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

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.

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:

```sas
/* 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;
```

Figure 2. CST Validation Process
data _null_;  
call symput('studyRootPath',cats("&_cstRoot","/cdisc-sdtm-3.1.3- 
&_cstVersion/sascstdemodata"));  
call symput('studyOutputPath',cats("&_cstRoot","/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.]
;  
quilt;

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

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 “resultdetails” 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.

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

Display 1. Example of a Validation Results Data Set (First part of columns)
How to validate clinical data more efficiently with SAS® Clinical Standards Toolkit

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.

<table>
<thead>
<tr>
<th>STUDYID</th>
<th>USUBJID</th>
<th>ASEQ</th>
<th>AETERM</th>
<th>AEDECOD</th>
<th>AEBODSYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S001F001</td>
<td>S001F002</td>
<td>1</td>
<td>No such term</td>
<td>ZORA Syndrome</td>
<td>GASTROINTESTINAL DIS...</td>
</tr>
<tr>
<td>S001F001</td>
<td>S001F003</td>
<td>2</td>
<td>ABDOMINAL CRAMP</td>
<td>Abdominal pain</td>
<td>GASTROINTESTINAL DIS...</td>
</tr>
<tr>
<td>S001F001</td>
<td>S001F003</td>
<td>3</td>
<td>RASH</td>
<td>Rash</td>
<td>SKIN AND SUBCUTANED...</td>
</tr>
<tr>
<td>S001F001</td>
<td>S001F005</td>
<td>4</td>
<td>ABDOMINAL CRAMP</td>
<td>Abdominal pain</td>
<td>GASTROINTESTINAL DIS...</td>
</tr>
<tr>
<td>S001F001</td>
<td>S001F005</td>
<td>5</td>
<td>HEARTBURN</td>
<td>Dyspepsia</td>
<td>GASTROINTESTINAL DIS...</td>
</tr>
<tr>
<td>S001F001</td>
<td>S001F005</td>
<td>7</td>
<td>PRURITIC RASH</td>
<td>Rash pruritic</td>
<td>SKIN AND SUBCUTANED...</td>
</tr>
<tr>
<td>S001F001</td>
<td>S001F005</td>
<td>6</td>
<td>RASH</td>
<td>Rash</td>
<td>SKIN AND SUBCUTANED...</td>
</tr>
<tr>
<td>S001F001</td>
<td>S001F007</td>
<td>9</td>
<td>HEADACHE</td>
<td>Headache</td>
<td>NERVOUS SYSTEM DISO...</td>
</tr>
<tr>
<td>S001F001</td>
<td>S001F007</td>
<td>8</td>
<td>HEARTBURN</td>
<td>Dyspepsia</td>
<td>GASTROINTESTINAL DIS...</td>
</tr>
</tbody>
</table>

Display 3. Part of the ADAE.sas7bdat
How to validate clinical data more efficiently with SAS Clinical Standards Toolkit

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.
Validating data

Validation completed
Validation took 3 seconds
79 records were examined across all datasets
2 of 2 datasets were validated
49 messages were generated
3,134 checks were performed

The full report is stored in: c:\Users\scwfe\Downloads\opencdisc-community\components\reports

Display 5. OpenCDISC Validation Results

<table>
<thead>
<tr>
<th>Domain</th>
<th>Label</th>
<th>Class</th>
<th>Source</th>
<th>Records</th>
<th>Errors</th>
<th>Warnings</th>
<th>Notices</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL</td>
<td>Global Metadata</td>
<td>--</td>
<td>--</td>
<td>79</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ADAE</td>
<td>Adverse Event Analysis Dataset</td>
<td>ADAM OTHER</td>
<td>adae.csv</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ADSL</td>
<td>Subject-Level Analysis</td>
<td>SUBJECT LEVEL ANALYSIS DATASET</td>
<td>adsl.csv</td>
<td>70</td>
<td>41</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>79</td>
<td>41</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>checklist</th>
<th>standard</th>
<th>standard version</th>
<th>check source</th>
<th>source id</th>
<th>check severity</th>
<th>checktype</th>
<th>codesource</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAM0653</td>
<td>CDISC-ADAM</td>
<td>***</td>
<td>SAS</td>
<td>SAS0053</td>
<td>Error</td>
<td>ValueConsistency</td>
<td>cstcheck_crossstdcomparedomains</td>
</tr>
</tbody>
</table>

Display 7. Validation check ADAM0653 (First part of columns)
How to validate clinical data more efficiently with SAS® Clinical Standards Toolkit

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.

```sas
%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:

Display 9. Messages of ADAM0653 (First 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.
   ```sas
   %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:

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

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

<table>
<thead>
<tr>
<th>resultcode</th>
<th>resultid</th>
<th>cbi</th>
<th>msg</th>
<th>msgno</th>
<th>sid</th>
<th>stcode</th>
<th>message</th>
<th>resultseverity</th>
<th>resultflag</th>
<th>_cst_remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CST0030</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>CST_SETPROPERTIES</td>
<td>The properties were processed from the PATH c:\sas\GlobalLibrary\standards\etc\data\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>CST0030</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>CST_CREATEDFROM</td>
<td>The SAS library was allocated to c:\sas\GlobalLibrary\standards\etc\data\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>CST0082</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>CST_CREATEDFROM</td>
<td>validation_cvals (SAS File and Library Reference) was created as requested</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>CST_METADATAVALIDATION</td>
<td>Process setup using the SASReferences C:sers\tom_Updata\LocalTempl\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>CST_METADATAVALIDATION</td>
<td>SASReferences data set was successfully validated</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>CST0080</td>
<td>2</td>
<td></td>
<td></td>
<td>2</td>
<td>CST_METADATAVALIDATION</td>
<td>SASReferences data set was successfully validated</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>CST_SETPROPERTIES</td>
<td>The properties were processed from the PATH c:\sas\GlobalLibrary\standards\etc\data\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>CST_SETPROPERTIES</td>
<td>The properties were processed from the PATH c:\sas\GlobalLibrary\standards\etc\data\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>CST_SETPROPERTIES</td>
<td>The properties were processed from the PATH c:\sas\GlobalLibrary\standards\etc\data\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS_STANDARDVALIDATION: 21</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>3</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS DRIVER: Unexpected</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>4</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS_DATE: 2050-01-01 00:00:00</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>5</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS TYPE: VALIDATION</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>6</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS SASREFERENCES C:sers\tom_Updata\LocalTempl\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>7</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS STUDYMetadata\checkMetadata\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>8</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS GLOBALLIBRARY c:\sas\GlobalLibrary\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>9</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS STANDARDIZATION SOURCE c:/sas\GlobalLibrary\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>CST0080</td>
<td>1</td>
<td></td>
<td></td>
<td>10</td>
<td>ADAM_VALIDATE</td>
<td>PROCESS CONTROLEDTERMINOLOGYSOURCE c:/sas\GlobalLibrary\</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>ADAM0653</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>ADAM_VALIDATE</td>
<td>STUDYID:GAS0STUDMO_DATAUSUBID=AMOPPDZASEED=14210T2M=No such term</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>CST0083</td>
<td>1</td>
<td></td>
<td></td>
<td>11</td>
<td>ADAM_VALIDATE</td>
<td>Validation process completed successfully</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>CST0083</td>
<td>1</td>
<td></td>
<td></td>
<td>12</td>
<td>ADAM_VALIDATE</td>
<td>STUDYID:GAS0STUDMO_DATAUSUBID=AMOPPDZASEED=14210T2M=No such term</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>CST0083</td>
<td>1</td>
<td></td>
<td></td>
<td>13</td>
<td>ADAM_VALIDATE</td>
<td>STUDYID:GAS0STUDMO_DATAUSUBID=AMOPPDZASEED=14210T2M=No such term</td>
<td>Info</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Display 12. Results of Validation Check 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.

**REGISTRATION 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:

```sas
%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
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

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.