PharmaSUG China2015 - Paper 24

How to validate clinical data more efficiently with SAS® Clinical Standards Toolkit

Wei Feng, SAS R&D, Beijing, China

ABSTRACT

How do you ensure good quality of your clinical data? Currently there are many tools that can handle simple validation, but if we meet cross-standard validation or other complicated validation, the work becomes very complex. On the other hand, although you can develop validation tools by yourself, it's hard to maintain, you have to do a lot of work to modify existing codes when you need to customize your own validation.

This paper presents SAS Clinical Standards Toolkit (CST), a powerful and flexible toolkit which can help you validating your clinical data, as well as supporting other work in clinical research activities. CST validation assesses the compliance of data, and the metadata describing the data. It assesses the consistency of values in a specific column, between columns, across records in a specific data set, and across data sets. After the validation process, it shows accurate and clear results to you. In addition, it can customize the solutions easily to meet your needs.

CST is a separately orderable component that is available at no additional charge to currently licensed SAS customers. It's free as long as you have SAS Foundation. The latest version 1.7 is available on SAS 9.4.

INTRODUCTION

SAS Clinical Standards Toolkit (CST) is a framework based on Base SAS that is used for supporting Health and Life Sciences industry data model standards. It provides support of multiple CDISC standards, including SDTM (3.1.2, 3.1.3, and 3.2), CRT-DDS, Define-XML 2.0, Dataset-XML, ODM, ADaM 2.1, CDASH 1.1, SEND 3.0, and validating XML files against an XML schema file.

CST contains three parts in file system:

- Framework macros - <SAS Foundation directory>/cstframework/sasmacro
- Global standards library - <CST installation root directory>/cstGlobalLibrary
- Sample library - <CST installation root directory>/cstSampleLibrary

Framework macros are the infrastructure of CST framework, they include all the utility macros that used by global standards library and sample library.

Global standards library is the metadata repository of CST. By default, it contains the metadata for the framework module and the metadata for each data standard that is provided with CST. Figure 1 shows the architecture of it.

Sample library includes sample data and sample programs that can be used directly or modified as needed.

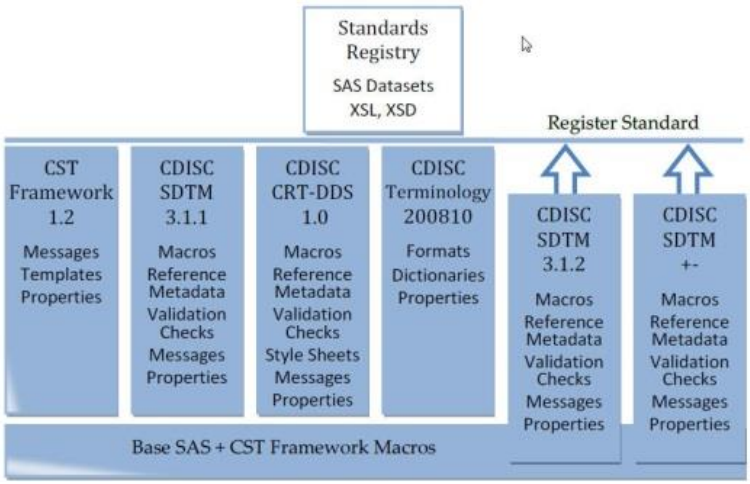


Figure 1. Architecture of CST components

VALIDATION PROCESS AND RESULTS

VALIDATION PROCESS

Validation process is one of the most important parts of CST. It provides a very easily used method - a SAS driver program to validate your clinical data. This driver program can be used directly or modified as needed.

The following steps in figure 2 show the execution flow in a typical SAS driver program to perform SDTM 3.1.3 validation process. This driver program can be found at the programs directory of sample library.

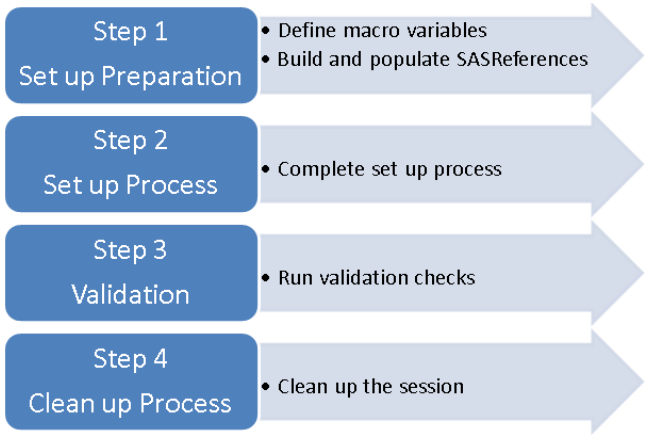


Figure 2. CST Validation Process

Define macro variables required by the validation process

This is the first stage of set up process. It prepares some useful macro variables. The sample codes are:

```
/* Define macro variables which used as substitution parameters later in the driver
program to reduce the number of code changes required. */
%let _cstStandard=CDISC-SDTM;
%let _cstStandardVersion=3.1.3
%let _cstCTPath=;
%let _cstCTMemname=;
%let _cstCTDescription=;

/* Initialize the minimum set of global macro variables used to run any CST
process. This includes the names of work data sets, default locations of files, and
metadata used to populate the process Results data set. */
%cst_setStandardProperties(_cstStandard=CST-FRAMEWORK,_cstSubType=initialize);

/* Choose the default controlled terminology that is associated with the
_cstStandard and _cstStandardVersion. */
%cst_getstandardsubtypes(_cstStandard=CDISC-
TERMINOLOGY,_cstOutputDS=work._cstStdSubTypes);
data _null_;
  set work._cstStdSubTypes (where=(standardversion="&_cstStandard" and
isstandarddefault='Y'));
  call symputx('_cstCTPath',path);
  call symputx('_cstCTMemname',memname);
  call symputx('_cstCTDescription',description);
run;

/* Set the studyrootpath and studyoutputpath. These are used to make the driver
programs portable across platforms and allow the code to be run with minimal
modification. These macro variables by default point to locations within the
cstSampleLibrary, set during install but modifiable thereafter. The
cstSampleLibrary is assumed to allow write operations by this driver module. */
%cstutil_setcstsroot;
```

```
data _null_;
  call symput('studyRootPath',cats("&_cstSRoot","/cdisc-sdtm-3.1.3-
&_cstVersion/sascstdemodata"));
  call symput('studyOutputPath',cats("&_cstSRoot","/cdisc-sdtm-3.1.3-
&_cstVersion/sascstdemodata"));
run;

/* (Optional) Set the workPath value. It provides the path to the Work directory
which is referenced within the sample study SASReferences data set path column. */
%let workPath=%sysfunc(pathname(work));
```

Build and populate the SASReferences data set

This is the second stage of set up process. It creates and populates SASReferences data set that is required for SDTM validation. The SASReferences data set defines the location and name of each input metadata source, input data source, and output file that is created by the validation process, including the Validation Control data set. The Validation Control data set contains the set of checks to include in the validation process. The sample codes are:

```
%let _cstSetupSrc=SASREFERENCES;

/* Create an empty SASReferences data set. */
%cst_createdsfromtemplate(_cstStandard=CST-FRAMEWORK,
_cstType=control,_cstSubType=reference, _cstOutputDS=work.sasreferences);

/* Populate the SASReferences data set. */
proc sql;
  insert into work.sasreferences
  values ("CST-FRAMEWORK" "1.2" "messages" "" "messages" "libref" "input" "dataset"
"N" "" "" 1 "" "")
  values ("CST-FRAMEWORK" "1.2" "template" "" "csttmplt" "libref" "input" "folder"
"N" "" "" 2 "" "")
  [etc.]
;
quit;
```

Call %cstutil_processsetup macro to complete set up process

```
%cstutil_processsetup();
```

This macro completes process setup. It ensures that all SAS librefs and filerefs are allocated; all system options, macro autocall paths, and format search paths are set; and all global macro variables that are required by the process have been appropriately initialized.

Call %sdtm_validate macro to run the validation checks

```
%sdtm_validate;
```

The sdtm_validate macro performs these tasks:

1. The macro looks up the Validation Control data set reference from SASReferences.
2. Metadata about the validation process, such as the standard/version, key files referenced, and process datetimes, is added to the process Results data set.
3. For each check in the Validation Control data set with a checkstatus > 0, this macro calls the check macro specified in the Validation Control codesource field. It passes all of the check metadata to the check macro.
4. After all of the checks are run, the results and metrics are saved to the file specified in SASReferences. Various SAS Work files are cleaned up if needed.

Clean up the session

This step is optional, and it is unnecessary with batch processing. The sample codes are:

```
%cstutil_cleanupcstsession(
  _cstClearCompiledMacros=0,
  _cstClearLibRefs=0,
  _cstResetSASAutos=0,
  _cstResetFmtSearch=0,
  _cstResetSASOptions=1,
  _cstDeleteFiles=1,
  _cstDeleteGlobalMacroVars=0);
```

VALIDATION RESULTS AND MATRICS

The primary outputs of each validation process are the Results data set and the Metrics data set. These data sets itemize and summarize the findings of the validation process. Display 1 and 2 shows SDTM 3.1.3 results data set.

In this results data set, column “checkid” shows which validation checks are run in this session; column “srcdata” shows which source data set the checks run on; column “resultseverity” and “message” show the result severity and related messages, the severity includes info, note, warning and error; column “actual” provides the actual column values that are in error; column “keyvalues” shows which records are in error; column “resultdetals” shows the basis or explanation for result in error. The “actual”, “keyvalues” and “resultdetails” will have value only when the severity is warning or error.

	resultid	checkid	result seq	seq no	srcdata	message	result severity	result flag	_cst_rc
1	CST0108		1	1	CST_SETPROPERT…	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cst-…	Info	0	0
2	CST0102		1	1	CST_CREATEDS	work.sasreferences was created as requested	Info	0	0
3	CST0200		1	1	CSTUTIL_PROCES…	Process setup is using this SASReferences: C:\Users\sasses\AppData\Local\Te…	Info	0	0
4	CST0200		1	1	CST_INSERTSTAN…	SASReferences data set was successfully validated	Info	0	0
5	CST0200		1	2	CSTUTIL_ALLOCAT…	SASReferences data set was successfully validated	Info	0	0
6	CST0108		1	1	CST_SETPROPERT…	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdi…	Info	0	0
7	CST0108		1	1	CST_SETPROPERT…	The properties were processed from the PATH c:/cstSampleLibrary/cdisc-sdtm-3…	Info	0	0
8	CST0200		1	1	SDTM_VALIDATE	PROCESS STANDARD: CDISC-SDTM	Info	0	0
9	CST0200		1	2	SDTM_VALIDATE	PROCESS STANDARDVERSION: 3.1.3	Info	0	0
10	CST0200		1	3	SDTM_VALIDATE	PROCESS DRIVER: SDTM_VALIDATE	Info	0	0
11	CST0200		1	4	SDTM_VALIDATE	PROCESS DATE: 2012-10-01T09:57:14	Info	0	0
12	CST0200		1	5	SDTM_VALIDATE	PROCESS TYPE: VALIDATION	Info	0	0
13	CST0200		1	6	SDTM_VALIDATE	PROCESS SASREFERENCES: C:\Users\sasses\AppData\Local\Temp\SAS Te…	Info	0	0
14	CST0200		1	7	SDTM_VALIDATE	PROCESS STUDYROOTPATH: c:/cstSampleLibrary/cdisc-sdtm-3.1.3-1.5/sascs…	Info	0	0
15	CST0200		1	8	SDTM_VALIDATE	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info	0	0
16	CST0200		1	9	SDTM_VALIDATE	PROCESS CSTVERSION: 1.5	Info	0	0
17	CST0200		1	10	SDTM_VALIDATE	PROCESS CONTROLLED TERMINOLOGY SOURCE: c:/cstGlobalLibrary/stand…	Info	0	0
18	CST0100	SDTM0101	1	1	SRCDATA.DM	No errors detected in SRCDATA.DM	Info	0	0
19	CST0100	SDTM0130	1	1	SRCDATA.DM	No errors detected in SRCDATA.DM	Info	0	0
20	CST0100	SDTM0210	1	1	SRCDATA.DM	No errors detected in SRCDATA.DM	Info	0	0
33	CST0100	SDTM0432	1	1	SRCDATA.AE	No errors detected in SRCDATA.AE	Info	0	0
34	SDTM0432	SDTM0432	1	2	SRCDATA.AE	MedDRA code present but corresponding MedDRA term missing, or vice versa	Warning	1	0
35	CST0100	SDTM0432	1	3	SRCDATA.AE	No errors detected in SRCDATA.AE	Info	0	0
36	CST0100	SDTM0432	1	4	SRCDATA.AE	No errors detected in SRCDATA.AE	Info	0	0
37	SDTM0432	SDTM0432	1	5	SRCDATA.AE	MedDRA code present but corresponding MedDRA term missing, or vice versa	Warning	1	0
38	CST0100	SDTM0432	1	6	SRCDATA.AE	No errors detected in SRCDATA.AE	Info	0	0
39	SDTM0433	SDTM0433	1	1	SRCDATA.TU	TUEVAL must be populated when TUEVALID is populated	Error	1	0
40	SDTM0433	SDTM0433	1	2	SRCDATA.TU	TUEVAL must be populated when TUEVALID is populated	Error	1	0
41	SDTM0434	SDTM0434	1	1	SRCDATA.TR	TREVAL must be populated when TREVALID is populated	Error	1	0
42	SDTM0435	SDTM0435	1	1	SRCDATA.RS	RSEVAL must be populated when RSEVALID is populated	Error	1	0
146	CST0102		1	1	CSTUTIL_SAVERES…	results.validation_metrics was created as requested	Info	0	0
147	CST0102		1	1	CSTUTIL_SAVERES…	results.validation_results was created as requested	Info	0	0

Display 1. Example of a Validation Results Data Set (First part of columns)

actual	keyvalues	resultdetails
AEDECOD=ZQRA Syndrome,AEPTCD=.	STUDYID=SASCSTDEMODATA,USUBJID=S001P002,AEDECO...	Has coding been done against the same version of MedDRA?
AEBODSYS=GASTROINTESTINAL DISO...	STUDYID=SASCSTDEMODATA,USUBJID=S001P002,AEDECO...	Has coding been done against the same version of MedDRA?
TUEVAL=,TUEVALID=RADIOLOGIST	STUDYID=SASCSTDEMODATA,USUBJID=S001P011,TUSEQ=2	
TUEVAL=,TUEVALID=RADIOLOGIST	STUDYID=SASCSTDEMODATA,USUBJID=S001P011,TUSEQ=3	
TREVAL=,TREVALID=RADIOLOGIST	STUDYID=SASCSTDEMODATA,USUBJID=S001P011,TRSEQ=2	
RSEVAL=,RSEVALID=INVESTIGATOR	STUDYID=SASCSTDEMODATA,USUBJID=S001P011,RSSEQ=3	

Display 2. Example of a Validation Results Data Set (Second part of columns)

CROSS-STANDARD VALIDATION CHECKS

The CDISC ADaM 2.1 standard validation requires the use of a number of cross-standard validation checks. These cross-standard validation checks include comparing data and metadata between two different standards, such as ADaM 2.1 and SDTM 3.1.3. To validate the clinical data between two different standards to ensure the consistency is always a problem that makes SAS programmers spend a lot of efforts on it.

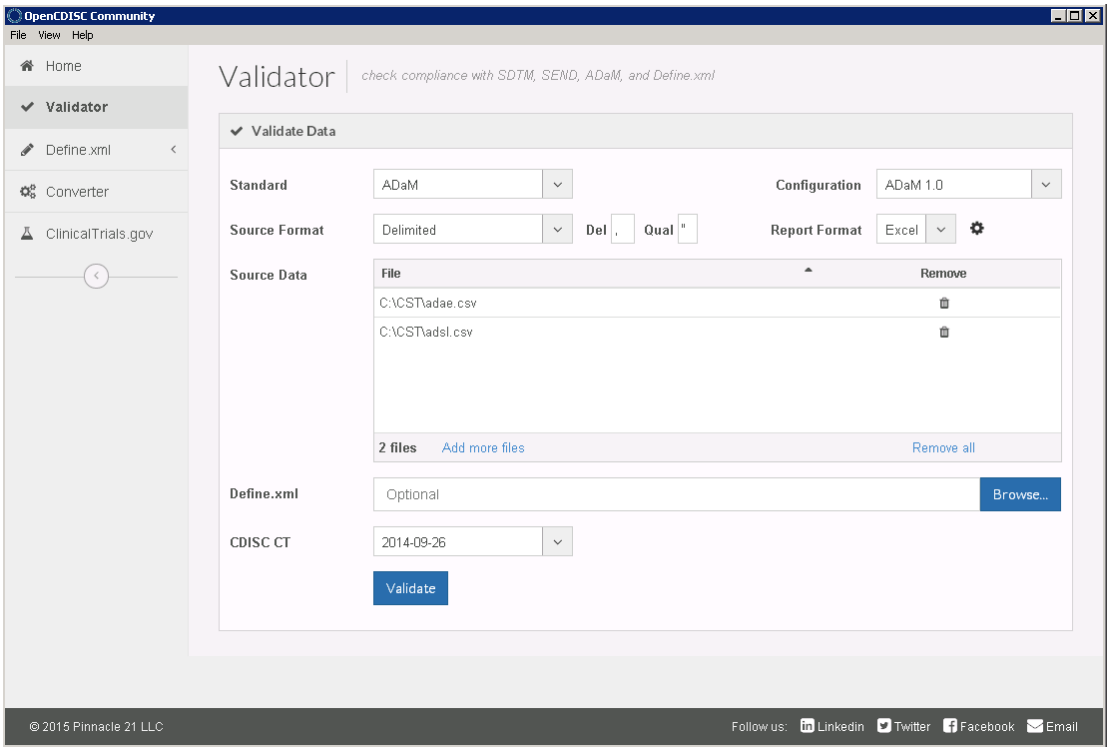
For example, we only want to ensure that all the adverse event terms (AETERM) in ADaM2.1 domain ADAE can be found in SDTM 3.1.3 domain AE. Display 3 shows part of the ADAE.sas7bdat.

	STUDYID	USUBJID	AESEQ	AETERM	AEDECOD	AEBODSYS
1	SASCSTDEMODATA	S001P002	1	No such term	ZQRA Syndrome	GASTROINTESTINAL DIS...
2	SASCSTDEMODATA	S001P003	2	ABDOMINAL CRAMP	Abdominal pain	GASTROINTESTINAL DIS...
3	SASCSTDEMODATA	S001P003	3	RASH	Rash	SKIN AND SUBCUTANEO...
4	SASCSTDEMODATA	S001P005	4	ABDOMINAL CRAMP	Abdominal pain	GASTROINTESTINAL DIS...
5	SASCSTDEMODATA	S001P005	5	HEARTBURN	Dyspepsia	GASTROINTESTINAL DIS...
6	SASCSTDEMODATA	S001P005	7	PRURITIC RASH	Rash pruritic	SKIN AND SUBCUTANEO...
7	SASCSTDEMODATA	S001P005	6	RASH	Rash	SKIN AND SUBCUTANEO...
8	SASCSTDEMODATA	S001P007	9	HEADACHE	Headache	NERVOUS SYSTEM DISO...
9	SASCSTDEMODATA	S001P007	8	HEARTBURN	Dyspepsia	GASTROINTESTINAL DIS...

Display 3. Part of the ADAE.sas7bdat

USING OPENCDISC TO VALIDATE

How to deal with this problem? As is well-known, the most commonly used open source validation tool for now is OpenCDISC community. The latest version 2.0.2 can be downloaded at its official website http://www.opencdisc.org/. Display 4 shows the validator of OpenCDISC community 2.0.2. This validator only supports three kinds of source data format, SAS transport (XPORT) v.5, text-delimited and dataset-XML.

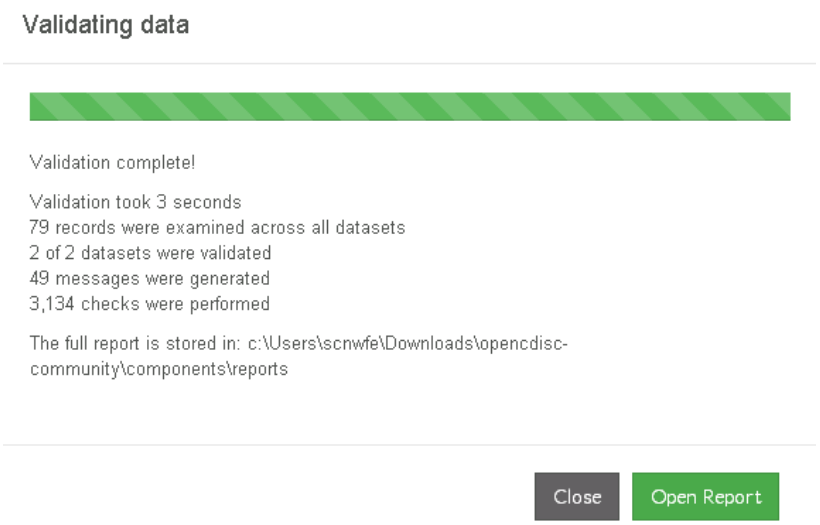


Display 4. Validator of OpenCDISC community 2.0.2

To resolve this problem, the following steps have to be taken:

1. Transform the format from SAS data set format to the format that validator can accept, text-delimited in this case.
2. Run the validation.
3. Check the validation results.

Display 5 and 6 shows the results of OpenCDISC validation. From the results, there is no error or warning on ADAE domain. Further check the report and the validation rules, we can find that there is even no such cross-standard check. And even more, the validator cannot specify the checks you need; it can only run all the checks at a time.



Display 5. OpenCDISC Validation Results

Processed Sources							
Domain	Label	Class	Source	Records	Errors	Warnings	Notices
GLOBAL	Global Metadata	--	--	--	0	0	0
ADAE	Adverse Event Analysis Dataset	ADAM OTHER	adae.csv	9	0	0	0
ADSL	Subject-Level Analysis	SUBJECT LEVEL ANALYSIS DATASET	adsl.csv	70	41	0	8
Total				79	41	0	8

Display 6. OpenCDISC Validation Report

USING CST TO VALIDATE

But CST provides a very neat solution on this problem. It encapsulates the most comprehensive check points into one data set validation_master. It provides several utility macros to support the cross-standard checks and it can specify the checks so that you don't need to run all the checks at a time.

The most important two macros that enable cross-standard comparisons are:

- cstcheck_crossstdcomparedomains.sas is for comparing the source data between two different standards
- cstcheck_crossstdmetamismatch.sas is for comparing the metadata between two different standards.

These macros are located at the framework macros location.

Cross-standard validation on source data

The cstcheck_crossstdcomparedomains macro compares values for one or more columns in one table with those same columns in another domain in another standard.

CST has the following validation check showed by display 7 and 8 that uses this macro whose check id is ADAM0653 in validation_master.sas7bdat of global standards library:

	checkid	standard	standard version	check source	source id	check severity	checktype	codesource
53	ADAM0653	CDISC-ADAM	***	SAS	SAS0653	Error	ValueConsistency	cstcheck_crossstdcomparedomains

Display 7. Validation check ADAM0653 (First part of columns)

usesource metadata	tablescope	columnscope	codelogic	code type	lookup type	lookup source
Y	class:ADAE	STUDYID+USUBJID+AESEQ+AETERM	%let _cstCheckVar=AETERM;proc sq...	2		

Display 8. Validation check ADAM0653 (Second part of columns)

Here are the sample codes from codelogic field of validation check ADAM0653. In this example, &_cstDSName (ADaM data set name) and &_cstCrossDataLib (SDTM library) are generated by the macro prior to execution of codelogic.

```
%let _cstCheckVar=AETERM;
proc sql noprint;
  create table work._cstproblems as
  select adam.studyid, adam.usubjid, adam.aeseq, adam.&_cstCheckVar
  from &_cstDSName as adam
  left join &_cstCrossDataLib..ae as sdtm on
  adam.studyid=sdtm.studyid and adam.usubjid=sdtm.usubjid and adam.aeseq=sdtm.aeseq
  where adam.&_cstCheckVar ne sdtm.&_cstCheckVar;
quit;
```

And CST also has the following rule description and error message for this check in messages.sas7bdat of global standards library:

	resultid	standard version	check source	sourceid	check severity	sourcedescription
51	ADAM0653	***	SAS	SAS0653	Error	Specified record not found in SDTM for this subject

Display 9. Messages of ADAM0653 (First part of columns)

messagetext	parameter1	parameter2	message details
Corresponding SDTM record not found based on STUDYID, USUBJID and AESEQ			

Display 10. Messages of ADAM0653 (Second part of columns)

The validation process is similar with the example SDTM 3.1.3 validation process, except only a few codes need to be updated for ADaM validation:

1. Some macro variables used as substitution parameters need to be updated or added.

```
%let _cstStandard=CDISC-ADAM;
%let _cstStandardVersion=2.1;
%let _cstTrgStandard=CDISC-SDTM;
%let _cstTrgStandardVersion=3.1.3;
```

2. SASReferences data set needs to be updated.
3. Validation macro adam_validate should be used for ADaM 2.1 checks.

Here are the sample codes that can be used to narrow the validation checks scope to run the specified validation checks chosen by you. Only ADAM0653 is run in this example:

```
data cntl_v.validation_control;
 set refcntl.validation_master;
 where (trim(checkid) in('ADAM0653'));
run;
%adam_validate;
```

Display 11 and 12 show the results data set. From the results, one record in domain ADAE doesn't compliance with ADAM0653, that is, the AETERM value "No such term" of this record in ADAE cannot be found in SDTM AE domain.

	resultid	checkid	result seq	seq no	srcdata	message	result severity	result flag	_cst_rc
1	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:\cstGlobalLibrary/standards/cst-fra...	Info	0	0
2	CST0200		1	1	CST_CREATEDSFROM...	The SAS libref csttmplt was allocated to c:\cstGlobalLibrary/standards/cst-framework-...	Info	0	0
3	CST0102		1	2	CST_CREATEDSFROM...	work.validation_sasrefs (SAS File and Library References) was created as requested	Info	0	0
4	CST0200		1	1	CSTUTIL_PROCESSSE...	Process setup is using this SASReferences: C:\Users\scnwfe\AppData\Local\Temp\2...	Info	0	0
5	CST0200		1	1	CST_INSERTSTANDAR...	SASReferences data set was successfully validated	Info	0	0
6	CST0200		1	2	CSTUTIL_ALLOCATESA...	SASReferences data set was successfully validated	Info	0	0
7	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:\cstGlobalLibrary/standards/cdisc-a...	Info	0	0
8	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:\cstSampleLibrary/cdisc-adam-2.1-1...	Info	0	0
9	CST0200		1	1	ADAM_VALIDATE	PROCESS STANDARD: CDISC-ADAM	Info	0	0
10	CST0200		1	2	ADAM_VALIDATE	PROCESS STANDARDVERSION: 2.1	Info	0	0
11	CST0200		1	3	ADAM_VALIDATE	PROCESS DRIVER: Unspecified	Info	0	0
12	CST0200		1	4	ADAM_VALIDATE	PROCESS DATE: 2015-07-30T04:18:21	Info	0	0
13	CST0200		1	5	ADAM_VALIDATE	PROCESS TYPE: VALIDATION	Info	0	0
14	CST0200		1	6	ADAM_VALIDATE	PROCESS SASREFERENCES: C:\Users\scnwfe\AppData\Local\Temp\2\SAS Temp...	Info	0	0
15	CST0200		1	7	ADAM_VALIDATE	PROCESS STUDYROOTPATH: c:\cstSampleLibrary/cdisc-adam-2.1-1.7/sascstdem...	Info	0	0
16	CST0200		1	8	ADAM_VALIDATE	PROCESS GLOBALLIBRARY: c:\cstGlobalLibrary	Info	0	0
17	CST0200		1	9	ADAM_VALIDATE	PROCESS CSTVERSION: 1.7	Info	0	0
18	CST0200		1	10	ADAM_VALIDATE	PROCESS CONTROLLED TERMINOLOGY SOURCE: c:/cstGlobalLibrary/standard...	Info	0	0
19	ADAM0653	ADAM0653	1	1	SRCDATA.ADAE	Corresponding SDTM record not found based on STUDYID, USUBJID and AESEQ	Error	1	0
20	CST0203		1	11	ADAM_VALIDATE	Validation process completed successfully	Info	0	0
21	CST0018		1	1	CSTUTILVALIDATIONS...	1+ validation warnings or errors reported	Info	0	0
22	CST0102		1	1	CSTUTIL_SAVERESULTS	results.validation_metrics was created as requested	Info	0	0
23	CST0102		1	1	CSTUTIL_SAVERESULTS	results.validation_results was created as requested	Info	0	0

Display 11. Results of Validation Check ADAM0653 (First part of columns)

actual	keyvalues	resultdetails
STUDYID=SASCSTDEMODATA,USUBJID=S001P002,AESEQ=1,AETERM=No such term	STUDYID=SASCSTDEMODATA,USUBJID=S001P002,AETERM=No such term,AESTDTC=.	

Display 12. Results of Validation Vheck ADAM0653 (Second part of columns)

Cross-standard validation on metadata

The cstcheck_crossstdmetamismatch macro identifies inconsistencies in metadata across registered standards.

An example ADaM validation check that uses this macro is ADAM0651. This check validates: ADaM column with a column name prefix of 'AE' not found in SDTM. If the validation process detects this issue, it reports error: ADaM column name starting with AE found having no like-named SDTM column.

The full codeLogic PROC SQL step for ADAM0651 can also be found at validation_master.sas7bdat of global standards library.

VALIDATION CUSTOMIZATION

Another problem that troubles the SAS programmers is updating the existing SAS code if the CDISC standards upgrades. Or sometimes the SAS programmers need to add their own standards defined by themselves just for testing or other purposes.

One of the significant benefits of CST is that you can customize the solution to meet your needs. From a validation perspective, this includes but not limits to:

- Modifying an existing standard or defining a new reference standard.
- Using any set of source data and metadata.
- Modifying the SAS validation checks for supported standards.
- Adding new validation checks for supported standards.
- Modifying existing validation check macros or adding new macros.
- Modifying the SAS Clinical Standards Toolkit messaging, including internationalization.
- Attempting to validate multiple studies in a single validation process.

REGISTERING A NEW VERSION OF A STANDARD

CST provides a method to register or add a new version of an existing standard very easily. For example, CDISC SDTM standard upgrades from version 3.1.3 to version 3.2. SDTM 3.2 needs to be added to global standards library for future use. The following codes can be used:

```
%cstutil_setcstgroot;
%cst_registerstandard(
  _cstRootPath=%nrstr(&_cstGRoot./standards/cdisc-sdtm-3.2-1.7),
  _cstControlSubPath=control,
  _cstStdDSName=standards,
  _cstStdSASRefsDSName=StandardSASReferences,
  _cstStdLookupDSName=standardlookup);
```

The first macro ensures that the macro variable that contains the global standards library path is set.

The second macro registers the standard by passing this information:

- The main path to the directory that contains the standard version's files.
- The path to the registration data sets that are used to populate the global standards library metadata data sets. This is the name of the subfolder in the _cstRootPath parameter value. This subfolder must exist before registering the standard.
- The names of the Standards and StandardSASReferences data sets. These data sets have the same structure as the data sets in the global standards library metadata directory. Both of these data sets are required to define a new standard or a new version of a standard.
- The name of the Standardlookup data set. This data set has the same structure as the data set in the global standards library/metadata directory. This data set is optional.

The entire register codes are located at programs directory of global standards library and it can be used directly or modified as needed.

CONCLUSION

CST framework provides a series of neat tools to accelerate and simplify the validation process. It contains the most comprehensive validation checks of CDISC standards and “out of the box” sample programs comparing with other validation tools. It can be customized flexibly to meet the users’ needs.

Furthermore, CST is a powerful and flexible toolkit which can help you not only validating your clinical data, but also supporting other work in clinical research activities. Although this paper only focuses on the validation process, other tools such as those for handling XML related standards or files can also be found at CST framework.

REFERENCES

SAS Institute: “SAS Clinical Standards Toolkit 1.7: User's Guide.” Available at http://support.sas.com/documentation/onlinedoc/clinical/index.html.

OpenCDISC: OpenCDISC community 2.0.2. Available at http://www.opencdisc.org/.

ACKNOWLEDGMENTS

First and foremost, I would like to show my deepest gratitude to my supervisor, Han Liu, who has provided me with valuable guidance in every stage of the writing of this thesis.

Second I shall extend my thanks to the persons who review this paper and give me valuable comments.

Last but not least, I'd like to thank my family, for their encouragement and support.

CONTACT INFORMATION

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

Name: Wei Feng
Enterprise: SAS Research and Development (Beijing) Co., Ltd.
Address: Motorola Plaza, No. 1 Wang Jing East Road
City, State ZIP: Beijing, 100102
Work Phone: (8610) 83193355-3524
Fax: (8610) 6310-9130
E-mail: wei.feng@sas.com
Web: www.sas.com