Paper 086-30

STORE YOUR OUTPUT AND DIGEST IT LATER

Frank Poppe, PW Consulting, the Netherlands

ABSTRACT

The appearance of output can be made to conform almost exactly to your wishes, and for a wide range of destinations. The appearance is influenced by the ODS styles, and for the destination you can select one of the standards available (HTML, PDF, RTF) or generate new ones through a tagset.

However, the order of the output is fixed. There is no way to move an overview table of results of different steps up front. You cannot start with a graphic that summarizes the results because the procedures have to run first. Within a procedure you cannot move elements around. This fixation of the output also makes the choices of style and destination final--when procedures have run, the output cannot be reproduced with another style, or for another destination, unless you preserve all the data.

Now there is the DOCUMENT Procedure (experimental in SAS® version 8, but now production). The output can be stored in its "naked" form--the data is frozen, but the style has not been applied yet--using a generic format that still can be fed to any destination. The storage can be used temporarily in the same job, or it can be in "deep freeze." Then you can reuse old test results in a new PDF document using current guidelines on style, without having to rerun statistical tests, without the need for the original data.

This paper will give an overview of the possibilities and the user interface.

INTRODUCTION

Before embarking on the ODS Document destination I first want to introduce the different elements of the ODS system – as I see it.
What happens behind the screens when a SAS procedure or data step produces output? Different parts of SAS, and in particular ODS, come into action, acting together or one after the other, to produce HTML, RTF or LaTeX, or whatever. This can be written to a file, or directly to a webserver, which will pass it on to someone's browser, or as an e-mail application to one or many addressees, to mention a few possibilities.
What I will describe here are the elements in the way I as a SAS programmer see them. What really exactly happens may be slightly differently, but I think this is practically speaking not very far of. I've based this description on the 'official' documentation (as laid down in the OnLineDocs), on the less official information that can be found in the Base Community pages on the SAS website, and on e-mail correspondence with some of the SAS developers that maintain those pages.
And of course on lots of experiments, trying to discover why things didn't work out the way I initially expected…

The elements are shown together in Figure 1 on the following page.
At the top in the rectangular boxes you see the four 'sources' that build up the output:

- ODS destination
- ODS style
- the table definition
- the data.

Above those boxes it is indicated how you can change and add items to those source. The data of course comes from the DATA and PROC steps. The other three can be managed through the TEMPLATE Procedure, with the DEFINE statement, for tagset, style and table, respectively.

With an ODS <*destination*> <*style*> … statement you bind destination and style, opening up something I prefer to call an output stream. When you then execute a PROC step that creates output, or run a DATA step using PUT _ODS_ statements, this creates an output object. In this output object the data is stored in a structured way. These are mostly tables but an output object can be a graph as well (think of the GRSEG entries in a graphic catalog).
If there are any output streams open, the output is send to these streams. Each streams does its own thing, transforming the output objects into its own format, taking into account the style specification where possible.
But the objects can also be stored in a document store, which can be made permanent by assigning it to a permanent libref (i.e., not WORK).
The remainder of this paper will deal with this document store.

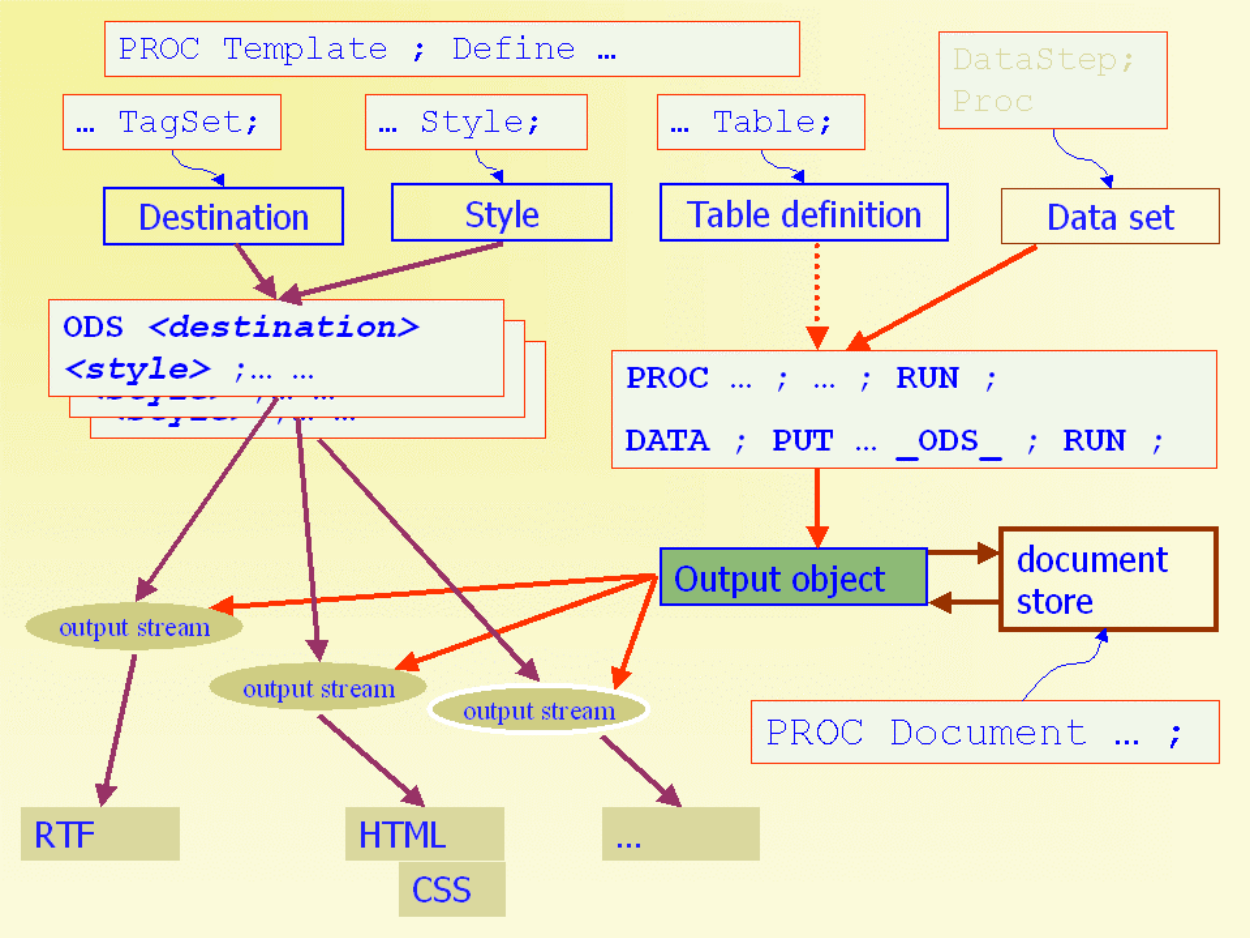

Figure 1. The elements of ODS

THE DOCUMENT STORE – IN SHORT

Let us now concentrate on the document store.

The picture shows that there 'output objects' are stored. The SAS procedure has done its work so the data is fixed forever in a structured way.

But you can also see from the picture that format (HTML, RTF, etc.) or style has not come into play yet. One might say the objects are still in a virginal state. The 'context' in which the objects were created is saved together with the store, but in such a way that we can decide later what we want done with them

In the following pages I will first show with some examples what we essentially can do with the document store. After that I will describe the main commands in some more details, without turning that in a complete syntax description.

EXPLORATION BY EXAMPLE

Let us first make sure that we have some in store. Somewhere in the past the following bits of code have run. Below I present all the code, with some short comments, so that it is clear what we will work with.

```
ODS _ALL_ Close ;
```

Close all the ODS destinations.

```
PROC SORT data = sashelp.class out = class ;
BY sex ;
RUN ;
```

These were some preparations. Even if there would be any output, that is not relevant to keep in store. The next piece of code *will* produce 'interesting' output, so now we open the document store.

```
ODS DOCUMENT
    name = sasuser.demo  (write)
    dir = ( path = /examples/stat label = 'The original result' )
;
```

The ODS Document statement opens the store, called *sasuser.demo*.

The SAS Explorer will not show anything when you look in the *sasuser* libref, but if you check the location where that libref points to, you will find this just created a file called *demo.sas7bitm*. This is a binary file, not readable by human eyes. The path we just defined with the *dir*= option, and anything that will happen later, exist only within that store.
Now we will produce some output. To be able to check later when all this was produced we put the current date and time in the title.

```
%LET date = %SYSFUNC ( date () , eurdfwkx. )  ;
%LET time = %SYSFUNC ( time () , hhmm8.2   )  ;

PROC GLM data = class ;
TITLE "GLM analyses, dated &date, &time" ;
BY sex ;
MODEL weight = height age  ;
OUTPUT out = sasuser.glm predicted = p ;
RUN ;
QUIT ;
```

Note that we closed all output destinations except the document store, so no readable output will have been produced. The Results window in SAS will show the elements produced by the Output statement, but even SAS itself will not know what to do with it (it is better not to double click the items).
Of course the Output statement did produce the data set *sasuser.glm*.

The next step is to do something with that data set. We set some graphic options to improve the readability of the picture.

```
GOPTIONS ftext = "Verdana" htext = 10 pt ;

SYMBOL1 value = dot  interpol = nont color = blue ;
SYMBOL2 value = plus interpol = none color = red  ;

PROC GPLOT data = sasuser.glm gOut = sasuser.plot ;
PLOT ( p weight ) * height / overlay name = 'demo1';
RUN ;
QUIT ;
```

This still will have produced nothing that we actually can look at, as there is no ODS destination open other then Document.
Now let us have a look at what we have. We will run the DOCUMENT Procedure to do that, and that in itself produces output: listings, etcetera. We do not want that output in our document store, so first we close the ODS Document destination, and then we open the destination where we want our information from PROC Document.

```
ODS Document Close ;

ODS RTF
    file = 'c:\temp\doc.rtf'
    style= sasDocPrinter
;
PROC DOCUMENT name = sasuser.demo ;
list / levels=all  ;
run ;
ODS RTF Close ;
```

The output of PROC Document has now been written to an RTF file. I have inserted that document in the one your are now reading, and the result is below (I only adjusted the margins and the font size).

Listing of: \Sasuser.Demo
Order by: Insertion
Number of levels: All

Obs	Path	Type
1	\examples#1	Dir
2	\examples#1\stat#1	Dir
3	\examples#1\stat#1\GLM#1	Dir

Listing of: \Sasuser.Demo
Order by: Insertion
Number of levels: All

Obs	Path	Type
4	\examples#1\stat#1\GLM#1\ByGroup1#1	Dir
5	\examples#1\stat#1\GLM#1\ByGroup1#1\Data#1	Dir
6	\examples#1\stat#1\GLM#1\ByGroup1#1\Data#1\NObs#1	Table
7	\examples#1\stat#1\GLM#1\ByGroup1#1\ANOVA#1	Dir
8	\examples#1\stat#1\GLM#1\ByGroup1#1\ANOVA#1\Weight#1	Dir
9	\examples#1\stat#1\GLM#1\ByGroup1#1\ANOVA#1\Weight#1\OverallANOVA#1	Table
10	\examples#1\stat#1\GLM#1\ByGroup1#1\ANOVA#1\Weight#1\FitStatistics#1	Table
11	\examples#1\stat#1\GLM#1\ByGroup1#1\ANOVA#1\Weight#1\ModelANOVA#1	Table
12	\examples#1\stat#1\GLM#1\ByGroup1#1\ANOVA#1\Weight#1\ModelANOVA#2	Table
13	\examples#1\stat#1\GLM#1\ByGroup1#1\ANOVA#1\Weight#1\ParameterEstimates#1	Table
14	\examples#1\stat#1\GLM#1\ByGroup2#1	Dir
15	\examples#1\stat#1\GLM#1\ByGroup2#1\Data#1	Dir
16	\examples#1\stat#1\GLM#1\ByGroup2#1\Data#1\NObs#1	Table
17	\examples#1\stat#1\GLM#1\ByGroup2#1\ANOVA#1	Dir
18	\examples#1\stat#1\GLM#1\ByGroup2#1\ANOVA#1\Weight#1	Dir
19	\examples#1\stat#1\GLM#1\ByGroup2#1\ANOVA#1\Weight#1\OverallANOVA#1	Table
20	\examples#1\stat#1\GLM#1\ByGroup2#1\ANOVA#1\Weight#1\FitStatistics#1	Table
21	\examples#1\stat#1\GLM#1\ByGroup2#1\ANOVA#1\Weight#1\ModelANOVA#1	Table
22	\examples#1\stat#1\GLM#1\ByGroup2#1\ANOVA#1\Weight#1\ModelANOVA#2	Table
23	\examples#1\stat#1\GLM#1\ByGroup2#1\ANOVA#1\Weight#1\ParameterEstimates#1	Table
24	\examples#1\stat#1\Gplot#1	Dir
25	\examples#1\stat#1\Gplot#1\Demo1#1	Graph

It is clear that asking for *levels=all* produces quite extensive information. There is a line for each level in the directory, and one for each item. The items all are in the path \examples\stat, like we specified when opening the store. Beyond that that the directory path is determined by first the procedure name and, if a BY statement was used, the BY group number. The rest of the path (and the depth of it) and the names of the items itself depend on the procedure. E.g., for the output of the GPLOT Procedure the *name*= option on the PLOT statement is obeyed.
And what is the '#1' doing behind each part of the path?
This is a sequence number: if code would run that, according to the naming logic just outlined, would produce the same path, the items would be stored in that path, with a '#2' (and so on) attached.

Now a store is only useful if we can do something with its contents. In this case, 'replaying' it. For that purpose there is the replay command. As default it will replay everything in the current document store. But you also can specify any of the paths listed in the overview above. If you specify a directory it will replay everything 'below' that point, if you specify an item, it will just give you that.

For the output this will produce, we now open the HTML destination, with the SASweb style selected.

```
ODS HTML file = 'doc.html' style = sasWeb ;
replay \examples\stat\glm\bygroup2\anova ;
replay \examples\stat\glm\byGroup1\anova ;
run ;
ODS HTML close ;
```

Figure 2 gives a screen shot of the result. We have asked for all the ANOVA results of PROC GLM, but the result from the second BY group (the males) is presented first, then the first group (female).
We can re-run this to any destination, with any style. Figure 3 gives an impression of how it looks like it we change the style to Sketch.
Note that the title (with the date and the time), stored with the output in the document store, remains the same.

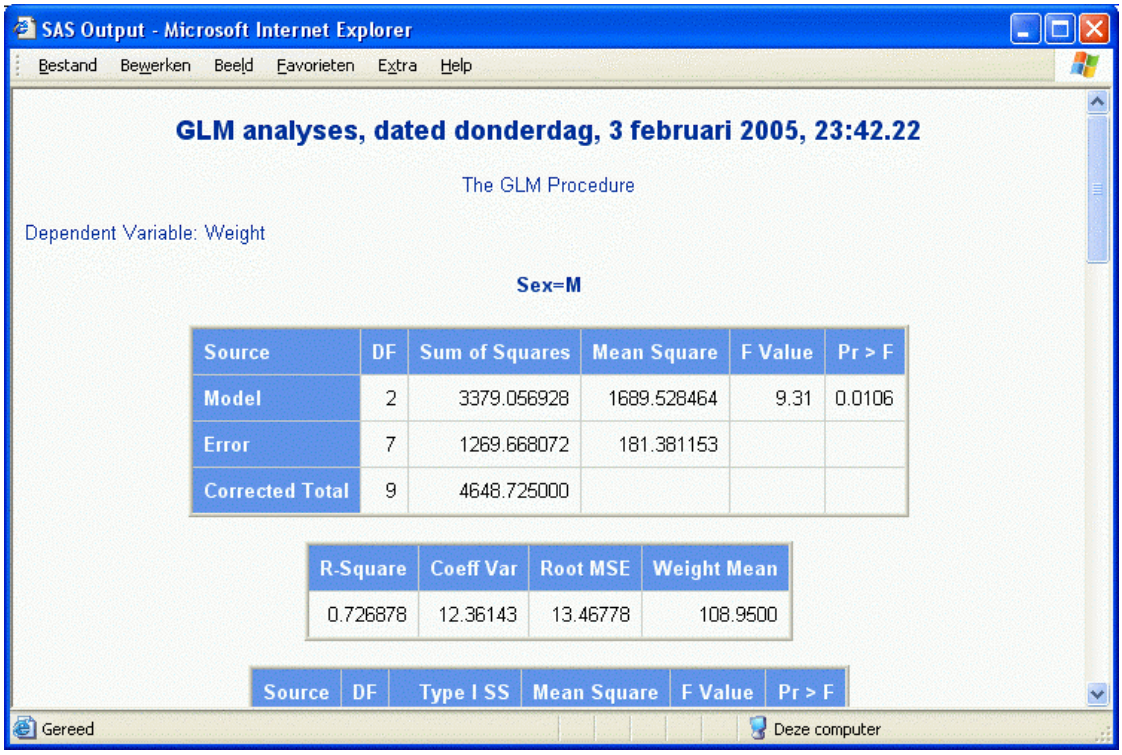

Figure 2. Results in SASweb style

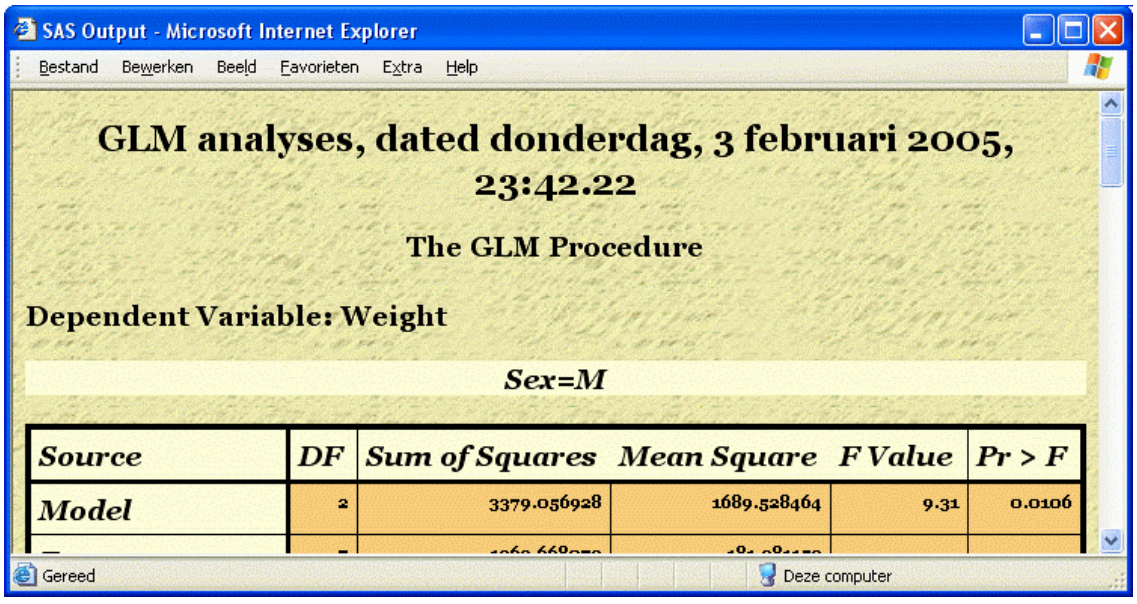

Figure 3. GLM results in another style

Let us further assume that this process is run regularly on each time updated data. The result as selected before, and in the order presented there, have to be produced each time, to be made available to the management.
For that we create a new document store, and from there link to the items we need from the store with our results.
The code for that might be like this.

```
PROC DOCUMENT ;
doc name=sasuser.execinfo ( write ) ;
note myNote 'Order is changed, first BY Group is M' ;
link \sasuser.demo\examples\stat\glm\bygroup2\anova to males ;
note MyNote 'Second BY group now is F' ;
link \sasuser.demo\examples\stat\glm\bygroup1\anova to females ;
ODS listing ;
list ;
run ;
ODS listing Close ;
ODS HTML body = 'exec.html' style = beige ;
replay ;
run ;
ODS HTML close ;
QUIT ;
```

A new document store is created, again in *SASuser* (this will create an *execinfo.sas7bitm* file). A note is inserted in the document store, under the name 'myNote'. Then the link is created, to a specific path in another document store, and the item is called 'males'. Then this is repeated for the females, creating another note and the item 'females'.
Just to check the *list* command is given, with output to the ODS Listing, and the whole thing is replayed to the HTML destination.
Be sure to add a *run* statement before closing a destination. As with nearly everything in SAS all statements are put in a queue and only executed when a *run* statement is encountered. Results are only written to the destinations that are open at that moment.
The result from the list command is cut from the Output window and pasted in below.

```
Listing of: \Sasuser.Execinfo\
             Order by: Insertion
             Number of levels: 1

               Obs       Path                                                Type
             ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
                    1    myNote#1                                            Note
                    2    males#1                                             Link
                    3    MyNote#2                                            Note
                    4    females#1                                           Link
```

Note that the second 'myNote' now has a sequence number of #2, and note too that the names are not case sensitive. From now on the document in sasuser.execinfo can be treated as an independent document, that can be replayed (e.g. to the PDF destination), moved, copied or made part of a bigger document.

THE STATEMENTS

After this exploration we now take a closer look at the statements that are available. This is only a short description; for a more complete and exact description turn to the documentation (see the References at the end).

DOC, DIR AND LIST

First we look at those statements enabling us to see what we have in store.
Doc without a parameter will give an overview of all the document stores that can be found in all the *librefs* that are currently defined. You can specify a document store with the *name=* option. That will make that document store the current one, and all further actions will act upon and in that document store.

Dir without a parameter will show the current path (when a document has been selected). With a parameter you change the current path. All further action will by default start from that path.
If a path starts with a \, it is interpreted as an 'absolute' path, so starting at the root of the document. Such a path can also be given with a document name. The document name is specified as libref.docstore. So it the first element of an absolute path contains a dot, it points explicitly to a document store, otherwise it refers the current document.

Path that do not start with a \ are relative to the current path.

List will give the contents, by default of the current path in the current document. It has several options. The option *details* will add more information about the size of the items, the date of creation, etc. With the option *levels*= the command can be instructed to follow the directory several levels 'down'.
The most interesting option is the *order*= option. Two of the possible values are not very surprising: *date* and *alpha*. The interesting third value is *insert*. Now most directories (like the ones you see with the Windows Explorer and the SAS Explorer) have the ability to show the contents in the order of several attributes like dates and names. But insertion order is usually not among them. If you move or copy an item to a directory, it does not matter if you drop it exactly between two items: as soon as you drop it it gets a place in the order of the attribute currently in force.
The document store however maintains an insertion order, and that is the default for commands like list and replay. In fact, we have seen that already in force; in the examples above the items were shown in the order we inserted them, not in any other order.
The next paragraph on moving and copying items shows how the insertion order can be exploited.

MAKE, MOVE, COPY AND LINK

The *make* command only creates one or more directory entries (as parameter you can give one or several paths). These have no content, but are placeholders into which you later can place items. You can add one option, with which you can specify where this directory item should be inserted. The possible values are *last* (the default, i.e. after all the existing entries), *first*, *before*= or *after*=. With the *before*= and *after*= values you can specify an item you want to use as a reference point to insert before or after.
Example:

```
make placeholder / first ;
```

will create a new entry called 'placeholder' before any of the existing entries, in the current path (which may be set with the *dir* command to somewhere deep down in the document store). Sequence number are added automatically and chronologically (meaning that if you add an item somewhere before an item with the same name, a listing in insertion order will show #2 before #1).

The *move* command has a similar syntax, but now you add the target location after the *to* keyword. The path or paths you move should exist already. Within the target location you can specify again with the same options as before where the new items should be inserted. The move command also support the *levels*= option. The default again is to move all levels.
This is a good place to introduce the special notation to denote the current location: the circumflex
Example:

```
move demo1 to ^ / last ;
```

This moves 'demo1', which should exist in the current directory, to the last position in the same directory.
Similarly to Unix notation, the double circumflex ^^ denotes the parent directory (It was apparently not possible to use the dot or double dot notation here).

The *copy* command has the same syntax as the *move* command, and will act the same, only of course it will make a copy in stead of moving items.

The *link* command is subtler. Basically it has the same syntax as *move* and *copy*, except that the source can only be a single path. The source of a link does not have to exist at the moment you create the link. When replaying a document that contains a link of which the source does not exist a warning is issued. The same happens when you list the contents with the follow option.
The link can also be specified with the *hard* option. This requires that the source already exists at that moment (and in the same document store). This way you create a new item, with its own place in the document store and its own name but pointing to the same data object. Perhaps a better way to call it would be a soft copy. The underlying data object will continue to exist as long as any hard links to it exist. In other words, deleting the source of a hard link will not delete the target (nor make it unusable). Deleting the source of a 'normal', symbolic, link will make the target unusable.

OTHER COMMANDS

The *replay* command has been introduced already. An option worth mentioning here is the *dest*= option. Only the ODS destinations that you specify there will receive the output, although more can have been opened. That way you can open a range of ODS destination, but using different replay commands with different *dest*= specifications you can determine what goes where.
The *rename* command obviously renames items.
Furthermore there is a range of commands to insert and modify notes, titles and footnotes.

CONCLUSIONS

The Output Delivery System with its different destinations is already a powerful instrument to get your output in the format and the style you require. The addition of the Document destination and the Document procedure is a powerful

tool to re-organize the different parts of the output of almost any SAS procedure to the requirements of an organization. A notable exception to this is that the REPORT Procedure currently is not supported.
The ability to store results and replay them later to different destination or in different style improves the efficiency.

REFERENCES

"SAS Output Delivery System: User's Guide" <http://support.sas.com/onlinedoc/912/docMainpage.jsp> (4feb2005), Select ►Base SAS ► SAS Output Delivery System: User's Guide ► The DOCUMENT Procedure ► The DOCUMENT Procedure.

"The ODS Document Resources" <http://support.sas.com/rnd/base/topics/odsdocument> (4feb2005). Part of the SAS Base Community webpages.

CONTACT INFORMATION

Any comments, questions or suggestions are welcome. Below you find the details.

Frank Poppe
PW Consulting
The Netherlands
Email: Frank.Poppe@pwcons.com
Web: http://www.pwcons.com
Mobile phone: +31 6 21218860
Fax: +31 8 4222 6612