Paper FC01

Implementation of the CDISC SDTM at the Duke Clinical Research Institute

Jack Shostak, Duke Clinical Research Institute (DCRI), Durham, NC

ABSTRACT

This paper discusses the current approach taken to implementing the *Study Data Tabulation Model* (SDTM) version 1.0 at the Duke Clinical Research Institute (DCRI). Possible routes to SDTM implementation are explored. The challenges of the DCRI SDTM implementation are discussed. Finally, some generic SDTM SAS® conversion macros are provided. The SAS code provided in this paper was developed with BASE SAS version 8.2 on UNIX, but the code should be compatible or portable to other operating systems and future releases of SAS. This paper assumes that the reader has some familiarity with the CDISC models and in particular the SDTM.

INTRODUCTION

Standard data definitions for the clinical trials industry have long been needed. Countless hours are spent by pharmaceutical companies, contract research organizations, and the FDA in learning and reconciling various proprietary database structures. Fortunately for us, the Clinical Data Standards Interchange Consortium (CDISC ™) is building clinical trials research database standards. The *Study Data Tabulation Model* (SDTM), the focus of this paper, is one such database standard that describes how to send the data tabulation datasets to the FDA. Please note that there are other equally important CDISC standards such as the analysis dataset models being developed by the CDISC Analysis Dataset Modeling (ADaM) team, the Operational Data Model (ODM), and the define.xml Case Report Tabulation Data Definition Specification.

CDISC defines the SDTM and other models, but CDISC does not tell you how to go about creating these data structures in your operational environment. The *CDISC SDTM Implementation Guide: Human Clinical Trials* is critical in understanding the SDTM, but it does not tell you specifically how to get from your “raw” data in your clinical data management system (CDMS) to the SDTM. This paper discusses one possible approach to implementing the SDTM and provides some tools to help with creating the SDTM. Just as there are many ways to accomplish a given task in SAS, there are certainly more ways to implement the SDTM than this paper describes. It is my hope that reading this paper will give you some ideas on how you can implement the SDTM within your organization.

APPROACHES TO CREATING THE SDTM

We considered three general approaches to creating the SDTM datasets: Build the SDTM entirely in the CDMS, build the SDTM entirely on the “back-end” in SAS, or take a hybrid approach and build the SDTM partially in the CDMS and partially in SAS.

BUILD THE SDTM ENTIRELY IN THE CDMS

It is possible to build the SDTM entirely within the CDMS. If the CDMS allows for broad structural control of the underlying database, then you could build your eCRF or CRF based clinical database to SDTM standards.

Advantages:

- Your “raw” database is equivalent to your SDTM which provides the most elegant solution.
- Your clinical data management staff will be able to converse with end-users/sponsors about the data easily since your clinical data manager and the und-user/sponsor will both be looking at SDTM datasets.
- As soon as the CDMS database is built, the SDTM datasets are available.

Disadvantages:

- This approach may be cost prohibitive. Forcing the CDMS to create the SDTM structures may simply be too cumbersome to do efficiently.
- Forcing the CDMS to adapt to the SDTM may cause problems with the operation of the CDMS which could reduce data quality.

BUILD THE SDTM ENTIRELY ON THE “BACK-END” IN SAS

Assuming that SAS is not your CDMS solution, another approach is to take the clinical data from your CDMS and manipulate it into the SDTM with SAS programming.

Advantages:

- The great flexibility of SAS will let you transform any proprietary CDMS structure into the SDTM. You do not have to work around the rigid constraints of the CDMS.
- Changes could be made to the SDTM conversion without disturbing clinical data management processes.
- The CDMS is allowed to do what it does best which is to enter, manage, and clean data.

Disadvantages:

- There would be additional cost to transform the data from your typical CDMS structure into the SDTM. Specifications, programming, and validation of the SAS programming transformation would be required.
- Once the CDMS database is up, there would then be a subsequent delay while the SDTM is created in SAS. This delay would slow down the production of analysis datasets and reporting. This assumes that you follow the linear progression of CDMS -> SDTM -> analysis datasets (ADaM).
- Since the SDTM is a derivation of the "raw" data, there could be errors in translation from the "raw" CDMS data to the SDTM.
- Your clinical data management staff may be at a disadvantage when speaking with end-users/sponsors about the data since the data manager will likely be looking at the CDMS data and the sponsor will see SDTM data.

BUILD THE SDTM USING A HYBRID APPROACH

Again, assuming that SAS is not your CDMS solution, you could build some of the SDTM within the confines of the CDMS and do the rest of the work in SAS. There are things that could be done easily in the CDMS such as naming data tables the same as SDTM domains, using SDTM variable names in the CTMS, and performing simple derivations (such as age) in the CDMS. More complex SDTM derivations and manipulations can then be performed in SAS.

Advantages:

- The changes to the CDMS are easy to implement.
- The SDTM conversions to be done in SAS are manageable and much can be automated.

Disadvantages:

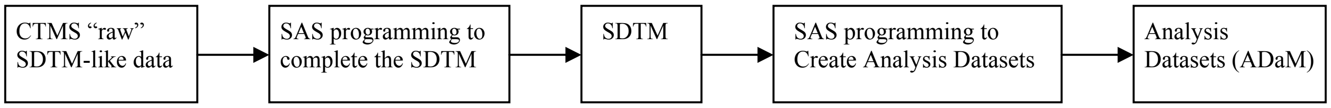
- There would still be some additional cost needed to transform the data from the SDTM-like CDMS structure into the SDTM. Specifications, programming, and validation of the transformation would be required.
- There would be some delay while the SDTM-like CDMS data is converted to the SDTM.
- Your clinical data management staff may still have a slight disadvantage when speaking with end-users/sponsors about the data since the clinical data manager will be looking at the SDTM-like data and the sponsor will see the true SDTM data.

CURRENT DCRI APPROACH TO CREATING THE SDTM

Currently the DCRI is using the hybrid approach to SDTM file creation. We chose this route because:

- It allowed us to do much (60-70%) of the SDTM conversion in the CDMS.
- The clinical data manager would be conversant in the SDTM structure.
- We could finish off the SDTM conversion in SAS which kept us from pushing the limits of our CDMS to get the remaining 30-40% of the SDTM conversion done there.
- It appeared to be the least expensive and quickest to implement.
- It leveraged the strengths of the CDMS and SAS.

Here is a graphic showing how the SDTM has been integrated into our operations:


CTMS "raw" SDTM-like data
SAS programming to complete the SDTM
SDTM
SAS programming to Create Analysis Datasets
Analysis Datasets (ADaM)

PROC CDISC can then be called to convert the SDTM or ADaM datasets into the CDISC ODM model.

CHALLENGES OF THE DCRI SDTM IMPLEMENTATION

Although we have been able to successfully convert several projects into the SDM 3.0 and SDTM 1.0 models, we have experienced some challenges.

PASSING RAW DATABASE PIECES THROUGH THE SDTM

Because we chose to implement the SDTM in the linear fashion shown above, our SDTM files were the source data for our analysis datasets. The good news with this approach is that only two databases are needed for regulatory submission: the SDTM datasets and the ADaM datasets. The disadvantage to this approach is that data elements in the CTMS database that were not part of the standardized SDTM domains must be retained in places such as the SUPPQUAL dataset so that all data are available for analysis dataset creation. Date variables are a good example of this problem. Simply passing the --DTC date variable to an analysis dataset program is probably not sufficient for date imputation as --DTC dates can "lose" date components which is discussed later in this paper.

CREATING TRIAL DESIGN MODEL DATASETS

With the exception of the Subjects Elements Table (SE) and the Subjects Visit Table (SV) domains, the rest of the Trial Design Model datasets are metadata. This metadata is not typically collected in the CDMS, so you have to create many of these datasets manually. Perhaps when a protocol authoring tool becomes available then this information can be passed as a data stream to the Trial Design Model datasets. A simple intermediate option is to build entry screens for the Trial Design Model datasets into your CDMS.

CREATING ANALYSIS DATASETS FROM "FINDINGS" DOMAINS

Findings domains, much like most of the SDTM, are highly normalized data structures. This is great because it allows them to be extended easily to accommodate new data fields. Unfortunately, it is often the case where you want your data to be denormalized for analysis purposes. For example, look at the following slice of an endpoint assessment CRF page:

Endpoint Assessment:

Assessment Date: _ _ / _ _ _ / _ _ _ _ (Day/Month/Year)

Since the last visit, has the patient experienced any of the following?

Death?	Yes ☐	No ☐
Recurrent Myocardial Infarction?	Yes ☐	No ☐
Stroke?	Yes ☐	No ☐

In a typical CDMS, you might expect to see this endpoint data represented like this for two patients:

subjectid	adate	death	rmi	Stroke
101001	12JUL2004	YES	NO	NO
101002	13AUG2004	NO	NO	YES

However, in the SDTM, you would see the same data represented something like this:

USUBJID	EPDATE	EPSEQ	EPTESTCD	EPTEST	EPSTRESC
101001	12JUL2004	1	DEATH	Death	YES
101001	12JUL2004	2	RMI	Recurrent Myocardial Infarction	NO
101001	12JUL2004	3	STROKE	Stroke	NO
101002	13AUG2004	1	DEATH	Death	NO
101002	13AUG2004	2	RMI	Recurrent Myocardial Infarction	NO
101002	13AUG2004	3	STROKE	Stroke	YES

As you can see, the SDTM turns variables into rows. Therefore, the SDTM data is sometimes not analysis friendly if you need to use these endpoint data as variables for a SAS procedure. We often had to transpose the normalized SDTM structures back into a flattened dataset for analysis purposes. Unfortunately this transposition cannot be fully automated based on the SDTM data alone. This is because the SDTM files do not by themselves carry enough metadata about the rows to perform the transposition. You do have the --TESTCD and --TEST variables which can serve as the SAS variable name and SAS label respectively. You do not have whether the resulting analysis variable should be numeric or character and no variable format is available.

This problem is addressed by the new CDISC define.xml standard. Define.xml will become the future replacement for define.pdf and data definition files which are sent to the FDA with regulatory submissions. The great advantage to define.xml is that presents the define.pdf information in a way that you can then process with a program. For instance, an automatic data transposition SAS macro could be written in SAS as follows. First, use PROC CDISC on define.xml to read the variable metadata for a given domain. Then use that define.xml metadata as parameters to a PROC TRANSPOSE to flip the SDTM domain data from a normalized to denormalized data structure. For further reading on value level metadata see the excellent discussion in the ADaM document *Statistical Analysis Dataset Model: General Considerations* and the define.xml specification at www.cdisc.org.

HOW MUCH TIME IS NEEDED FOR SDTM IMPLEMENTATION?

SDTM implementation will probably take more time than you originally planned. You need to plan time for training your staff on the SDTM model. CDISC offers SDTM training courses that you can take. Then, you need to plan how you want to implement the SDTM in your organization. Whether you pick one of the three previous mentioned approaches or not, you need an implementation plan. If your data management staff will be helping with the SDTM conversion, they will need to take time to figure out how to get the CDMS to make SDTM structures. This may involve changes to working processes. If your SAS programmers will be helping with the conversion, plan on automating as much of the conversion programming as you can. Add additional time for SDTM conversion programming specifications, programming, and validation. Finally, for your first implementation, expect delays simply due to the CDISC model learning curve. Even with formal training on the SDTM, it takes time to get to the point where you think in terms of interventions, findings, events and the SDTM model as a whole.

HOW DO I KNOW MY SDTM IS VALID SDTM?

At this time, I do not believe there is a SDTM validation tool. Therefore, even if you build a SDTM database, how do you know it is valid SDTM? Unlike the ODM, there is no schema available to validate the SDTM against. See later in this paper for a SAS macro that may help somewhat in validating your SDTM files. Eventually a comprehensive SDTM validation tool will be needed.

A FEW SDTM CONVERSION SAS MACROS

Some of the following SAS macros were developed to help convert our CDMS SDTM-like database into a final SDTM database. You may find some of the ideas presented here to be of use even if the code itself is not applicable in your environment.

AUTOMATIC SDTM VARIABLE LABELER AND PARTIAL SDTM VALIDATOR

As mentioned above, there is no SDTM validator that I am aware of yet. However, we wanted something to help determine if the variables for a given SDTM domain were correct. The following SAS macro applies the SDTM standard labels to the SDTM domains and also checks to make sure that all “required” and “expected” SDTM variables are present. This is by no means a complete SDTM validator, but it is helpful in checking SDTM files.

```
/*********************************************************************************
 Purpose:  When given a SDTM domain and SDTM class, the CDISC_LABEL macro
           will apply the SDS v3.1 standard label to the domain variables.  If a
           specific domain variable exists then that label will be applied first.
           If the domain is newly defined, then the generic domain level label
           will be applied secondarily.  As a side function, the CDISC_LABEL macro
           will check for "expected" or "required" variables from the SDS v3.1
           standard. If an expected/required variable is not found in the domain,
           then a SAS warning message will be generated.  If a variable is found
           in the domain that is not part of the SDS v3.1 standard, then a NOTE will
           be generated in the SAS log.

 Assumption:  This program assumes that the DOMAINS.SAS7BDAT dataset exists in the
              present working directory with the specified domain dataset.
              DOMAINS.SAS7BDAT is a SAS dataset that was created by running a simple PROC
              IMPORT on the CDISC provided Excel file of the SDTMv1.0/SDSv3.1 data tables.
              This file is called "SDS V3 1 Tables 20041012.xls" and can be found under
              the members only section of www.cdisc.org.

 Parameters:  domain = valid SDTM domain name.
              class  = SDTM class.  It may be set to nothing (for DM),
                       INTERVENTIONS, EVENTS, or FINDINGS.

 Sample Calls: %cdisc_label(domain=EVENTS,domain=AE)
               %cdisc_label(domain=INTERVENTIONS,domain=CM)
               %cdisc_label(domain=,domain=DM)
**************************************************************/
%macro cdisc_label(class=,domain=);
options ls = 256;

**** GET DOMAIN CONTENTS AND KEEP VARIABLE_NAME IN VARINFO DATASET;
proc contents
   data=&domain
   out=varinfo
   noprint;
run;

data varinfo;
   set varinfo(keep=name);

   drop name;
   length variable_name $ 255;
   variable_name = upcase(name);
run;

proc sort
   data=varinfo;
      by variable_name;
run;

**** GET GENERAL VARIABLES AS WELL AS DOMAIN SPECIFIC VARIABLES FROM DOMAINS DATASET.;
data cdisc_labels;
   set "./domains.sas7bdat"(keep=Observation_Class Domain_Prefix Variable_Name
                                 Variable_Label Core);
   **** KEEP DOMAIN SPECIFIC AND OBERVATION CLASS VARIABLES;
   if (domain_prefix = "&domain")  or
      ("&class" = "INTERVENTIONS" and observation_class="Interventions-General") or
      ("&class" = "EVENTS" and observation_class="Events - General") or
      ("&class" = "FINDINGS" and observation_class="Findings - General") or
      ("&class" in ("FINDINGS","EVENTS","INTERVENTIONS") and
      observation_class="All Classes");

   **** FIRST ORDER LABELS ARE DOMAIN SPECIFIC OR COMMON TO ALL CLASSES;
   if (domain_prefix = "&domain" or observation_class = "All Classes") then
      order = 1;

   **** GENERAL DOMAIN CLASSES ARE SECOND ORDER;
   if substr(variable_name,1,2) = "--" then
      do;
         substr(variable_name,1,2) = "&domain";
         order = 2;
      end;
run;

proc sort
   data=cdisc_labels;
      by variable_name order;
run;

**** KEEP DOMAIN SPECIFIC VARIABLE LABEL IF IT EXISTS OVER THE GENERAL CLASS LABEL;
data cdisc_labels;
   set cdisc_labels;
      by variable_name order;

      if first.variable_name;
run;

**** MERGE THE CDISC SDTM VARIABLE LABELS WITH THE DOMAIN DATASET;
data varinfo;
   merge varinfo(in=indomain) cdisc_labels(in=incdisc);
      by variable_name;

      **** NOTE FOR VARIABLES THAT ARE IN THE DOMAIN BUT ARE NOT PART OF THE SDTM;
      if indomain and not incdisc then
         put "NOTE: &domain dataset variable not in CDISC SDS v3.1 " variable_name=;

      **** CHECK FOR EXPECTED OR REQUIRED VARIABLES;
      if incdisc and core in ("Exp","Req") and not indomain then
         put "WARN" "ING: &domain required SDS v3.1 variable not in dataset "
             variable_name=;

      if indomain and incdisc;
run;

**** BUILD A SERIES OF MACRO VARIABLES HOLDING VARIABLE_NAME IN VARS# AND;
**** VARIABLE_LABEL IN LABEL#.;
**** COUNT THE NUMBER OF VARIABLES TO PROCESS AND STORE IT IN MACRO VARIABLE  VARS.;
%let vars = 0;
data _null_;
   set varinfo end=eof;

   retain vars 1;
   call symput("var"     || left(put(vars,best3.)), put(variable_name,$8.));
   call symput("label"   || left(put(vars,best3.)), put(variable_label,$40.));
   if eof then
      call symput("vars", put(vars,8.));
   vars + 1;
run;
```

```
**** ASSIGN CDISC LABELS TO DOMAIN VARIABLES;
%if "&vars" ne "0" %then
   %do;
      data &domain;
         set &domain;

         %do j = 1 %to &vars;
            label &&var&j = "&&label&j";
         %end;
      run;
   %end;
%mend cdisc_label;
```

For example, if I call the %cdisc_label macro like this for an old SDS v3.0 DM dataset:

```
%cdisc_label(class=,domain=DM)
```

I get back a DM dataset in the SAS work space with all of the Submission Data Standards v3.1 variable labels applied. I also see the following in my SAS log:

```
WARNING: DM required SDS v3.1 variable not in dataset variable_name=ARM
WARNING: DM required SDS v3.1 variable not in dataset variable_name=ARMCD
NOTE: DM dataset variable not in CDISC SDS v3.1 variable_name=BRTHDTM
NOTE: DM dataset variable not in CDISC SDS v3.1 variable_name=BRTHDTMP
NOTE: DM dataset variable not in CDISC SDS v3.1 variable_name=REFDTC
NOTE: DM dataset variable not in CDISC SDS v3.1 variable_name=REFDTM
NOTE: DM dataset variable not in CDISC SDS v3.1 variable_name=REFDTMP
WARNING: DM required SDS v3.1 variable not in dataset variable_name=RFENDTC
WARNING: DM required SDS v3.1 variable not in dataset variable_name=RFSTDTC
NOTE: DM dataset variable not in CDISC SDS v3.1 variable_name=TRTCD
NOTE: DM dataset variable not in CDISC SDS v3.1 variable_name=TRTGRP
etc…
```

AUTOMATIC SUPPQUAL MERGING TOOL

The SDTM SUPPQUAL domain contains additional variables and qualifiers that go with the parent domain but are not part of the standard variables of the general observation class for the parent domain. The challenge in using the SUPPQUAL dataset is that it is highly normalized. In SUPPQUAL, the variable names are stored in the QNAM variable and QNAM is qualified with the variable label QLABEL and result QVAL. To rejoin the SUPPQUAL data with the parent domain, you need to employ USUBJID, RDOMAIN, IDVAR, and IDVARVAL.

The following example uses a few variables from a sample SDTM concomitant medications (CM) dataset and some associated SUPPQUAL CM data.

PARTIAL CM DATASET:

studyid	usubjid	Domain	cmseq	cmdecod
101	101001	CM	1	POTASSIUM
101	101001	CM	2	MORPHINE

SUPPQUAL DATASET:

studyid	usubjid	qnam	qlabel	Qval	rdomain	qorig	qeval	idvar	idvarval
101	101001	CM_LLT	WHODRUG Lower Level Term	POTASSIUM CHLORIDE	CM	CRF		CMSEQ	1
101	101001	CM_AEYN	Was Medication Taken for an AE?	NO	CM	CRF		CMSEQ	1
101	101001	ITT	Intent to Treat Flag	YES	DM	DER			
101	101001	CM_YN	Did the patient receive any conmed?	YES	CM	CRF			
101	101001	CM_LLT	WHODRUG Lower Level Term	MORPHINE SULFATE	CM	CRF		CMSEQ	2
101	101001	CM_AEYN	Was Medication Taken for an AE?	NO	CM	CRF		CMSEQ	2

Here is a SAS macro that will bring the SUPPQUAL and given domain datasets back together:

```
/*****************************************************************
 Purpose:  When given a SDTM domain, the MERGESUP macro
           automatically merges the relevant SUPPQUAL dataset
           variables back into the specified domain.

 Assumption:  This program assumes that SUPPQUAL and the specified
              SDTM domain are present in SAS work.

 Parameters:  domain = valid SDTM domain name.

 Sample Call: %mergesup(domain=CM)
*************************************************************/
%macro mergesup(domain=);

**** GET ONLY SPECIFIED DOMAIN DATA OR DATA FOR QUALIFIERS THAT APPLY ACROSS DOMAINS;
proc sort
   data=suppqual;
   where rdomain = "&domain" or idvar = "";
      by usubjid idvarval;
run;

**** LOAD IDVAR VARIABLE INFORMATION FROM SUPPQUAL INTO MACRO VARIABLES;
**** _IDS_     = NUMBER OF DISTINCT NON MISSING IDVARS;
**** _MISSIDS_ = NUMBER OF MISSING IDVARS;
**** _NUMIDS_  = TOTAL NUMBER OF IDVAR = NUMBER OF DISTINCT NON MISSING IDVARS + 1 IF
****             THERE ARE ANY MISSING IDVARS;
%let _ids_=;
%let _missids_=0;
%let _numids_=0;
%let _idvars_=;
proc sql noprint;
   select count(distinct idvar), nmiss(idvar)
      into :_ids_,  :_missids_
      from suppqual;
quit;
%let _numids_ = %eval(&_ids_ + (&_missids_ > 0));


**** BUILD IDVARS MACRO STRING FOR MACRO LOOP BELOW;
proc sql noprint;
   select distinct idvar
      into :_idvars_ separated by ' ,'
      from suppqual;
quit;

**** FOR EACH IDVAR, MERGE THE SUPPQUAL DATA BACK AGAINST THE DOMAIN DATA;
%if &_numids_ > 0 %then
  %do;
    %do i = 1 %to &_numids_;
      **** SELECT THE SUPPQUAL IDVAR OF INTEREST;
      %let idvar=%scan("&_idvars_",&i,",");

      **** TRANSPOSE SUPPQUAL DATA BY IDVARVAL TO MAKE IT FLAT AGAIN;
      proc transpose
        data=suppqual
        out=s_temp;
          where idvar="&idvar";
        by usubjid idvarval;
        var qval;
        id qnam ;
        idlabel qlabel;
      run;
```

```
        **** MERGE SPECIFIC IDVAR DATA FROM SUPPQUAL BACK WITH REGULAR SDTM DOMAIN;
        proc sort
          data=s_temp(drop=_name_
                    %if "&idvar" ne "" %then
                      rename=(idvarval=&idvar);
                    %else
                      idvarval;);
            by usubjid
               %if "&idvar" ne "" %then &idvar;;
        run;

        proc sort
          data=&domain;
            by usubjid
               %if "&idvar" ne "" %then &idvar;;
        run;

        data &domain;
          merge &domain s_temp;
            by usubjid
               %if "&idvar" ne "" %then &idvar;;
        run;
      %end;
    %end;
  %mend mergesup;
```

So, if you call the %mergesup macro above like this:

```
  %mergesup(domain=CM)
```

You get the following CM SAS dataset returned to you with the relevant SUPPQUAL variables merged in:

studyid	usubjid	domain	cmseq	cmdecod	itt	cm_yn	cm_llt	cm_aeyn
101	101001	CM	1	POTASSIUM	YES	YES	POTASSIUM CHLORIDE	NO
101	101001	CM	2	MORPHINE	YES	YES	MORPHINE SULFATE	NO

The PROC CONTENTS of the new CM file in SAS work looks like this:

CM Dataset with SUPPQUAL variables merged in				
Variable	Type	Len	Pos	Label
CMSEQ	Num	8	0	Sequence Number
CM_AEYN	Char	200	750	Was Medication Taken for an AE?
CM_LLT	Char	200	550	WHODRUG Lower Level Term
CM_YN	Char	200	350	Did the patient receive any conmed?
DOMAIN	Char	2	68	Domain Abbreviation
ITT	Char	200	150	Intent to Treat Flag
STUDYID	Char	20	8	Study Identifier
USUBJID	Char	40	28	Unique Subject Identifier
CMDECOD	Char	80	70	Standardized Medication Name

AUTOMATIC SAS FORMATTED VARIABLE TO SDTM CHARACTER VARIABLE CONVERTER

The SDTM does not allow for the logical equivalent of SAS variables with user defined formats. In the SDTM all variables contain the decoded text string values. So, instead of having an AE severity field coded with numeric values 1, 2, or 3 for “Mild”, “Moderate”, or “Severe” you have the SDTM AESEV variable containing the text string “MILD”, “MODERATE”, or “SEVERE”. Since our CDMS data had CDISC named variables with permanent SAS formats attributed we wanted an easy way to replace that variable with the text string equivalent. The following SAS macro replaces SAS variables having user defined permanent formats with the decoded text string equivalent. The original coded variable is then stored in the _*variablename* variable in SAS which could be renamed and placed in the SUPPQUAL dataset if desired.

```
  /*****************************************************************************
   Purpose:  This SAS macro takes a SAS dataset and makes decoded character
             variables out of all non-date variables with permanent SAS formats.
             The original variable is stored as "_variable" where "variable"
             then holds the text equivalent decode for the SDTM.

   Assumptions:  dset specified as macro parameter is a valid dataset.
                 This macro was developed with SAS 8.2 and would have to be modified
```

```
                To work with long format names in version 9.

 Parameters:  dset = SAS dataset name.

 Sample Call: %cdisc_char(dset=AE)
*************************************************************/
%macro cdisc_char(dset=);

**** GET DATASET CONTENT;
proc contents
   data=&dset
   noprint
   fmtlen
   out=varinfo;
run;

**** BUILD A SERIES OF MACRO VARIABLES HOLDING VARIABLE NAME (VAR),
****   VARIABLE LABEL (LABEL), VARIABLE FORMAT (FORMAT), AND FORMAT LENGTH (LFORMAT);
**** COUNT THE NUMBER OF VARIABLES TO PROCESS AND STORE IT IN VARS MACRO VARIABLE;
%let vars = 0;
data _null_;
   set varinfo end=eof;

   **** KEEP ONLY FORMATTED VARIABLES BUT NOT DATES;
   where format not in ('',"DATE","DATETIME");

   retain vars 1;

   call symput("var"     || left(put(vars,best3.)), put(name,$8.));
   call symput("label"   || left(put(vars,best3.)), put(label,$40.));
   call symput("format"  || left(put(vars,best3.)), compress(put(format,$8.)||"."));
   call symput("lformat" || left(put(vars,best3.)), put(formatl,8.));

   if eof then
      call symput("vars", put(vars,8.));

   vars + 1;
run;

**** MAKE SDTM CHARACTER EQUIVALENTS.;
**** RENAME THE OLD VARIABLES TO _VARIABLES.  ASSIGN LENGTH, VALUE, AND LABEL
****    TO NEW VARIABLE.;
%if "&vars" ne "0" %then
   %do;
      data &dset;
         set &dset (rename=( %do i = 1 %to &vars;
                               &&var&i = _&&var&i
                            %end; ))
                    ;
            %do j = 1 %to &vars;
               length &&var&j $ &&lformat&j.;
               &&var&j = upcase(put(_&&var&j,&&format&j));
               label &&var&j = "&&label&j";
            %end;
         run;
    %end;
%mend cdisc_char;
```

For example, assume you have an AE dataset with the following variable:

PROC CONTENTS OF AE DATASET VARIABLE AESEV					
Variable	Type	Len	Pos	Format	Label
AESEV	Num	8	24	C_SEV.	Severity/Intensity

Also assume that the C_SEV format is 1=”Mild”, 2=”Moderate”, 3=”Severe” and that AESEV contains a numeric 1, 2, or 3.

Then, if you run the %cdisc_char SAS macro like this:

```
   %cdisc_char(dset=AE);
```

Then you end up with a SAS dataset that has the following AE severity variables:

PROC CONTENTS OF AE DATASET VARIABLES AESEV AND _AESEV					
Variable	Type	Len	Pos	Format	Label
AESEV	Char	8	731		Severity/Intensity
_AESEV	Num	8	24	C_SEV.	Severity/Intensity

AESEV here is a text string containing "MILD", "MODERATE", or "SEVERE". The original SAS formatted numeric version of AESEV has been placed into _AESEV. By doing this the coded SAS variables can be kept and used for analysis dataset calculations where necessary. To create a valid SDTM AE file, these additional underscore variables could be renamed and placed in the SUPPQUAL dataset and dropped from the domain specific dataset with the following SAS code:

```
   **** DROP ALL LEADING UNDERSCORE VARIABLES;
   data ae;
      set ae;
         drop _:;
   run;
```

ISO 8601 COMPLIANT DATE CONVERTER

In the SDTM version 1.0, dates are represented as ISO 8601 compliant text strings. This is a good approach to date implementation as it stores dates in an easy to read non-proprietary format. To assist with this date formatting, SAS has supplied the IS8601* formats and informats for us as an XML engine enhancement to SAS 8.2. Unfortunately, although SAS requires a full valid ISO 8601date for these formats to work the SDTM allows for partial dates. The following is a SAS macro that will handle partial dates and create a SDTM ISO 8601 compliant date when given the pieces of the date variables as macro parameters.

```
   /*****************************************************************************
    Purpose:  Create --DTC date when supplied with the parts of a date.
              If a date part is left unstated or it contains a missing value,
              then the --DTC result will be truncated at that point per the SDTM
              v1.0 specification.

    Assumption:  This program assumes that this macro is called from
                 within a datastep.

    Parameters:  dtcdate = name of the desired SDTM --DTC date variable.

                 year, month, day, hour, minute, second =
                    numeric variable names representing the respective date part.

    Sample Call: %dtc_date(dtcdate=brthdtc, year=birthm, month=birthd, day=birthy)
   *****************************************************************************/
   %macro dtc_date(dtcdate=, year=., month=., day=., hour=., minute=., second=.);

      length &dtcdate $ 19;

      if (&year = .) then
         &dtcdate = "";
      else if (&year ne . and &month = .) then
         &dtcdate = put(&year,z4.);
      else if (&year ne . and &month ne . and &day = .) then
         &dtcdate = put(&year,z4.)   || "-" ||
                    put(&month,z2.);
      else if (&year ne . and &month ne . and &day ne . and &hour = .) then
         &dtcdate = put(&year,z4.)   || "-" ||
                    put(&month,z2.)  || "-" ||
                    put(&day,z2.);
      else if (&year ne . and &month ne . and &day ne . and &hour ne .
              and &minute = .) then
         &dtcdate = put(&year,z4.)   || "-" ||
                    put(&month,z2.)  || "-" ||
                    put(&day,z2.)    || "T" ||
                    put(&hour,z2.);
      else if (&year ne . and &month ne . and &day ne . and &hour ne .
               and &minute ne . and &second = .) then
         &dtcdate = put(&year,z4.)    || "-" ||
                    put(&month,z2.)   || "-" ||
                    put(&day,z2.)     || "T" ||
                    put(&hour,z2.)    || ":" ||
                    put(&minute,z2.);
      else if (&year ne . and &month ne . and &day ne . and &hour ne .
               and &minute ne . and &second ne .) then
         &dtcdate = put(&year,z4.)    || "-" ||
                    put(&month,z2.)   || "-" ||
                    put(&day,z2.)     || "T" ||
                    put(&hour,z2.)    || ":" ||
                    put(&minute,z2.) || ":" ||
                    put(&second,z2.);
   %mend dtc_date;
```

For example if you have a SAS work dataset containing experimental drug infusion start date and time then you could call the above SAS macro from the data step like this:

```
  %dtc_date(dtcdate=exstdtc,
            year=infusy,month=infusm,day=infusd,hour=infushr,minute=infusmn,second=infussc);
```

And you would get the EXSTDTC variable back as follows:

Obs	infusm	infusd	infusy	infushr	infusmn	infussc	exstdtc
1	2	13	2005	12	31	22	2005-02-13T12:31:22
2	3	.	2005	12	2	13	2005-03
3	4	25	.	6	22	3	

For the first observation you could have used the SAS supplied IS8601DT format to get the EXSTDTC result, but that would not work for the next two observations because they are partial dates. Also note how observations two and three are truncated in EXSTDTC because date components are missing. Partial dates pose an operational challenge in creating your SDTM and subsequent analysis datasets. If you impute dates in the analysis datasets (think ADaM) from the --DTC dates in the SDTM, you may have lost valuable date information during the SDTM conversion. For instance in observation three in the above dataset it could be very easy to determine what the INFUSY variable should be, but there is no place to store the remaining known date pieces in the SDTM. One possible solution to this problem would be to place those date pieces or a date field holding the date pieces in the SUPPQUAL dataset. Then when analysis datasets need to impute dates, the known pieces of the dates are still available in SUPPQUAL.

STUDY DAY CALCULATOR

All dates in the SDTM have a standard study day calculation. The following simple SAS macro performs that derivation.

```
  /*******************************************************************************
   Purpose:  Create --DY STDM compliant study day variable for a given --DTC date.

   Assumptions:  This program assumes that this macro is called from within a
                 datastep and the RFSTDTC variable is present along with the --DTC
                 date.  Also, it is assumed that the --DTC date is complete and of
                 the form YYYY-MM-DDTHH:MM:SS.  This SAS macro was developed with
                 SAS version 8.2 with the IS8601DT format available.

   Parameters:   dtcdate = name of the SDTM --DTC date variable which requires the
                           --DY study day variable.

   Sample Call:  %studyday(dtcdate=aestdtc)
  ********************************************************************************/
  %macro studyday(dtcdate=);
    if datepart(input(&dtcdate,is8601dt.)) >= datepart(input(rfstdtc,is8601dt.)) > . then
      %substr("&dtcdate",2,%length(&dtcdate)-3)dy =
             datepart(input(&dtcdate,is8601dt.)) - datepart(input(rfstdtc,is8601dt.)) + 1;
    else if .< datepart(input(&dtcdate,is8601dt.)) < datepart(input(rfstdtc,is8601dt.)) then
      %substr("&dtcdate",2,%length(&dtcdate)-3)dy =
             datepart(input(&dtcdate,is8601dt.)) - datepart(input(rfstdtc,is8601dt.));
  %mend studyday;
```

So, for example the following SAS code would yield the subsequent PROC PRINT result.

```
data dm_and_ae;
   aestdtc = "2004-04-14T10:00:00";
   rfstdtc = "2004-04-13T12:30:00";

   %studyday(dtcdate=aestdtc)
run;

proc print;
run;
```

obs	aestdtc	rfstdtc	aestdy
1	2004-04-14T10:00:00	2004-04-13T12:30:00	2

CONCLUSION

This paper shows just one way that the SDTM can be implemented and there are certainly many more implementations possible and already underway. I hope that some of the ideas presented in this paper may be of some use to those getting started with SDTM implementations. The SDTM implementation at the DCRI is an ongoing process and we continue to study ways in which to implement the CDISC models more effectively.

In the long term, the models being published by CDISC will make the conduct and analysis of clinical trials a much more efficient process. In the end, much money will be saved and resources can be reallocated to drug discovery activities. However, with that said, I believe that those who implement the SDTM now will probably incur increased operational cost and some production delays as they adapt their operations to using the SDTM. I believe that eventually when the SDTM is accepted as a stable standard then you will see more technology vendors supplying automated tools to work with the SDTM. You will see the CDMS vendors making software that produces SDTM files at the push of a button. You will see automated analysis tools that manipulate and analyze the SDTM files. Some of this work has already been completed, PROC CDISC at SAS for example, but for now the CDISC SDTM and other models are a bit ahead of the software that will need to use them.

REFERENCES

Study Data Tabulation Model version 1.0,
Study Data Tabulation Model Implementation Guide: Human Clinical Trials version 1.0,
Statistical Analysis Dataset Model: General Considerations Version 1.0 and other CDISC standards found at http://www.cdisc.org/standards/index.html

ACKNOWLEDGMENTS

Many thanks to the fine people of the Submission Data Standards team at CDISC for their continued devotion and assistance.

RECOMMENDED READING

[1] The CDISC message boards at http://www.cdisc.org/discussions/discussions.html.
[2] SAS (2004), *The CDISC Procedure for SAS Software* at http://support.sas.com/rnd/base/index-xml-resources.html
[3] Kenney, Susan (2005), *Strategies for Implementing SDTM and ADaM Standards.* Proceedings of the PharmaSUG 2005 Conference.
[4] Friebel, A., Helton, E., Bastos, E., Kilhullen, M. and Boone, J. (2004) *Defining and Validating CDISC Data Standards to XML in SAS® Technology*. Proceedings of the PharmaSUG 2004 conference at http://www.lexjansen.com/pharmasug/2004/SASInstitute/SAS3.pdf
[5] Sattler, Jim (2004), *A Table-Driven Solution for Clinical Data Submission*. Proceedings of SAS User's Group International (SUGI) 2004 Conference.

CONTACT INFORMATION

Any comments that you may have are valued and encouraged. Contact the author at:

Jack Shostak
Duke Clinical Research Institute

Address	P.O Box 17969
City state ZIP	Durham, NC 27715
Work Phone:	919-668-8832
Email:	jack.shostak@duke.edu