

Data Definition Tables – Definition and Automation

Matthew J. Becker, PharmaNet, Inc., Cary, NC

Kitty Moses, Microcebus, Inc., Durham, NC

ABSTRACT

One of the elements of a successful FDA electronic submission is the data definition table (DDT). Three types of data definition tables are needed for a submission: 1) a list of all datasets and what type of information each contains (eg: Adverse Events, Demographics, etc.); 2) for each dataset in the database, a list of all variables, with variable type, variable format, and, if the variable is derived, algorithm; and 3) an alphabetical list of all the variables in the database, along with the dataset in which each resides. Using macros originally developed by Iza Peszek and Frank Liu, we have taken a step towards automating DDT generation for a submission. We will discuss the components and process used to arrive at the ultimate goal: automated DDTs!

INTRODUCTION

The FDA requires a number of items relating to the transfer of data in support of a submission to the agency: SAS transport format files of each dataset, with a size of 25MB or less (uncompressed); case report tabulations; annotated CRFs (case report forms); data definition tables, etc. Many of us have had the experience of supplying these items to the FDA. One of the most cumbersome items is the data definition table, defined in Item 11 in the November 1999 version of "Providing Regulatory Submissions to the Center for Biologics Evaluation and Research (CBER) in Electronic Format – Biologics Marketing Applications." This paper will discuss the process of automating the three types of data definition tables required by CBER.

DATA DEFINITION TABLES - DEFINED

Within Item 11, point 3 details what is expected in the documentation of datasets. The two PDF files required by the FDA and described in point 3 are `define.pdf` and `blankcrf.pdf`. `Blankcrf.pdf` contains the annotated CRFs, and `define.pdf`, the data definition tables. We will be discussing `define.pdf`.

The data definition tables define all datasets in the database. Three types of data definition tables are requested:

- For all datasets, dataset name, description of the dataset, and location within the electronic submission of the transport file
- For each dataset, a table of all the variables, variable label, variable type, variable code, and variable comments
- An alphabetic list of all variables in the study datasets and the dataset(s) in which each can be found.

SYSTEM

We are running on WinNT servers, Citrix terminal server client, SAS V6.12 in production. SAS V8.2 is in validation stage.

AUTOMATION PROCESS – DDT TYPE 1

The first DDT in the submission requirements is the listing of all dataset names in the database, along with a description of each dataset and where the dataset is located within the electronic submission. Assuming the dataset labels are clear and concise, creating this DDT is straightforward. Figure 1 contains an example DDT and Figure 2 contains the code we used to create this DDT.

Figure 1

Raw Datasets for Study ABC		
Dataset	Description of Dataset	Location
ADE	Adverse Events	Crt/datasets/ABC/ade.xpt
DEMOG	Demography	Crt/datasets/ABC/demog.xpt
...

Figure 2

```
proc sql;
  create table dsenti as
  select distinct memname, memlabel as dmlabel
  from dictionary.tables
  where libname="OUTRAW"
  order by memname;
quit;

*** Create the study ABC raw rtf file ***;

data _null_;
  file 'ddt_arawABC.rtf';
  set dsenti end=eof;
  by memname;
  if _n_=1 then do;
    %mrtf(0);
    %mrtf(1,b=1,h=a,v=a);
    put &bl "Raw Datasets for Study ABC" &e;
    %mrtf(3,1 2 2,b=1,h=a,v=a);
    put &bl "Dataset"
        &cl "Description of Dataset"
        &cl "Location" &e;
  end;
  memname=upcase(memname);
  put &bl memname
      &cl dmlabel
      &cl ' '
      &e;
  if eof then do;
    %mrtf(100);
  end;
run;
```

The first Proc SQL step pulls the MEMNAME (dataset name) and MEMLABEL (dataset label, renamed to DMLABEL) from DICTIONARY.TABLES, where the libname is OUTRAW. OUTRAW points to our raw data library for the ABC study.

At this point, we have 2 out of the 3 columns required for the first data definition table. The last column is the location of the dataset's transport file within the submission. PharmaNet, Inc.'s publishing group added this field once the electronic submission was complete. If, however, the location was known, a SAS dataset with all of the raw dataset names and location directory could be created and merged with the DSENTI dataset in Figure 2. The dataset name would be the link (merge variable) between DSENTI and the SAS dataset containing the location.

The MRTF macro is a derivative of the RTF macro created by Iza and Robert Peszek. Utilizing MRTF, a rich text format (RTF) document is created containing unformatted text, control symbols and words, and groups. For those familiar with Word tables, the RTF file created above is much like a Word table. The call %mrtf(1,b=1,h=a,v=a) creates a single cell with horizontal and vertical cell lines. Within this cell, we then place the text "Raw datasets for Study ABC." Next, the call %mrtf(3,1 2 2,b=1,h=a,v=a) creates a row with 3 cells. At this point, we can place text within each of these cells (in this case, the dataset, description of the dataset, and location in the submission of the transport files). The 3 cells continue to be created upon each new row until the call to %mrtf(100) which says, "I'm done, close the RTF file creation."

Please reference "Automate the Creation and Manipulation of Word Processor Ready SAS Output", SAS Observations, 1998, Izabella Peszek and Robert Peszek for greater understanding of the (M)RTF invocation. The paper can be found online at:

<http://www.sas.com/service/doc/periodicals/obs/obswww13/index.html>

Limitations or potential limitations include:

- Missing, unclear, or verbose dataset labels
- 200-character limit with SAS V6.12.

AUTOMATION PROCESS – DDT TYPE 2

Figure 3

Study ABC – Demographics Dataset Variables				
Variable	Label	Type	Codes	Comments
PATID	Unique patient identifier	Char		Demographics page 1
SEX	Gender of subject	Char	m = male f = female	Demographics page 1

The second DDT within the submission requirements is the listing, by dataset, of all variables, variable labels, variable types, variable format codes, and comments pertaining to that variable (see Figure 3). The information in the first 3 required columns can be pulled from the SAS dictionary of the SAS database. The last two columns, codes and comments, are more difficult to populate.

The Codes column lists the format codes for a variable. These values can be found within the format catalog. The comments are read in from an Excel file (see explanation below.) Figure 4 contains the code used to pull the formats and comments into a SAS dataset containing the variable, variable label, and variable type.

Figure 4

```

**** Read Excel file containing variable
comments;

Data varcomm;
  infile "ddtauto.csv" lrecl=2056 pad dlm=',';
  Length memname $8 name $8 comment $200;
  input memname $ name $ comment $;
Run;

**** Create dataset of formats;
proc format library=library cntlout=dsfmt;
run;

```

```

proc sql;
  ** Create dataset of dataset information;
  create table dsenti as
  select distinct a.*
  from (select memname, name, uppercase(type) as
         dtype, length, trim(left(format))
         as fmt, uppercase(label) as dlabel
        from dictionary.columns
        where libname="WORK") as a;
  ** Create dataset of 'formatted' format codes;
  create table formatds as
  select case
  when type='N' then trim(left(fmtname)) ||
    '.'
  when type='C' then '$' ||
    trim(left(fmtname)) || '.'
  else ''
  end as fmt,
  (right(put(left(start),$4.)) || '= ' ||
   label) as coding
  from dsfmt;
  ** Merge format codes onto dataset information
  by format name;
  create table var_fmt as
  select distinct a.*, b.coding
  from dsenti a left join (select *
                          from formatds
                          where fmt ne '') b
  on a.fmt=b.fmt
  order by memname, name, dtype, length, fmt;

  ** Merge variable comment information onto
  dataset information;
  create table var_fmt as
  select distinct a.*, b.comment
  from var_fmt a left join (select *
                          from varcomm) b
  on a.memname=b.memname and
  a.name=b.name
  order by memname, name;
quit;

**** Create DDT for individual derived datasets
****;

data _null_;
  file 'ddt_iderABC.rtf';
  set var_fmt end=eof;
  by memname name;
  length tempstr $200;
  if _n_=1 then do;
    %mrtf(0);
  end;
  if first.memname then do;
    %mrtf(1,b=1,h=a,v=a);
    tempstr="Study ABC - " ||
      trim(left(memname)) ||
      "Dataset Variables";
    put &bl tempstr &e;
    %mrtf(5,3 4 2 3 4,b=1,h=a,v=a);
    put &bl "Variable"
      &cl "Label"
      &cl "Type"
      &cl "Codes"
      &cl "Comments"
      &e;
  end;
end;

```

```

if first.name then %mrtf(5,3 4 2 3 4,b=1,v=a);

if last.name then do;
  %mrtf(5,3 4 2 3 4,b=1,v=a);
  %*put &nl;
  %mrtf(5,3 4 2 3 4,b=1,h=a,v=a);
end;

if first.name then put &bl name
                      &cl dlabel
                      &cl dtype
                      &cl coding
                      &cl comment
                      &e;

else put &bl
        &cl
        &cl
        &cl coding
        &cl
        &e;

if eof then do;
  %mrtf(100);
end;
Run;

```

WALK-THROUGH OF FIGURE 4 CODE

The first block of code in Figure 4 reads in the variable comments from a comma-delimited Excel file. The Comments column lists either the CRF page where that variable can be found OR the algorithm used to derive the variable. One of the biggest issues was documentation of the derivation algorithms. Many discussions were held on the best way to link the derivations with the variable. How could we get the algorithm text into the final dataset to be sent through the RTF macro? Several techniques were discussed:

- Dataset name, variable name, and algorithm contained within:
 - o Text file (ASCII)
 - o Excel file
 - o SAS dataset
- Inclusion of variable and algorithm in the program header.

Including the variable and algorithm in the program header was not time-efficient in this particular case, although it was a great idea and was left as an enhancement to the current system (see ENHANCEMENTS section below). Using the SAS dataset to house the information was the most matter-of-fact, but it had the 200-character limitation in SAS V6.12. Therefore, the text file and Excel file were the finalists; we chose Excel due to ease of entry.

Three variables are included in the Excel file: MEMNAME (dataset name), NAME (variable name), and COMMENT (variable comment). The variables MEMNAME and NAME will be used to merge the variable comments on with the additional variable information.

```

Data varcomm;
  infile "ddtauto.csv" lrecl=2056 pad dlm=',';
  Length memname $8 name $8 comment $200;
  input memname $ name $ comment $;
Run;

```

The next block of code creates a SAS dataset from the format catalog. This dataset will be used in a later Proc SQL step to merge the format codes onto the variable(s) which have associated formats.

```

proc format library=library cntlout=dsfmt;
run;

```

Following the creation of the SAS dataset containing the formats, a Proc SQL statement is executed. This section of code provides the majority of the data manipulation necessary to create the final dataset for DDT Type 2.

The first SQL statement creates a dataset containing: dataset name, variable name, variable type, variable length, variable format, and the label of the variable. All of this information is pulled from the DICTIONARY.COLUMNS SAS View.

```

create table dsenti as
select distinct a.*
  from (select memname, name, upcase(type) as
         dtype, length, trim(left(format))
         as fmt, upcase(label) as dlabel
        from dictionary.columns
        where libname="WORK") as a;

```

This CREATE statement generates a dataset containing 3 of the 5 columns necessary for DDT Type 2 (variable name, variable description, and variable type). The dataset name is included in the header of the DDT Type 2 output.

Next, another CREATE statement is executed to create a working SAS dataset of format names and values. The resulting dataset contains two variables: FMT and CODING. FMT contains the format name. If the variable type is character, the format name is preceded with a '\$.' It is extremely important to create the variable FMT with the '\$' preceding character variables and the '.' assigned to the end of the format name string. This ensures a "match" when merging back to the format catalog dataset.

In addition, a variable named CODING is created which contains the format(s) for a specific format name. Depending on the number of entries for a format, the resulting SAS dataset will contain the same number of records as that format has entries. For example, if the format GENDERCD has two entries: m=MALE and f=FEMALE, the resulting dataset will have two records.

```

create table formatds as
select case
  when type='N' then trim(left(fmtname)) ||
    '.'
  when type='C' then '$' ||
    trim(left(fmtname)) || '.'
  else ''
end as fmt,
(right(put(left(start),$4.)) || '= ' ||
  label) as coding
from dsfmt

```

Following the creation of the format dataset, another CREATE statement is executed which merges the format "codes" onto each variable which is assigned that format.

```

create table var_fmt as
select distinct a.*, b.coding
  from dsenti a left join (select *
                          from formatds
                          where fmt ne '') b
  on a.fmt=b.fmt
  order by memname, name, dtype, length, fmt;

```

Finally, the variable COMMENT that was read from the Excel file is merged on.

```

create table var_fmt as
select distinct a.*, b.comment
  from var_fmt a left join (select *
                          from varcomm) b
  on a.memname=b.memname and
  a.name=b.name
  order by memname, name;

```

The last section of Figure 4 is the DATA _NULL_ step that creates the RTF file. The resulting RTF file then has to be manually split into by-file sections.

Limitations or potential limitations:

- All derived variable definitions have to be documented in a file readable by SAS!
- 200-character limit on codes and comments.
- Missing, unclear, or verbose variable labels.

AUTOMATION PROCESS – DDT TYPE 3

The third DDT in the submission requirements is the listing of all variables in all datasets. It consists of all variable names in the database and their source datasets, sorted by variable name; an example of a type 3 DDT is shown in figure 5.

Figure 5

Study ABC – All Derived Dataset Variables	
Variable	Dataset
ACTION	ADE
ADE	ADE
AGE	CHEM
AGE	DEMOG
ALBUM	CHEM
...	...

The Proc SQL step in Figure 6 pulls NAME (variable name) and MEMNAME (dataset name) from DICTIONARY.COLUMNS, where the libname is OUTRAW (the raw data for the ABC study); these are the 2 columns that are to be included in the DDT. The file is sorted by NAME and MEMNAME, and then is read into the DATA _NULL_ step that creates the RTF file.

Figure 6

```
proc sql;
  create table vars as
  select distinct a.*
  from (select memname, varnum, name
        from dictionary.tables
        where libname="OUTRAW") as a
  order by memname;
quit;

*** Create the study ABC raw rtf file ***;
proc sort data=vars; by name memname; run;
data _null_;
  file 'ddt_vrawABC.rtf';
  set vars end=eof;
  by name memname;
  if _n_=1 then do;
    %mrtf(0);
    %mrtf(1,b=1,h=a,v=a);
    put &bl "Study ABC: All Raw Dataset
Variables"
    &e;
    %mrtf(2,2 2,b=1,h=a,v=a,w=4);
    put &bl "Variable"
    &cl "Dataset "
    &e;
  end;
  put &bl name
    &cl memname
    &e;
  if eof then do;
    %mrtf(100);
  end;
run;
```

ENHANCEMENTS

The following enhancements have been made to the original DDT automation process:

- Automation of the COMMENTS column: variable derivation algorithms can be included in the program header and pulled into the documentation.
- By formatting the existing derived data file, we can now add the formats and format decode columns into the document automatically.
- We have automated the process of combining all the derived data documentation files into one document with page breaks after each data file.
- An Excel workbook can be created with each data file in one sheet.

LESSONS LEARNED

If your submission will require DDTs, do not ignore them. Trying to assemble type 2 DDT codes and comments after the fact can take an inordinate amount of time; if you gather this information up front, you will keep time and aggravation to a minimum.

Ensure that dataset labels and variable labels are clear, concise, and complete. Utilizing the SAS dictionary views to pull this information is a major component of the automation concept.

CONCLUSION

Creating data definition tables (DDTs) can be one of the most time consuming parts of supplying electronic data to the FDA. By using the SAS software and macros created by Iza Peszek, Robert Peszek, and Frank Liu, we were able to drastically reduce the amount of manual effort to create DDTs.

REFERENCES

SAS is a registered trademark of SAS Institute Inc.

F. Liu, "A Simple Way to Create a Clinical Trial Study Data Dictionary", Annual Proceedings of SAS User Group International, SUGI 21, 1996.

R. Peszek, I. Peszek, "Automate the Creation and Manipulation of Word Processor Ready SAS Output," SAS Observations, 1998. <http://www.sas.com/service/doc/periodicals/obs/obswww13/index.html>

ACKNOWLEDGMENTS

The authors would like to thank Iza Peszek and Frank Liu for the use of their macros. In addition, the authors would like to thank Monte Jarvis and PharmaNet, Inc. for their support in this endeavor. Finally, the authors would like to express their gratitude to Syamala Kasichainula, Eugene Yeh, Angela Ringelberg, and Janice Englund for their contributions to this project.

CONTACT INFORMATION

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

Matthew J. Becker
PharmaNet, Inc.
1001 Winstead Drive, Suite 505
Cary, NC 27513
Work Phone: 919-465-4717
Fax: 609-720-5034
Email: mbecker@pharmanet.com

And/Or:

Kitty Moses
Microcebus, Inc.
1523 Hermitage Court
Durham, NC 27707
Email: kitty.moses@britsys.net