Paper 176-2012

A Standard SAS® Program for Corroborating OpenCDISC Error Messages

John R Gerlach, TAKE Solutions, Princeton, NJ USA

Ganesh Sankaran Thankgavel, TAKE Solutions, Princeton, NJ USA

ABSTRACT

The freeware application OpenCDISC does a thorough job of assessing the compliance of SDTM domains to the CDISC standard. However, the application generates error and warning messages that are often vague, even confusing. Thus, it is beneficial to corroborate the CDISC compliance issues using SAS. Moreover, realizing that a validation check generates similar messages across CDISC data libraries, the SAS code should be similar, as well. Thus, paper explains a comprehensive standard SAS program that contains concise and reusable code that facilitates the process of corroborating OpenCDISC (OC) reports.

INTRODUCTION

This paper is a continuation of another paper, "Resolving OpenCDISC Error Messages Using SAS" (Redner and Gerlach, 2010) which claims that OpenCDISC messages are often cryptic and, thus, proposes the idea of corroborating OC messages in order to resolve CDISC issues. However, this paper goes beyond the scope of discussing techniques for corroborating OpenCDISC messages by recognizing the similarities inherent in the validation process across CDISC data libraries and proposing a standard program.

Consider the validation error message SD0002 that states "Null value in variable marked as Required," which identifies any Required variable (e.g. STUDYID) that contains a null value. The *Issue Summary* section of an OC report indicates the frequency; whereas, the *Details* section shows the actual instances. Since the OC report is an Excel book, you can ascertain the scope of the problem better using the Filter (Data) option. Nonetheless, it often becomes expedient to corroborate the OC report in order to understand the issue and to identify the problem records. Moreover, recognizing that a validation check is an invariant in the process, the programming technique to corroborate the OC analysis should be similar. Extending this idea further, the corroborating process becomes a standard SAS program containing reusable code. Imagine having a standard SAS program that allows you to address any OC issue with minimal modification.

THE TEMPLATE PROGRAM

Because the process of corroborating OC issues requires focused attention that may require modification of code, the proposed standard SAS program is intended to be run inside Display Manager, not as a batch job, Also, consider how such a program might be used in a Net-Meeting that is, corroborating an OC message while explaining how to resolve the CDISC issue. The SAS program itself can be a powerful presentation tool.

The standard program consists of several preliminary components, specifically: defining the SDTM data library; assigning a main title for reports; generating a concise abstract of the data library; and defining several very useful formats. The $NULLF and NULLF formats allow you to produce frequency distributions showing whether variables are either Null or Not Null. The VISITF format converts the VISIT numeric variable into a dichotomous variable, that is, valid or not valid visit values based the appropriate range of values.

```
libname sdtm "< Location of CDISC Data Library >";

title1 "OpenCDISC Issues for Study ABC-001 -- &sysdate9.";

proc contents data=sdtm._all_ nods;
   title2 'Abstract of SDTM Data Library';
run;

proc format;
   value $nullf  ' ' = 'Null'
             other = 'Not Null';
   value nullf      . = 'Null'
             other = 'Not Null';
   value visitf 1-10 = 'Valid'
             other = 'Not Valid';
run;
```

For this discussion, we will focus only on ***SDnnnn*** messages that pertain to the Study Data, for which there are about 100 validation checks. Despite the numerous types of validation checks, there are only a handful of programming techniques that are used in the standard program, including:

- Produce a listing based on the WHERE statement
- Produce a frequency distribution – Either single or juxtaposed distributions
- Perform a SQL Correlative Sub-query
- Perform By-group processing using FIRST / LAST logic
- Utilize SQL Dictionary tables on the CDISC Data Library
- Utilize study-specific metadata.

Consider the next segment of the standard program that addresses OC messages SD0001 and SD0002. In order to corroborate the first message, the SQL procedure with Dictionary tables is used to identify any empty domains. The second message addresses the aforementioned issue concerning Required variables, in this case, the ARMCD and ARM variables found in the DM domain. Notice how the TITLE2 statement is used to generate a sub-title for each corroborating report. Thus, by using Display Manager, you can focus on a specific OC issue, such as submitting the highlighted code below. Finally, the remaining portion of this segment illustrates the general format of each OC issue, that is, a TITLE statement and some SAS code.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0001: No Records in Data Source';

   proc sql;
      select memname, nobs
         from dictionary.tables
         where libname eq 'SDTM' and nobs eq 0;
   quit;

/* ------------------------------------------------------------------------------------ */
   title2 'SD0002: Null Value in Variable Marked as Required';

   proc freq data=sdtm.dm;
      tables armcd * arm / list missing;
   run;

/* ------------------------------------------------------------------------------------ */
   title2 'SDnnnn: < Error Message >';

   < SAS Code >
```

Imagine another OC report that indicates Error SD0002 (Null value in variable marked as required); however, it pertains to several variables in the CO (Comments) domain. Simply include the appropriate code in that segment of the standard program, as shown below. Thus, the standard program evolves as it corroborates OC issues.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0002: Null Value in Variable Marked as Required';

   proc freq data=sdtm.dm;
      tables armcd * arm / list missing;
   run;

   proc freq data=sdtm.co;
      tables studyid * domain * coseq * coval / list missing;
      format studyid domain coval $nullf. coseq nullf.;
   run;
```

Now, assume that your experience with OpenCDISC introduces an error you have never experienced, such as Error SD0005, which pertains to the sequence variable not having unique values for a subject. In this case, the CE

(Clinical Events) domain has a serious flaw. Once again, we consider how to corroborate the OC report and to produce a plausible report to help resolve the issue. Using By-group processing, we can locate these observations easily, which can be reported to Data Management.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0005: Non Unique Value for --SEQ Variable';

   proc sort data=sdtm.ce out=ce;
      by usubjid ceseq;
   run;

   data sd0005;
      set ce;
         by usubjid ceseq;
      if not (first.ceseq and last.ceseq);
   run;

   proc print data=sd0005(obs=100);
      id usubjid;
   run;
```

In regards to the Findings domains, all subjects should have at least one baseline observation, except those subjects who failed screening (i.e. ARMCD = 'SCRNFAIL') or were not assigned to a treatment arm (i.e. ARMCD = 'NOTASSGN').   In this case, OC reports an issue with the LB (Laboratory) domain. First, the DM (Demography) domain is used to ascertain each subject's status with respect to treatment arms.   Then, the LB domain is processed using By-group processing in order to impute a Flag variable that indicates whether a subject had a baseline reading at all.   Finally, the DM and LB domains are merged to produce the reporting data set for the PRINT procedure (not shown) that fully corroborates the OC report.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0006: No Baseline Result in <Domain> for Subject';

   proc sort data=sdtm.dm out=dm;
      by usubjid;
   run;

   proc sort data=sdtm.lb(keep=usubjid lbseq lbblfl) out=lb;
      by usubjid lbseq;
   run;

   data lb01;
      retain lbflag;
      set lb;
         by usubjid;
      if first.usubjid  then lbflag = 0;
      if lbblfl eq 'Y'  then lbflag = 1;
      if last.usubjid and not lbflag
         then output;
   run;

   data lb02;
      merge dm(in=dm) lb(in=lb);
         by usubjid;
      if dm and lb;
   run;
```

Let's consider an unusual issue. The Trial Visits (TV) domain defines planned visits for a clinical study, which are denoted by an integer value. Thus, a visit number in another domain, whether planned or unplanned, should not be outside the range of those values found in the TV domain; that is, an unplanned visit should be a decimal number whose integer value is in proximity of a planned visit.   Moreover, the decimal component should not exceed three decimal places, such as visit numbers: 1.1, 2.13, and 3.123).

The EG (ECG Test Results) domain seems to have bogus Visit Numbers consisting of more than 3 decimal places. The following code employs the FREQ procedure along with the VISITF format that produces a bi-level distribution of the continuous numeric variable VISITNUM, which indicates that many of its values are not valid. However, this code does not resolve the issue! There's an interesting caveat with OpenCDISC concerning visit numbers consisting of very large integers. Apparently, the integer value is converted into E-notation; consequently, OC discerns that that these values exceed the 3-decimal limit when, in fact, the values are very large integers. Nonetheless, the values are neither defined by the TV domain nor reasonable.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0010: VISITNUM Should Not Be Formatted to More Than 3 Decimal Places';

   proc freq data=sdtm.eg(obs=5000);
      tables visitnum;
      format visitnum visf.;
   run;
```

OC message SD0012 identifies records where the onset study day of an observation does not occur prior to the end study day of the same observation, assuming that both the onset and end study days are not null. The PRINT procedure with an appropriate WHERE statement produces a suitable report for investigating this matter.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0012: Begin Day Must Be Less Than or Equal to End Day';

   proc print data=sdtm.cm;
      var usubjid cmseq cmstdy cmendy;
      where cmstdy gt cmendy and cmstdy is not null and cmendy is not null;
   run;
```

OpenCDISC has the dubious reputation of having vague, short messages. Consider OC messages SD0017 that states, "Invalid value for –TEST variable" and SD0018 that states, "Invalid value for –TESTCD variable." One might think that such messages indicates a problem with the value of the variables, for example, LBTEST or LBTESTCD. In that case, the code segment would be very similar to that which is appropriate for corroborating OC SD0040, "Inconsistent value for name of measurement, test, or examination." Upon further study of the OC internal documentation, it turns out that messages SD0017 and SD0018 pertain simply to the length of the variables –TEST and –TESTCD, forty and eight bytes, respectively. Moreover, the –TESTCD variable may not start with a number or contain special characters. It is left as an exercise for the reader to write the segment code to corroborate OC message SD0018. Hint: Use similar SQL query with ANYPUNCT and ANYDIGIT functions.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0017: Invalid Value for --TEST Variable';

   proc sql;
      select memname, name, length
         from dictionary.columns
         where libname eq 'SDTM' and name like '__TEST' and length(name) gt 40;
   quit;

/* ------------------------------------------------------------------------------------ */
   title2 'SD0018: Invalid Value for --TESTCD Variable';

   < Segment code >
```

OpenCDISC messages SD0026 and SD0029 pertain to missing units when a value is provided for original and standardized variables, respectively. The following code segment corroborates an issue with the Laboratory (LB) domain such that there are original response values (LBORRES is not null) for which there are no units (LBORRESU is null). The code includes the lab test code and name for further clarity. Why? Well, for this study, some of the lab tests reasonably do not have units for their respective response values, such as the lab test for specific gravity or the "Absence / Presence" of a yeast infection.

The messages SD0029 and SD0030 are the converse to messages SD0026 and SD0029; that is, there are missing values for which the units are provided.

```
/* ------------------------------------------------------------------------------------- */
   title2 'SD0026: Missing Units on Value';

   proc freq data=sdtm.lb;
      tables lbtestcd * lbtest * lborres * lborresu / list missing;
      format lborres $nullf.;
      where lborres is not null and lborresu is null;
   run;
```

Here's another example of the an OC message that can be misleading. Upon further study of this validation check in the context of the CDISC standard, it involves more than just the start and end dates; that is, it involves the reference period. The OC internal document for SD0031 states:

> "Identifies records that violate the condition [Start Date/Time of Observation or Start Relative to Reference Period is not null], limited to records where [End Date/Time of Observation or End Relative to Reference Period is not null]."

A bit convoluted. If the objective were to produce a detailed listing using the PRINT procedure, suffice it to say that much care would be needed when writing the WHERE statement. An alternative method might be to use the FREQ procedure, shown below.

```
/* ------------------------------------------------------------------------------------- */
   title2 'SD0031: Start Date Expected When End Date Provided';

   proc freq data=sdtm.cm;
      tables cmstdtc * cmstrf * cmendtc * cmenrf / list missing;
      format _character_ $nullf.;
   run;
```

Here's an example of why the OC Issues program is not run in batch mode. The following code segment was written for another study. However, the permissible variables VSTPT and VSTPTNUM are not found in the VS domain pertaining to this study; hence, this segment is not applicable. More importantly, the OC Issues program represents an evolving collection of code segments. In a previous study, it was necessary to corroborate an issue concerning time point variables.

```
/* ------------------------------------------------------------------------------------- */
   title2 'SD0032: TIMEPOINT (Character) Missing When TIMEPOINT Number Provided';

   proc freq data=sdtm.vs;
      tables vstpt * vstptnum / list missing;
   run;
```

As part of the validation process, OpenCDISC utilizes the Define-XML document to ensure the harmony between the CDISC data and the Define-XML document. In fact, there are numerous checks to ensure harmony. One check pertains to variables having code lists. OC message SD0037 fires when a variable contains a value which is not found in the code list as found in the Define-XML document.

Many sites have there own application for generating the Define-XML document. Typically, these applications utilize a collection of metadata data sets that determine the contents of the Define-XML document. These metadata data sets can be used along with a given domain to corroborate this OC issue. The following code segment uses the CODELISTS metadata data set and the AE domain to corroborate a problem with the variables AEGRPID and AEACN.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0037: Value for <Variable> Not Found in <List> User-Defined Code List';

   proc freq data=meta.codelists;
      tables codelist * codeitem / list missing;
   run cancel;

   proc freq data=sdtm.ae(obs=1000);
      tables aegrpid;
      tables aeacn;
   run;

   proc print data=meta.codelists;
      var codelist codeitem code decoded_value;
      where codeitem eq 'GRPID';
    * where codeitem eq 'ACN';
      format _all_ $15.;
   run;
```

There is a 1-1 relationship between Test Code and Test Name. The code segment below addresses this issue with respect to the QS (Questionnaire) and LB (Laboratory) domains. For the QS domain the FREQ procedure generates a data set that the following Data step processes using By-group processing in order to identify those test codes that have multiple test names. The same method is used for the LB domain focusing on a particular laboratory test category (i.e. LBCAT).

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0040: Inconsistent Value for Name of Measurement, Test, or Examination';

   proc freq data=sdtm.qs;
      tables qstestcd * qstest / list noprint out=qs;
   run;

   data sd0040;
      set qs;
         by qstestcd;
      if not (first.qstestcd and last.qstestcd);
   run;

   proc freq data=sdtm.lb;
      tables lbcat * lbtestcd * lbtest / noprint out=lbtests01;
      where lbcat eq 'CHEMISTRY';
   run;

   data lbtest02;
      set lbtests01;
         by lbtestcd;
      if not (first.lbtestcd and last.lbtestcd);
   run;
```

There should be a 1-1 relationship between the variables VISITNUM and VISIT. That is, for each value of VISITNUM, there should be only one value for VISIT; otherwise, OpenCDISC fires off Error SD0051. Since the TV domain is relatively small, the FREQ procedure generates a complete report putting the values of VISITNUM and VISIT in juxtaposition.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0051: Inconsistent Value for Visit Name';

   proc freq data=sdtm.tv;
      tables visitnum * visit / list;
   run;
```

The OC message SD0052 concerns the existence of a variable in the Define-XML document that is not found in the CDISC data library. Similar to corroborating OC message SD0037, it is necessary to identify all the variables in the DEFINEXML metadata data set that was used to create the Define-XML document and compare that to the SDTM data library. The following code segment does the job easily.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0052: Variable in Define-XML not in Data Set;

   proc sort data=meta.definexml(keep=domain name role mandatory) out=definexml;
      by domain name;
      where length(domain) eq 2;
   run;

   proc sql;
      create table domains as
      select memname as domain length=2, name, type, label
         from dictionary.columns
         where libname eq 'SDTM' and length(memname) eq 2
         order by memname, name;
   quit;

   data sd0052;
      merge domains(in=d1) define(in=d2);
         by domain name;
      if d1 and d2
         then source = 'Both  ';
         else if d1
            then source = 'Domain';
            else source = 'Define';
   run;
```

OpenCDISC identifies variables whose attributes do not comply to the CDISC standard. Usually this error occurs when the domain contains variables that are not defined by the standard, such as so-called Plus/Minus variables or variables that are intended for a SUPPQUAL domain. The following code investigates the CO domain, which contains the variable SUBJID, which is found in the DM domain, not the CO domain.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0055: SDTM / Data Set Variable Type Mismatch';

   proc contents data=sdtm.co;
   run;

   proc print data=meta.cdisc;
      var name type label;
      where domain eq 'CO';
   run;
```

Similar to OC message SD0051, SD0061 concerns the existence of a domain in the Define-XML document that is not found in the CDISC data library. Once again, it is necessary to use the DEFINE metadata data set that was used to create the Define-XML document and compare that to the SDTM data library. The following code segment does the job easily.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0061: Domain Referenced in Define-XML, But Data Set is Missing';

   proc sql;
      create table domains as
      select memname as domain
         from dictionary.tables
         where libname eq 'SDTM'
         order by memname;
   quit;

   proc sort data=meta.define(keep=domain) out=define nodupkey;
      by domain;
   run;
```

```
    data sd0061;
        merge domains(in=meta) define(in=define);
           by domain;
        if not (meta and define)
           then do;
              if meta
                 then source = 'Meta  ';
                 else if define
                    then source = 'Define';
              end;
           else source = 'Both  ';
     run;
```

Only those subjects found in the DM (Demography) domain are considered part of the study. Conversely, if the CO (Comments) domain contains subjects who are not found in the DM domain, OpenCDISC declares the subject invalid. Using a nested query in the SQL step below easily corroborates this issue. Assuming that the DM domain contains only randomized subjects, the CO domain includes several subjects who were not randomized.

```
  /* ---------------------------------------------------------------------------------- */
     title2 'SD0064: Invalid Subject';

     proc sql;
        create table sd0064 as
        select distinct usubjid
           from sdtm.co
           where usubjid not in(select usubjid from sdtm.dm);
     quit;
```

The Subject Visits (SV) domain contains all actual visits. Consequently, if the PE (Physical Examination) domain contains a subject visit that is not found in the SV domain, it is considered invalid. The code segment below produces two frequency distributions on the SV and PE domains for comparison. On occasion, it may be useful to produce a detailed report based on a specific subject.

```
  /* ---------------------------------------------------------------------------------- */
     title2 'SD0065: Invalid Subject Visit / Visit Number';

     proc freq data=sdtm.sv;
        tables visitnum * visit / list missing;
     run;

     proc freq data=sdtm.pe;
        tables visitnum * visit / list missing;
        format visitnum visit.;
     run;
```

OC message SD0067 pertains to Element Codes (ETCD) that are defined in the TE (Trial Elements) domain. The TE domain contains one record for each type of element in the Trial Arms (TA) domain. Although an trial element can appear more than once in the TA domain, it appears only once in the TE domain. The Subject Elements (SE) domain differs from the Trial domains in that it stores actual, rather than planned, elements for each subject. In this situation, the Subject Elements (SE) domain contains values for the variable ETCD that are not found in the TE domain, which poses a problem. The following code segment produces a listing of the TE domain and a frequency of the ETCD variable in the SE domain for comparison.

```
  /* ---------------------------------------------------------------------------------- */
     title2 'SD0067: Invalid ETCD';

     proc print data=sdtm.te;
        title3 'TE Domain';
     run;

     proc freq data=sdtm.se;
        tables etcd;
        title3 'SE Domain';
     run;
```

The following situation concerns Supplemental domains. Here again, it is important to understand the CDISC standard, specifically that SUPPQUAL domains store information that references a record in its respective Parent domain. It is imperative to understand that a record in a SUPPQUAL domain is defined as "one record per IDVAR, IDVARVAL, and QNAM." Typically, the IDVAR contains the Sequence variable and the IDVARVAL contains the value of the Sequence variable. The QNAM is simply the name of the information being stored.

In this example, the SUPPSE domain contains records for which there is no reference (Parent) record. To corroborate this problem, we sort the Parent (SE) domain and process the SUPPSE domain so that both data sets have the sequence variable SESEQ. Upon merging the data sets, we identify those observations for which there is a record in SUPPSE, but not in SE. The number of observations in the data set SD0078 should corroborate the OpenCDISC report.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0078: Referenced Record Not Found';

   proc sort data=sdtm.se out=se;
      by usubjid seseq;
   run;

   data suppse;
      set sdtm.suppse(keep=usubjid idvarval);
      seseq = input(idvarval,best.);
      drop idvarval;
   run;

   proc sort data=suppse;
      by usubjid seseq;
   run;

   data sd0078;
      merge se(in=se) suppse(in=suppse);
         by usubjid seseq;
      if suppse and not se
         then output;
   run;
```

According to OpenCDISC, an adverse event (AE) should not occur after a subject's latest disposition date. In some studies, this may be appropriate if there is a follow-up period; nonetheless, it is useful to identify those AE records. The code segment below identifies the latest disposition date for each subject, as found in the Disposition (DS) domain. Then, another SQL step joins the AE domain and the summarized DS domain such that it identifies those AE records whose dates occur after the latest disposition date.

OC messages SD0080, SD0081, and SD0082 are similar, that is, an observation occurring after a subject's latest disposition date; however, they differ with respect to CDISC Observation Class: Events, Findings, and Interventions, respectively. Consequently, the segment code would be similar, albeit processing class specific domains.

```
/* ------------------------------------------------------------------------------------ */
   title2 'SD0080: AE Start Date Must Be LE Latest Disposition Date';

   proc sql;
      create table ds as
      select usubjid, max(dsstdtc) as dsstdtc
         from sdtm.ds
         group by usubjid
         order by usubjid;
   quit;

   proc sql;
      create tables ae as
      select ae.usubjid, dsstdtc, aeseq, aestdtc, aesev, aeser, aedecod
         from sdtm.ae, ds
         where ae.usubjid eq ds.usubjid and aestdtc gt dsstdtc
         order by usubjid, aeseq;
   quit;

   options pageno=1;
```

```
     proc print data=ae;
        id usubjid dsstdtc;
        var aeseq aestdtc aesev aeser aedecod;
        by usubjid dsstdtc;
        format aedecod $30.;
    run;

 /* ---------------------------------------------------------------------------------- */
     title2 'SD0081: Observation Date Must Be LE Latest Disposition Date';

     < Similar to SD0080 >

 /* ---------------------------------------------------------------------------------- */
     title2 'SD0082: Exposure Date Must Be LE Latest Disposition Date';

     < Similar to SD0080 >
```

A simple PROC step below produces a comprehensive report that identifies those subjects who do not have a reference start date and includes their treatment assignments. The WHERE statement could be expanded in order to exclude screen failures or those subjects who were not assigned to a treatment group. Also, notice the abbreviated notation in the VAR statement.

```
  /* ---------------------------------------------------------------------------------- */
     title2 'SD0087: RFSTDTC Cannot Be Null For Randomized Subject';

     proc print data=sdtm.dm;
        var usubjid armcd arm rf:;
        where rfstdtc is null;
     run;
```

According to CDISC, the planned study day visit (VISITDY) should be null for unplanned visits, which can be discerned from the variable SVUPDES (Unplanned Visit Description). In this situation, the Subject Visits (SV) domain has instances where the VISITDY variable is not null for unplanned visits. The following code segment corroborates this issue.

```
  /* ---------------------------------------------------------------------------------- */
     title2 'SD1019: VISITDY should be null for unplanned visits';

     proc freq data=sdtm.sv;
        tables visitnum * vist * visitdy * svupdes /list missing;
        format visitdy nullf. svupdes $nullf.;
        where visitdy is not null and svupdes is not null;
     run;
```

CONCLUSION

It is important to corroborate an OpenCDISC report in order to understand the CDISC issue and to proceed to resolve it. The *OC Issues* Standard SAS program provides a powerful tool that can be used repeatedly to corroborate any OpenDISC analysis. The code segments in the standard program are ordered according to their OpenCDISC naming convention, from SD0001 to SD*nnnn*, which allows you to locate the code segment needed for corroboration that is submitted inside SAS Display Manager.

As the CDISC standard evolves, including new domains and variables, OpenCDISC will encompass more validation checks; consequently, the standard program will evolve, as well. Thus, the program is self-documenting and expandable. Finally, the standard program offers a challenging exercise in writing SAS code that would corroborate all OpenCDISC messages.

REFERENCES

Redner, Virginia R. & Gerlach, John R. 2010. “Resolving OpenCDISC Error Messages Using SAS” *Proceedings of the Pharmaceutical SAS Users Group 2010 Conference.*

http://www.lexjansen.com/cgi-bin/xsl_transform.php?x=psug11&s=pharmasug&c=pharmasug

ACKNOWLEDGMENTS

The authors wish to thank Ilango Ramanujam, VP at TAKE Solutions, for his support in writing this paper.

RECOMMENDED READING

- CDISC Study Data Tabulation Model Implementation Guide, Version 3.1.2. http://www.cdisc.org.
- OpenCDISC Application & Documentation. http://www.opencdisc.org.

CONTACT INFORMATION

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

Name:	John R. Gerlach	Ganesh Sankaran Thankgavel
Enterprise:	TAKE Solutions, Inc.	Take Solutions, Inc.
Address:	502 Carnegie Center, Suite 100	502 Carnegie Center, Suite 100
City, State ZIP:	Princeton, NJ	Princeton, NJ
Work Phone:	609-720-1002, x239	609-720-1002
Fax:	609-720-1003	609-720-1003
E-mail:	John.Gerlach@takesolutions.com	Ganesh.Sankaran@takesolutions.com
Web:	www.takesolutions.com	www.takesolutions.com

APPENDIX A – ABRIDGED TEMPLATE PROGRAM

```
/* ====================================================================================
   Program : OC_Issues.sas

   Study   : ABC001 - 003
   Date    : YYYY-MM-DD

   Purpose : Corroborate OpenCDISC messages.

   Notes   : Run inside SAS Display Manager.

   ==================================================================================== */

   Libname sdtm '< Location of CDISC Data Library >';

   title1 'OpenCDISC Issues';

   proc format;
      value $missf  ' ' = 'Null'
                  other = 'Not Null';
      value missf     . = 'Null'
                  other = 'Not Null';
      value visitf 1-10 = 'Valid'
                  other = 'Not Valid';
   run;

   proc contents data=sdtm._all_ nods;
      title2 'Abstract of SDTM Data Library';
   run;

/* ------------------------------------------------------------------------------------ */
   title2 'SD0001: No Records in Data Source';

   proc sql;
      select memname, nobs
         from dictionary.tables
         where libname eq 'SDTM' and nobs eq 0;
   quit;

/* ------------------------------------------------------------------------------------ */
    title2 'SD0002: Null Value in Variable Marked as Required';

    proc freq data=sdtm.dm;
       tables armcd * arm / list missing;
    run;

            :      :      :      :      :      :      :      :      :


/* ------------------------------------------------------------------------------------ */
    title2 'SD0031: Start Date Expected When End Date Provided';

    proc print data=sdtm.cm;
       var usubjid cmstdtc cmendtc cmenrf cmstrf;
       where cmstdtc is null and cmendtc is null and cmenrf is not null;
    run;


            :      :      :      :      :      :      :      :      :


/* ==================================================================================== */
```