PharmaSUG2011 – Paper CD07

Resolving OpenCDISC Error Messages Using SAS®

Virginia Redner, Merck & Company, Inc., Upper Gwynedd, PA
John R. Gerlach, SAS / CDISC Analyst; Hamilton, NJ

ABSTRACT

The Clinical Data Interchange Standards Consortium (CDISC) Study Data Tabulation Model (SDTM) defines the required standard for the submission of human clinical trial data to a regulatory agency. OpenCDISC Validator is a free, open-source tool that is embraced by many pharmaceutical and biotechnology companies and used to assist in validating clinical data in accordance with the ever-evolving CDISC standards. Although OpenCDISC does a thorough job of validating CDISC domains, the application produces error messages that can be difficult to understand, let alone resolve. Interpreting the output and determining the necessary course of action is not always straightforward. Certainly, creating and validating clinical data requires a multi-faceted approach that includes having knowledge of the CDISC standard, experience with Base SAS®, as well as an understanding of clinical data. OpenCDISC introduces another factor to that endeavor. This paper gives an overview of the OpenCDISC application and discusses examples of programming techniques for resolving data issues.

INTRODUCTION

In 2004 the Clinical Data Interchange Standards Consortium (CDISC) recommended the use of the SDTM standard for submitting clinical data to a regulatory agency. Since that time the pharmaceutical and biotechnology industries have worked tirelessly to implement this standard for the submission of clinical data and its related metadata in order to facilitate the review process of determining the safety and efficacy of a drug. The OpenCDISC community was established to build a framework for the implementation of the CDISC Standard. In fact, this community of professionals created the OpenCDISC Validator tool that performs numerous checks on clinical data to ensure compliance with the standard. This tool validates both collected data as well as its respective metadata. The latest version of the tool also validates analysis data in accordance with the CDISC Analysis Data Model (ADaM) guidelines. Future enhancements are underway to include data from earlier trial phases using the Standard for Exchange of Nonclinical Data (SEND) guidelines.

OpenCDISC is a fine validation tool, especially when considering the price – It's free. However, using the application poses a real challenge with respect to its concise error and warning messages. The messages afford little insight into resolving data issues. It's nice to know that a particular error occurred 14,316 times; however, it is more important to understand the error, where and why it happened. Is the problem systemic? For example, the Subject Visits domain seems to be a mess, but only because it contains non-randomized subjects who don't belong there. Least of all, a lot of time is spent trying to decipher messages some of which seem extraneous. In order to use OpenCDISC efficiently, it is necessary to realize the multi-disciplined nature of the validation process, which goes beyond the application, specifically: CDISC, SAS, and clinical data. Also, the user should understand how OpenCDISC functions with respect to validation checks and data/metadata issues.

Admittedly, CDISC requirements for standardization are extensive, and always evolving. The validation process involves different types of checks to ensure compliance, including the metadata describing the clinical data. In fact, the clinical data, stored as SAS transport (XPT) data sets, must match that which is specified in the Case Report Tabulation – Data Definition (Define-XML) document. Not surprisingly, OpenCDISC uses both transport data sets and the Define-XML to perform the validation. Besides metadata checks, the application also checks for appropriate controlled terminology values (e.g. F,M, or U for the variable SEX) and standard formats, such as using the ISO 8601 format for date / time values. Of course, OpenCDISC does not guarantee 100% compliance. However, it does a fine job detecting most data issues that would otherwise delay a submission. Since most companies are just beginning to implement CDISC standards, this application is a timely and important asset to conformance of the clinical trials data with the submission standards.

OPENCDISC – AN OVERVIEW

First of all, it is important to develop a proper perspective of the validation process, such as recognizing whether a validation check pertains to the actual data or to its metadata. The latter pertains obviously to issues concerning variable labels, lengths and data type; whereas, the data should be either valid or reasonable. Another important perspective is the scope of a validation check. Is it domain-specific or class-specific? Perhaps the check is

performed on all domains, just SUPPQUALs, or only the RELREC. Perhaps the validation check pertains to timing variables found only in a subset of SDTM domains.

OpenCDISC employs a simple naming convention for its validation rules. The first two characters identify the type of validation followed by a four-digit number. For example CT0006, the sixth validation check, pertains to the variable COUNTRY with respect to Controlled Terminology; that is, appropriate values denoting a proper abbreviation (e.g. USA). The four types of validation checks include:

CT	Controlled Terminology
SD	Study Data
DD	Data Definition
OD	Operation Data Model

OpenCDISC breaks down validation checks into the following categories:

Consistency	There are dependencies between certain variables. For example, units should be assigned to test results, albeit not always. Another example is the bijective mapping between such variables as VISITNUM and VISIT.
Controlled Terminology	Some variables require only specific values based on pre-defined code lists. For example, the aforementioned variable COUNTRY uses the Code List C66786 and the variable SEX uses the Code List C66731, which contain pre-ordained values.
Format Compliance	The values of certain variables must follow a specific format, such as the ISO 8601 standard used for dates, time, and periods of time.
Referential Integrity	There are dependencies across domains. An obvious example would be any record in a SUPPQUAL for which there must be a record in the respective parent domain.
Limits	The values of certain variables must be within specified limits, such as the variable AGE which must be greater than zero.

There are several other categories that are outside the scope of this paper. Also, this paper discusses only validation checks pertaining to Study Data (SD) and Controlled Terminology (CT). The reader is advised to consult documentation on the OpenCDISC application for further information.

Besides having categories of validation checks, OpenCDISC determines whether a data issue warrants a Warning or Error message. Also, juxtaposed with the Error Type, the application ascribes a severity level. Errors always have High severity; whereas Warnings have either Low or Medium severity. Errors must be corrected; however, Warnings should be corrected in order to assist with the submission, even though some warning may be acceptable depending on the study.

OpenCDISC generates output in either Excel or XML format, which includes a detailed description of each validation rule, as well as the following partial listings in Tables 1-3.

Table 1: Dataset Summary – Data issues found for each domain indicating the frequency of occurrence.

Processed Sources							
Name	Label	Class	Source	Records	Errors	Warnings	Infos
AE	Adverse Event	Events	ae.xpt	243	243	669	0
BS	Biospecimens	Findings	bs.xpt	220	0	454	0
CM	Concomitant Medication	Interventions	cm.xpt	460	334	1074	0
CO	Comment	Special-Purpose Domain	co.xpt	781	197	798	0
DM	Demographics	Special-Purpose Domain	dm.xpt	23	0	51	0
DS	Disposition	Events	ds.xpt	68	0	107	0

EG	ECG	Findings	eg.xpt	26904	0	116937	0
EX	Exposure	Interventions	ex.xpt	1580	48	15795	0

Table 2: Issue Summary – Listing errors and warnings per validation check.

Issue Summary		
Rule ID	Message	Found
Errors		
CT0020	Value for IECAT not found in IECAT controlled terminology codelist	35
CT0038	Value for CMSTRF not found in STENRF controlled terminology codelist	167
CT0039	Value for CMENRF not found in STENRF controlled terminology codelist	167
SD0002	Null value in variable marked as Required	34
Total		1215
Warnings		
CT0012	Value for EGSTRESC not found in EGSTRESC controlled terminology codelist	26904
CT0013	Value for EGTEST not found in EGTEST controlled terminology codelist	26904
CT0014	Value for EGTESTCD not found in EGTESTCD controlled terminology codelist	22254
CT0016	Value for COEVAL not found in EVAL controlled terminology codelist	1
Total		168029

Table 3: Details – Lists the actual messages and each occurrence of non-compliance.

Name	Record	Variable	Value	Rule ID	Message	Category	Type	Severity
SC	1	SCTESTCD	ANCJPN	CT0033	Value for SCTESTCD not found in SCCD controlled terminology codelist	Terminology	Warning	Medium
SC	2	SCTESTCD	FULLBSLN	CT0033	Value for SCTESTCD not found in SCCD controlled terminology codelist	Terminology	Warning	Medium
PY				SD0001	No records in data source	Presence	Warning	Medium
CO	241	COVAL	null	SD0002	Null value in variable marked as Required	Presence	Error	High

DATA ISSUES

Much of the output from OpenCDISC is fairly easy to interpret and indicates what needs to be fixed in the study data. For other errors and warnings, however, it can be very difficult to decipher the message and, more importantly, to determine what needs to be done in order to resolve the matter.

The following examples offer some techniques in corroborating and understanding data issues –

CTnnnn – Value for <variable> not found in <variable> Controlled Terminology Code List

There are over 70 validation checks pertaining to Controlled Terminology (CT) that have become an integral part of the CDISC standard. All CT errors boil down to the following statement – There are values in a given variable that are not allowed. Obviously, the variable should contain only those values found in an associated code list. Thus, it is a matter of comparing the values of a variable with its pre-defined code list. However, even that task can be tedious and time-consuming. To learn more about dealing with this particular aspect of the validation process, read the paper, "Validating Controlled Terminology in SDTM Domains" by Gerlach (See References).

Most variables that do not adhere to the respective code list generate merely a Warning message with Medium severity, except for several variables (e.g. AGEU, SEX) that generate an Error message with High severity. Also,

there are variables related to an adverse event that might contain a null value, but still be incorrect, when the AESER variable denoting a Serious Adverse Event contains the value 'Y'. In other words, at least one of the SAE related variables (e.g. AESHOSP) should contain the value 'Y' as well.

SD0002 – Null value in a variable marked as Required

This is a compliance issue. Required variables in SDTM must contain a reasonable value in order for the record to be meaningful. Percentage-wise there are not as many Required variables as Expected and Permissible variables. However, when a Required variable contains a null value, it is a de facto Error with High severity. Some Required variables (e.g. STUDYID, DOMAIN) store a constant value while others may have numerous unique values (e.g. USUBJID). Thus, to corroborate the number of instances, it is not always appropriate to use the FREQ procedure. Instead, consider applying a user-defined format that dichotomizes a discrete variable into 'Missing' and 'Not Missing' levels, as follows.

```
proc format;
  value $missf  ' ' = 'Missing'
            other = 'Not Missing';
  value missf  . = 'Missing'
          other = 'Not Missing';
run;

proc freq data=sdtm.lb;
  tables usubjid / missing;
  format usubjid $missf.;
run;
```

SD0002 - Null Value in Variable Marked as Required

USUBJID	Frequency	Percent
Missing	3	1.00
Not Missing	297	99.00

SD0003 – Invalid ISO 8601 value

Timing variables are found in most SDTM domains; therefore, this validation check can generate a lot of errors. However, the error is likely to be systemic; that is, perhaps the values do not conform to the ISO 8601 standard due to a poorly written assignment statement. Timing variables are text variables; consequently, such variables could contain regular text (e.g. "Yesterday"), instead of an actual date. But even valid dates might not be correct, such as the date value "15JAN2011" because it does not conform to the format yyyy-mm-ddThh:mm:ss.

Investigating data issues for timing variables can be a bit tricky for two reasons. A timing variable may employ multiple formats, such as yyy-mm-dd and yyyy-mm-ddThh:mm:ss. Also, the ISO 8601 standard allows partial dates and times, such as 2011-03 (March, 2011). For this discussion, let's assume that the values are complete or null, no partial dates. Consider the following code that validates the timing variables (LBDTC, LBENDTC) in the LB domain. First, it's important to process the timing variables only when they contain a value. The LENGTH function determines whether the value is a date only or a date-time value. Also, the INPUT functions employ the INFORMATs E8601DA and E8601DT to read ISO 8601 date and date time values, respectively. If either timing variable has a bogus value, the observation will become part of the report. The subsequent report shows a systemic problem with the assignment of the LBDTC variable. For example, the value 2009-09-09t15:20:00 should be 2009-09-09T15:20:00.

```
data rep;
  set lptetda.lb(keep=usubjid lbseq lbdtc lbendtc);
  if lbdtc ne ''
    then do;
```

```
    if length(lbdtc) gt 10
      then do; if input(lbdtc, ?? e8601dt.) eq . then error=1; end;
      else do;  if input(lbdtc, ?? e8601da.) eq . then error=1; end;
    end;
  if lbendtc ne ''
    then do;
    if length(lbendtc) gt 10
      then do; if input(lbendtc, ?? e8601dt.) eq . then error=1; end;
       else do; if input(lbendtc, ?? e8601da.) eq . then error=1; end;
    end;
  if error  then output;
  keep usubjid lbseq lbdtc lbendtc;
run;
```

SD0003 - Invalid ISO 8601 Value (Partial Listing)

LBDTC	LBENDTC
2009-09-09t15:20:00	
2010-04-02t08:37:00	
2009-10-09t10:03:00	
2009-09-09t15:20:00	
2009-05-13t09:32:00	

Error code SD1011 has the same message as SD0003. So, why are there two codes? Well, SD0003 pertains to actual dates and times; whereas, SD1011 pertains to time intervals (e.g. P2Y denoting a period of 2 years). It is left to the reader to address how to investigate SD1011 errors.

SD0007 – Inconsistent value for standard unit

Imagine having lab tests that do not have consistent units. How can that be? Well, perhaps different sites used different units when performing the lab tests, which should not be an issue for the standardized variable. Perhaps the records from a particular site failed to specify the units, thus it is null. In any case, there is a lack of consistency in the data. The following FREQ step will produce a listing of all the lab tests juxtaposed with their units. We're interested in finding lab tests having different units, not merely how often the error occurs.

```
proc freq data=sdtm.lb;
  tables lbtestcd * lbstresu / noprint out=tests;
run;
data tests2;
  set tests;
  by lbtestcd lbstresu;
  if not (first.lbtestcd and last.lbtestcd);
run;
```

SD0007 - Inconsistent Value for Standard Unit (Partial Listing)

LBTESTCD	LBSTRESU
BILI	
BILI	micromol/L
GLUC	
GLUC	mmol/L
PROT	
PROT	gm/L

SD0009 – AE is serious but no qualifier set to 'Y'

Whenever there is a Serious Adverse Event (SAE), the AESER variable contains the value 'Y' per its Code List C66742. However, if that be the case, then at least one of several SAE related variables must contain the value 'Y'. In order to corroborate the OpenCDISC report, you might consider using the FREQ procedure with the related SAE variables nested in the Tables statement juxtaposed with the AESER variable such that the report will show the related variables containing null values. Likewise, the WHERE statement can be used to process only those records representing a serious adverse event.

```
proc freq data=sdtm.ae;
  tables aescong * aesdisab * aesdth * aeshosp * aeslife
    * aecontrt * aescan * aesmie * aesod / list missing;
  where aeser eq 'Y';
run;
```

SD0009 - AE is Serious But No Qualifier Set to 'Y'

AESCONG	AESDISAB	AESDTH	AESHOSP	AESLIFE	AECONTRT	AESCAN	AESMIE	AESOD	Frequency	Percent
N	N	N	N	N	N	N	N	N	4	28.57
N	N	N	Y	N	Y	N	N	N	8	57.14
N	N	Y	N	N	N	N	N	N	2	14.29

SD0010 – VISITNUM should not be formatted to more than 3 decimal places

There are Scheduled and Unscheduled visits in a clinical study. Unscheduled visits are denoted by using a decimal value whose value indicates a scheduled visit. For example, the unscheduled visit 3.1 occurred just after scheduled visit 3. Even if the values of VISITNUM match with those found in the Subject Visits (SV) domain, they are still incorrect if the value exceed three decimal places. But, how do you identify those observations whose values are Real numbers (e.g. 3.141592) that erroneously represent an unscheduled visit? Often there are at least six ways to do anything in SAS, such as using the logarithmic function to discern the precision of a real number, which is left as an exercise to the reader.

```
data rep;
  set sdtm.<domain>;
  if int(visitnum) ne visitnum
    then do;
      * Discern precision of visit number;
      * Output observation if visit number exceeds three decimal places;
run;
```

SD0026 – Missing units on value

Open CDISC identifies records in the Findings domains where there is an original response but no units specified. There are instances, however, where there might be a response, such as a lab test having a response of 'MODERATE', 'TRACE', or 'BLQ' (Below Level of Quantification), for which there are no units. Also, let's not forget actual test (e.g. Specific Gravity) for which there are no units. Consider the following SAS code that processes the LB domain using the INPUT function to discern whether there is a numeric value for which there should be units assigned.

```
data rep;
  set sdtm.lb;
  if input(lborres, ?? best.) ne . and lborresu eq ' '
    then output;
run;
```

SD0037 – Value for [Variable] not found in [Codelist] user-defined codelist

This data issue indicates an error in the Define-XML document, specifically the code list associated with a particular variable does not contain an actual value found in the data. For example the variable LBCAT (Category for Lab Test) contains the value "CHEMISTRY" which does not exist in the code list found in the Define-XML document. Thus, the error is not with the LB domain, but the Define-XML document. Whatever application generated the Define-XML document, it failed to include a complete code list for the variable, hence the error.

SD0036 – Missing character result when original result provided

This is a compliance issue that involves the –ORRES variable, which stores the original response to a test. Whether this value is numeric or text, it needs to be copied or derived into the variable –STRESC; however, the variable –STRESN will remain null when the value is non-numeric. Using the $MISSF format with the FREQ procedure easily produces a corroborating report, which includes the test code for further granularity. The subsequent Data step identifies actual records that manifest this problem.

```
proc freq data=sdtm.lb;
  tables lbtestcd * lborres * lbstresc / list missing;
  format lborres lbstresc $missf.;
run;
data rep;
  set sdtm.lb;
  if lborres ne '' and lbstresc eq ''
    then output;
run;
```

SD0040 – Inconsistent value for name of measurement, test, or examination

There should be a 1-1 mapping between test codes (–TESTCD) and test names (–TEST). For example, lab test codes ALB, CREAT, and GLUC should have one lab test name: Albumin, Creatinine, and Glucose, respectively. In order to find those tests that have more than one code or vise versa, use the Freq procedure with a subsequent Data step, as follows.

```
proc freq data=sdtm.lb;
  tables lbtestcd * lbtest / out=tests;
run;

data rep;
  set tests;
    by lbtestcd;
  if not (first.lbtestcd and last.lbtestcd);
run;
```

SD0040 - Inconsistent Value for Name of Measurement, Test, or Examination

LBTESTCD	LBTEST	COUNT	PERCENT
MONOLE	Monocytes-Leukocytes	3	0.02735
MONOLE	Monocytes/Leukocytes	140	1.27621

SD0064 – Invalid subject

The study population is identified as those subjects found in the Demography (DM) domain. Thus, invalid subjects would be those identified in other domains who do not exist in the DM domain. Again, the objective is to corroborate the OpenCDISC report with a more suitable report that might show a systemic problem. An obvious solution would be to use a nested query in the SQL procedure, as follows.

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

If there are numerous occurrences of invalid subjects across multiple domains, this solution could be expanded into a macro solution using Dictionary tables along with the nested query.

SD0070 – No Exposure record found for subject

No exposure record for a randomized subject?! It's conceivable, but worth verifying. Well, obviously, the subject(s) will not be found in the Exposure (EX) domain. Yet, both the EX and DM domains are needed to identify those subjects. A nested query should do the job, as follows. Notice that the same programming was employed to address the prior data issue.

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

SD0078 – Referenced record not found

This error message concerns referential integrity. Every record in a SUPPQUAL data set must refer to a record in its respective parent domain. For example, a record in SUPPAE is supposed to supplement a record in the AE domain; otherwise, OpenCDISC considers it extraneous since there is no parent record. But, how do you pinpoint the problem? Well, what references a record in a SUPPQUAL to its parent domain? Typically, but not always, it's the –SEQ variable as found in the IDVAR and IDVARVAL variables. In this case, the values of USUBJID and IDVARVAL in the SUPPAE domain should point to a record in the AE domain having the same values of USUBJID and AESEQ. But, there's no such record. Why not? Well, perhaps the supplemental record indicates a subject not included in the study population or the wrong adverse event. If the supplemental record is extraneous, then perhaps the process that created the SUPPQUAL domain should be investigated. The following SQL step may be used to identify the orphan SUPPQUAL records.

```
proc sql number;
  select catx(' ', usubjid, idvarval) as suppkey
       from sdtm.suppae
       where calculated suppkey not in
            (select distinct catx(' ', usubjid, aeseq)
                 from sdtm.ae);
quit;
```

SD0048 - Referenced Record Not Found

Row	Suppkey (USUBJID IDVARVAL)
1	8669-037_000100006 2
2	8669-037_000100006 3

SD0080 – AE start date must be less than or equal to latest disposition date

This data issue affords an opportunity to use a correlated subquery in SQL, since the comparison is subject-specific. The SQL step identifies those subjects who have an AE start date that occurs after the last disposition date. Typically, the AE start date is incorrect (e.g. year) or the Disposition (DS) domain is incomplete.

```
proc sql;
  create table rep as
  select distinct usubjid
    from sdtm.ae
    where aestdtc gt (select max(dsdtc)
      from sdtm.ds
      where ae.usubjid eq ds.usubjid);
quit;
```

SD0090 – AESDTH='Y' expected when AEOUT='FATAL'

Obviously, the variables AESDTH (Results in Death) and AEOUT (Outcome of Adverse Event) are related. If the outcome of the adverse event is fatal, then the variable AESDTH should contain the value 'Y'. A simple frequency distribution of these variables should reveal everything about this pair of variables, as follows.

```
proc freq data=sdtm.ae;
  tables aesdth * aeout / list missing;
run;
```

SD0090 - AESDTH='Y' Expected When AEOUT='FATAL'

AESDTH	AEOUT	Frequency
N	NOT RECOVERED/NOT RESOLVED	72
N	RECOVERED/RESOLVED	156
N	RECOVERED/RESOLVED WITH SEQUELAE	3
N	RECOVERING/RESOLVING	6
N	UNKNOWN	4
Y	FATAL	2

SD1017 – Invalid VISITNUM

With respect to planned visits, the Visit Number must match entries in the Trial Visits domain. You might consider using the SQL procedure with a nested query, but let's consider an alternative technique. Typically, there are not

many planned visits; hence, why not create a format, then use it in a Data step? Although the data issue pertains to planned visits only, what about the unplanned visits? Let's expand the Boolean expression such that it checks planned visits only by making sure that the value is an integer, as follows.

```
data pvisitf;
  retain fmtname 'pvisitf'  type 'N' label 'Y';
  set sdtm.tv(keep=visitnum rename=visitnum=start) end=eof;
  output;
  if eof
    then do; start = .;  label = '';  hlo = 'O'; output; end;
run;
proc format cntlin=pvisitf;
run;
data rep;
  set sdtm.lb;
  if int(visitnum) eq visitnum and put(visitnum,pvisitf.) ne 'Y'
    then output;
run;
```

Now, what's wrong with the proposed solution? Recall that the TV domain represents "One record per Planned Visit per Arm." Therefore, it's possible that the values of VISITNUM might occur more than once, across different treatment arms, which would foul up the Control Input data set for the Format procedure. Thus, the process (format) may need to be treatment arm specific or, perhaps, using unique visit numbers might suffice. Again, it's important to keep in mind that this effort is multi-disciplined.

CONCLUSION

OpenCDISC does a fine job validating SDTM domains; however, the error messages can be difficult to interpret. In order to fully comprehend a data issue, it is often necessary to produce a corroborating report that pinpoints the troubled records or, better yet, identifies a systemic problem. Also, it is important to recognize that the task of validating CDISC domains is multi-disciplined: CDISC, SAS, and clinical data, as well as the OpenCDISC application itself. After a brief overview of the OpenCDISC application, methods for corroborating data issues were discussed, thereby affording a better understanding of the issues identified by the tool, as well as possible resolutions.

REFERENCES

Agostinelli, Robert; "Thanks CDISC – Free Utilities." Proceedings of the PharmaSUG Conference, 2010.

Gerlach, John R; "Validating Controlled Terminology in SDTM Domains." Proceedings of the CDISC Interchange Conference, 2011.

Loonan, Peter; Kottam, Sandeep; Tripuraneni, Sree K; "SAS and Open Source Tools for CDISC SDTM Compliance Checks for Regulatory Submissions." Proceedings of the NESUG Conference, 2010.

Open CDISC http://www.opencdisc.org/node.

Van Reusel, Peter; De Leeuw, Nico; "SDTM Validation – How Can We Do It Right?" Proceedings of the PhUSE Conference, 2009.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact either author at:

Ginger Redner
Merck & Co., Inc.
UG-1D88
351 N. Sumneytown Pike
North Wales, PA 19454-2505
267-305-7634
Virginia_Redner@merck.com

John R. Gerlach
SAS / CDISC Analyst
Hamilton, NJ
609-672-5034
JRGerlach@optonline.net