

## **The Output Bundle: A Solution for a Fully Documented Program Run**

Carl Herremans, MSD (Europe), Inc., Brussels, Belgium

### **ABSTRACT**

Within a biostatistics department, it is required that each statistical result reported to the regulatory authorities is traceable (as per CFR21 Part 11). Hereto, it is necessary to know exactly what resources were used to produce the results. Typical resources are input data (e.g. SAS datasets, Excel files, ...) and programs (included SAS programs, macros, ...). Each resource has to be identified in a unique way in order to allow exact reproduction of the reported results.

Keeping manual track of the resources and reported results is a tedious and labor intensive task. The author has developed a SAS/AF application to allow end users to execute their SAS code in a documented fashion. The application is able to execute the SAS program and keep track of all resources used and all output created. The information will be stored in an Excel file referred to as the *output bundle*.

### **INTRODUCTION**

One of the challenges facing the community of SAS® program developers in the pharmaceutical industry is the documentation of their programs; what are the exact resources used and what are the outputs generated? This is a requirement per good business sense in order (i) to respond quickly to questions from regulatory authorities related to the results reported, and (ii) to prove the validation state of all programs used to obtain these results.

Programs developed within the departments of Clinical Biostatistics and Statistical Programming are typically one-off SAS® programs for a specific study instead of being part of integrated software [see ACDM/PSI guidelines]. Due to the one-off nature of the SAS® programs, documentation is a significant cost in the overall program development. It is possible to automate the documentation process as demonstrated by F. Coppin and C. Herremans (2003) and (2004) during the program development, validation and maintenance stages. However, an important piece of information of the one-off SAS® programs is only available at the moment of the production run which is the exact list of what input data (e.g. SAS datasets, Excel files, ...) and programs (included SAS programs, macros, ...) are used and what results are produced (e.g. SAS datasets, listings, graphics, Excel files, ...). These single pieces have to be uniquely identified at the moment the program is executed, and can be lumped together in a document referred to as *output bundle*.

This paper presents and discusses the automated construction of the output bundle. It explains how the necessary information is collected after program execution takes place and is lumped together into the output bundle.

### **STANDARD DIRECTORY STRUCTURE**

The output bundle fits in the context of the programming development environment within pharmaceutical companies supporting the process of a New Drug Application [NDA]. The development environment comprises several applications as well as a set of conventions. One of the conventions is a standard directory structure [SDS].

Within each SDS, all programs, input and output datasets as well as SAS log- and SAS lst-files belong to a specific directory. For each SDS (i.e. for each study) an autoexec.sas file is available where filenames, libnames, SAS-options and a set of global macro variables are defined (see, A. Dynder, B. R. Cohen, G. Cunningham (2000)).

### **APPLICATION**

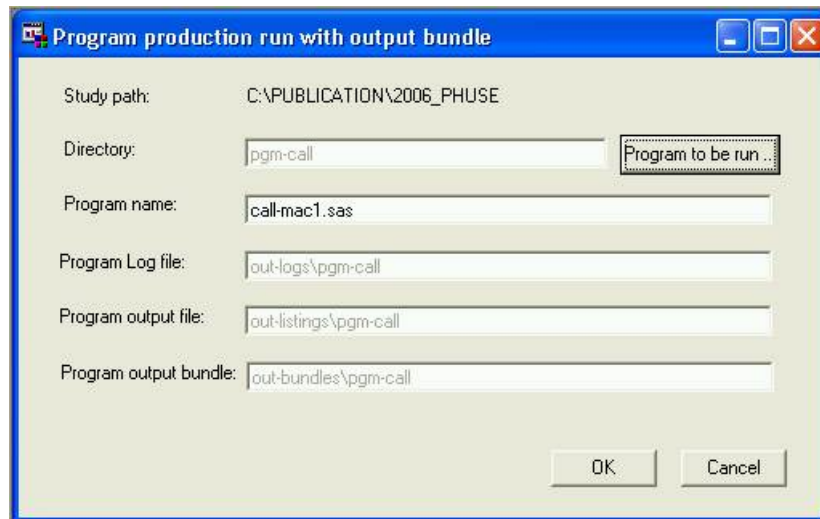
The output bundle application is accessible to the end user through a standard programming toolbar [F. Coppin and C. Herremans (2003) and (2004)]. It is a customized SAS toolbar available to the user from the SAS version 9 enhanced editor (See Figure 1). This toolbar supports different stages of the programming development life cycle (PDLC): development, testing and production. The addition of the output bundle is a natural extension to this programming toolbar.

## PhUSE 2006



**Figure 1:** Output bundle application harbored on the standard programming toolbar.

Figure 2 displays the GUI of the output bundle application allowing the user to select the one-off SAS program to be executed. Based on the path of the program selected, the corresponding locations for the LST files, LOG files and output bundles is derived according to the conventions of the standard directory structure [SDS]. Once the user confirms his selection, the program is executed and the corresponding output bundle is produced.



**Figure 2:** Output bundle GUI allowing the user to select the one-off SAS program to be executed.

### APPLICATION FLOW

Five steps are defined in the application flow:

1. Characteristics of the SAS session
2. Execution of the one-off SAS program
3. Extraction of the resources used and the outputs created based on the SAS LOG
4. Unique version identification of the resources used and the outputs created
5. Creation of the output bundle

### CHARACTERISTICS OF THE SAS SESSION

The characteristics of the SAS session prior to the one-off SAS program execution have to be determined. This covers, for example; the SAS options set as well as libnames and filenames assigned. These are typically defined by a global study autoexec.sas file.

### EXECUTION OF THE ONE-OFF SAS PROGRAM

The second step consists of the execution of the selected one-off SAS program. At this stage, specific SAS system options are activated by the application to ensure that the necessary information about the resources used and outputs created is displayed in the SAS LOG [see, H. Ament (2005)].

## PhUSE 2006

Examples of these options are:

- NOTES: display the NOTES in the SAS LOG window
- SOURCE: display the source code in the SAS LOG window when executed
- SOURCE2: display the source code in the SAS LOG window for programs included

### EXTRACTION OF THE RESOURCES USED AND THE OUTPUTS CREATED

Resources used and outputs created are identified based on the information available in the SAS LOG. In the following sections, two cases are discussed in more detail to illustrate the methodology applied to extract the necessary information from the SAS LOG.

#### USED SAS MACROS

Macros can be compiled in the SAS session in two manners: (i) using the SAS autocall option and (ii) using an explicit *include* of the macro. The output bundle application has to be able to deal with both methods.

1. In SAS 8, the macros loaded through a SAS autocall were very difficult to trace. Thanks to a new option in SAS 9 [AUTOLOCDISPLAY], a NOTE is displayed the moment the SAS autocall macro is used.

```
4  %mac1;
   MAUTOLOCDISPLAY(MAC1):  This macro was compiled from the autocall file
                           F:\Publication\2006_phuse\macrolib\mac1.sas
```

Based on this NOTE, the exact origin of the compiled SAS MACRO can be extracted. In this example it can be seen that the macro mac1 is compiled from the source file "F:\Publication\2006\_phuse\macrolib\mac1.sas".

2. The Macros compiled by explicit include of the file are easy to retrieve based on the SOURCE2 SAS NOTE. As illustrated below, an explicit NOTE appears stating the include of a file located at "F:\Publication\2006\_phuse\macrolib\mac1.sas".

```
5  %include "&protpath\macrolib\mac1.sas";
   NOTE: %INCLUDE (level 1) file F:\Publication\2006_phuse\macrolib\mac1.sas
        is file F:\Publication\2006_phuse\macrolib\mac1.sas.
6  +%macro mac1;
7  +  %put this is a demonstration macro;
8  +%mend mac1;
   NOTE: %INCLUDE (level 1) ending.
```

#### INPUT/OUTPUT SAS DATASETS

The input/output SAS datasets can be extracted from the notes available in the SAS LOG. In the following example, the input/output dataset information can be easily obtained from the SAS LOG notes:

```
6  proc sort data=sashelp.shoes
7         out=sasuser.shoes;
8         by region;
9  run;
```

```
NOTE: There were 395 observations read from the data set SASHELP.SHOES.
NOTE: The data set SASUSER.SHOES has 395 observations and 7 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time          0.12 seconds
      cpu time           0.05 seconds
```

However, the issue with this approach is that not all procedures display a NOTE with the input/output dataset. Typical examples of this are the PROC IMPORT and the PROC SQL procedures. The corresponding SAS NOTES for the PROC SQL is displayed below:

## PhUSE 2006

```
1   proc sql;
2       create table shoes as
3       select *
4           from sashelp.shoes;
NOTE: Table WORK.SHOES created, with 395 rows and 7 columns.

5   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          1.07 seconds
      cpu time           0.16 seconds
```

Based on this information it is possible to identify the SAS dataset created but no information is displayed in the SAS LOG about the SAS dataset used. In order to retrieve the input dataset, a scan of the source code is necessary.

### RESULT OF EXTRACTION

The result of the SAS LOG analysis is an explicit list of resources used and outputs created. Their exact location is identified by their full filename, i.e., the filename and the folder name where the file is stored. For example, F:\Publication\2006\_phuse\macrolib\mac1.sas is the full path for the file "mac1.sas" which is stored in the folder "F:\Publication\2006\_phuse\macrolib".

In order to have a machine independent mapping of the full filename, the drive letter must be translated into its UNC-name which consists of the server name and the share name. If, for example, the F-drive corresponds to \\MSD0001\SHARE02, the corresponding machine independent full filename is \\MSD0001\SHARE02\Publication\2006\_phuse\macrolib\mac1.sas. The collection of this UNC-name can be done using following code:

```
71  filename uncname pipe 'net use f: ';
72
73  data _null_;
74      infile uncname length=len;
75      input tt $varying70. len;
76      if tt='Remote name'
77          then do;
78              UNC_name = trim(substr(tt,12));
79              put UNC_name =;
80          end;
81  run;
```

```
NOTE: The infile UNcname is:
      Unnamed Pipe Access Device,
      PROCESS=net use f:,RECFM=V,LRECL=256
```

```
UNC_name=\\MSD0001\SHARE02
```

### UNIQUE VERSION IDENTIFICATION OF THE RESOURCES USED AND THE OUTPUTS CREATED

The machine independent full filename is not sufficient for a unique identification as this file might change over time. Typically the outputs are overwritten after each run of the one-off SAS programs. The full filename needs to be extended with a counter to identify the exact version of the file. In order to make the application independent of other third-party software, the last modification date of the file is used as the unique identifier. For example, the macro mac1 is uniquely identified by its full filename 'F:\Publication\2006\_phuse\macrolib\mac1.sas' and its last modification date 'May 01, 2006, 10:53:09 PM'.

The files' last modification date can be retrieved using the Microsoft API's *FileTimeToLocalFileTime* and *FileTimeToSystemTime* stored in the kernel32.dll. These API's can be accessed using the SAS® Modulen function which allows to call directly a specific routine or module that resides in an external dynamic link library (DLL). Excellent tutorials on the use of the Microsoft API's from SAS® using the MODULEN function can be found in Roper (2001) and Johnson (2005).

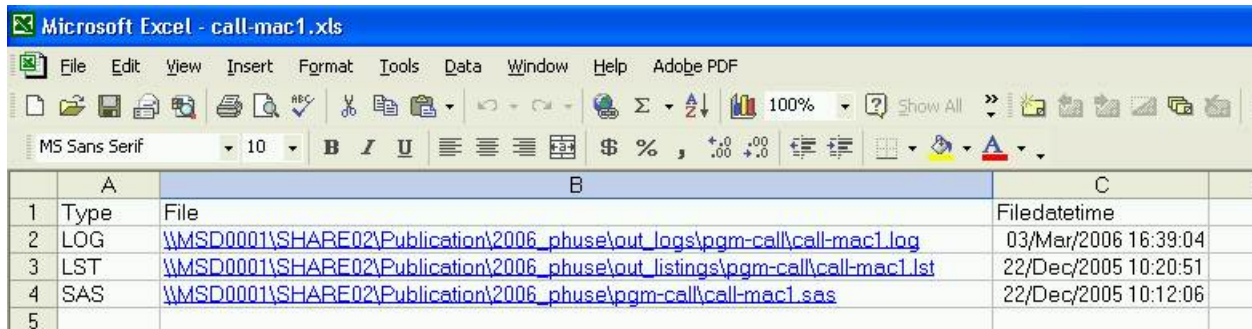
### CREATION OF THE OUTPUT BUNDLE

The collected information (described in the previous sections) form the content of the output bundle. MS-Excel format was chosen to store this information and is easily accessible to the end user in one single database-like file.

The following sheets are distinguished within the output bundle:

Sheet name	Description
SystemFiles	this sheet contains the location of the program executed and the LOG/LST files created
Inputfile	this sheet contains the full filename for the resources used by the program (SAS datasets, MS-Excel sheets, ...)
Outputfile	this sheet contains the full filename for the results created by the program
Macrovariables	this sheet contains the global macro variable and their values present prior to the execution of the one-off SAS program
Libname	this sheet contains the libnames present prior to the execution of the one-off SAS program
Filename	this sheet contains the filename present prior to the execution of the one-off SAS program
Format	this sheet contains the formats present prior to the execution of the one-off SAS program
Option	this sheet contains the SAS options set just prior to the execution of the one-off SAS program
Systeminfo	contains the SAS version, operating system and username of the account that creates the output bundle

An example of the resulting output bundle for the program call-mac1.sas is illustrated in Figure 3..



**Figure 3:** Sheet SystemFiles for the Output bundle of the program call-mac1.sas.

In order to create this multi-sheet Excel file, the SAS 9 Excel libname engine was used.

```
libname bundle EXCEL "&Bundlepath\&bundle_name..xls";
```

Note that by default, Excel will assign a date format (only showing the date) to the column containing the date time stamp. The date time can be visualized by applying the custom excel format "dd/mmm/yyyy hh:mm:ss" to the column containing the date time values. If these sheets are imported in SAS without explicitly applying the appropriate date time format, only the date part is preserved. In order to solve this issue, the option SASDATEFMT must be set when reading the excel sheets into SAS.

```
libname bundle EXCEL "&Bundlepath\&bundle_name..xls";
```

```
data systemfiles;
  set bundle.systemfiles (SASDATEFMT=(Filedatetime=datetime18.));
run;
```

## CONCLUSION

This paper proposes a solution to automate the program documentation for the production run of a one-off SAS program. By integrating this solution on the standard programming toolbar, the output bundle application was made available to the user community of one-off SAS® programs in a user friendly way.

The cost of this application consists of the design, development and maintenance of the SAS/AF® application and in the observance/compliance to certain programming conventions. The return of investment is the automated documentation of the program executed.

## TECHNICAL DETAILS

The proposed standard programming toolbar is developed using the SAS/AF® software in SAS® 9.1. As shown, only GUIs are developed using SAS/AF. The source code is a mixture of macro, open and SCL SAS® codes.

Several functionalities of the toolbar require an interaction with the OS. As this toolbar was developed for Windows XP platform, Microsoft Windows APIs are used in order to manage these interactions.

## REFERENCES

- ACDM/PSI, 'Computer Systems Validation in Clinical Research – A Practical Guide', 1st Edition, Aug 1998.
- A. Dynder, B. Cohen, G. Cunningham (2000), 'SAS Macro for Creating a Standard directory Structure', Proceedings of the Pharmaceutical industry SAS® Users Group Conference 2000, application Development paper 1.
- C.A. Roper (2001), 'Accessing and Utilizing the Win32 API From SAS', Proceedings of the Twenty-Sixth annual SAS Users Group International Conference [SUGI], Systems Architecture section, paper 281-26.
- F. Coppin and C. Herremans (2003), 'A standard SAS® programming toolbar: A step forward for GPP/SOP compliant SAS® program development', Proceedings of the Annual Conference of the Pharmaceutical industry SAS® Users Group [PharmaSUG], Application Development section, p. 59-63.
- F. Coppin and C. Herremans (2004), ' SAS®-generated PTEs (Program Testing Environments) for one-off programs used in Clinical Trials', SAS Forum international
- H. Ament (2005), 'Macro MDU: a flowchart of used programs, datasets files based upon the Final Run', Proceedings of the First Pharmaceutical Users Software Exchange [PHUSE], Applications and Software Development, Paper as04
- D. H. Johnson (2005), 'SAS® with the Windows API', Proceedings of the Thirtieth annual SAS Users Group International Conference [SUGI], Systems Architecture section, paper 248-30.

## ACKNOWLEDGMENTS

The author wishes to thank Amy Gillespie and Frederic Coppin for giving him the opportunity to develop this application and Jane Kang for her detailed review of the application.

## CONTACT INFORMATION

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

Carl Herremans  
Statistical Programming Analyst  
Merck Sharp & Dohme (Europe), Inc.  
Clos du Lynx 5  
1200 Brussels [Belgium]  
Tel: ++32 27766309  
E-mail: Carl\_Herremans@merck.com

SAS and all other SAS Institute Inc. products or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.