PhUSE 2015

Paper AD06

The data listing facility: Data is just a few clicks away

Luigi De Vittorio, OCS Consulting BV, 's-Hertogenbosch, Netherlands

ABSTRACT

Data listings, or monitoring listings, are part of the statistical reporting process to assure the quality of data for Clinical Trials. The Data Listing Facility is a tool that was implemented within the dedicated repository (SAS Drug Development) of a pharmaceutical company to generate several monitoring listings of clinical data (CDASH and SDTM) for clinical trials on vaccines. The tool is interactive and flexible and it was implemented using only SAS programs and macros. Each data listing is controlled by a specific program and can be developed and updated on its own, making maintenance and validation easier. Execution is ruled by an interface window which allows the users to select the required parameters. The tool allows every user to easily generate listings in an Excel format and receive the output file via email.

INTRODUCTION

Data listings (also known as monitoring listings) are normally part of the documentation that must be generated, reviewed and validated for a clinical trial together with tables and figures. These listings present a picture of the data collected for a clinical trial and they play an important role in assuring that the quality of the data is the highest possible. This is a key point for the whole pharmaceutical industry, in fact as stated in the Good Clinical Data Management Practices: "Data collected during a clinical trial must have as few errors as possible to be able to support the findings or conclusions drawn from that trial."

The data listing facility is a tool that was implemented within the SAS® Drug Development (SDD 3.5) environment of a pharmaceutical company that allows to interactively generate data listings for clinical trials on vaccines studies.

The following sections will show the functioning of the tool and the user experience with it:

- Structure and functioning
- Input and Output
- Execution
- A look at the code: what happens in the background?
- Conclusion

STRUCTURE AND FUNCTIONING

The main characteristic about this tool is that it uses only SAS programs. The structure of the tool can be summarized in the picture below. Four components can be identified within the tool

- Program dl__userinterface.sas
- Program dl__driver.sas.
- Listings programs
- Macro programs

PROGRAM DL__INTERFACE.SAS

This program is the main component of the tool. It's the program that the user can execute in SDD using the "Job Results Viewer" functionality. It presents a graphical user interface in which the user can specify certain parameters which affect the execution (e.g. the study protocol and the listings).

PROGRAM DL__DRIVER.SAS

After the parameters have been selected, dl__interface.sas includes a second program named dl_driver.sas, whose aim is to include all macros, set up the libnames and filenames, execute the listing program(s) that the user has previously selected, generate the results of the execution and send them via email to the user.

LISTING PROGRAMS

The core of the tool consists of a collection of SAS programs, each of whom is developed to generate a specific monitoring listing. During the execution of the tool the user selects the listing(s) he wants to have an output for and the tool executes those listings in series (multiple listings can be run during one execution). The tool allows to handle two separate two categories of listings: general listings (so called standard listings) and study-specific listings.

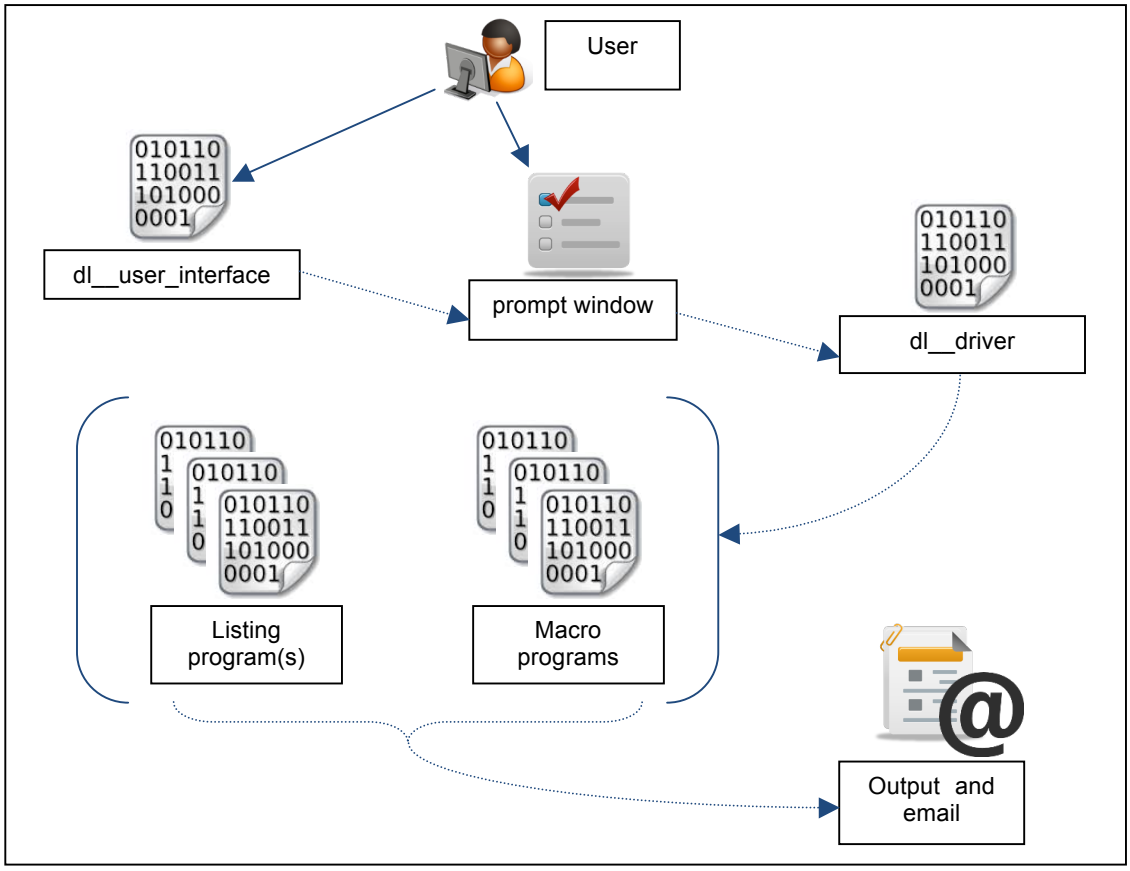
- General listings are intended to be executed for every study. These listings must be programmed in a way that makes them as much generalized as possible. They should be able to handle differences within studies, such as datasets structure.

- Study-specific listings are intended to be executed for one study only. The code for this program is less flexible and generalized.

General listings may seem harder to program compared to study-specific listings, however macro programs can make the code development much easier. Macros are often used to overcome differences between studies (e.g. by using dynamic keep statements to avoid errors in case a specific variable does not exist for a specific dataset). Eventually, since these programs can work for every study, there is no need to re-use or adapt the code to work. On the other hand, programming a study-specific listing can be easier (e.g. hardcoding can be allowed in this case), but code is strictly study related, and in the event that a listing became required for a different study, the code already developed may not be completely reusable, as code re-using could generate unexpected issues and should be performed carefully.

MACRO PROGRAMS

The tool makes also use of a collection of SAS user defined macros, each of whom is stored within a dedicated program. These macros have been implemented to assist the development and the execution, by providing frequently used functionalities (e.g. defining the libraries, generating a cover page in the output file, etc.) and improving the accessibility. They help clinical programmers by standardizing the code, making development and the resolution of issues easier and quicker.



Picture 1: Schematic description of the functioning of the tool

INPUT AND OUTPUT

INPUT DATA

The tool mostly generates listings based on standard CDISC CDASH data, but not only. Listings and programs have been generated to include and integrate data coming from different sources, such as: standard CDISC SDTM data, non-standard clinical data, raw data converted into SDD with appropriate tools, Excel spreadsheets read in by the listing using SDD functionality, Oracle tables, etc.

OUTPUT DATA

All data listings are generated as .xml file saved with a .xls extension. The .xls output file is stored in an archive .zip file to reduce the file size by up to 99%.

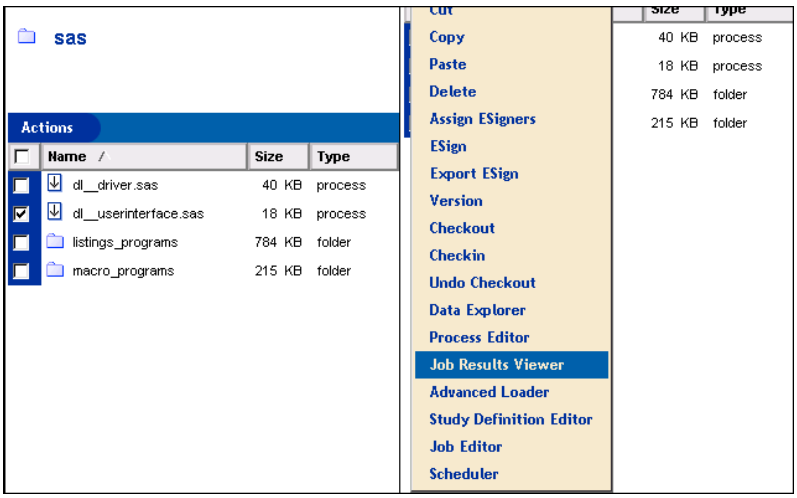
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
1	Center	Subject	Date of Visit 1	Gender	Date of Birth	Numeric Age of Subject	Age Units	Group	Ethnicity	Race	Other Race	Height	Height Unit	Height in CM	Weight	Weight Unit	Weight in KG	Previous Vaccination	D
41	101	1011019	2013-12-05	M	1961-05-28	52	YEARS	1	NOT HISPANIC OR LATINO	WHITE		69	IN	175	146	LB	66,2		Y
42	101	1011020	2013-12-05									73	IN	185	163	LB	74		Y
43	101	1011020	2013-12-05	M	1968-09-23	45	YEARS	1	NOT HISPANIC OR LATINO	BLACK OR AFRICAN AMERICAN		73	IN	185	163	LB	74		Y
44	101	1011021	2013-12-06									59	IN	150	155	LB	70,3		Y
45	101	1011021	2013-12-06	F	1990-01-15	23	YEARS	1	HISPANIC OR LATINO	WHITE		59	IN	150	155	LB	70,3		Y
46	101	1011022	2013-12-06									67	IN	170	256	LB	116,2		Y
47	101	1011022	2013-12-06	F	1982-01-04	31	YEARS	1	NOT HISPANIC OR LATINO	BLACK OR AFRICAN AMERICAN		67	IN	170	256	LB	116,2		Y
48	101	1011023	2013-12-09									67	IN	170	253	LB	114,8		Y
49	101	1011023	2013-12-09	F	1972-06-10	41	YEARS	1	NOT HISPANIC OR LATINO	WHITE		67	IN	170	253	LB	114,8		Y
50	101	1011024	2013-12-09									65	IN	165	173	LB	78,5		Y
51	101	1011024	2013-12-09	F	1969-01-19	44	YEARS	1	HISPANIC OR LATINO	WHITE		65	IN	165	173	LB	78,5		Y
52	101	1011025	2013-12-12									71	IN	180	210	LB	95,3		Y
53	101	1011025	2013-12-12	M	1966-06-05	47	YEARS	1	NOT HISPANIC OR LATINO	WHITE		71	IN	180	210	LB	95,3		Y
54	102	1021001	2013-12-05									171	cm	171	58,8	kg	58,8		Y
55	102	1021001	2013-12-05	F	1971-05-16	42	YEARS	1	NOT HISPANIC OR LATINO	WHITE		171	cm	171	58,8	kg	58,8		Y
56	102	1021002	2013-12-10									181	cm	181	73,3	kg	73,3		Y

Picture 2: Example of data listing output Excel spreadsheet

EXECUTION

THE PROMPT WINDOW

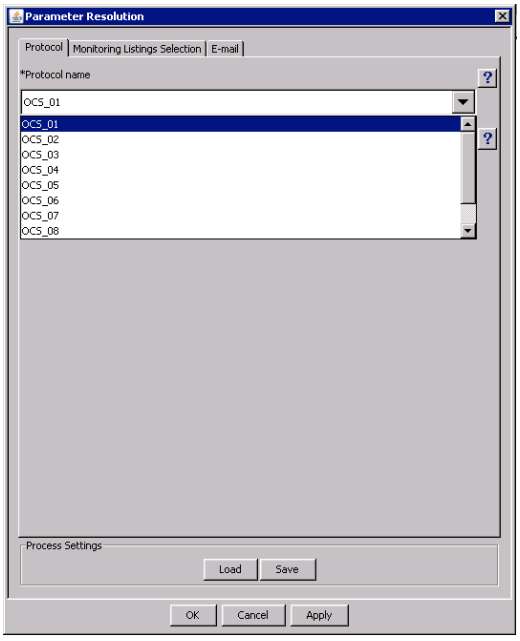
To execute the tool the user executes the main program dl__user_interface.sas from its dedicated location in SDD using the “Jobs Result Viewer” functionality.



Picture 3: Execution of the tool

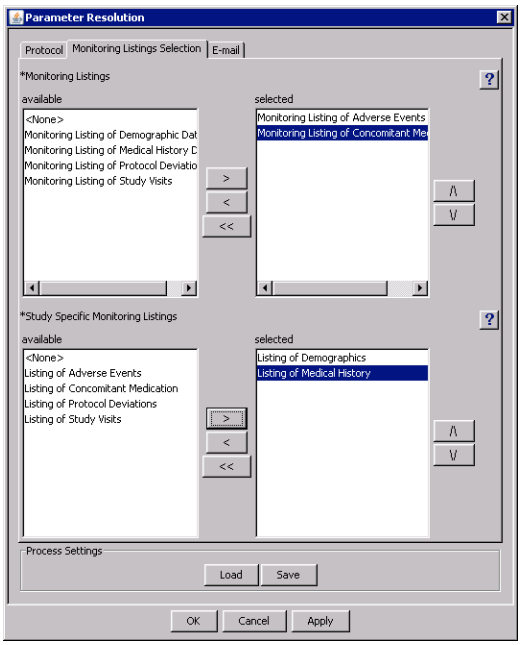
During the execution of this program SDD builds a prompt window which is presented to the user to let him select some parameters which are required to complete the execution:

- Protocol
  This window allows the user to select the study for which he wants to generate the listing(s). This list is generated over a metadata table stored in a dedicated location. Only one study can be selected for the execution.



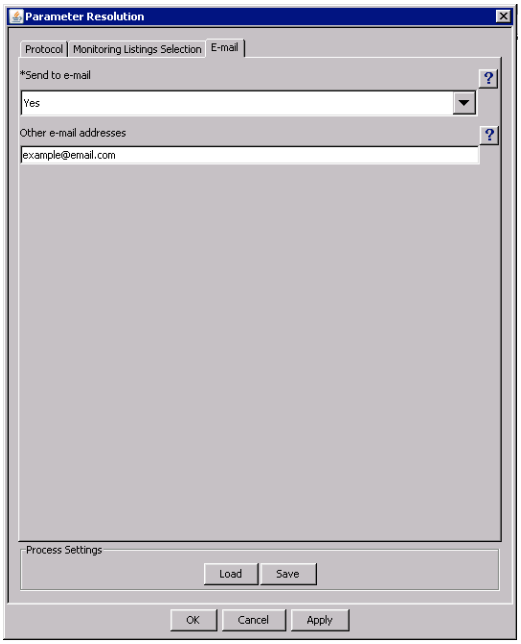
Picture 4: Prompt window, Protocol Selection Tab

- Monitoring listings selection
  On this page the user can select the monitoring listing(s) that he wants to generate. There are two distinct sections that user can select from: one for the general listings and one for the study-specific listings. Study-specific listings are grouped in categories, the user can select one or more categories and then all listings that are assigned to this category in the selected study are executed.
  Each of these lists is generated from a dedicated Excel file stored in a dedicated location. These files are read in by the dl__user_interface.sas program, every change that is performed to the files (e.g. a new listing is added), is reflected in the lists shown in the prompt window. Multiple listings can be selected for both categories for the execution.



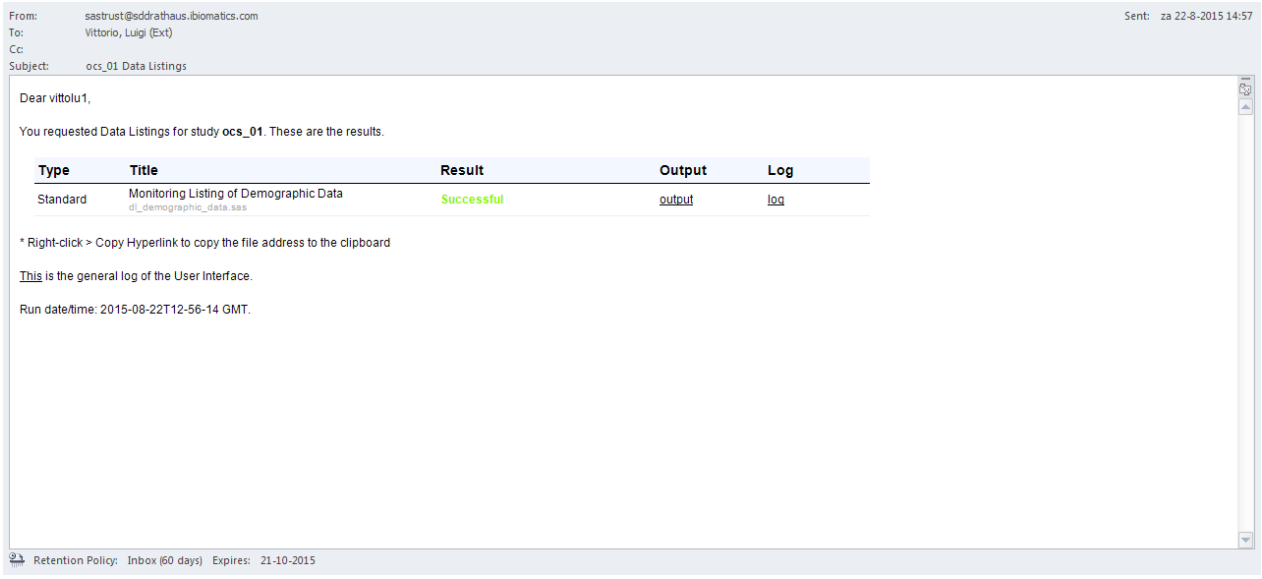
Picture 5: Prompt window, Listing Selection Tab

- E-mail
  The user can specify whether he wants to receive the results of the execution via email. He can also specify further email recipients in a dedicated form. The list of the users is stored in a metadata table and automatically updated.



Picture 6: Prompt window, Email Tab

OUPTUT
When the code execution for the selected listing(s) is completed, the Tool analyses the execution results, generates an execution report and sends it via email (like in the example below) to the user (unless the user specifies he does not want to receive the output).



Picture 7: Example of a received email

Depending on the execution's result the user can receive different message and attachments in the email. Execution can terminate in the following cases:

Possible scenario	Message in the email
The tool could not find the program in the appropriate location. There is no log file and output file available for the user.	Program not found
Program is executed. Errors occur during execution and prevent SAS from generating an output file. Only the log is available for the user.	No output created
Program is executed. Errors occur during execution, but do not prevent SAS from generating an output file. Both log and output file are available for the user, however output is not reliable, since errors occurred during execution.	Output with errors
Program is executed. No errors occur during execution. Both log and output file are available for the user.	Successful

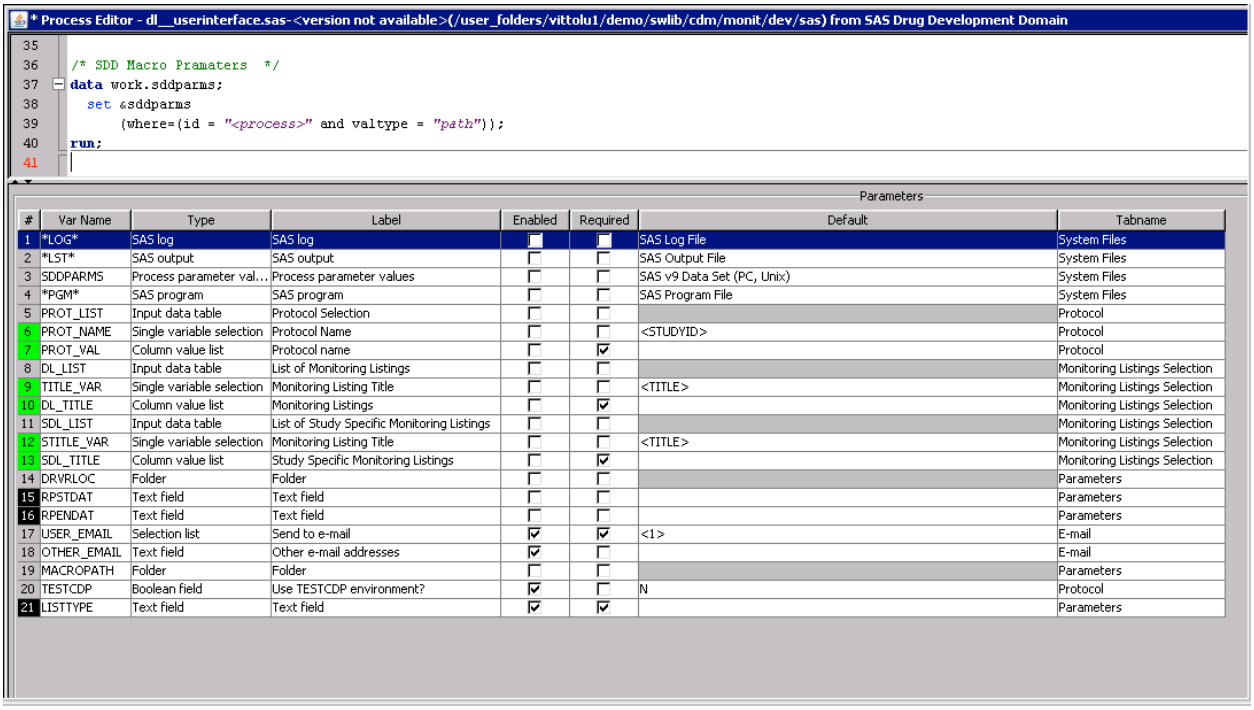
Table1: Output's results scenarios

A LOOK AT THE CODE: WHAT HAPPENS IN THE BACKGROUND

This paragraph provides the reader with a deeper understanding of the SAS code that runs in background during the execution of the tool; also an example of data listing program and of macro program are provided.

PROGRAM DL__INTERFACE.SAS

As already stated the aim of the main program of the tool is to build and display the user interface window for the selection of the study and of the listings to generate. The program makes use of several parameters set up in SDD parameters section (e.g. to locate the files containing the list of study protocols, general listings and study-specific listings).



Picture 8: SDD parameters for dl__user_interface

Eventually the program includes subprogram dl__driver.sas.

```
%macro resolve_pw;
  filename macro "&macropath.";
  filename drvrloc "&drvrloc.";
  %include drvrloc(dl__driver.sas);
%mend resolve_pw;
```

PROGRAM DL__DRIVER.SAS

This program is the most complex of the tool. It performs several steps: it prepares the running environment, runs general and standard and study specific data listings, checks the result after each run for presence of errors, and finally sends an e-mail to the user with links to the output and log files. Some patches of code are provided below to clarify how this program works.

- Define three macro variables that can be used to put notes, warnings and errors in the log without getting them to pop up when they are not triggered.

```
%let note=%sysfunc(compress(###NO TE));
%let warn=%sysfunc(compress(###WAR NING));
%let err =%sysfunc(compress(###ER ROR));
```

- Derive the macro variables protocol and protocol_path, containing the name of the protocol and the location of it in SDD.

```
%global protocol;
%global protocol_path;
%global valsdd;

data _null_;
  set work.protlist;
  call symputx("protocol", lowcase(studyid));
  call symputx("protocol_path", lowcase(catx("/", "data", cdpname, studyid)));
  call symputx("cdpname", cdpname);
run;

data _null_;
  set work.sddparms;
  call symputx("valsdd", lowcase(scan(value, 4, "/")));
  call symputx("programpath", value);
run;
```

- Redirect the log file to a specific text file.

```
filename _genlog sasxbamw "&genlog" user="&_sddusr_" pass="&_sddpw_";

proc printto log=_genlog new;
run;
```

- Obtain a list of all the SAS programs in the folder where the data listing SAS programs are stored

```
%dl_folder_contents(folder=&dl_fold, dsout=work.dl_prog);

proc sort data=work.dl_prog (keep = name
                             rename = (name = pgmname));
  by pgmname;
run;

/* Generate a list of all the standard listings that are
   selected. */
proc sort data=work.selected_standard;
  by pgmname;
run;

data work.requested_dls;
  merge work.dl_prog (in = exists)
        work.selected_standard (in = selected);
```

```
  by pgmname;
  prog_exists = (selected and exists);
  if selected;
run;
```

- Combine the study specific data listings and the standard data listings

```
data work.all_listings;
  set work.requested_dls   (in = standard)
      work.requested_ssdls (in = specific);

  if standard then dl_sdl = 1;
  if specific then dl_sdl = 2;
run;
```

- Obtain a list of all SAS macro programs and include each program file.

```
%sasdrugdev_login(url         = https://&servername./sddremote,
                  sdduserid   = &_sddusr_,
                  sddpassword = &_sddpw_);

%sasdrugdev_getobjects(sddpath       = &programpath./macro,
                       recursivelevel= 1,       /* No subfolders.      */
                       typedef       = process, /* Only program files. */
                       dsname        = work.list);

%sasdrugdev_logout();

data _null_;
  set work.list (where = (iscontainer = 0));
  call execute('%include macro('||strip(scan(name, 1, '.'))||');');
run;
```

- Execute the data listings by including the programs. Each listing program contains a macro and a call to the macro, so including the program automatically executes it.

```
%global progname;
%let progname=&pgmname;
%global listtitle;
%let listtitle=&title;
%global filename zipfile;

%let filename=&outloc/&progname._&_sddusr_._&datetimestamp..xls;
%let filepath=&outpath/&progname._&_sddusr_._&datetimestamp..xls;
%let logfile =&outloc/&progname._&_sddusr_._&datetimestamp..log;
%let zipfile=&outloc/&progname._&_sddusr_._&datetimestamp..zip;

/* Redirect the log file. */
filename _logfile sasxbamw "&logfile" user="&_sddusr_" pass="&_sddpw_";
%put Redirecting log to &logfile;
proc printto log=_logfile new;
run;

%put Redirected from &genlog;
%include &type._list(&progname..sas);
```

- Compress the listing output report in zip format to reduce the size of excel file(listing output report).

```
data _null_;
  call symput('_sddremote_'
              ,strip(tranwrd("&_sdddav_",'/webdav','/sddremote'))
              );
run;
%sasdrugdev_login(url=%str(&_sddremote_.),sdduserid=%str(&_sddusr_.),sddpasswor
d=%str(&_sddpw_.));

%let path=%sysfunc(pathname(work));
%let filename1=&outloc./&progname._&_sddusr_._&datetimestamp..zip;

filename outzip webdav "&filename1" user="&_sddusr_" password="&_sddpw_" ;
filename listout webdav "&filename" user="&_sddusr_" password="&_sddpw_" ;
```

- Sending the output via e-mail.

```
filename email email to="%sysfunc(strip(&email))" &cc. type="text/html"
subject="&protocol Data Listings";

data _null_;
  file email;
  set work.email_header
      work.email_middle
      work.email_footer;
  put line $;
run;
```

LISTING PROGRAMS

The core of the tool is made by the collection of data listings programs that are executed by the users. Every listing must have its dedicated program. The following code is an example of a simple data listing which generates the list of subject which have a duplicated subject code.

```
/*************************************************************************
  Name of Program:  dl_duplicate_subjects.sas

  Purpose of Program:  Generate the Listing of Duplicate Subjects
*************************************************************************/

%macro duplicate_subjects;

  /* Define the name out the output file. */
  %let progname=dl_duplicate_subjects;
  %let listtitle=Listing of Duplicate Subjects;
  %let hasdata = %sysfunc(exist(prot.dm));

  %if (&hasdata = 1) %then %do;

    proc sort  data  =  prot.dm
               out  =   work.dm;
      by subjectkey studyevent studyeventrepeatkey form formrepeatkey;
      where form = "DM";
    run;

    proc sql;
      create table work.dm_all as
      select *
      from work.dm
      group by dmsubjcd, sex, brthdat
      having count(*) > 1;
    quit;
```

```
    %dl_merge_sites(dsin=work.dm_all, dsout=work.dm_all);

    /* End of steps */

    data _null_;
      if 0 then set dm_all nobs=nobs;
      call symputx("numobs",nobs);
      stop;
    run;

    %if (&numobs = 0) %then  %let hasdata = 0;
  %end;

  %dl_setup;
  %dl_standard_titles;
  %dl_cover_page;

  %if (&hasdata = 1) %then %do;

    %dl_merge_form_status(dsin=work.dm_all, dsout=work.dm_all);

    /* Determine the COLSTR macro variable that is used in PROC REPORT. */

    %dl_determine_colstr(dsin=work.dm_all,
                         vars=pfmnemonic subjid dmsubjcd sex brthdat svcomplete
                              svc_date frozen locked first_entry_datetime
                              last_entry_datetime upload_datetime);

    %dl_start_page(sheetname=&listtitle,
                   frozen_rowheaders=2,
                   frozen_headers   =1);

    proc sort data=work.dm_all;
      by dmsubjcd subjid;
    run;

    proc report data=work.dm_all headline headskip pspace=2 split='@'
                    nowd missing
                    style(column)={font_size=7pt}
                    style(header)={font_size=7pt font_weight=bold};
      column &colstr;
      define  pfmnemonic    / display %dl_col(35, string) 'Center';
      define  subjid        / display %dl_col(35, string) 'Subject';
      define  dmsubjcd      / display %dl_col(35)         'Subject Code';
      define  sex           / display %dl_col(35)         'Gender';
      define  brthdat       / display %dl_col(80)         'Birth date';
      &define_svlocfr;
      &define_timing;
    run;
  %end;

  %if (&hasdata = 0) %then  %do;
      %dl_no_data_page(listtitle=&listtitle);
  %end;

  %dl_end;

%mend duplicate_subjects;

%duplicate_subjects;
```

Here it is possible to see how macros are intensively used in the development of data listing programs. In fact %dl_setup, %dl_standard_titles, %dl_cover_page, %dl_end, %dl_merge_form_status, %dl_merge_sites, %dl_start_page and %dl_no_data_page are all macros that have been defined within this tool to perform specific operations.

MACRO PROGRAMS

As said earlier, macro programs are programs that define standard functionalities that can be used by every program. An example is the macro% dl_has_var, that can be used to check the presence of a variable in dataset.
The macro returns a number which is the position of the variable in that dataset, so it returns 0 if the variable is not present and a value>0 otherwise.

```
%macro dl_has_var(ds, var);

  %put ### Macro DL_HAS_VAR - start;
  %put ### ;
  %put ### Parameters:;
  %put ###   - 1     : &ds;
  %put ###   - 2     : &var;

  %let varnum=0;
  %let dsid=%sysfunc(open(&ds));
  %if &dsid > 0 %then %do;
    %let varnum=%sysfunc(varnum(&dsid, &var));
    %let dsid=%sysfunc(close(&dsid));
  %end;

  %put &varnum;
  %put ### Macro DL_HAS_VAR - end;

%mend dl_has_var;
```

CONCLUSION

Having been involved in the development of the data listing facility for over a year and a half allows me to provide my user experience with this tool. Overall I can tell that this tool is a powerful way to generate monitoring listings for clinical studies, which a core part of the process of every clinical trial essential for the pharmaceutical industry. The advantages of using this tool are: the flexibility given by the prompt window; the possibility of generating multiple listings on demand; the use of standard macros which allows makes code development easier; the usefulness of receiving the output directly via email. The disadvantages reside mostly in the Java engine which allows SDD 3.5 to execute programs and jobs, in fact it can happen that, depending on the server utilization the execution, the execution of the tool is considerably slowed down.

REFERENCES

Good Clinical Data Management Practices: http://www.clinplus.com/-Data-and-MetaData-DZS-software/?Tag=data%20listings

CONTACT INFORMATION

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

Luigi De Vittorio
OCS Consulting
Ruwekampweg 2G
5222 AT 's-Hertogenbosch
The Netherlands
+31 (0)73 523 6000
LuigiD@ocs-consulting.com

http://ocs-consulting.nl/