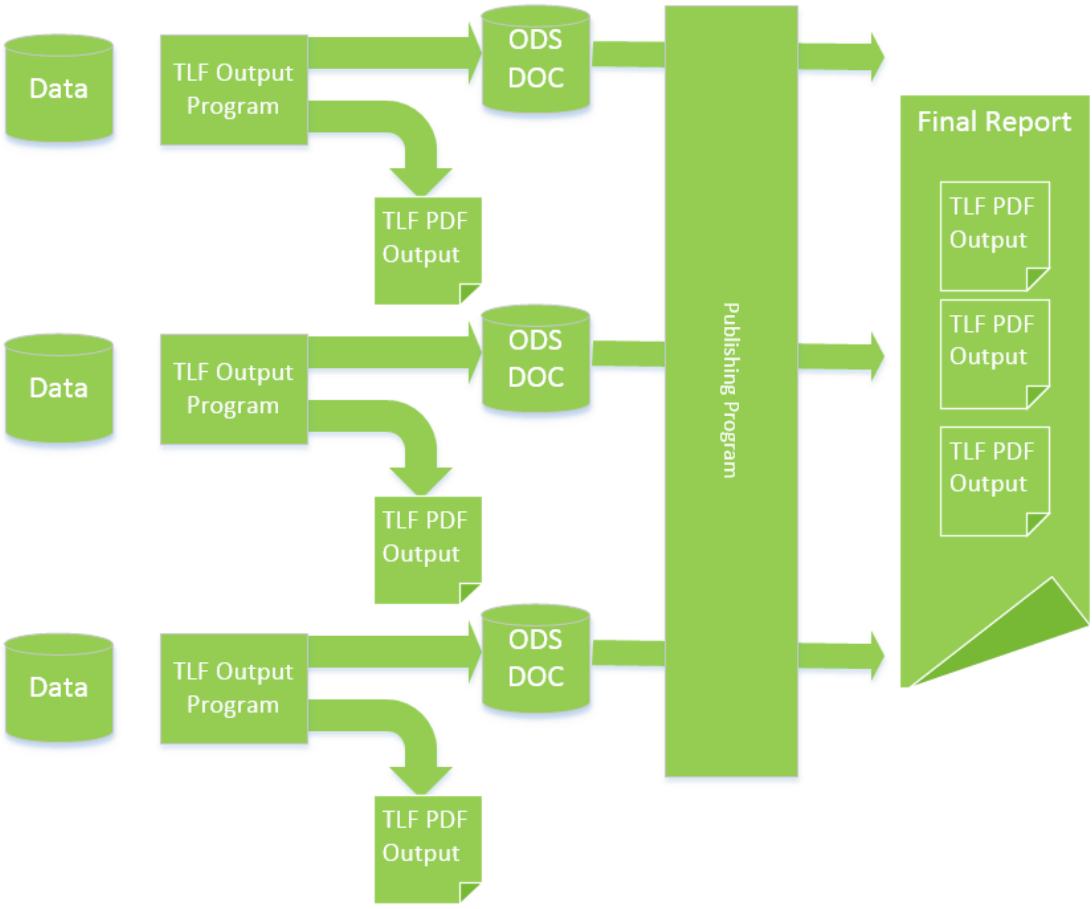
I removed all dependencies to other tools and techniques than SAS Base for this paper. This leads to the following technical decisions to simplify the description:

- All outputs are produced in PDF
- The output format of the publishing process is a PDF
- The interface for the TOC meta data is stored in a simple text files
- The TOC structure has only 2 levels

In other contexts these decisions may not very comfortable. Please note, that these decisions are not fixed and it should be easy for a SAS programmer to adapt the idea of the paper and write with the help of the examples here their own program code to satisfy slightly other requirements.

OVERVIEW

The general procedure is to produce the single outputs by running all outputs programs successfully and produce a ODS output into two streams (PDF and DOCUMENTS). In this texts I will name this programs TLF-programs (TLF stands for Table-Listing-Figures). They have to use a SAS procedure that supports the ODS system (e.g. PROC REPORT). It is recommended to use a template to guarantee a unique layout over all pages in the final document. How to produce the template and all other prerequisites is described in the preparation paragraph.



The program that creates the final report implements the publishing process and I call it publishing program within this paper. It produces the first the TOC and appends all outputs from the TLF Programs. The meta data for the TOC is also used to loop through all outputs, set the appropriate titles and replay them into the final report.

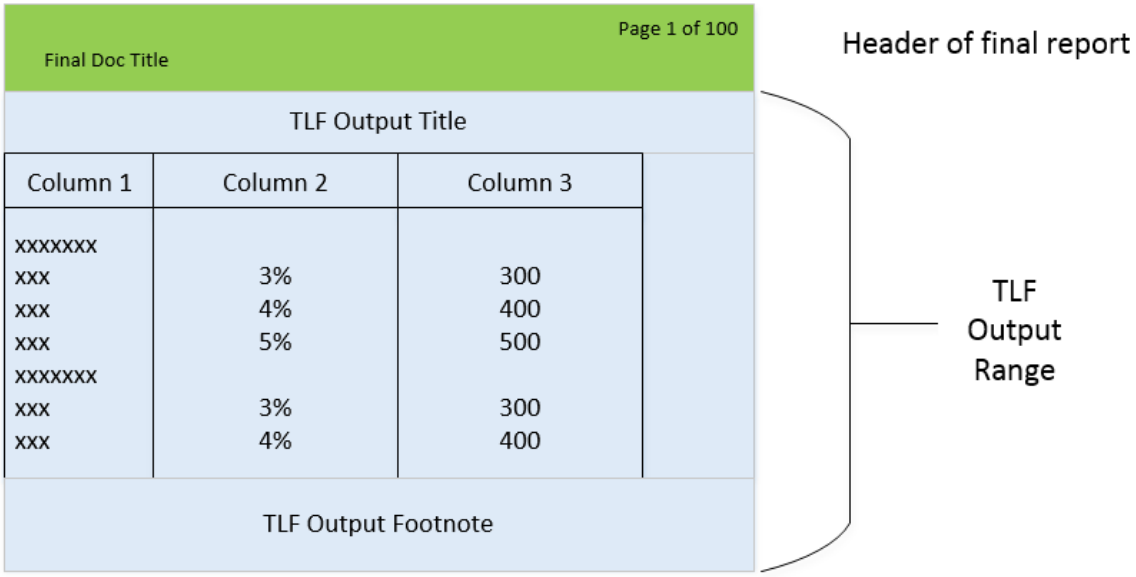
PREPARATION

Each program within the trail has to follow some rules during the output creation to support the generation of the whole appendix document.

THE LAYOUT OF THE FINAL REPORT

The layout for the final PDF reports starts with a Page header. In this example, I reserve three header rows. At the bottom of the header, a line will be drawn to separate the header from the TLF program report range. The first header row of the TFL program is title 5.

In principle, the margins of the TLF template could be set to the height of the headers and the line of the final report. But it is hard to calculate this as a top margin in inch exactly. It is much easier to delegate this calculation to SAS by setting four title rows with the font and font size that you want and start the TLF outputs with title 5. SAS will set the title for all output exactly as you need this. You only have to use the same TITLEFONTSIZE and TOPMARGIN in all your templates.



At least one template for the TOC, one template for the TLF outputs and one for the publishing program are needed. Set the margins to the same value for all templates. The TOC template is defined as a table without any paddings and margins with one column (see the code example for details).

```
PROC TEMPLATE;
    DEFINE STYLE TOCASREPORT / STORE = work.TEMPLAT;
        PARENT = STYLES.pearl;

        STYLE TABLE FROM TABLE /
        RULES       =none
        FRAME       =void
        CELLPADDING = 0PT
        CELLSPACING = 0PT
        BORDERWIDTH = 0PT;

        STYLE HEADER FROM HEADERSANDFOOTERS / BACKGROUND = WHITE VJUST=T;

        REPLACE BODY FROM DOCUMENT /
            TOPMARGIN    = 0.4in
            BOTTOMMARGIN = 1in
            LEFTMARGIN   = 1in
            RIGHTMARGIN  = 1in ;
    END;
RUN;
```

For your TLF programs, it is only important to have appropriate values for the table. E.G.

```
            RULES       =GROUPS
            FRAME       =HSIDES
            CELLPADDING = 2PT
            CELLSPACING = 0.75PT
            BORDERWIDTH = 1PT;
```

The publishing template needs extra definitions for the three title lines. Please note the syntax of the inline statements depends on the final output format. That means the syntax in the example below is only valid for PDF outputs. However, this is a usual behavior of the ODS system and you will find the syntax for the other formats in the SAS help.

```
            STYLE ContentTitle
               from IndexTitle/pretext=
               "&_headerFirstPage.^{newline 3}Table of Contents ^{newline 2}
(*ESC*){leaders  }Page^{newline 2}"
               JUST=L FONT=FONTS('TitleFont');
               REPLACE PrintedContentsLabel
               "Sort of a post-posttext for the CONTENTS" /
               POSTTEXT = " (*ESC*){leaders .  }(*ESC*){tocentrypage}"
               PRETEXT = "(*ESC*){tocentryindent 2em}";
            REPLACE ContentItem from IndexItem
               "Controls the leafnode item in the Contents file." /
               RIGHTMARGIN = 0%
               LEFTMARGIN = 0%
               JUST = L;
            REPLACE ContentFolder from IndexItem
               "Controls the generic folder definition in the Contents file." /
               RIGHTMARGIN = 0%
               LEFTMARGIN = 0%
               JUST = L
               LISTENTRYANCHOR = OFF
               FOREGROUND = COLORS('confolderfg');
            REPLACE ContentProcName from IndexProcName
               "Controls the proc name in the Contents file." /
               RIGHTMARGIN = 0%
               LEFTMARGIN = 0%
               JUST = L;
```

PROVIDING THE TOC META DATA

The publishing program needs to know where each TLF output is stored. To have a TOC with different hierarchy levels the program needs in addition the level and the title of the output for the TOC. Usually this information is provided by the programming and reporting environment. For this paper, we assume that this is stored in a simple dataset with three columns:

- the hierarchy level,
- the title of the output,
- and where the outputs are stored. (complete path to the storage file and the name of the storage)

The hierarchy should be given like the heading numbers in office programs. E.g. 1.1, 1.2, 2.1 and 2.2 shall be interpreted as two chapters on the main level and the each chapter itself has two subchapters. The chapter 1 and 2 has no outputs (the document storage is empty). With this simple definition, it is possible to cover the requirements for mostly all reports.

```
DATA toc;
  FORMAT chapter $20.
         toc_entry $100.
         document_storage $500.
         document_storage_name $50.
        level 8.
         page 8.;
RUN;
```

STORE THE TLF OUTPUTS

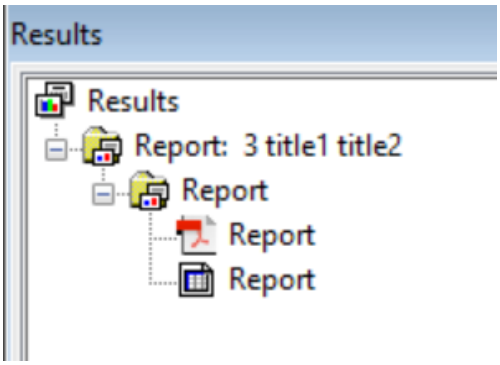
As discussed in the preparation section the TLF programs has to reserve space for the tiles of the final report. We reserved the first four titles for the final report and each TLF output has to start with the fifth tile line. Please note that it is possible to use output format specific escape sequences inside the PROC report but this leads to strange effects by replaying the content into other formats.

```
%LET MyTLFTitle1=My TLF Output Title;

TITLE;
OPTIONS NODATE NONUMBER;
ODS ESCAPECHAR="^";
ODS NOPROCTITLE;

%LET _m_newlines=;
%DO _m_i=1 %TO 4;
  %LET _m_newlines=&_m_newlines(*ESC*)n;
%END;

TITLE 5="&_m_newlines.&MyTLFTitle1";
```

During the output creation, the TLF report was printed as a PDF representation and an ODS DOCUMENT storage. We need the PDF representation later on to calculate the number of pages only. The content itself will be replayed from the document store.
To create the document storage in parallel to the PDF does not affect the performance. The calculation and processing will be done by SAS only once. For the performance, only the extra time for writing the second representation to the hard disk takes into account. The advantage of the ODS representation is that during the publishing process it is easy to add page numbers, headers and footers of the final report without parsing a specific output format (e.g. the PDF files). That means the programmer does not need a specific knowledge of the output format. The manipulation of these parts of the final report can easily done later on with normal SAS report features.

STORAGE OF THE TLF PROGRAM OUTPUT

The reports are stored in the ODC DOCUMENT stores. The location is specified in the definition of the ODS stream. On the filesystem, SAS creates a file with the extension " sas7bitm". In the results viewer of the SAS IDE the ODS stream output is shown and it is possible to view the content in the output window by choosing the context menu "Replay".
For the publishing process the chapter number and level, the title for the TOC and the location of the DOCUMENT store on the filesystem needs to be stored in one observation of the meta data file toc (See section providing the TOC META DATA above).



The code to write the outputs for the TLF Programs has only to open both streams by specifying the template (style) and the location of the result file for the output stream.

```
ODS PDF FILE="&_m_outpath.\&_m_filename..pdf" STYLE=&pdfstyle. STARTPAGE=NOW;
ODS DOCUMENT NAME=&_m_output.(WRITE);
ODS PROCLABEL="&chapter. &title1 &title2";
PROC REPORT DATA=indata NOWD CONTENTS='' NOCENTER
            STYLE (column)=[linkcolor=_undef_]
            ;

    FOOTNOTE1 j=l '^{style [URL="#toc" color=red linkcolor=_undef_]internal link to ToC }';
    FOOTNOTE8 j=r "&_exec_programname";
    DEFINE _v_holder / noprint order;
    BREAK BEFORE _v_holder /  CONTENTS="" PAGE;
RUN;
ODS _ALL_ CLOSE;
```

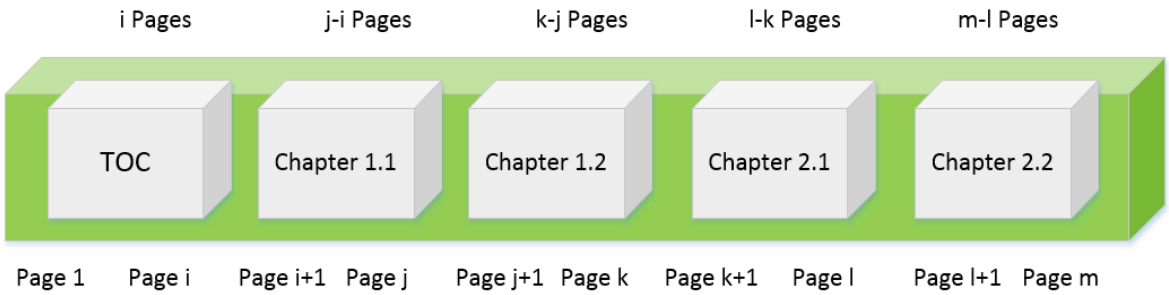
CALCULATION OF THE TOC

COUNT THE PAGES FOR EACH TLF OUTPUT.

We need to know the number of pages for each TLF program output. The easiest way is to count the pages of the PDF representation. The PDF representation was rendered by SAS and has exactly the number of pages that the replayed output in the final report needs. Here is a simple data step that counts the page breaks in the PDF file. The file reference INFIL shall points to the PDF file. It will be interpreted as a normal text even it has the PDF format. With regular expressions the number of occurrences of the page breaks ('/TYPE /PAGE') will be counted.

```
DATA _NULL_;
      LENGTH _v_countpages 8. _v_lastendtext $10. text2 $1048.;
      RETAIN _v_countpages 0 _v_lastendtext '';
        _v_reg=prxparse('/(\/Type *\/Page\b)/i');

      INFILE in RECFM=n LRECL=1024  ;
      INPUT text $1024.  ;

      text2=_v_lastendtext || text;
      _v_lastendtext=substrn(text,length(text)-9);

      start = 1;
      stop = length(text2);

      CALL prxnext(_v_reg, start, stop, text2, position, length);

      DO WHILE (position > 0);
        _v_countpages=_v_countpages+1;
         found = substr(text2, position, length);
         CALL prxnext(_v_reg, start, stop, text2, position, length);
      END;
      CALL symput ("&_r_numberOfPages",_v_countpages);
RUN;
```

For each PDF the number of pages has to be updated in the TOC. Please do not forget to produce also a TOC PDF and count the pages for the TOC PDF first and add the number of pages to the page count of all TLF outputs.



PUBLISHING

The publishing has now to replay all contents of the ODS DOCUMENT storages into one PDF. This is called replaying the content. The publishing program has to parse through all outputs and replay them into the PDF file in the correct sequence. First the table of content followed by all chapters.

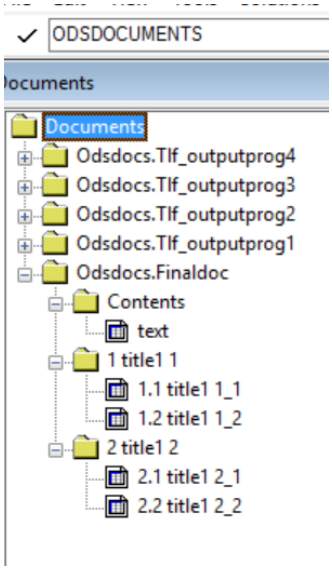
REPLAYING THE OUTPUTS

The publishing program opens a PDF Stream as followed sets the appropriate titles per output and replays the content with PROC DOCUMENT. The following code snippet shows the code to replay the content.

```
OPTIONS NOCENTER PDFPAGEVIEW=FITWIDTH PDFPAGELAYOUT=CONTINUOUS;
OPTION ORIENTATION= landscape ;
ODS DECIMAL_ALIGN=YES ;
ODS PDF FILE="&_m_outpath.\&_m_filename..pdf" STYLE=&finalstyle. STARTPAGE=NOW;

TITLE1 "&company (*ESC*){leaders  }%str( Page )^{thispage} of ^{lastpage}";
TITLE2 "Trial No.: T.ENTIMO";
TITLE3 "(*ESC*){style[textdecoration=underline]Table of Contents: ^{leaders  }}";

PROC DOCUMENT;
  DOC NAME=&libnameDocStore..&toc.;
  REPLAY;
QUIT;
```

WORKING WITH PROC DOCUMENTS

PROC DOCUMENT gives the programmer to work with the document stores without printing them. The ODS Document viewer (Type ODSDOCUMENTS in the IDE command window) shows the structure and properties of each document store and its content.



We define a new storage and replay first the toc into this storage:

```
PROC DOCUMENT;
   DOC NAME=&libnameDocStore..finalDoc;
   ODS pdf anchor='toc';
   REPLAY contentfinal;
QUIT;
```

The following PROC DOCUMENT call runs in a loop over the TOC meta data. With make and setlabel the nodes for chapter 1 and 2 are created. The variable `makedocnode` stores the unique name of the node with a path notation syntax, as it is used for the directory structure in filesystems. Please use the property function of the context menu to explore the naming convention. The nodes are numbered during creation. This leads to node names like table#1 for node of chapter 1 and table#1\lastnode#1 for the chapter 1.1.
The copy statement of PROC DOCUMENT (in the example below) copies the output of the TLF program, that is saved in the &`document_storage_name` storage. The source node name is per default `\Report#1\Report#1` because we did not specify a node name during the creation. The target node is located in a new storage named finaldoc. The `makedocnode` variable holds the path to the next chapter without any outputs (e.g. \table#1, \table#2, ...). These chapters are called "Directories". The variable `targetDocNode` holds the path to the current directory node (e.g. \table#1, \table#2, ...) where the report should be linked to. The PROC DOCUMENTS statements set label (for the label that is shown in the viewer) and objtitle (for the titles that replay will use) edit the properties of a node. The code that derives the names and locations from the meta data is not included in the paper because it is very simple data step with some string manipulations.

```
PROC DOCUMENT;
       DOC NAME=&libnameDocStore..finalDoc;
      .....
      %IF %LENGTH(&document_storage_name) eq 0 %THEN %DO;
          make &makedocnode.;
          setlabel &makedocnode. "&chapter &toc_entry";
       %END;
       /* Create note with TLF output */
       %ELSE %IF %LENGTH(&document_storage_name) ne 0 %THEN %DO;
          * copy TLF output;
          copy \&libnameDocStore..&document_storage_name.\Report#1\Report#1
               to &targetDocNode.\lastnode;
          setlabel &targetDocNode.\lastnode "&chapter &toc_entry";
           *set title of page;
          obtitle1 &targetDocNode.\lastnode "(*ESC*){leaders  }%str( Page )^{thispage}
of ^{lastpage}";
          obtitle2 &targetDocNode.\lastnode "Trial No.: T.ENTIMO";
          obtitle3 &targetDocNode.\lastnode
"(*ESC*){style[textdecoration=underline]Chapter &chapter &toc_entry ^{leaders  }}";
       %END;
       /*page break after report*/
       OBPAGE   &targetDocNode.\lastnode ;
       %END;
QUIT;
```

The OBPAGE statement after each TLF output produce a page break. When the store is replayed, this forces the ODS system to start each output on a new page.

All data and information are now copied into the finaldoc ODS storage. This can now easily replayed into a PDF stream

```
ODS pdf file="&pdfout." style=&styleREPORT. startpage=now bookmarkgen
             bookmarklist=show NOCONTENTS

PROC DOCUMENT;
       DOC NAME =&libnameDocStore..finalDoc;
       REPLAY &_m_mainnode.#&_m_mainnodecount.;
QUIT;
```

If you want to extract information that are stored in a ODS Storage, the following code example is helpful. It stores the properties in a dataset:

```
ODS OUTPUT properties=singleDocProperties;
    PROC DOCUMENT;
       DOC NAME =&libnameDocStore..&document_storage_name.;
        list / details levels=all;
    QUIT;
ODS _ALL_ CLOSE;
```

CONCLUSION

SAS provides with the DOCUMENT stores and PROC DOCUMENTS a well-engineered persistent storage technique for the SAS output delivery system. It enables programmers to use results without rerunning the calculation programs and focus the work of the programmer on the layout and not on the content of the report.
The one and only drawback of the ODS storage format is, that this is a proprietary SAS technique. The reports in ODS storages not accessible from any other programming language.

CONTACT INFORMATION

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

Sven Prasse
Entimo AG
Stralauer Platz 33-34
10243 Berlin
Germany

Work Phone: +49 30 52 00 24 100
Fax: +49 30 52 00 24 101
Email: pr@entimo.de
Web: www.entimo.de

Please contact me via email for source code examples.