

## Paper PO-11

**Validating Controlled Terminology in SDTM Domains**

John R. Gerlach, Independent Contractor, Hamilton, NJ

**ABSTRACT**

Creating and validating SDTM domain data sets have become rather perfunctory tasks involving regulatory submissions. Unfortunately, both tasks (creating and validating), especially the latter, often lack a comprehensive method concerning Controlled Terminology, which is an important extension of the SDTM standard. The lack of using Controlled Terminology or the absence of validating appropriate values undermines the content and compliance of SDTM domains, and thereby jeopardizes the submission. Thus, a validation tool should include a mechanism that ensures the proper use of Controlled Terminology, which affects almost all SDTM domains, including the SUPPQUAL and RELREC domains. This paper discusses an efficient and dynamic SAS® solution that validates Controlled Terminology in SDTM domains.

**INTRODUCTION**

There are different types of validation checks that help ensure the integrity of a SDTM domain, such as: adherence to SDTM metadata, consistency checks, and format compliance, to name a few. For example, SDTM domains must comply with respective CDISC metadata. Also, the value of the variable DOMAIN must be consistent and date variables must follow the ISO 8601 standard (e.g. yyyyymmddThhmmss). More robust validation applications will check for value limits, such as a start date cannot occur after its respective end date, and even perform referential integrity checks, such that a subject who suffered an adverse event should exist in the Demographic (DM) domain.

An important extension of SDTM domains concerns controlled terminology, which represents a discrete set of values for a given variable. These sets of values may be sponsor-defined or originate from an external published source, similar to MedDRA and WHO dictionaries. There is a growing collection of standard code lists that can be applied when creating SDTM domains and utilized when validating them. In fact, currently, there are over 65 code lists containing over four thousand discrete values.

Notice the difference between the validation of controlled terminology and other types of validation checks. With controlled terminology, the variable must contain a value from a respective code list; otherwise, the domain could be deemed as non-compliant to the standard. It's that simple. The issue is not about metadata or referential integrity or reasonable values, such as a Reference Start Date preceding an adverse event. It's simply about a collection of discrete values, perhaps even sponsor-defined, that are allowed for a given variable. For example, the variable DOMAIN must contain a value that is found in a code list; however, it must also be consistent within a domain. Thus, it becomes obvious that controlled terminology pertains to nominal responses, such as: a lab test, the route of administration of a drug, or the abbreviation of a country where the subject resides.

**THE CODELIST FILE**

The Open CDISC application uses a text file that contains header records identifying each code list, along with its records representing the collection of appropriate values. For example, below are several values from the code lists C67154 and C65047, lab tests and lab test codes, respectively. Not surprisingly, there are 581 codes found in each code list (See Appendix A).

The proposed SAS solution uses a similar file, albeit discarding header records and most of the data, which is deemed superfluous information. Why? Well, consider what is needed to validate the LBTESTCD variable in the LB domain. It should contain values such as "HDL" and "LDL." Similarly, the variable LBTEST should contain values such as "HDL Cholesterol" and "LDL Cholesterol." Now, imagine if there were a format catalog (dynamically generated) representing all code lists containing those appropriate, respective values. In that case, it would be efficient to utilize only the code list identifier (e.g. C65047) and its discrete values (e.g. HDL).

**Code List File**


---

CODELIST	CODE	VALUE	VALUE_DESC
C67154	C61041	HDL Cholesterol	HDL Cholesterol
	C61042	LDL Cholesterol	LDL Cholesterol
	C80187	HDL-Cholesterol Subclass 2	HDL-Cholesterol Subclass 2

	C80188	HDL-Cholesterol Subclass 3	HDL-Cholesterol Subclass 3
C65047	C61041	HDL	HDL Cholesterol
	C61042	LDL	LDL Cholesterol
	C80187	HDL2	HDL-Cholesterol Subclass 2
	C80188	HDL3	HDL-Cholesterol Subclass 3
C66786	C17653	VGB	VIRGIN ISLANDS, BRITISH
	C17654	MKD	REPUBLIC OF MACEDONIA
	C17668	CZE	CZECH REPUBLIC
	C17669	SVK	SLOVAKIA
	C17704	TKL	TOKELAU
	C17733	PLW	PALAU
	C17739	ASM	AMERICAN SAMOA
	C17740	WSM	SAMOA
		:	:
	C64377	SCG	SERBIA
	C64378	MNE	MONTENEGRO
	C83609	BLM	SAINT BARTHELEMY
	C83610	MAF	SAINT MARTIN, FRENCH

To emphasize the point, consider the variable COUNTRY found in the DM domain, which must contain pre-ordained 3-letter abbreviations, which is found in the code list C66786 above. Thus, imagine a SAS format called \$C66786F that contains such values that was created by a Control Input data set, as shown below. Notice the HLO variable that represents erroneous values.

**Control Input Data Set**

FMTNAME	TYPE	LABEL	START	HLO
C66786F	C			O
		Y	ABW	
		Y	AFG	
		Y	AGO	
		Y	AIA	
		Y	ALA	
		Y	ALB	

In fact, the SAS solution generates a single, large Control Input data set that creates the format catalog representing all the code lists. Consequently, the process of validating controlled terminology becomes a matter of using the appropriate format for a given validation check, which requires metadata.

**THE METADA7TA**

The The proposed SAS solution for validating controlled terminology modeled after Open CDISC, an open source, free application that ensures compliance to the CDISC standard. In fact, the following metadata for the solution emulates the conventions found in Open CDISC, which includes:

- RULEID Validation Rule Identifier
- CLIST Code List Identifier
- DOMAIN Domain or Type of Domain (e.g. Events)
- NAME Domain Variable
- TYPE Type of Check: Warning, Error
- SEVERITY Low, Medium, High
- TITLE Validation Description Used for Title

The following Data step creates a data set that contains all the relevant metadata needed for the application, which represent over 70 validation checks. Notice that the variable RULEID is imputed using a simple assignment statement. Also, the variable TITLE is slightly modified to include single quotes, which becomes part of the title in the report.

```

data ctmeta;
  length clist $6 domain $15 name $8 type severity $1 title $60;
  infile cards missover;
  input clist domain name type severity title &$;
  ruleid = 'CT' || put(_n_,z4.);
  title = "" || trim(title) || "";
cards;
C66767 Events          ACN          W M      Action Taken with Study Treatment
C66769 AE             AESEV         W M      Severity / Intensity Scale for ...
C66780 TS             TSVAL         W M      Age Span
C66781 DM             AGEU          E H      Age Unit
C66781 TS             TSVAL         W M      Age Unit
C66786 DM             COUNTRY       W M      Country
C78731 DA             DATEST        W M      Drug Accountability Test Name
C78732 DA             DATESTCD      W M      Drug Accountability Test Code
C66734 All            DOMAIN        W M      Domain Abbreviation
C74558 DS             DSCAT         W M      Category for Disposition Event
C71151 EG             EGMETHOD    W M      ECG Test Method
C71150 EG             EGSTRESC     W M      ECG Result
C71152 EG             EGTEST        W M      ECG Test Name
C71153 EG             EGTESTCD     W M      ECG Test Code
C66790 DM             ETHNIC        W M      Ethnic Group
C78735 Findings        EVAL          W M      Evaluator
C78735 SUPPQUAL        QEVAL         W M      Evaluator
C71113 Interventions  DOSFRQ        W M      Frequency
C66726 Interventions  DOSFRM        W M      Pharmaceutical Dosage Form
;
                                < More MetaData >

```

Half the metadata is used to define and execute the validation checks, obviously. However, all of it is used to formulate the title in the reporting component of the application. Consider the following titles in the validation report on the EG domain concerning the Control Terminology check CT0013, which denotes 'ECG Test Name' Not Found in Controlled Terminology Codelist C71152. The title includes even the Type and Severity of the validation check, as follows.

### Report Titles

---

```

                                CDISC SDTM Validation of EG Domain
CT0013: 'ECG Test Name' NOT Found in Controlled Terminology Codelist C71152
      ( Type: Warning / Severity: Medium )

```

---

Most of the validation checks concern a unique variable that is found in only one domain, such as the variable DSCAT (Disposition Category) found in the DS domain. However, there are other checks that apply to more than one domain, such as the –BLFL (Baseline Flag) variable that is found in several Findings domains, including: EG (Electrocardiogram), LB (Laboratory Tests), PE (Physical Examinations), QS (Questionnaires), and VS (Vital Signs). Notice also the validation check on the variable DOMAIN, which is found in all domains.

Keep in mind that this application pertains to controlled terminology, not CDISC metadata (e.g. labels) or other relevant validation checks. Finally, to facilitate this discussion, it is assumed that all controlled terminology is based on actual code lists that have not been modified according to accommodate sponsor-specific terminology. summarizes your paper and ties together any loose ends. You can use the conclusion to make any final points such as recommendations predictions, or judgments.

### SUMMARY REPORTS

A validation application requires a good reporting mechanism to convey useful information for resolving issues. The following macro %genrep offers a concise method of reporting errors via a frequency distribution. It contains two parameters denoting the domain and variable being validated. Initially, the SQL step determines whether there are any errors (after performing the validation step, of course). If there are errors, then the FREQ procedure generates an output data set containing a frequency distribution of those errors, which is used by the PRINT procedure to produce the report.

```

%macro genrep(dom, nam);
  proc sql noprint;
    select count(*) into :errors
    from errors;

```

```
quit;
%if &errors.
  %then %do;
    proc freq data=errors;
      tables &nam. / noprint out=rep missing;
    run;
    proc print data=rep noobs;
      var &nam. count percent;
      format &nam. $30. count comma8. percent 5.2;
      sum count percent;
      title1 "CDISC SDTM Validation of &dom. Domain";
      title2 "&ruleid: &title. NOT Found in Controlled Terminology Codelist &clist.";
      title3 "( Type: &type. / Severity: &severity. )";
    run;
    proc datasets library=work nolist;
      delete errors;
    quit;
  %end;
%mend genrep;
```

The following report clearly shows a problem with the EGTEST variable in the EG domain, specifically, there are instances of the 'ECG Test Name' that are not found in the pre-ordained controlled terminology, as found in the code list C71152. In fact, this particular error represents a Warning with Medium severity. The erroneous values are listed along with their frequency of occurrence, thereby indicating the gravity of the problem.

According to the following report, the variable AEBODSYS in the AE domain indicates errors. However, upon inspection, the values seem appropriate, at least from a clinical perspective. Nonetheless, with respect to the CDISC standard that employs code lists, there's a problem with the data. Obviously, it would be nice to know how the values differ from the code list.

---

CT0037: 'CDISC System Organ Class' NOT Found in Controlled Terminology Codelist C66783  
( Type: Warning / Severity: Medium )

AEBODSYS	COUNT	PERCENT
< Blank >	2	20.00
Cardiac disorders	2	20.00
Gastrointestinal disorders	1	10.00
Infections and infestations	1	10.00
Musculoskeletal and connective	1	10.00
Nervous system disorders	1	10.00
Renal and urinary disorders	1	10.00
Vascular disorders	1	10.00
	=====	=====
	10	100.0

---

Recall that the code lists have been stored in a SAS format catalog whose character formats are named using the code list identifier ending with the letter F, since user-defined formats cannot end with a number. In fact, the report title contains the code list identifier. Thus, a simple macro can be used to generate a listing of those values found in a code list, as follows.

```
%macro view_clist(cl);
  proc format library=work cntlout=&cl.;
    select &cl.F;

  run;
  proc print data=&cl.;
    var start;
    title1 "Code List %upcase(&cl.)";
  run;
%mend view_clist;

%view_clist(c66783);
```

The macro generated the following listing showing all the values found in the Code List C66783. It becomes obvious that the difference between the apparent errors and the code list is case sensitivity. Perhaps, the variable AEBODSYS should be revised so that its values contain text in uppercase only.

Code List C66783

```

1  "CONGENITAL, FAMILIAL AND GENETIC DISORDERS"
2  "INJURY, POISONING AND PROCEDURAL COMPLICATIONS"
3  "NEOPLASMS BENIGN, MALIGNANT AND UNSPECIFIED (INCL
4  "PREGNANCY, PUERPERIUM AND PERINATAL CONDITIONS"
5  "RESPIRATORY, THORACIC AND MEDIASTINAL DISORDERS"
6  BLOOD AND LYMPHATIC SYSTEM DISORDERS
7  CARDIAC DISORDERS
8  EAR AND LABYRINTH DISORDERS
9  ENDOCRINE DISORDERS
10 EYE DISORDERS
11 GASTROINTESTINAL DISORDERS
12 GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIO
13 HEPATOBILIARY DISORDERS
14 IMMUNE SYSTEM DISORDERS
15 INFECTIONS AND INFESTATIONS
16 INVESTIGATIONS
17 METABOLISM AND NUTRITION DISORDERS
18 MUSCULOSKELETAL AND CONNECTIVE TISSUE DISORDERS
19 NERVOUS SYSTEM DISORDERS
20 PSYCHIATRIC DISORDERS
21 RENAL AND URINARY DISORDERS
22 REPRODUCTIVE SYSTEM AND BREAST DISORDERS
23 SKIN AND SUBCUTANEOUS TISSUE DISORDERS
24 SOCIAL CIRCUMSTANCES
25 SURGICAL AND MEDICAL PROCEDURES
26 VASCULAR DISORDERS
27 **OTHER**

```

Of course, the errors might indicate a more serious matter. For example, in another report (not shown) the value "Oral" is found in the variable DOSFRM, which is not found in the Code List C66726. In fact, "Oral" denotes a route of medication, not a dose form, such as: capsule, douche, or tablet. In another case, a value of a given variable might be simply a truncated version of the code list value, such as the vital sign "Pulse" instead of "Pulse Rate."

Consider another possibility. The following report lists errors that occur in every observation in the CM domain. Upon inspection of the Code List C71113, which contains over 50 unique values, you learn that the variable CMDOSFR (Dose Frequency) uses a different coding convention (descriptive text); whereas, the code list contains abbreviations, such as PRN (As Needed) and BID (Twice Daily). Mystery solved!

```

CDISC SDTM Validation of CM Domain
CT0018: 'Frequency' NOT Found in Controlled Terminology Codelist C71113
( Type: Warning / Severity: Medium )

```

CMDOSFR	COUNT	PERCENT
	143	2.97
AS NEEDED	324	6.73
FOUR TIMES A DAY	52	1.08
ONCE A DAY	3,453	71.73
ONCE A WEEK	24	0.50
ONCE EVERY MONTH	1	0.02
THREE TIMES A DAY	193	4.01
TWICE A DAY	624	12.96
	=====	=====
	4,814	100.0

In all cases, the validation of controlled terminology in a CDISC domain is little more than finding out whether the values assigned to a variable are defined in a respective code list, then assessing the discrepancies, if any. Whether the problem is a matter of case-sensitivity, using different conventions, or outright bogus values, the first step is to compare the values stored in a variable to its respective code list. It is left to the reader to study the following reports that are intended to illustrate the power of this application.

**CDISC SDTM Validation of VS Domain**

CT0031: 'Route of Administration' NOT Found in Controlled Terminology Codelist C66729  
( Type: Warning / Severity: Medium )

VSPOS	COUNT	PERCENT
	3,242	64.84
SEMI-RECUMBENT	4	0.08
SITTING	1,744	34.88
STANDING	7	0.14
SUPINE	3	0.06
	=====	=====
	5,000	100.0

**CDISC SDTM Validation of SC Domain**

CT0033: 'Subject Characteristic Code' NOT Found in Controlled Terminology Codelist C74559  
( Type: Warning / Severity: Medium )

SCTESTCD	COUNT	PERCENT
FULLBSLN	2,093	49.18
MEDSTAT	1,929	45.32
RACEOTH	234	5.50
	=====	=====
	4,256	100.0

**CDISC SDTM Validation of EG Domain**

CT0050: 'Unit' NOT Found in Controlled Terminology Codelist C71620  
( Type: Warning / Severity: Medium )

EGORRESU	COUNT	PERCENT
	35	35.35
beats/min	33	33.33
degrees	31	31.31
	=====	=====
	99	100.0

**CDISC SDTM Validation of LB Domain**

CT0050: 'Unit' NOT Found in Controlled Terminology Codelist C71620  
( Type: Warning / Severity: Medium )

LBORRESU	COUNT	PERCENT
	121	5.06
/HPF	78	3.26
/LPF	78	3.26
10[3]/mm[3]	841	35.20
10[6]/mm[3]	112	4.69
fL	112	4.69
gm/dL	464	19.42
mIU/mL	480	20.09
mg/dl	36	1.51
microIU/mL	47	1.97
mmol/l	20	0.84
	=====	=====
	2,389	100.0

**THE PROCESS**

After creating the format catalog and the metadata data set, a macro performs the actual validation process. The macro first determines the number of validation checks, which is accomplished easily by the SQL step below. Then, a %DO loop performs each validation check beginning with the acquisition of the metadata, which is accomplished using another SQL step, specifically the INTO operator of the SELECT clause. Notice how several of the variables use a format (e.g. \$severf.) to assign several of the macro variables.

```
proc sql noprint;
    select count(*) into :nct from ctmeta;
quit;
%do ctcheck = 1 %to &nct.;
    proc sql noprint;
        select ruleid, clist, domain, name, put(type,$typef.), put(severity,$severf.),
```

```

title into :ruleid, :clist, :domain, :name, :type, :severity, :title
from ctmeta
where ruleid eq 'CT' || put(input("&ctcheck.",best.),z4.);
quit;
%let clist = &clist.;
%let domain = &domain.;
%let title = &title.;
%let type = &type.;
%let severity = &severity.;

```

The macro variables are uniquely assigned for each validation check. The following table contains the values for validation check CT0022, which pertains to code values for laboratory tests, such as CREAT and PSA denoting Creatinine and Prostate-Specific Antigen, respectively.

MACRO VARIABLE	DESCRIPTION	EXAMPLE
RULEID	Validation Rule	CT0022
CLIST	Code List	C65047
DOMAIN	SDTM Domain	LB
NAME	Domain Variable	LBTESTCD
TYPE	Warning or Error Check	Warning
SEVERITY	Medium or High Severity	Medium
TITLE	Report Title	Laboratory Test Code

After defining the macro variables, it is necessary to discern the validation check, which determines how the subsequent SQL step will be formulated, since there are several possible scenarios, as follows:

- Unique variable (LBTESTCD)
- Variable suffix found in a domain class (\_\_ACN variable found in the Events class)
- Variable found in several domains (DOMAIN).

Recall the metadata. Because most of the validation checks concern a unique variable found in a specific domain, the following code considers that scenario and proceeds to create a macro variable NVAR, which must be valued either one or zero, since the variable either exists or it doesn't. Then, a Data step processes the domain, checking the values of the variable, by asking the simple question: "Does the value exist in the code list specific to that validation check?" If the value is not found in the code list, it outputs that observation to the ERRORS data set. For those cases when the value is null, the variable is assigned the text "< Blank >". The LENGTH statement is used to ensure that the variable can store the Blank message for those variables that store less than eight bytes, such as COUNTRY and 'Yes / No' variables, for example.

```

%if %length(&domain.) eq 2 or &domain. eq RELREC or &domain. eq SUPPQUAL
%then %do;
proc sql noprint;
select count(name) into :nvar
from dictionary.columns
where libname eq 'SDTM' and memname eq "&domain." and name eq "&name.";
quit;
%if &nvar. eq 1
%then %do;
data errors;
length &name. $200;
set sdtm.&domain.(keep=&name.);
if put(&name.,$&clist.f.) ne 'Y'
then do;
if &name. eq '' then &name. = '< Blank >';
output;
end;
run;
%genrep(&domain., &name.) ;;
%end;
%end;

```

Keep in mind that the data determines whether a validation check is applicable. Of course, validation checks concerning the DM domain will always be executed; whereas, those checks pertaining to the Trial Summary (TS) domain might be unsuitable since the data library lacks that domain. If there is an appropriate domain and there are errors, then the %genrep macro generates the report as explained earlier.

Those validation checks that concern more than one domain require an alternative method. First, it is necessary to identify those domains that apply, if any. The following code accomplishes this task using the SQL procedure again. Notice the LIKE operator that handles those suffix-named variables (e.g. \_\_ACN, \_\_ORRESU).

```
%else %do;
  proc sql;
    create table doms as
    select memname, name
      from dictionary.columns
     where libname eq 'SDTM' and name like "&name.";
  quit;
  proc sql noprint;
    select count(*) into :ndoms
      from doms;
  quit;
```

In the event that there are domains germane to the validation check, a Data \_NULL\_ step defines pair-wise macro variables denoting both the SDTM domain and respective variable. For the variable \_\_BLFL (Baseline Flag), there will be as many pairs as there are domains in the Findings class that contain such a variable; whereas, for the variable DOMAIN, there will be as many pairs as there are domains in the data library.

```
%if &ndoms. gt 0
  %then %do;
    data _null_;
      set doms end=eof;
      call symput('domain' || left(put(_n_,3.)), memname);
      call symput('name' || left(put(_n_,3.)), name);
      if eof
        then call symput('ndoms', left(put(_n_,best.)));
    run;
```

Once those Domain / Name pairs have been defined, a %DO loop iterates such that an identical Data step performs the validation check by asking the same simple question, "Does the value exist in the code list specific to that validation check?" If not, the observation is written to the ERRORS data set and a report is generated accordingly.

```
%do i = 1 %to &ndoms.;
  data errors;
    length &name. $200;
    set sdtm.&&domain&i..(keep=&&name&i..);
    if put(&&name&i..,%cclist.f.) ne 'Y'
      then do;
        if &&name&i.. eq ''
          then &&name&i.. = '< Blank >';
        output;
      end;
  run;
  %genrep(&&domain&i..,&&name&i..) ;;
%end;
%end;
%end;
```

## PROXY CODE LISTS

In CDISC, there are a handful of variables, such as AESER (Serious Adverse Event) whose response is either 'Yes' or 'No' and are populated as 'Y' or 'N' respectively. Also, there are variables that may contain either 'Yes' or a Null value, such as a Baseline flag (BLFL) variable, where a 'No' response is unnecessary to indicate a baseline reading.

Consider the collection of Record Qualifier variables that clarify the cause and outcome of a serious adverse event, which includes: AESCAN (Cancer Related), AESCONG (Congenital Defect), AESDISAB (Permanent Disability) AESDTH (Death), AESHOSP (Hospitalization), AESLIFE (Life Threatening), AESOD (Overdose), AESMIE (Other Medically Important Event), and AECONTRT (Concomitant Treatment). In fact, according to the CDISC Implementation Guide, if the AESER variable contains the value 'Y', then at least one of these variables will have a 'Y' response, such that the other variables may contain either 'N' or 'Null'.

Here's the rub. Unfortunately, there is no code list readily available to accommodate such variables. There is the Code List 66742 that supports the responses Yes, No, Unknown, and Non-Applicable, but it does not recognize the Null value as valid. Thus, the Code List 66742 can be used to validate the variables AESER and IORRES, for example, but it seems inadequate for the other aforementioned variables, such as AESCAN and AEHOSP. In fact, the Baseline Flag (--BLFL) and the Derived Flag (--DRVFL) variables share a similar problem, since they may

contain either Yes or a Null value only.

Since the application cannot use the Code List 66742 for variables that may contain Null values, it creates two formats: CTYNLF (Yes, Null values only) and CTYNNF (Yes, No, Null values only) to accommodate these variables. In fact, these formats are defined along with the \$TYPEF and \$SEVERF formats that are used to formulate the titles in the Error reports. The proxy code lists CTYNL and CTYNN are specified in the metadata just like any other code list to be applied accordingly.

```
proc format;
  value $typef      'E' = 'Error'
                  'W' = 'Warning';
  value $severf    'L' = 'Low'
                  'M' = 'Medium'
                  'H' = 'High';
  value $ctynnf    'Y' = 'Y'
                  'N' = 'Y'
                  ' ' = 'Y';
  value $ctynlf    other = 'O';
                  'Y' = 'Y'
                  ' ' = 'Y';
                  other = 'O';
run;
```

## DESIGN ISSUES

The proposed SAS solution offers an efficient method for validating SDTM variable that contain pre-defined values, which is only one aspect of determining the integrity of SDTM domains. For example, recall the variable AESER and its associated qualifier variables (e.g. AESCAN). This utility can determine whether the values of the several variables are appropriate, but what about their consistency? According to the CDISC standard, if there is a Serious Adverse Event, then at least one of the qualifier variables must contain the value 'Y'. Obviously, this validation check is outside the scope of the application. Besides limitations, several design issues are now explained.

**Design Issue #1: Discerning Applicable Validation Checks** – Compare the following SQL steps with respect to efficiency. Given a SAS data library containing a typical collection of SDTM domain, the second step takes over 7 times longer to process. Why? Simply because the WHERE clause requires that all domains must be considered; whereas, the first SQL step specifies that only one domain is processed. Since most of the validation checks involve a specific variable belonging to one domain, the value of the macro variable NVAR will be 1; otherwise, the validation check involves a variable (e.g. DOMAIN, \_\_ACN) found in multiple domains. The second SQL step could accommodate both scenarios; however, the processing time would be increased significantly.

```
proc sql noprint;
  select count(name) into :nvar
  from dictionary.columns
  where libname eq 'SDTM'
        and memname eq "&domain." and name eq "&name.";
quit;
```

NOTE: PROCEDURE SQL used (Total process time):  
**real time**                    **0.56 seconds**

```
proc sql noprint;
  select count(name) into :nvar
  from dictionary.columns
  where libname eq 'SDTM' and name eq "&name.";
quit;
```

NOTE: PROCEDURE SQL used (Total process time):  
**real time**                    **6.54 seconds**

**Design Issue #2: Producing Comprehensive Error Reports** – Recall the previous report for the variable RACE in the DM domain. Notice that the report shows only erroneous values along with their frequency. However, the report does not put the errors in context of the whole domain, that is, the correct values as well. In this case, the DM domain represents 357 subjects; thus, about 20 percent of the values are incorrect, indicating a more severe matter than realized otherwise.

CDISC SDTM Validation of DM Domain  
 CT0029: 'Race' NOT Found in Controlled Terminology Codelist C74457  
 ( Type: Warning / Severity: Medium )

RACE	COUNT	PERCENT	
< VALID >	230	80.06	←
< BLANK >	10	1.57	
MULTI-RACIAL	117	18.37	
	=====	=====	
	357	100.0	

A possible enhancement might include an additional line that shows the number of correct values as a category. Perhaps the first line might have the value "< VALID >" denoting correct values, along with its frequency, 230 instances or just over 80% of the data.

**Design Issue #3:** Discriminating Vital Sign Tests – The application validates the values in the VSSTRESC variable; however, it actually pertains only to the Vital Sign called "Frame Size" (i.e. VSTESTCD equals FRMSIZE) such that the valid values are Large, Medium, and Small. The application needs to perform this validation check for that Vital Sign only; otherwise, it generates error records for all other values, which are typically numeric in value, such as: pulse rate, systolic and diastolic blood pressure.

**CONCLUSION**

Although the Open CDISC application effectively validates variables whose values consist of controlled terminology, the proposed SAS solution offers an alternative method that allows greater flexibility, such as defining proxy code lists. Also, this utility affords a deeper understanding of this important aspect of producing valid SDTM domains. And, no doubt, existing code lists will grow and new code lists will be created.

**REFERENCES**

- CDISC <http://www.cdisc.org/>.
- Gerlach, John R.; "Formats As A Programming Tool." An invited paper at the 6<sup>th</sup> Annual Southeast SAS Users Group Conference, 1998.
- Kilhullen, Michael; "Implementing CDISC Data Models in the SAS Metadata Server."
- Moresino, Cecilia; "Controlled Terminology;" CDISC Italian User Group; November, 2007.
- Open CDISC <http://www.opencdisc.org/node>.

**CONTACT INFORMATION**

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

Name: John R. Gerlach  
 Enterprise: SAS / CDISC Analyst  
 City, State ZIP: Hamilton, NJ  
 Work Phone: 609-672-5034  
 E-mail: [jrgerlach@optonline.net](mailto:jrgerlach@optonline.net)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

APPENDIX A – CONTROLLED TERMINOLOGY CODE LISTS

Category	Description	#	%
C65047	Laboratory Test Code	581	14.1
C66726	Pharmaceutical Dosage Form	168	4.1
C66727	Completion/Reason for Non-Completion	16	0.4
C66728	Relation to Reference Period	7	0.2
C66729	Route of Administration	112	2.7
C66731	Sex	4	0.1
C66732	Sex of Participants	3	0.1
C66733	Size	3	0.1
C66734	Domain Abbreviation	45	1.1
C66735	Trial Blinding Schema	3	0.1
C66736	Trial Indication Type	5	0.1
C66737	Trial Phase	12	0.3
C66738	Trial Summary Parameter Test Code	24	0.6
C66739	Trial Type	8	0.2
C66741	Vital Signs Test Code	15	0.4
C66742	No Yes Response	4	0.1
C66767	Action Taken with Study Treatment	7	0.2
C66768	Outcome of Event	6	0.1
C66769	Severity/Intensity Scale for Adverse Events	3	0.1
C66770	Units for Vital Signs Results	14	0.3
C66780	Age Span	8	0.2
C66781	Age Unit	5	0.1
C66783	CDISC System Organ Class	26	0.6
C66784	Common Terminology Criteria for Adverse Events	5	0.1
C66785	Control Type	3	0.1
C66786	Country	246	6.0
C66787	Diagnosis Group	1	0.0
C66788	Dictionary Name	7	0.2
C66789	Not Done	1	0.0
C66790	Ethnic Group	4	0.1
C66797	Category for Inclusion/Exclusion	2	0.0
C67152	Trial Summary Parameter Test Name	24	0.6
C67153	Vital Signs Test Name	15	0.4
C67154	Laboratory Test Name	581	14.1
C71113	Frequency	50	1.2
C71148	Position	10	0.2
C71150	ECG Result	110	2.7
C71151	ECG Test Method	22	0.5
C71152	ECG Test Name	46	1.1
C71153	ECG Test Code	46	1.1
C71620	Unit	310	7.5
C74456	Anatomical Location	303	7.3
C74457	Race	5	0.1
C74558	Category for Disposition Event	3	0.1
C74559	Subject Characteristic Code	7	0.2
C74561	Skin Type	3	0.1
C76348	Marital Status	9	0.2
C76351	Skin Classification	6	0.1
C78731	Drug Accountability Test Name	2	0.0
C78732	Drug Accountability Test Code	2	0.0
C78733	Specimen Condition	8	0.2
C78734	Specimen Type	41	1.0
C78735	Evaluator	15	0.4
C78736	Reference Range Indicator	4	0.1
C78737	Relationship Type	2	0.0
C78738	Never/Current/Former Classification	3	0.1
C85491	Microorganism	868	21.0
C85492	Method	65	1.6
C85494	PK Parameter Units of Measure	208	5.0
C85495	Microbiology Susceptibility Testing Result Categor	7	0.2
C87162	Common Terminology Criteria for Adverse Events V4.	6	0.1
		=====	=====
		4,129	100.0