Paper 215-25

# Automatically Converting Data Set Specifications on Excel® to a SAS® Program Used to Assign Data Set Attributes – An Approach to Global Data Mart Building Process

Melanie Paules, SmithKline Beecham Pharmaceuticals, Collegeville, PA
Pilita Canete, EDP Contract Services, Bala Cynwyd, PA
Shi-Tao Yeh, EDP Contract Services, Bala Cynwyd, PA

## ABSTRACT

The Global Data Mart (GdMart) is a collection of SAS® datasets in a standard structure that are used for reporting clinical data. The data mart building process entails mapping data fields from the Oracle® database into SAS datasets based on mapping specifications. The data mart mapping specifications are held in Microsoft® Excel spreadsheets.

This paper describes an automated approach to building an attribute dictionary and the code to assign the attributes to the datasets during the data mart building process. The approach includes two steps. The first step uses a nested SAS program with SAS macros to read the dataset specification from Excel worksheets and build an attribute dictionary that is transported to UNIX. The second step uses a SAS macro to assign the attributes to the data mart data sets within the mapping programs. The outcome is an attribute dictionary in the form of a SAS dataset and a SAS macro that can be invoked within the mapping programs to assign the attributes to the reporting datasets.

The SAS product used in this paper is SAS BASE® on PC and UNIX platforms.

## SECTION 1: INTRODUCTION

SmithKline Beecham (SB) Pharmaceuticals conducts clinical trials. The study data collected from clinical studies are stored in an Oracle database. SB has a process to create data marts in the form of SAS datasets for supporting clinical reporting needs. The study specific data marts include a collection of SAS datasets with a standard structure called the Global Data Mart (GdMart). This process provides an effective support for the reporting of the clinical study data. It is an implementation of a data warehousing solution that enables development of programming code to be shared between different projects and protocols for the SAS data sets of the GdMart structure.

To build the data mart, data fields are mapped from the Oracle database structures into SAS datasets. The data items in the GdMart datasets are either mapped one-to-one from Oracle raw data or derived based on the definitions described in the mapping specifications. The resulting SAS datasets form the reporting database. The reporting SAS programs read from and report from these SAS datasets.

The data mart building process starts from the mapping specifications. The mapping specifications for each dataset are stored in Microsoft Excel file as worksheets. Each worksheet defines a dataset with the attributes and the derivations for mapping the items from the Oracle database structure. The SAS mapping programs are coded based on the specifications. A special team is tasked to maintain the GdMart spreadsheets and ensure standardization across the organization.

This paper describes an automated approach to assist in the data mart building process. This approach uses a nested SAS program with SAS macros to read the dataset specification from Excel and build an attribute dictionary. This attribute dictionary is used by another SAS macro that is used to assign the attributes to the datasets during the data mart building process. The whole data mart building process is shown in the Figure 1.

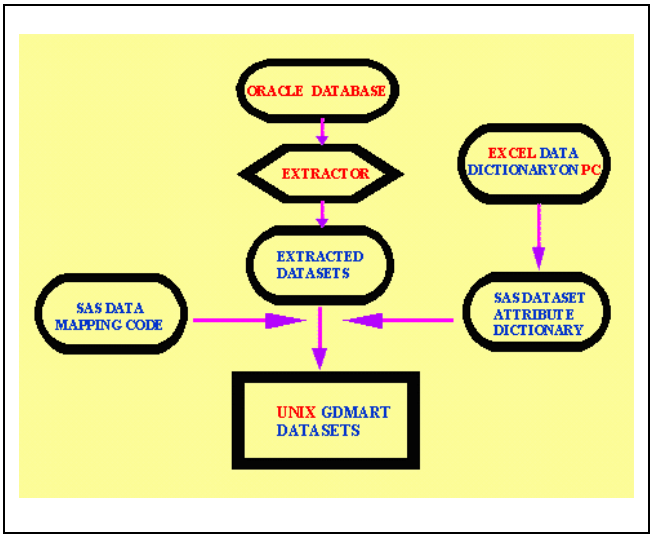


**Figure 1. Flowchart of Data Mart Building Process**

This paper is comprised of four sections. Section 1 includes the abstract and the introduction, Section 2 describes the dataset specification. Section 3 is devoted to the building of the attribute dictionary with sample code. Section 4 deals with final output from the automation and conclusion.

## SECTION 2: DATA SET SPECIFICATION

The data mart specifications are stored as an Excel worksheet that contains data definitions and attributes. It consists of multiple worksheets, one for each dataset, with standard column header. It can be treated as the

metadata or the data dictionary and is used as the first step of data mart building process. The worksheet column header with its descriptions are as follows:

| ***Column Name*** | ***Column Description*** |
|---|---|
| Variable Name | SAS variable name. (8 characters max.) |
| Variable Label | Label for a SAS variable. (40 characters max.) |
| Key Type/Order | Identifies the variable as being a Key variable. |
| SAS Type | SAS variable attribute. NUM, or CHAR. |
| SAS Length | SAS variable attribute length |
| SAS Format | SAS variable attribute. |
| SAS Mapping | Identifies whether the variable is a one-to-one map from an Oracle table or "DERIVED". |
| SB Derivation | Description of specifications to derive the variable |
| Description of Variable and Derivation | A plain English description of the variable and its derivation |

The following Figures are sample Excel worksheets.

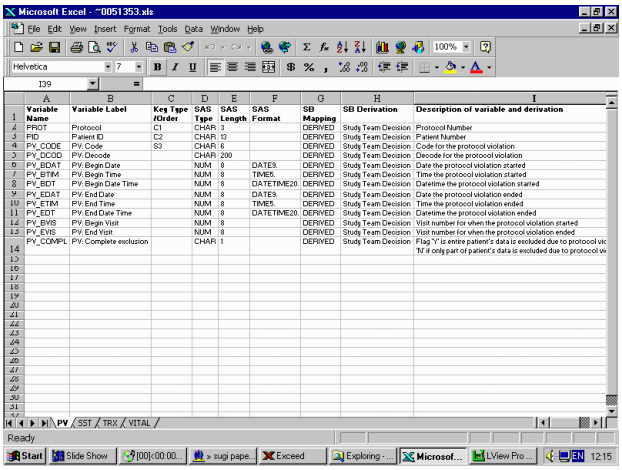


**Figure 2. A Sample Mart Worksheet for Dataset PV**

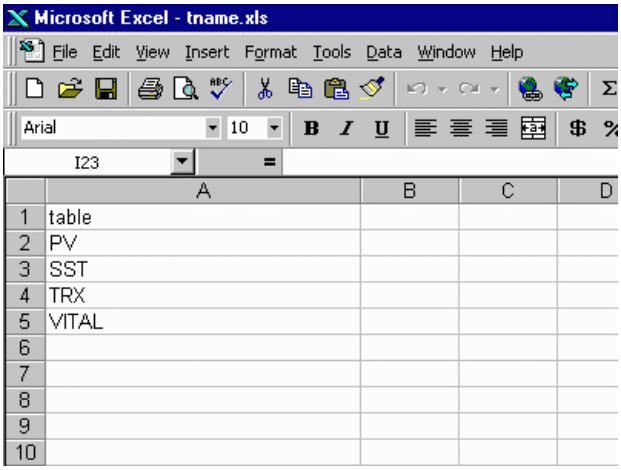


**Figure 3. A Sample Mart Worksheet for Table List**

The dataset specification not only defines data items, it also defines derived data. The data derivations can be categorized in three groups:

| | |
|---|---|
| 1 | Combining variables together, |
| 2 | Decoding data values using SAS format ,SAS function or dynamic code list link, |
| 3 | Using data manipulation steps for complex derivation. |

## SECTION 3: SAS CODE TO ACCESS EXCEL FILE

The purpose of the following SAS code is to access Excel files and to convert the Excel worksheets into a UNIX SAS data set. The following code consists of two main parts: Part1 (MACRO DEFINITION) describes the two main macros used in the program. Part 2 (MAIN CODE) contains the main code.

### Part 1 (macros)

There are two main macros used in this code:

MACRO1: **getxls** - a macro that uses PROC ACCESS to convert a specified sheet in an Excel file into a (work) SAS data set

MACRO2: **gtable** - a macro that loops through each worksheet in the Excel file by invoking the **getxls** macro to create the (work) SAS data sets and then append them into one final data set.

These macros will be described in greater detail later. It suffices to say at this point that these two macros are what drive the main code.

```
*******************************************;
* MACRO 1:  getxls
*******************************************;
%macro getxls(xlfile=,dsout=&sheet,sheet=);
```

```
proc access dbms=xls;
   create work.test.access;
   path="&xlfile";     * a full path Excel file;
   worksheet="&sheet"; * specify the sheet to
                         be read;
   scantype=yes; * scan entire column to
                   determine variable type;
   getnames=yes; * use first row as header;
   assign=yes;   * use header as variable names;
   mixed=yes;    * allow num-to-char conversion
                   in mixed data columns;

   create work.test.view;
   select all;
run;

proc access viewdesc=work.test
   out=work.&dsout;
run;

proc datasets nolist;
  delete test /memtype=all;
run;

%mend getxls;
*******************************************;
* MACRO 2: gtable
*******************************************;
%macro gtable(dsin=,xlin=);

data _null_;
   set &dsin end=eof;
   call symput
        ("sheet"||left(put(_n_,8.)),table);
   cnt+1;
   if eof then call symput('totpnl',cnt);
run;
*******************************************;
* create work data set for each sheet
*******************************************;
%do i=1 %to &totpnl;

%getxls(xlfile=&xlin,dsout=&&sheet&i,sheet=&&she
et&i);

  %let pnum=%eval(&pnum + 1);
  %global f&pnum;
  %let f&pnum=&&sheet&i;

  data &&sheet&i;
      set &&sheet&i;
      table="&&sheet&i";
  run;
%end;
%mend gtable;
```

**Part 2 (Main Code)**

To run the main code, the following assumptions are made:

| | |
|---|---|
| 1. | EXCEL IS NOT OPEN |
| 2. | Excel file to be read (i.e. the main file) exists |
| 3. | in case of multisheet Excel file, a separate Excel file containing the sheet names also exists |
| 4. | Excel sheet names (or tab names) are limited to 8 characters |

.

***Reminder: For this macro to run, make sure that the Excel file to be read is not open***

The main code consists of 5 steps:

**STEP1:**

MACRO1 (getxls) is run to read the file containing the table or worksheet names and create a corresponding data set called 'tname'. (Of course this name is user defined. We just used 'tname' to stand for table names).

**STEP2:**

Having tname as input, MACRO2 (gtable) is run to create separate data sets for each sheet in the Excel file.

**STEP3:**

Variables are renamed according to the GDMART standards. Remember that from an Excel file, the first row is used as the variable names and therefore may not be the desired names.

**STEP4:**

The separate data sets are appended to create the final SAS data set (the meta-data or the attribute dictionary)

**STEP5:**

Finally, the final SAS data set is uploaded to UNIX.

**MACRO1: getxls:**

GETXLS macro uses PROC ACCESS to read an Excel file. In brief, PROC ACCESS reads the Excel file by creating a view descriptor from the Excel file and creating a SAS data set from this descriptor. (This paper will not discuss PROC ACCESS in detail. For more information on PROC ACCESS refer to: Getting Started with the SAS/ACCESS Software Version 6, 1st Ed. or SAS/ACCESS PC File formats, Reference V6, 1st Ed.).

- GETXLS is driven by 3 positional parameters:
  - xlfile = the full path Excel file
  - dsout = the name of the output SAS data set
  - sheet = the name of the sheet in the Excel file to be read (in case of multi-sheet Excel file)

The following PROC ACCESS options are set to:
- read the first row of the Excel file as the variable names (getnames=yes, assign=yes)
- ensure that mixed data columns are converted to character data (mixed=yes)

- **Reason for choice:**

PROC ACCESS is chosen for this paper because of its file-independence and ease of use. In other words, unlike DDE which entails specifying column lengths and data types (making it file-dependent), PROC ACCESS asks only for the Excel file name. All other file specifics like column lengths and types are automatically determined as the file is read. It is therefore, suitable for iteration in case of multi-sheet Excel file.

***Limitation: Currently, the SAS engine supports only up to Excel95. As such, PROC ACCESS is limited to Excel file with 16 worksheets. Until SAS comes up with an engine that is compatible with Excel97 or higher, this macro works only on version Excel95 with 16 sheets or less.***

- **Invocation:**

**%getxls(xlfile=, dsout=,sheet=);**

**- Example:**

To read the worksheet name in the Excel file c:\tables.xls and create a 'tname' SAS data set:
**%getxls (xlfile=c:\tables.xls, dsout=tname, sheet=table);**

- **Output:**

The output is a work data set specified in 'dsout'

## MACRO2: gtable

GTABLE macro iterates through each worksheet in the Excel file by invoking the getxls macro to read the multi-sheet Excel file. It creates an individual (work) SAS data set for each worksheet in the Excel file. It expects the following parameters:

dsin = data set containing a list of sheet names (in our case the data set 'tname')
xlin = the full path Excel file to be read (i.e. the main Excel file containing the attributes information)

- **Assumption:**

To be able to read the sheet correctly, this macro assumes that the sheet name is listed in the input data set 'dsin'. So care must be taken to match the data in 'dsin' perfectly with all the sheet names. For example, suppose we want to read an Excel file containing 3 worksheets named:

PV (for protocol violation)
SST (for study session times)
TRX (for treatments)

The Excel file c:\tname.xls should contain 3 rows:
pv, sst, and trx

(Order doesn't matter. i.e. it could be sst, pv, trx)

- **Invocation:**

**%gtable(dsin=, xlin=);**

**- Example:**

To read the mart file c:\mart.xls that is associated with 'tname' data set:

**%gtable(dsin=tname, xlin=c:\mart.xls)**

**- Output:**

GTABLE creates individual data sets for each sheet in the Excel file.

## MAIN CODE

```
************ PART 2: Main Code ***************;
%let xlpath=u:\ailet\gdmart\;

%macro mainxls(xlfnl=);
%global &xlfnl;
*****************************************;
* STEP1:  get the data set names
*****************************************;

%getxls(xlfile=&xlpath.tname.xls, dsout=tname,
sheet=tname);

*****************************************;
* STEP2: create the SAS data set
*****************************************;
%let pnum=0;
%let i=0;

%gtable(dsin=tname,xlin=&xlpath.mart.xls);
*****************************************;
* STEP3: rename variables according to standards
*****************************************;
%let i=0;

%do i=1 %to &pnum;
data &&f&i;
   set &&f&i;
   if VARIABLE = ' ' then delete;
   rename VARIABLE = var
          VARIABL0 = varlabel
          KEYTYPE_ = key
          SASTYPE  = type
          SASLENGT = length
          SASFORMA = format
          SBMAPPIN = mapping
          SBDERIVA = deriv
          DESCRIPT = descrip;
run;
%end;
*****************************************;
* STEP4:  create final file
*****************************************;
%let i=0;
data &xlfnl;
   set %do i=1 %to &pnum;
       &&f&i %end;; cnt+1;
run;

proc sort data=&xlfnl out=&xlfnl.(drop=cnt);
  by table cnt;
run;
*****************************************;
* STEP5:
* upload the attrib data to unix.
*****************************************;
options comamid=tcp remote=unixnode;
```

```
%let unixnode=upu150;
filename rlink
'h:\SAS612\connect\SASlink\tcpunix.scr';

signon; rsubmit;

proc upload data=sugi out=outdata.sugi;

endrsubmit; signoff;

%mend mainxls;

%mainxls(xlfnl=sugi);
```

CONTENTS PROCEDURE

| | | | |
|---|---|---|---|
| Data Set Name: | WORK.SUGI | Observations: | 71 |
| Member Type: | DATA | Variables: | 10 |
| Engine: | V612 | Indexes: | 0 |
| Created: | 12:45 Tuesday, September 14, 1999 | Observation Length: | 275 |
| Last Modified: | 12:45 Tuesday, September 14, 1999 | Deleted Observations: | 0 |
| Protection: | | Compressed: | NO |
| Data Set Type: | | Sorted: | YES |
| Label: | | | |

-----Engine/Host Dependent Information-----

| | |
|---|---|
| Data Set Page Size: | 8704 |
| Number of Data Set Pages: | 3 |
| File Format: | 607 |
| First Data Page: | 1 |
| Max Obs per Page: | 31 |
| Obs in First Data Page: | 25 |

-----Alphabetic List of Variables and Attributes-----

| # | Variable | Type | Len | Pos | Format | Informat | Label |
|---|---|---|---|---|---|---|---|
| 8 | DERIV | Char | 19 | 71 | $19. | $19. | SB DERIVATION |
| 9 | DESCRIP | Char | 156 | 90 | $156. | $156. | DESCRIPTION OF VARIABLE AND DERIVATION |
| 6 | FORMAT | Char | 11 | 52 | $11. | $11. | SAS FORMAT |
| 3 | KEY | Char | 8 | 31 | $8. | $8. | KEY TYPE /ORDER |
| 5 | LENGTH | Num | 8 | 44 | 6. | 6. | SAS LENGTH |
| 7 | MAPPING | Char | 8 | 63 | $8. | $8. | SB MAPPING |
| 10 | TABLE | Char | 29 | 246 | | | |
| 4 | TYPE | Char | 5 | 39 | $5. | $5. | SAS TYPE |
| 1 | VAR | Char | 9 | 0 | $9. | $9. | VARIABLE NAME |
| 2 | VARLABEL | Char | 22 | 9 | $22. | $22. | VARIABLE LABEL |

-----Sort Information-----

| | |
|---|---|
| Sortedby: | TABLE |
| Validated: | YES |
| Character Set: | ANSI |

## SAS CODE GENERATOR

The SAS macro, **attribu,** performs the following tasks:

1. Reads in the attribute dictionary SAS dataset created from Excel file
2. Uses macro arguments to subset the dataset
3. Constructs the SAS attribute statements
4. Sets up the sort order
5. Writes SAS attribute statements and sort order,as SAS macro program, to a tempory file
6. Invokes the SAS macro
7. Removes the temporary file

```
%macro attribu(dsin=);

%let dname = "attri ;" ;
%let din = "&dsin ;" ;
%let b1 ='%macro ';
%let b2 = 'data ';
%let sortby = 'proc sort;by  ';
%let semic = ';' ;
%let e1 ='%mend;';
%let setd = 'set ';
%let blank = ' ';
%let d1 = "data outmart.&dsin;";
%let d2 = "set &dsin;";

libname data '/home/yehs1/data/';

*********************************************;
* Read in the converted text file from Excel
* study Mart specification sheet
*********************************************;

  data attrfile(drop=deriv mapping descrip
table);
   set data.sugi;
   if table = upcase("&dsin");
   rename var=vars format=sformat type=vtype
length=leng;
run;
*********************************************;
* construct attrib statement for each variable
* in dsin txt file
*********************************************;

  data attrfile;
   set attrfile;
   length c1 $95. formats $20.;
   if vtype = 'NUM' then vtype=' ';
   else if vtype = 'CHAR' then vtype ='$';
   if sformat ^= ' ' then formats = ' format='
|| sformat || ';' ;
   else formats = '; ';
   c1 = 'attrib ' || vars || ' length=' ||
compress( vtype || leng) ||
        ' label=' || "'" || trim(varlabel) ||
"'" || formats ;
*********************************************;
* count number of variables in dsin txt file
* each variable is assigned as macro var nni for
* each attrib atatement
*********************************************;
  data attrfile;
   set attrfile nobs=nobs;
   call symput('_nobs', put(nobs,3.));
   run;

  data t1;
   set attrfile;
   %do i = 1 %to &_nobs;
     if _n_ = &i then do;
       call symput("n&i", put(c1, $95.));
       %let nn&i = "&&n&i";
     end;
   %end;

*********************************************;
* count number of variables in dsin txt file to
* be sorted by and set up a by var list
*********************************************;
data t2 (keep=vars key);
 set attrfile;
```

```
 if key = ' ' then delete;

proc sort;by key;

data t2;
 set t2 nobs = nobs2;
 call symput('_nobs2', put(nobs2,3.));

data t2;
   set t2;
   %do i = 1 %to &_nobs2;
     if _n_ = &i then do;
          call symput("v&i", put(vars, $8.));
          %let vv&i = "&&v&i";
     end;
   %end;

run;

*********************************************;
* create temp file for each dsin macro code and
* save it as ofile and put macro name  and
* attrib statements
*********************************************;
data _null_;
 file 'attri.SAS';

 put &blank;
 put &b1 &dname;
 put &blank;
 put &b2 &din;

 %do i = 1 %to &_nobs;
  put  &&nn&i ;
 %end;

*********************************************;
* put set data statement and sort var list
*********************************************;
  put &setd &din;
  put &blank;
  put &sortby;

  %do i = 1 %to &_nobs2;
  put  &&vv&i ;
  %end;

 put &semic;
*********************************************;
* put mart dataset creation and end statements
*********************************************;
put &d1;
put &d2;
put &semic;
put &e1;
put &blank;
run;

%attri;
run;
  x 'rm attri.SAS' ;
run;
%mend attribu;
```

You can invoke this macro at the end of data mart building program. Example of usage and SAS code generated are shown as follows:

**%attribu(dsin=pv);**

```
%macro pv ;

data pv ;
attrib PROT     length=$3 label='Protocol';
attrib PID      length=$13 label='Patient ID';
attrib PV_CODE  length=$6 label='PV: Code';
attrib PV_DCOD  length=$200 label='PV: Decode';
attrib PV_BDAT  length=8 label='PV: Begin Date' format=DATE9.     ;
attrib PV_BTIM  length=8 label='PV: Begin Time' format=TIME5.     ;
attrib PV_BDT   length=8 label='PV: Begin Date Time' format=DATETIME20.;
attrib PV_EDAT  length=8 label='PV: End Date' format=DATE9.     ;
attrib PV_ETIM  length=8 label='PV: End Time' format=TIME5.     ;
attrib PV_EDT   length=8 label='PV: End Date Time' format=DATETIME20.;
attrib PV_BVIS  length=8 label='PV: Begin Visit';
attrib PV_EVIS  length=8 label='PV: End Visit';
attrib PV_COMPL length=$1 label='PV: Complete exclusion';
set pv ;

proc sort;by
PROT
PID
PV_CODE
;
%mend;
```

## SECTION 4: CONCLUSION

This paper describes an automated approach to the assignment of data set attributes during the data mart building process. The data mart specifications are stored in Excel and the macros described in the first section of this paper provide a method of converting the specification into an attribute dictionary (meta-data). The macros show the technique for accessing Excel on the PC and uploading SAS data sets to UNIX.

The maintenance of the specifications in Excel provides a single location for changes or updates in the specifications. The changes to the specifications are then automatically applied to the data mart data sets by rerunning the build program. The macros described in the second section of this paper provide a method of generating macro code using the attribute dictionary to automatically assign the attributes to the data mart data sets during the build of the mart. This approach provides a centralized method of keeping the data mart data sets consistent with the specifications.

## REFERENCES

[1]. SAS Institute, Inc: *SAS/ACCESS Software for PC File Formats*, Version 6, 1st Edition, p.260

[2]. SAS Institute, Inc: *Getting Started with SAS/ACCESS Software*, Version 6, p.98



Authors

**Melanie Paules**
**(610)917-5104(W)**
**E-mail:**
**melanie_a_paules@sbphrd.com**

SmithKline Beecham Pharmaceuticals

**Pilita Canete**
**(610)917-6909(W)**
**E-mail:**
**pilita_L_canete@sbphrd.com**

EDP Contract Services™

**Shi-Tao Yeh, Ph. D**
**(610)917-5883(W)**
**E-mail: shitaoyeh@us.sina.com**

EDP Contract Services™