A SAS macro to check SDTM domains against controlled terminology

Guido Wendland

UCB Biosciences GmbH, Monheim, Germany

The SAS macro presented here allows programmers to check the compliance of SDTM domains with controlled terminology. The macro is based on two input metadata Excel files: one file contains variables/variable groups (e.g. --ACN) together with their corresponding 'codelist' term (e.g. ACN). The other is a reference list of all 'codelist' terms and their controlled terminology values (e.g. DOSE INCREASED, DOSE NOT CHANGED, etc.). Both files are based on the OpenCDISC standard checks for controlled terminology but could be customized to the sponsor's needs. The primary output consists of a list of all values that could not be found in the controlled terminology. Furthermore, the corresponding entries that have not been successfully mapped are also provided. Multiple studies and domains can be checked simultaneously. Therefore, programmers can use the macro at various stages, e.g. during the SDTM development process of a single domain, or when preparing multiple studies for pooling.

Keywords: Controlled terminology, Codelists, ExcelXP, Metadata, SDTM

Introduction

The Clinical Data Interchange Standards Consortium (CDISC) Study Data Tabulation Model (SDTM) defines a standard format for the exchange of clinical data among various stakeholders such as pharmaceutical companies and regulatory agencies. Controlled terminology (CT) is an integral part of SDTM. According to the SDTM 3.1.2 Implementation Guide,[1] some character variables should conform to a CT. CDISC regularly publishes codelists (i.e. list of variable values) via the National Cancer Institute (NCI) homepage.[2] These CDISC codelists are a major component of the CDSIC SDTM 3.1.2 Validation Rules.[3] The Food and Drug Administration (FDA) requests that 'Data values for CDISC standards-specified variables should use the CDISC Controlled Terminology'.[4]

Most of the CDISC codelists are extensible, where sponsors can define and add values to existing codelists. Sponsors may also define separate sponsor-specific or study-specific codelists.

The examples in this paper focus on SDTM 3.1.2 CT, but a different set of CT could be applied to the Analysis Dataset Model (ADaM) domains developed from SDTM domains.

The paper outlines the technical implementation of checking compliance to CT with SAS®. This involves the following components:

Correspondence to: Guido Wendland, UCB Biosciences GmbH, Alfred-Nobel-Straße 10, 40789 Monheim, Germany. Email: guido.wendland@ucb.com

- input metadata: two metadata datasets are used as input files;
- input SDTM data: the SDTM data are scanned for variables that have to comply with CT;
- for each variable, the list of allowed values is derived;
- the violation dataset(s) is/are created with one observation per violation;
- the checks are extended to include multiple libnames and domains;
- reporting of the results.

The process is outlined in a flowchart (Fig. 1).

The focus is on the technical programming aspects; therefore, the text contains a couple of programming blocks. Keyword macro parameters which can be specified by the user are denoted in capital letters, while temporary macro variables are in lower case and start with an underscore ('_').

Good macro programming practice and techniques (e.g. restore options at the end, declare local macro variables, etc.) were followed in the development of this utility macro, using practices previously described.[5,6]

Metadata Sources

The macro uses metadata from two files. Both files are originally Excel files that are converted into SAS datasets.

The first file contains the codelists applied to SDTM variables. Fig. 2 presents the structure of the CT file delivered by CDISC. Each codelist can be identified by the value in the column 'codelist name' and by its short form which is displayed in the column 'CDISC Submission Value' on the first row of each codelist name.

DOI 10.1179/175709311X13152986403310

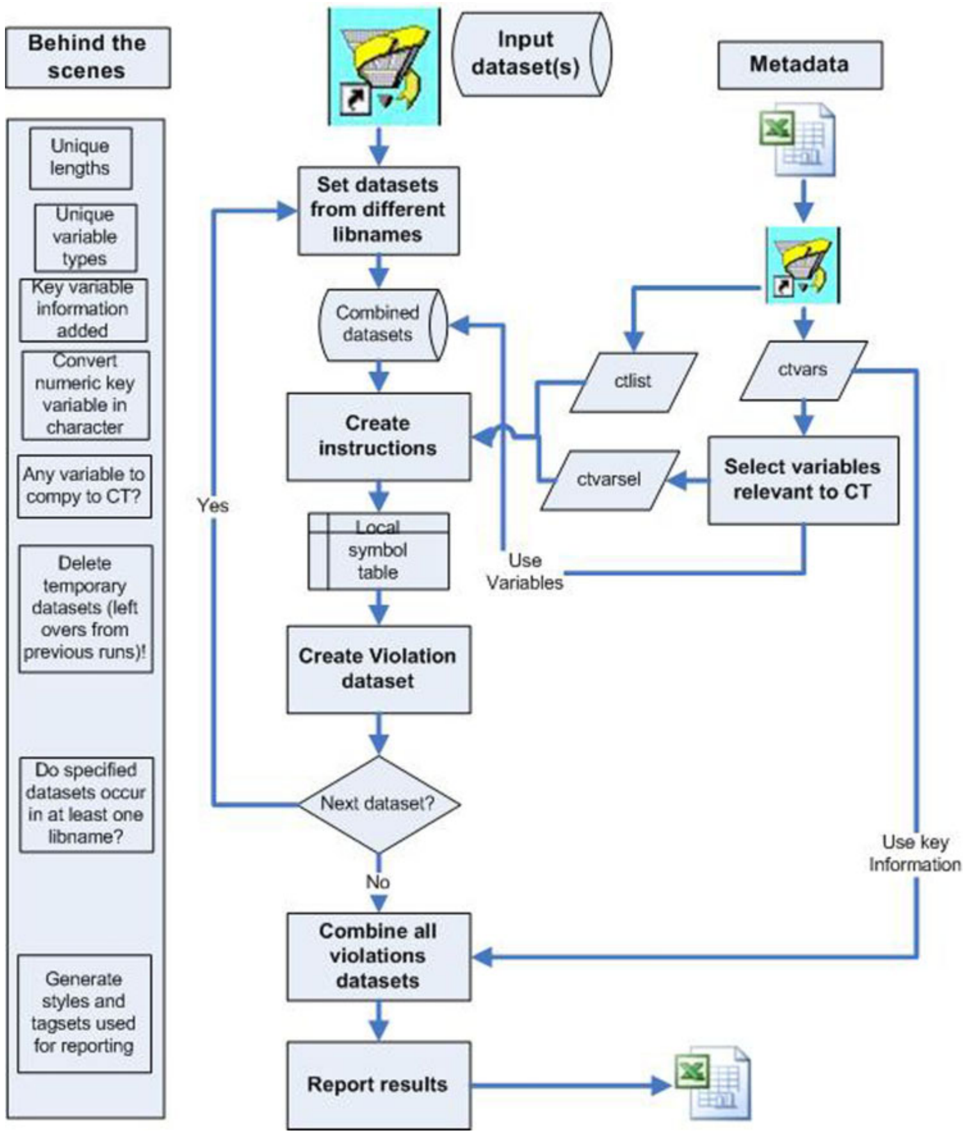

Behind the scenes
Unique lengths
Unique variable types
Key variable information added
Convert numeric key variable in character
Any variable to compy to CT?
Delete temporary datasets (left overs from previous runs)!
Do specified datasets occur in at least one libname?
Generate styles and tagsets used for reporting
Input dataset(s)
Metadata
Set datasets from different libnames
Combined datasets
Create instructions
Local symbol table
Create Violation dataset
Next dataset?
Yes
No
Combine all violations datasets
Report results
ctlist
ctvars
ctvarsel
Select variables relevant to CT
Use Variables
Use key Information

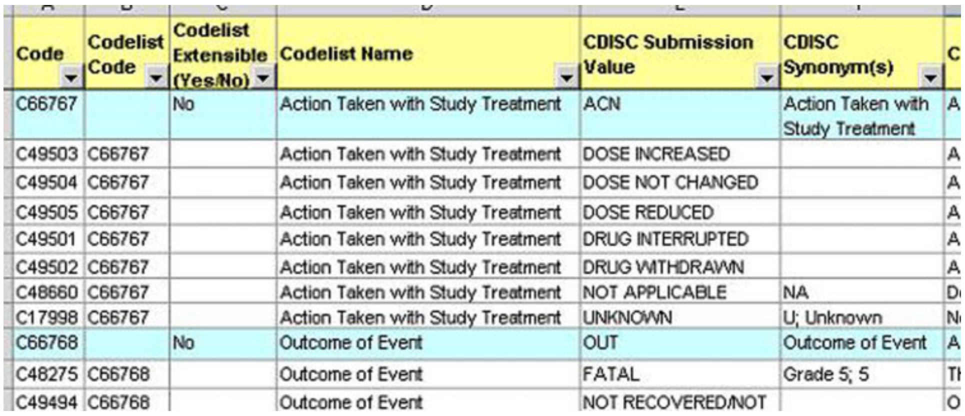
Figure 1 Flowchart of the SAS macro

This Excel file is converted into a SAS dataset (Fig. 3), let us call it **ctlist.sas7bdat.**

The second metadata file contains the link between the variables and the CT they have to comply with. A template Excel file will be available open request. Again an Excel file is converted into SAS dataset (**ctvars.sas7bdat**). An excerpt of the SAS dataset is shown in Fig. 4.

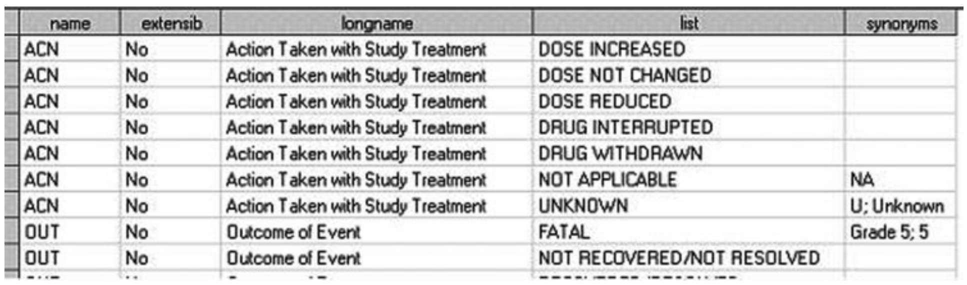
If multiple users apply the macro, the input metadata files would be preferably maintained centrally, e.g. to adapt to new CT or add sponsor specific rules.

The three highlighted variables are relevant for the mapping of dataset variables to codelists. The other variables contain additional information about any corresponding CT violations which will be used later for reporting. The variable RuleID (CT0001–CT0076)



Code	Codelist Code	Codelist Extensible (Yes/No)	Codelist Name	CDISC Submission Value	CDISC Synonym(s)	C
C66767		No	Action Taken with Study Treatment	ACN	Action Taken with Study Treatment	A
C49503	C66767		Action Taken with Study Treatment	DOSE INCREASED		A
C49504	C66767		Action Taken with Study Treatment	DOSE NOT CHANGED		A
C49505	C66767		Action Taken with Study Treatment	DOSE REDUCED		A
C49501	C66767		Action Taken with Study Treatment	DRUG INTERRUPTED		A
C49502	C66767		Action Taken with Study Treatment	DRUG WITHDRAWN		A
C48660	C66767		Action Taken with Study Treatment	NOT APPLICABLE	NA	D
C17998	C66767		Action Taken with Study Treatment	UNKNOWN	U; Unknown	N
C66768		No	Outcome of Event	OUT	Outcome of Event	A
C48275	C66768		Outcome of Event	FATAL	Grade 5; 5	T
C49494	C66768		Outcome of Event	NOT RECOVERED/NOT		O

Figure 2 Excerpt from CDISC provided controlled terminology (Excel)



name	extensib	longname	list	synonyms
ACN	No	Action Taken with Study Treatment	DOSE INCREASED	
ACN	No	Action Taken with Study Treatment	DOSE NOT CHANGED	
ACN	No	Action Taken with Study Treatment	DOSE REDUCED	
ACN	No	Action Taken with Study Treatment	DRUG INTERRUPTED	
ACN	No	Action Taken with Study Treatment	DRUG WITHDRAWN	
ACN	No	Action Taken with Study Treatment	NOT APPLICABLE	NA
ACN	No	Action Taken with Study Treatment	UNKNOWN	U; Unknown
OUT	No	Outcome of Event	FATAL	Grade 5; 5
OUT	No	Outcome of Event	NOT RECOVERED/NOT RESOLVED	

Figure 3 Excerpt of SAS dataset containing controlled terminology

corresponds to the CT-related validation rules outlined by OPEN CDISC[7] which are applied by the FDA when reading the data into their clinical data repository.

The metadata file addresses the fact that some variables contained in multiple domains are only distinguishable by their domain prefixes. If the same codelist applies to such a class of variables, the variable name is prefixed by a double dash (–). Figure 4 shows two examples of this (for RuleID='CT0001' and 'CT0016') highlighted in blue.

Some codelists only apply conditionally depending on values of other variables. In these cases, a 'where condition' needs to be applied, as indicated for RuleID=CT0003 and CT0005. Due to its set-up, this is a common occurrence for the Trial Summary (TS) domain but not limited to it.

The metadata files can be modified for the specific needs of the sponsor, e.g. if the sponsor has a restricted set of values for some variables or to add extensions for extensible codelists.

Scan Input Dataset for Variables that Have to Comply with CT

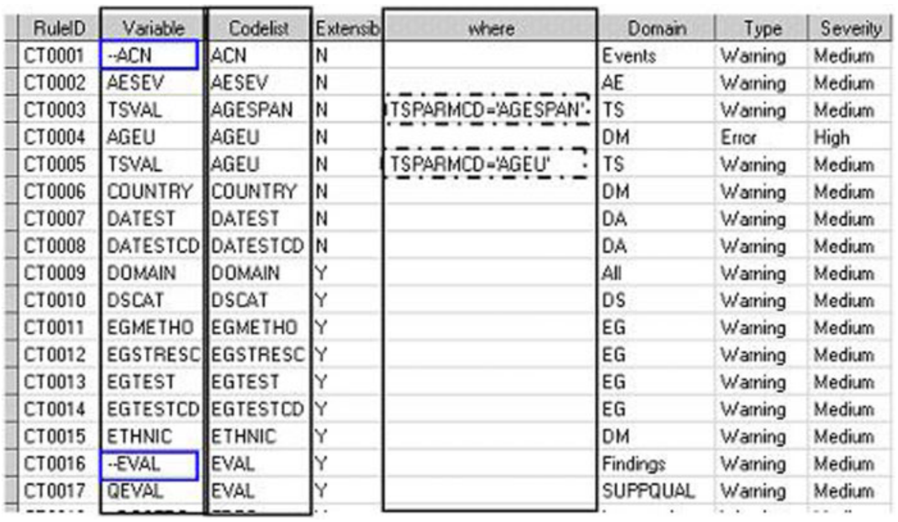
The presented macro operates on an SDTM input dataset that should be checked for CT compliance. For simplicity, it is assumed first that the input dataset contains data from a single study and the single domain AE.

The first step is to identify the variables which need to comply with CT. To get these variables, we create two lists of variables and then determine those which are contained in both lists. The first variable list is retrieved from a modification of dataset **ctvars. sas7bdat** where for all values of 'variable', the double dash is replaced by the corresponding domain, e.g. AEACN, AESEV, AEOUT, etc. This can be achieved using:

```
proc sql noprint;
  select distinct(variable)
    into : list1 separated by ' '
    from ctvars_mod;
quit;
```

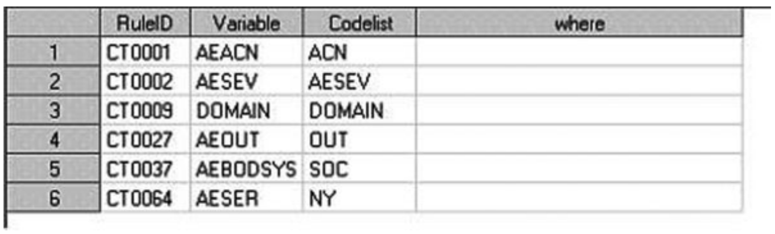
The second variable list contains the variables in the SDTM dataset under investigation: AESEQ, AETERM, AEDECOD, etc. Let us assume that the two lists are stored as space-separated macro variables &list1 and &list2. To store the intersection of the variable lists in the macro variable &_intersect, the following code is applied:

```
%local _intersect;
 %let _count1=1;
  %do %until (%qscan(&list1.,&_count1.)
= %STR( ) ) ;
    %let _count2=1;
    %do %until (%qscan(&list2.,&_-
count2.) = %STR( ) ) ;
      %if %upcase(%qscan(&list1.,&_
count1.))=%upcase(
      %qscan(&list2.,&_count2.)) %then
```



RuleID	Variable	Codelist	Extensib	where	Domain	Type	Severity
CT0001	--ACN	ACN	N		Events	Warning	Medium
CT0002	AESEV	AESEV	N		AE	Warning	Medium
CT0003	TSVAL	AGESPAN	N	TSPARMCD='AGESPAN'	TS	Warning	Medium
CT0004	AGEU	AGEU	N		DM	Error	High
CT0005	TSVAL	AGEU	N	TSPARMCD='AGEU'	TS	Warning	Medium
CT0006	COUNTRY	COUNTRY	N		DM	Warning	Medium
CT0007	DATEST	DATEST	N		DA	Warning	Medium
CT0008	DATESTCD	DATESTCD	N		DA	Warning	Medium
CT0009	DOMAIN	DOMAIN	Y		All	Warning	Medium
CT0010	DSCAT	DSCAT	Y		DS	Warning	Medium
CT0011	EGMETHO	EGMETHO	Y		EG	Warning	Medium
CT0012	EGSTRESC	EGSTRESC	Y		EG	Warning	Medium
CT0013	EGTEST	EGTEST	Y		EG	Warning	Medium
CT0014	EGTESTCD	EGTESTCD	Y		EG	Warning	Medium
CT0015	ETHNIC	ETHNIC	Y		DM	Warning	Medium
CT0016	--EVAL	EVAL	Y		Findings	Warning	Medium
CT0017	QEVAL	EVAL	Y		SUPPQUAL	Warning	Medium

Figure 4 Link between SDTM variables and codelists



	RuleID	Variable	Codelist	where
1	CT0001	AEACN	ACN	
2	CT0002	AESEV	AESEV	
3	CT0009	DOMAIN	DOMAIN	
4	CT0027	AEOUT	OUT	
5	CT0037	AEBODSYS	SOC	
6	CT0064	AESER	NY	

Figure 5 Relevant variables for CT compliance (example AE)

```
      %let _intersect=&_intersect
      %qscan(&list2.,&_count2.);
      %let _count2=%eval(&_count2.+1);
    %end;
  %let _count1=%eval(&_count1.+1);
%end;
```

This list of variables can now be used to subset dataset ctvars (Fig. 4). Furthermore, we can drop all variables containing descriptive attributes which are not needed to derive the violations. These will be merged back in for creating the reports. An example for the resulting SAS dataset **ctvarsel.sas7bdat** is shown in Fig. 5.

Retrieving the Codelist Values

In order to find observations in the SDTM dataset whose variable values do not comply with CT, we need to formulate the violation conditions. In **ctvarsel.sas7bdat** (Fig. 5), we need to add the list of corresponding 'codelist' values from the other metadata dataset **ctlist.sas7bdat**. In the code presented below, it is required that all variables are of character type. Ideally, the list elements should be enclosed in quotes, so the list can be used together with the 'in' operator.

```
proc sql;
 create table ctvarmod as
  select a.*, b.list
    from ctvarsel as a
      left join ctlist as b
      on a.codelist=b.name
      order by variable, where;
quit;
```

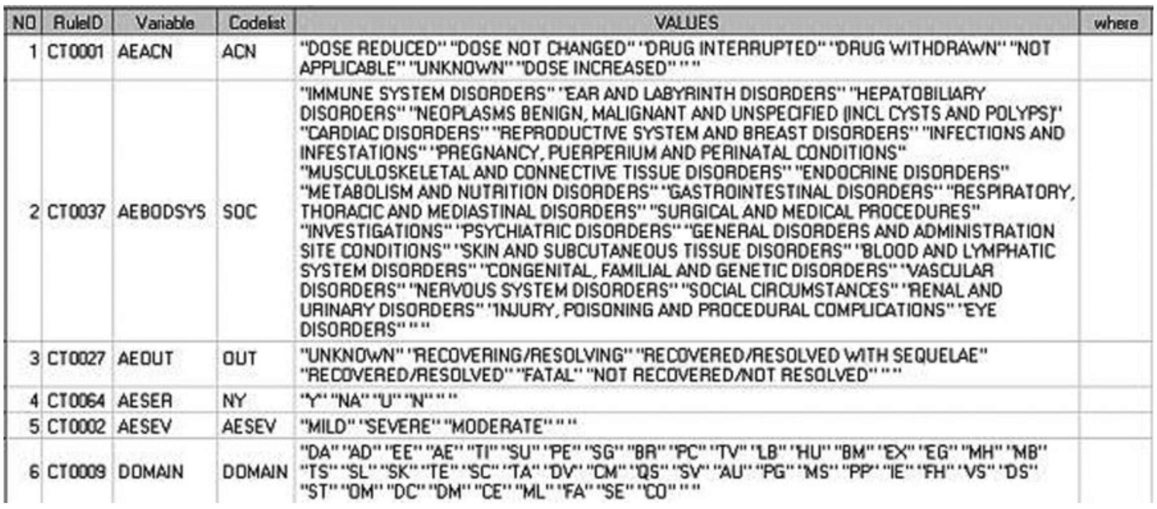
Bear in mind that for variables in **ctvarsel.sas7bdat**, multiple observations could be present with different where conditions (e.g. see Fig. 3 for the TSPARMCD values in the TS domain). Therefore, we need to maintain all combinations of 'variable' and 'where'. Furthermore, assuming that they are always allowed empty character values are added to the end of each list.

```
data ctvarmd;
 retain no ruleid codelist variable
 values where;
 length values $32767;
 set ctvarmod;
  by variable where;
   if first.where then values="";
     values=catx(' " "',strip(values),
     strip(list));
   if last.where then do;
       values='"'||strip
      (values)||'"'||'""';
      no+1;
      output;
    end;
 drop list;
run;
```

The resulting SAS dataset (**ctvarmd.sas7bdat**) contains the selected variables together with their 'allowed' values (variable VALUES) and a row number (NO):

Creating the CT Violation Datasets

The violations dataset combines the SDTM dataset and the metadata needed to formulate the CT



NO	RuleID	Variable	Codelist	VALUES	where
1	CT0001	AEACN	ACN	"DOSE REDUCED" "DOSE NOT CHANGED" "DRUG INTERRUPTED" "DRUG WITHDRAWN" "NOT APPLICABLE" "UNKNOWN" "DOSE INCREASED" ""	
2	CT0037	AEBODSYS	SOC	"IMMUNE SYSTEM DISORDERS" "EAR AND LABYRINTH DISORDERS" "HEPATOBILIARY DISORDERS" "NEOPLASMS BENIGN, MALIGNANT AND UNSPECIFIED (INCL CYSTS AND POLYPS)" "CARDIAC DISORDERS" "REPRODUCTIVE SYSTEM AND BREAST DISORDERS" "INFECTIONS AND INFESTATIONS" "PREGNANCY, PUERPERIUM AND PERINATAL CONDITIONS" "MUSCULOSKELETAL AND CONNECTIVE TISSUE DISORDERS" "ENDOCRINE DISORDERS" "METABOLISM AND NUTRITION DISORDERS" "GASTROINTESTINAL DISORDERS" "RESPIRATORY, THORACIC AND MEDIASTINAL DISORDERS" "SURGICAL AND MEDICAL PROCEDURES" "INVESTIGATIONS" "PSYCHIATRIC DISORDERS" "GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS" "SKIN AND SUBCUTANEOUS TISSUE DISORDERS" "BLOOD AND LYMPHATIC SYSTEM DISORDERS" "CONGENITAL, FAMILIAL AND GENETIC DISORDERS" "VASCULAR DISORDERS" "NERVOUS SYSTEM DISORDERS" "SOCIAL CIRCUMSTANCES" "RENAL AND URINARY DISORDERS" "INJURY, POISONING AND PROCEDURAL COMPLICATIONS" "EYE DISORDERS" ""	
3	CT0027	AEOUT	OUT	"UNKNOWN" "RECOVERING/RESOLVING" "RECOVERED/RESOLVED WITH SEQUELAE" "RECOVERED/RESOLVED" "FATAL" "NOT RECOVERED/NOT RESOLVED" ""	
4	CT0064	AESER	NY	"Y" "NA" "U" "N" ""	
5	CT0002	AESEV	AESEV	"MILD" "SEVERE" "MODERATE" ""	
6	CT0009	DOMAIN	DOMAIN	"DA" "AD" "EE" "AE" "TI" "SU" "PE" "SG" "BR" "PC" "TV" "LB" "HU" "BM" "EX" "EG" "MH" "MB" "TS" "SL" "SK" "TE" "SC" "TA" "DV" "CM" "QS" "SV" "AU" "PG" "MS" "PP" "IE" "FH" "VS" "DS" "ST" "OM" "DC" "DM" "CE" "ML" "FA" "SE" "CO" ""	

Figure 6 Relevant variables for CT compliance (example AE) and their 'allowed' values

violation conditions (**ctvarmd.sas7bdat**). This is implemented via a %syscall set() statement to convert the values of the data step variables of dataset ctvarmd.sas7bdat into macro variables. These macro variables are then used in subsetting if statements to only keep observations of the SDTM dataset that contains a CT violation. Note that the where condition is only added to the if-statement if the variable "where" contains a value (Fig. 6).

```
data violdset (keep=keyinfo source
domain ruleid variable value);
   length source variable domain $8
ruleid $40 value $200 keyinfo $1000;
   set dataset;
     %let _id=%sysfunc(open(ctvarmd));
     %let _nobs=%sysfunc(attrn(&_id,
     NOBS));
     %syscall set(_id);
     %do _k=1 %to &_nobs.;
        %let rc=%sysfunc(fetchobs
        (&_id,&_k));
/**** Create the instructions text ****/
        ruleid="&ruleid.";
        variable="&variable.";
        value = &variable;
        domain= "%upcase(dataset)";
        if &variable not in (&values. )
        %IF %NRBQUOTE(&where) ne %THEN
        and &where.;
            then output;
     %end;
     %let _id=sysfunc(close(&_id));
run;
```

Extension: Check CT over Multiple Studies and Domains

What we have seen so far covers the simple case, where just one SDTM domain dataset is to be checked. This functionality of the macro can be expanded in two directions:

- to a list of specified datasets or all SDTM domain datasets available for a libname (macro parameter &DSETLIST);
- to a list of libnames indicating SDTM domain data coming from different studies (macro parameter &LIBNAMES).

This would make the tool interesting not only for a single programmer dealing with a particular SDTM domain, but also for project programmers who manage SDTM Domains over multiple studies or projects.

Preparation for extending to multiple datasets

Generally speaking, the extensions are implemented applying appropriate macro-looping, starting with splitting up the lists (libnames and datasets) into single elements, e.g. where &DSETLIST is the list of user-specified datasets:

```
%let _dscount=1;
%do %until
  (%qscan(%quote(&DSETLIST.),&_dscount.)
   = %str( ) );
  %let _dset&_dscount.=%upcase
  (%qscan(%quote(&dsetlist.),&_ds
  count.));
  %let _dscount=%eval(&_dscount.+1);
%end;
```

Furthermore, a couple of additional checks should be implemented, including:

- check that all specified libnames exist;
- check that at least one of the specified datasets is contained in at least one libname.

To illustrate the latter, further assume that &_dslist is the list of datasets contained in the specified libnames and &_dscnt is the count of elements in this list. The list of datasets can be retrieved from the view sashelp.vcolumn.

The following code block checks for each dataset of &DSETLIST whether or not it is contained in any of the libnames. If the dataset is found, it is listed in the local macro variable &_found and added to the list of datasets already found earlier during the looping.

```
%let _count1=1;
  %do %until
  (%qscan(&DSETLIST.,&_count1.) =
   %STR( ) ) ;
    %let _count2=1;
    %do %until (%qscan(&_dslist.,&
    _count2.) = %STR( ) ) ;
      %if
      %upcase(%qscan(&DSETLIST.,&
      _count1.))= %upcase(%qscan(&
      _dslist.,&_count2.))
        %then %let _flag=1;
        %if &_count2=&_dscnt %then %do;
        /** Last element of list reached
        ***/
            %if &_flag ne 1 %then
                 %let _notfound
                 =&_notfound
          %upcase(%qscan(&DSETLIST.,
          &_count1.));
          %let _flag=0; /** Reset _flag
          for next macro variable of &
          DSETLIST. ***/
      %end;
       %let _count2=%eval
       (&_count2.+1);
    %end;
   %let _count1=%eval(&_count1.+1);
%end;
%if &_notfound ne %then %do;
  %put;
  %put %STR(ER)ROR: The following data
   set(s) "&_notfound." are not found in
   any of the specified libnames.
   Macro is aborted.;
  %put;
  %goto exit1;
%end;
```

Combining datasets from different libnames (e.g. studies)

In order to combine datasets from different libnames, the same steps have to be implemented for all the datasets (e.g. AE, EG, LB, VS, etc.), so we frame all the following steps in this chapter with a %do-loop over all datasets:

```
%do _i=1 %to &_dscount.-1;
...
%end;
```

When setting together SDTM datasets, their common variable attributes — label, type, and length — have to be accounted for. This paper focuses on the length of the variables as this is probably the most critical attribute when dealing with SDTM. The following statements were used to create a macro variable &_lengthv that contains a length statement. It accounts for the maximum length of the variable observed across all libnames. This statement occurs within the looping through the datasets indicated by loop variable &_i:

```
proc sql noprint ;
  select distinct(compress(upcase(
  name))||" $"||strip(put(max(
  length),best.)))
    into : _lengthv separated by' '
     from sashelp.vcolumn
     where libname in (quoted list of
     libnames) and
                upcase(memname)=upca
                se("&&_dset&_i..") and
                lowcase(type)="char"
     group by name;
quit;
```

Useful information for later reporting includes the key variable values for which a CT violation is observed. To prepare for this, the key variables are retrieved from sashelp.vcolumn. In contrast to the macro variable &_lengthv (see previous code snippet), we need to retrieve separate keys for each different libname because the key variables could differ by libname. So we need to loop through the specified libnames. Assume that &_j is the loop variable for libnames, &_libcount.-1 is the total number of libnames specified, and &&_lib&_j. are the single libnames. In the following sql statement, we store the key variables in the macro variable &&_keys&_j:

```
%do _j=1 %to &_libcount-1;
/* &&_keys&_j. would otherwise not be
  created if sql-query does not select
  any rows */
   %let _keys&_j.=;
   proc sql noprint ;
     select name
       into : _keys&_j. separated by' '
       from sashelp.vcolumn
       where libname in ("%upcase(&&_
       lib&_j.)") and upcase(memname)=
          upcase("&&_dset&_i..") and
          sortedby>0
       order by sortedby
     ;
    quit;
%end;
```

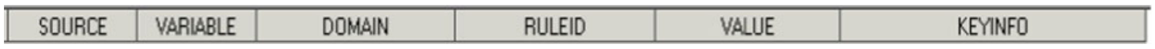
The assignment of &&_keys&_j. could be refined by allowing user-specified keys (e.g. via macro parameter KEYS) or by providing default keys if the input datasets are not sorted. In both cases, any variables not in the corresponding dataset should be dropped from that list.

Now we can combine the SDTM domain datasets from the (various) libnames. The following dataset _&&_dset&_i. collects all identical domains from the various libnames. The libname from which the data originated is stored in the dataset variable SOURCE and the key variable information is collected in the variable KEYINFO. For the purpose of the following data step, any numeric variables (collected in &_numlist in a previous step) are converted to character to avoid the following note:

```
NOTE: Numeric values have been converted
to character values at the places given
by:…
```

The code is displayed in Fig. 7.

As a result, the SDTM domain violation datasets can be created from combining violations from all specified libnames. All these files contain the same variables:

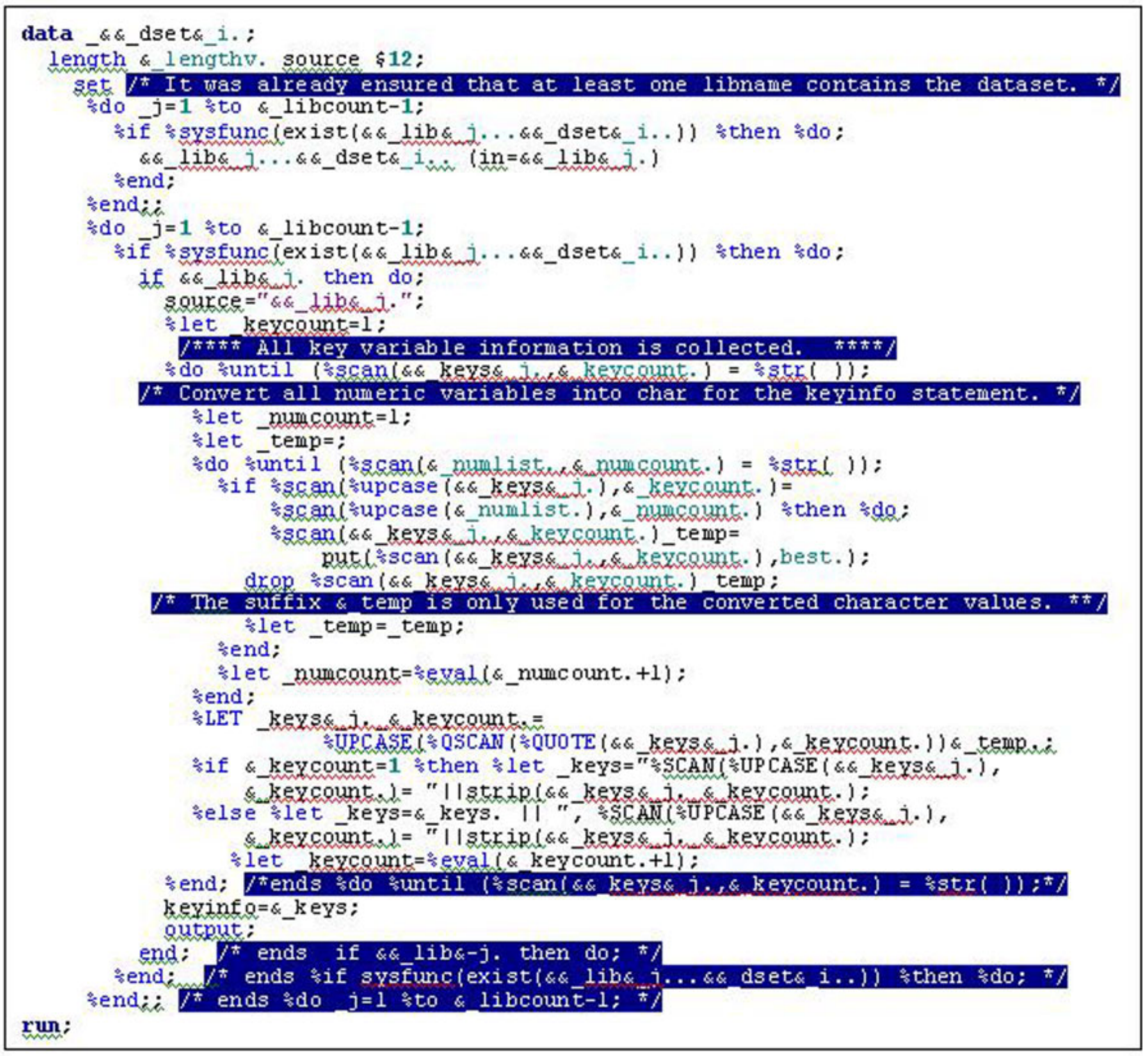


SOURCE	VARIABLE	DOMAIN	RULEID	VALUE	KEYINFO

Combining the violation datasets and preparing the reporting datasets

In the next steps, the single violations datasets all contain the same variables:

- create an overall violations dataset **violdset.sas7bdat**;
- merge **violdset.sas7bdat** with **ctvars.sas7bdat** by RULEID to get additional violation information (e.g. codelist, type of warning, etc.). Resulting dataset is **violdset2.sas7bdat**;
- create useful messages for all violations including the variable KEYINFO.



```
data _&&_dset&_i.;
  length &_lengthv. source $12;
    set /* It was already ensured that at least one libname contains the dataset. */
     %do _j=1 %to &_libcount-1;
       %if %sysfunc(exist(&&_lib&_j...&&_dset&_i..)) %then %do;
         &&_lib&_j...&&_dset&_i.. (in=&&_lib&_j.)
       %end;
     %end;;
     %do _j=1 %to &_libcount-1;
       %if %sysfunc(exist(&&_lib&_j...&&_dset&_i..)) %then %do;
         if &&_lib&_j. then do;
           source="&&_lib&_j.";
           %let _keycount=1;
            /**** All key variable information is collected.  ****/
           %do %until (%scan(&&_keys&_j.,&_keycount.) = %str( ));
         /* Convert all numeric variables into char for the keyinfo statement. */
             %let _numcount=1;
             %let _temp=;
             %do %until (%scan(&_numlist.,&_numcount.) = %str( ));
               %if %scan(%upcase(&&_keys&_j.),&_keycount.)=
                   %scan(%upcase(&_numlist.),&_numcount.) %then %do;
                   %scan(&&_keys&_j.,&_keycount.)_temp=
                       put(%scan(&&_keys&_j.,&_keycount.),best.);
                 drop %scan(&&_keys&_j.,&_keycount.)_temp;
          /* The suffix & temp is only used for the converted character values. **/
                 %let _temp=_temp;
               %end;
               %let _numcount=%eval(&_numcount.+1);
             %end;
             %LET _keys&_j._&_keycount.=
                       %UPCASE(%QSCAN(%QUOTE(&&_keys&_j.),&_keycount.))&_temp.;
             %if &_keycount=1 %then %let _keys="%SCAN(%UPCASE(&&_keys&_j.),
                 &_keycount.)= "||strip(&&_keys&_j._&_keycount.);
             %else %let _keys=&_keys. || ", %SCAN(%UPCASE(&&_keys&_j.),
                 &_keycount.)= "||strip(&&_keys&_j._&_keycount.);
               %let _keycount=%eval(&_keycount.+1);
           %end; /*ends %do %until (%scan(&&_keys&_j.,&_keycount.) = %str( ));*/
           keyinfo=&_keys;
           output;
         end;  /* ends  if &&_lib&-j. then do; */
       %end;  /* ends %if sysfunc(exist(&&_lib&_j...&&_dset&_i..)) %then %do; */
     %end;; /* ends %do _j=1 %to &_libcount-1; */
run;
```

Figure 7 Combining datasets from multiple libnames maintaining the source libname and key information

After completing these steps, we have a dataset that contains one observation per CT violation and all accompanying information.

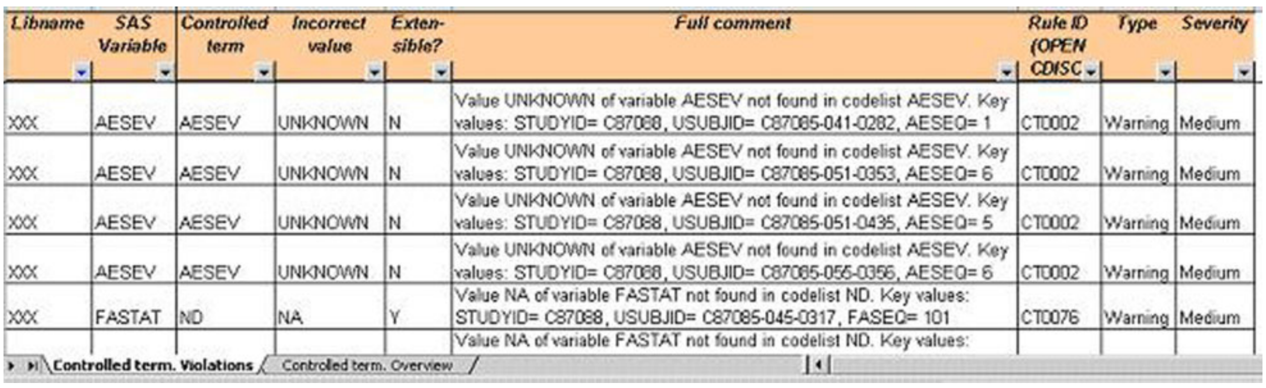
For the user of the macro, it will be also helpful to know the list of allowed values for those CT's that were violated against. This dataset — let us call it **ctsel.sas7bdat** — is a subset of **ctlist.sas7bdat** restricted to those codelists occurring in **violdset2.-sas7bdat**.

Reporting

For reporting, we chose a format that allows the user to easily view and navigate through the findings. SAS provides a lot of different output alternatives via the output delivery system (ODS). One of these alternatives, ods tagsets.ExcelXP, was chosen as the output destination, because it supports a well-arranged report.

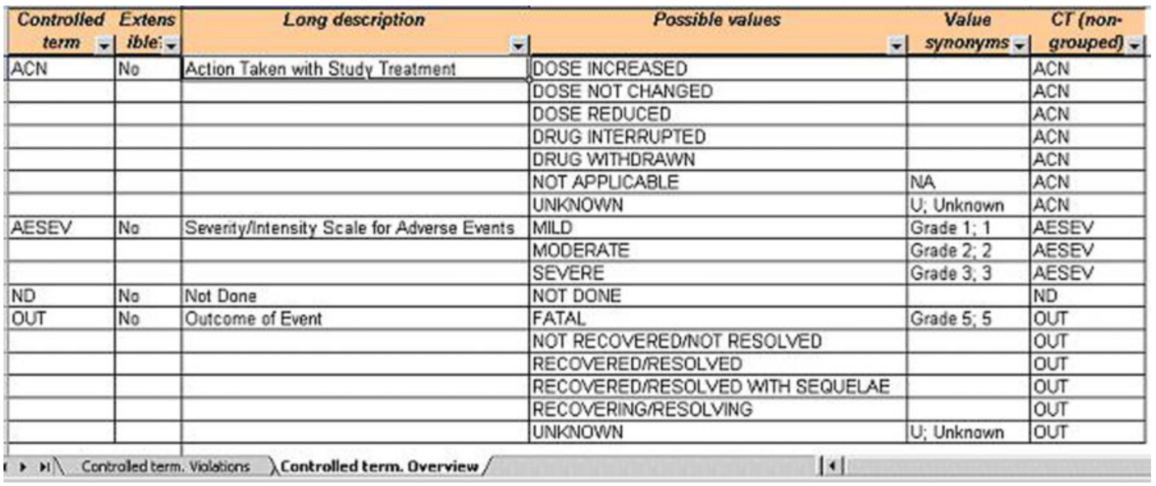
ODS tagsets.ExcelXP creates an extensible markup language (xml) file containing one or multiple worksheets that can be opened and modified with Excel.[8] It allows the user to control many of the Excel features from within SAS, e.g. autofilters, frozen headers, printer orientation, etc. The layout of the Excel worksheets is controlled via a 'style'. Styles are created by the procedure proc template and stored in itemstores. They can be provided to a group of programmers.

```
ods listing close;
 ods tagsets.ExcelXP file="outputfile.xml"
 path="output path" style=UCB2;
 ods tagsets.ExcelXP
   options (frozen_headers="yes" frozen
   _rowheaders="2" pages_fitwidth="1"
    pages_fitheight="200" absolute_col
    umn
    _width="8,8,8,22,5,60,10,8,8,8" row
    _repeat="1"
    fudge_factor="0.5"
    autofilter="Yes"
    sheet_name="Controlled term.
    Violations"
   orientation="landscape" formu
   las="no"
   autofit_height="Yes"
   center_vertical="Yes"
   center_horizontal="Yes" gridlines="Yes");
 proc report nowd data=chk_all1;
```



Libname	SAS Variable	Controlled term	Incorrect value	Exten-sible?	Full comment	Rule ID (OPEN CDISC	Type	Severity
XXX	AESEV	AESEV	UNKNOWN	N	Value UNKNOWN of variable AESEV not found in codelist AESEV. Key values: STUDYID= C87088, USUBJID= C87085-041-0282, AESEQ= 1	CT0002	Warning	Medium
XXX	AESEV	AESEV	UNKNOWN	N	Value UNKNOWN of variable AESEV not found in codelist AESEV. Key values: STUDYID= C87088, USUBJID= C87085-051-0353, AESEQ= 6	CT0002	Warning	Medium
XXX	AESEV	AESEV	UNKNOWN	N	Value UNKNOWN of variable AESEV not found in codelist AESEV. Key values: STUDYID= C87088, USUBJID= C87085-051-0435, AESEQ= 5	CT0002	Warning	Medium
XXX	AESEV	AESEV	UNKNOWN	N	Value UNKNOWN of variable AESEV not found in codelist AESEV. Key values: STUDYID= C87088, USUBJID= C87085-056-0356, AESEQ= 6	CT0002	Warning	Medium
XXX	FASTAT	ND	NA	Y	Value NA of variable FASTAT not found in codelist ND. Key values: STUDYID= C87088, USUBJID= C87085-045-0317, FASEQ= 101	CT0076	Warning	Medium
					Value NA of variable FASTAT not found in codelist ND. Key values:			

Figure 8 Excerpt from a report of violations



Controlled term	Extensible:	Long description	Possible values	Value synonyms	CT (non-grouped)
ACN	No	Action Taken with Study Treatment	DOSE INCREASED		ACN
			DOSE NOT CHANGED		ACN
			DOSE REDUCED		ACN
			DRUG INTERRUPTED		ACN
			DRUG WITHDRAWN		ACN
			NOT APPLICABLE	NA	ACN
			UNKNOWN	U; Unknown	ACN
AESEV	No	Severity/Intensity Scale for Adverse Events	MILD	Grade 1; 1	AESEV
			MODERATE	Grade 2; 2	AESEV
			SEVERE	Grade 3; 3	AESEV
ND	No	Not Done	NOT DONE		ND
OUT	No	Outcome of Event	FATAL	Grade 5; 5	OUT
			NOT RECOVERED/NOT RESOLVED		OUT
			RECOVERED/RESOLVED		OUT
			RECOVERED/RESOLVED WITH SEQUELAE		OUT
			RECOVERING/RESOLVING		OUT
			UNKNOWN	U; Unknown	OUT

Figure 9 Reported reference lists for variables that violated CT

```
    label source ="Libname" ...;
    columns source variable codelist
    value extensib comment domain type
    severity ruleid;
  run;
ods tagsets.ExcelXP
    options (... absolute_column_
    width="8,5,30,30,30,10"
    ... sheet_name="Controlled term.
Overview");
  proc report nowd data=cntlcode (keep='"
name
  extensib
  longname list synonyms name1);
     label name="Controlled term" ...;
     define name / group;
     define extensib / group;
     define longname / group;
  run;
ods tagsets.ExcelXP close;
ods listing;
```

Output is stored in two separate worksheets in the same workbook. The first lists all CT violations (Fig. 8).

The second worksheet displays the codelists that were violated against (Fig. 9).

Conclusion

The use of CT is an integral part of the SDTM domain dataset creation process. The macro presented here is tailor-made for checking compliance with the used CT at the time SDTM domain datasets are created with SAS.

The implemented checks can be easily extended by modifying the metadata input datasets. Furthermore, users have full control over the range of SDTM domain data that is checked: a single SDTM domain dataset or all SDTM Domains for all studies in a particular project.

The principle shown here could also be applied to ADaM datasets.

References

1 Available from: http://meta-x.com/cdisc/doc/SDTM%20Implementation%20Guide%20V3.1.2.pdf [cited 2011 Jul 26].
2 Available from: http://evs.nci.nih.gov/ftp1/CDISC/SDTM/SDTM%20Terminology.xls [cited 2011 Jul 26].
3 Available from: http://www.opencdisc.org/projects/validator/cdisc-sdtm-3.1.2-validation-rules [cited 2011 Jul 26].
4 Available from: http://www.fda.gov/downloads/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/UCM254113.pdf [cited 2011 Jul 26].
5 Available from: http://datasavantconsulting.com/roland/macrotips.html [cited 2011 Jul 26].
6 Gregory M. Techniques for writing robust SAS macros. Pharm Programm 2009;**2**(1):5–13.
7 Available from: www.opencdisc.org/ [cited 2011 Jul 26].
8 Available from: http://support.sas.com/rnd/base/ods/odsmarkup/excelxp_demo.html [cited 2011 Jul 26].