Introduction to SAS® Clinical Standards Toolkit

Andreas Mangold, HMS Analytical Software, Heidelberg, Germany
Nicole Wächter, HMS Analytical Software, Heidelberg, Germany

ABSTRACT
The SAS® Clinical Standards Toolkit is a framework for the validation of CDISC data, especially SDTM, and the generation of define.xml. It has been launched by SAS Institute as an add-on to SAS Base at no additional charge in mid 2009. Currently it includes the validation of files according to the SDTM model (including most of the published WebSDM™ and Janus checks), as well as the creation of metadata according to CRT-DDS for the submission. Since all the metadata and rules are stored in SAS tables, modified standards or new versions of existing standards can be integrated by the user. Also, SAS Institute has announced the integration of further standards and standard versions.
This tutorial gives an introduction to the SAS Clinical Standards Toolkit. Programming examples will be provided along with some background about the modular structure of the toolkit. Participants will be enabled to assess usability for their application.

BACKGROUND
Pharmaceutical companies increasingly apply industry-wide clinical data standards a) to meet the requirements of the US Food and Drug Administration (FDA) when submitting New Drug Applications (NDA) electronically (eCTD submission) and b) to ease the data exchange with partners and CROs. A global standards organization is the Clinical Data Interchange Standards Consortium (CDISC) that published the Study Data Tabulation Model (SDTM) of human clinical study data tabulations for submission to regulatory authorities. The FDA stores CDISC SDTM data plus their accompanying metadata in a so-called Janus data warehouse. The metadata of the content and structure of the submitted clinical data are described in a machine readable XML document named Case Report Tabulation Data Definition Specification (CRT-DDS or define.xml). The XML schema for the define document is also based on an extension of another standard, the CDISC Operational Data Model (ODM).
Before loading clinical data and the define.xml into the Janus warehouse, several validation checks are performed deploying the WebSDM software. Conformance of the submission deliverables (data and metadata) to SDTM and CRT-DDS is verified. After successfully loading the files into Janus, additional validation checks are employed within the Janus reviewer tools (Janus checks).
For the pharmaceutical industry it is now a necessity to generate and validate the FDA submission deliverables. A SAS macro based framework, the SAS Clinical Standards Toolkit (CST), alleviates the process of becoming compliant with regulatory standards.
The SAS Clinical Standards Toolkit 1.2 initially provides a set of standards and functionality aimed to generate the define.xml (from study metadata) and to perform validation checks against implemented standards. Currently the SAS Clinical Standards Toolkit 1.2 supports the CDISC-SDTM 3.1.1, CDISC-CRTDDS 1.0 and CDISC-Terminology-200810 standards and has implemented most WebSDM 2.6, the Janus and several SAS-developed checks to back up the SDTM compliance. Support for SDTM 3.1.2 and terminology 201003 is available as a preproduction update and it is announced that it will become productive in the end of 2010.
Exceeding these standards, the toolkit has also the capability to be extensible and configurable to add new validation checks, new versions of standards, custom standards and upcoming standards. To manage that, the modular designed CST is built as a framework plus various pluggable standard modules that contribute to the centrally managed process runs. By providing an extensive process library of utility programs and data the toolkit enables the user to build robust processes to accomplish extensions to the above listed tasks and standards.

TUTORIAL OVERVIEW
As a start, we will introduce the software architecture, system requirements and versions of the SAS Clinical Standards Toolkit. The main parts of the tutorial will explain four different sample programs for the areas of SDTM validation and define.xml generation, one basic sample and one more advanced sample for each area. More information can be found in the user’s guide (reference 1).

INTRODUCTION TO THE SOFTWARE ARCHITECTURE

SYSTEM REQUIREMENTS
The SAS Clinical Standards Toolkit is available for SAS 9.1.3 on Microsoft Windows and SAS 9.2 on Microsoft Windows (not
64 bit) and UNIX. The only requirement from the part of SAS licenses is SAS BASE. An installed Java virtual machine is needed for the creation and validation of define.xml.

For SAS 9.2, the installation medium will be delivered, upon request, free of charge from SAS Institute. A download for the installation under SAS 9.1.3 is available from the SAS website (reference 2).

VERSIONS AND THEIR SUPPORT FOR STANDARDS

At the time of preparation of this paper, version 1.2 of the SAS Clinical Standards Toolkit is the most current version. It supports validation for SDTM 3.1.1 and uses the CDISC terminology 2008-10.

There is a preproduction software update for the toolkit which gives partial support (other than updated validation checks) for version 3.1.2 of SDTM and for version 2010-03 of the CDISC terminology. This update is available from the SAS Institute website and also includes enhanced reporting capabilities (reference 3).

SAS announced version 1.3 of the toolkit for the end of 2010 and that it will give full support of SDTM 3.1.2, including documented checks from WebSDM™ 3.0 and OpenCDISC. OpenCDISC (www.opencdisc.org) is an open source community focused on building extensible frameworks and tools for the implementation and advancement of CDISC Standards. SAS has also announced that CDISC standard ADaM may be supported in the future: “Once ADaM 2.1 and ADaM Implementation Guide, Version 1.0 are finalized, provision of a SAS representation of ADaM will be considered.” (Reference 1)

THE GLOBAL STANDARDS LIBRARY

Figure 1 shows an overview of the components of the SAS Clinical Standards Toolkit.

- The central component is the standards registry, which contains two SAS datasets, in which the installed standards are registered, as well as several XML files. One of the SAS datasets lists the individual standards and their installed versions. The other one lists all the resources used by a standard.

- The four blocks to the left represent the accompanying standards, whereby the actual Clinical Standards Toolkit Framework (CST Framework 1.2) is likewise registered as a standard. Every standard consist of some or all of the following components:
  - Macros: Standard-specific SAS macros (for example, carrying out the generation of define.xml)
  - Reference metadata for the description of data tables and columns according to the CDISC standard
  - Test rules for compliance testing (“Validation Checks”) lodged in SAS Tables
  - Text messages used in validation reports (“Messages”)
  - Properties (name/value pairs in text files) for controlling the program execution.

- The framework enhances Base SAS by a few so-called framework macros, which provide auxiliary functions such as adding new standards or generating files from metadata.
DIRECTORY STRUCTURE
The SAS Clinical Standards Toolkit is being installed in three locations: global standards library (see Figure 2), samples (Figure 3) and framework macros (Figure 4). User defined data and programs should not be placed into these locations. The only exception to this is the implementation of new standards.

![Folder structure of global standards library](image)

The global standards library (Figure 2) contains the standards registry, some XML files and one subfolder per standard version. During installation, there is a prompt for the location of the global standards library. In a productive environment, this is a shared resource for many users. So this should reside outside the normal SAS installation.

For every standard version, there is one additional folder structure (Figure 3) which contains the files belonging to the standard and a sample study where relevant (SDTM and CRTDDS only). The samples are also used for installation qualification. These folders are installed on the level of SASFoundation for SAS 9.2.

As always in SAS, reusable macros are stored in a folder called sasmacro below the product folder (called cstframework) which in turn resides below the sasroot folder (Figure 4).
Figure 3: Folder structure for samples per standard

- ClinicalStandardsToolkitFramework
- SASClinicalStandardsToolkitCRTDDS10
- SASClinicalStandardsToolkitSDTM311
- 9.2
  - _uninstCSTSDTM
  - CDISCSDTMImplementationGuide3.1.1
- sample
  - cdisc-sdtm-3.1.1
    - sascstdemodata
      - control
      - data
      - metadata
      - programs
      - results
      - terminology
  - standards
- SASClinicalStandardsToolkitTerminology200810
- SASFoundation

Figure 4: Folder structure for framework macros

INSTALLATION
The following applies to the installation under SAS 9.2:
Installation is done with the deployment wizard like for any other SAS product. The following points are worth mentioning:

- SAS Foundation has to be installed together with the toolkit even if it was installed before.
- A path to the global standards library has to be provided in the course of the installation process. This might be local or shared. In a productive environment, it must be shared and read only.

After installing the product, an installation qualification procedure should be followed (reference 4).
The following components, inputs and outputs are used in the validation process, as shown in Figure 5:

- **In the reference control** dataset (called SASReferences), the following input and output components of the process are specified.
- **Source (Study) data** and the respective **metadata** as input
- **Reference metadata, properties** (values of macro variables) and the **validation rules** (checks to be run) for controlling the validation process
- Optional: **Controlled Terminologies** with standardized dictionaries for checking encoded values
- The results file (**Results Dataset**) contains the results of the applied validation checks, making possible violations against the SDTM Standard visible
- The metrics file (**Metrics Dataset**) contains a summary of counts for the number of validations carried out, the number of violations and the number of warnings.
- Both result files can be utilized for extended **reporting** (preproduction in version 1.2 of the toolkit).

**SIMPLE EXAMPLE**

The following example shows the most basic steps in validating SDTM data. It uses demo data delivered with the SAS Clinical Standards Toolkit and control datasets prepared for this sample. The next section will explain more about control data sets.

```sas
/*-- root location of the process input and output --*/
%let studyRootPath=C:\projects\PhUSE\demol;

/*-- load basic configuration to macro variables --*/
%cst_setStandardProperties(
   _cstStandard=CST-FRAMEWORK
   ,_cstStandardVersion=1.2
   ,_cstSubType=initialize
 );
%cst_setStandardProperties(
   _cstStandard=CDISC-SDTM
 );
```
Have a look at the results dataset and the metrics dataset in the library named results. Here is an excerpt from the results data set:

<table>
<thead>
<tr>
<th>Result identifier</th>
<th>Validation check id</th>
<th>Seq. no.</th>
<th>Source data</th>
<th>Resolved message text from message file</th>
<th>Result severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CST0108</td>
<td>1</td>
<td>CST_SETPROPERTIES</td>
<td>The properties were processed from the PATH C:\Programme\SAS\cstGlobalLibrary\standards/cst-framework\programs\initialize.properties</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>CST0108</td>
<td>1</td>
<td>CST_SETPROPERTIES</td>
<td>The properties were processed from the PATH C:\Programme\SAS\cstGlobalLibrary\standards/cdisc-sdtm-3.1.1\programs\initialize.properties</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>CST0200</td>
<td>1</td>
<td>SDTM_VALIDATE</td>
<td>PROCESS STANDARD: CDISC-SDTM</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>CST0200</td>
<td>2</td>
<td>SDTM_VALIDATE</td>
<td>PROCESS STANDARDVERSION: 3.1.1</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>CST0200</td>
<td>3</td>
<td>SDTM_VALIDATE</td>
<td>PROCESS DRIVER: SDTM_VALIDATE</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>CST0200</td>
<td>4</td>
<td>SDTM_VALIDATE</td>
<td>PROCESS DATE: 2010-10-12T13:38:05</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>CST0200</td>
<td>5</td>
<td>SDTM_VALIDATE</td>
<td>PROCESS TYPE: VALIDATION</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>CST0200</td>
<td>6</td>
<td>SDTM_VALIDATE</td>
<td>PROCESS SASREFERENCES: C:\projects\PhUSE\demo1\control\sasreferences.sas7bdat</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>CST0100</td>
<td>SDTM0011</td>
<td>1</td>
<td>WORK._CSTSRCCOLUMN_METADATA</td>
<td>No errors detected in source data</td>
<td>Info</td>
</tr>
<tr>
<td>SDTM0015</td>
<td>SDTM0015</td>
<td>1</td>
<td>SUPPAE</td>
<td>Variable IDVAR appears in dataset but is not in SDTM standard</td>
<td>Warning</td>
</tr>
<tr>
<td>SDTM0015</td>
<td>SDTM0015</td>
<td>2</td>
<td>SUPPAE</td>
<td>Variable IDVARVAL appears in dataset but is not in SDTM standard</td>
<td>Warning</td>
</tr>
<tr>
<td>CST0100</td>
<td>SDTM0019</td>
<td>1</td>
<td>WORK._CSTSRCCOLUMN_METADATA</td>
<td>No errors detected in source data</td>
<td>Info</td>
</tr>
<tr>
<td>SDTM0452</td>
<td>SDTM0452</td>
<td>1</td>
<td>SRCDATA.AE</td>
<td>AE is Serious but no qualifiers set to 'Y'</td>
<td>Note</td>
</tr>
<tr>
<td>CST0029</td>
<td>SDTM0453</td>
<td>1</td>
<td>CSTCHECK_NOTINCODE LIST</td>
<td>Format catalog WORK.FORMATS in fmtsearch could not be found</td>
<td>Info</td>
</tr>
<tr>
<td>CST0033</td>
<td>SDTM0453</td>
<td>2</td>
<td>CSTCHECK_NOTINCODE LIST</td>
<td>Format search path has been set to WORK.FORMATS SRCFMT.FORMATS CSTFMT.CTERMS</td>
<td>Info</td>
</tr>
<tr>
<td>CST0100</td>
<td>SDTM0453</td>
<td>3</td>
<td>SRCDATA.AE.AESER</td>
<td>No errors detected in source data</td>
<td>Info</td>
</tr>
</tbody>
</table>
Records with result id CST0108 and CST0200 are messages from the framework which document the environment. The other records result from validation checks where the column “validation check id” refers to the corresponding validation check in table control.validation_control (see next table) and the corresponding message in table sdtmmsg.messages (see table after the next).

<table>
<thead>
<tr>
<th>Validation check identifier</th>
<th>Source of check</th>
<th>Severity of check</th>
<th>Category of check</th>
<th>SAS macro module name</th>
<th>Domains to which check applies</th>
<th>Columns to which check applies</th>
<th>SAS format name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDTM0011</td>
<td>Janus</td>
<td>Note</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0012</td>
<td>JanusFR</td>
<td>Error</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0013</td>
<td>Janus</td>
<td>Note</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td><em>ALL</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0014</td>
<td>SAS</td>
<td>Note</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td><em>ALL</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0015</td>
<td>Janus</td>
<td>Warning</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td><em>ALL</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0019</td>
<td>JanusFR</td>
<td>Warning</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td><em>ALL</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0020</td>
<td>SAS</td>
<td>Warning</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td><em>ALL</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0022</td>
<td>SAS</td>
<td>Note</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td><em>ALL</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0023</td>
<td>SAS</td>
<td>Error</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td><em>ALL</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0030</td>
<td>SAS</td>
<td>Note</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0031</td>
<td>SAS</td>
<td>Error</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0032</td>
<td>SAS</td>
<td>Note</td>
<td>Metadata</td>
<td>cstcheck_metamismatch</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDTM0452</td>
<td>Janus</td>
<td>Note</td>
<td>ColumnValue</td>
<td>cstcheck_column</td>
<td>AE</td>
<td>AESER</td>
<td>$NY</td>
</tr>
<tr>
<td>SDTM0453</td>
<td>JanusFR</td>
<td>Error</td>
<td>Cutterm</td>
<td>cstcheck_notincodelist</td>
<td>AE</td>
<td>AESER</td>
<td>$NY</td>
</tr>
</tbody>
</table>

The table above shows the definition of the validation checks used in this example (not all columns are shown). Note that:

- Each check has a unique identifier, but the same check may be included from different sources with different severity, so it is always important to select checks.
- There are different kinds of checks. E.g. some checks operate only on metadata, some directly on column values and others compare column values to controlled terms.
- Every check uses a SAS macro and an additional piece of SAS code also stored in the validation control dataset (not shown in the above table). For instance, the code for check SDTM0452 is:
  ```sas
  if (upcase(&_cstColumn)="Y" and (upcase(AESCAN) ne "Y" and upcase(AESCONG) ne "Y" and upcase(AESDISAB) ne "Y" and upcase(AESDTH) ne "Y" and upcase(AESHOSP) ne "Y" and upcase(AESLIFE) ne "Y" and upcase(AESMIE) ne "Y" and upcase(AESOD) ne "Y")) then _cstError=1;
  ```
- Checks can be designed for groups of domains and groups of columns with a very flexible notation (e.g. “check all columns ending in STDY or ENDY in all domains aside from DS”)
- The validation control dataset is complemented by the messages dataset shown in the table below.

<table>
<thead>
<tr>
<th>Result identifier</th>
<th>Rule description from checksource</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDTM0011</td>
<td>Identifies a column that was described in the domain description but not included in the SAS dataset for that domain</td>
<td>Variable &amp;_cstparm1 in description file not in dataset</td>
</tr>
<tr>
<td>SDTM0012</td>
<td>Identifies a column listed in the domain description as Required (‘Req’) but not included in the SAS dataset for that domain</td>
<td>SDTM required variable &amp;_cstparm1 not found</td>
</tr>
<tr>
<td>SDTM0013</td>
<td>Identifies a column listed in the domain description as Expected (‘Exp’) but not included in the SAS dataset for that domain</td>
<td>SDTM expected variable &amp;_cstparm1 not found</td>
</tr>
<tr>
<td>SDTM0014</td>
<td>Identifies a column listed in the domain description as Permissible (‘Perm’) but not included in the SAS dataset for that domain</td>
<td>SDTM permissible variable &amp;_cstparm1 not found</td>
</tr>
<tr>
<td>SDTM0015</td>
<td>Identifies a column that appears in the SAS dataset but is not listed in the domain description</td>
<td>Variable &amp;_cstparm1 appears in dataset but is not in SDTM standard</td>
</tr>
<tr>
<td>SDTM0019</td>
<td>Identifies a variable where datatype in (study specific) description is not consistent with datatype implicit in SAS dataset</td>
<td>Description file/dataset variable type mismatch for &amp;_cstparm1</td>
</tr>
<tr>
<td>SDTM0020</td>
<td>Column order does not match standard</td>
<td>Column order does not match standard for &amp;_cstparm1</td>
</tr>
<tr>
<td>Result identifier</td>
<td>Rule description from checksources</td>
<td>Message text</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>SDTM0022</td>
<td>Column length &lt; length defined in standard</td>
<td>Column length &lt; length defined in standard for &amp;_cstparm1</td>
</tr>
<tr>
<td>SDTM0023</td>
<td>Column length &gt; length defined in standard</td>
<td>Column length &gt; length defined in standard for &amp;_cstparm1</td>
</tr>
<tr>
<td>SDTM0030</td>
<td>Column label inconsistent with label defined in standard</td>
<td>Column label inconsistent with label defined in standard for &amp;_cstparm1</td>
</tr>
<tr>
<td>SDTM0031</td>
<td>Column format found but column not subject to controlled terminology</td>
<td>Column not subject to controlled terminology for &amp;_cstColumn</td>
</tr>
<tr>
<td>SDTM0032</td>
<td>Column format found but format name mismatch with standard controlled terminology name</td>
<td>Column format name mismatch with standard for &amp;_cstparm1</td>
</tr>
<tr>
<td>SDTM0452</td>
<td>Identifies records where Serious Event (AESER)=&quot;Y&quot; but none of Involves Cancer (AESCAN), Congenital Anomaly or Birth Defect (AESCONG), Persist or Signif Disability/Incacitypacity (AESDISAB), Results in Death (AESDTH), Requires or Prolongs Hospitalization (AESHOSP), Is Life Threatening (AESLIFE), Other Medically Important Serious Event (AESMIE), or Occurred with Overdose (AESOD) equals 'Y'</td>
<td>AE is Serious but no qualifiers set to 'Y'</td>
</tr>
<tr>
<td>SDTM0453</td>
<td>Identifies records where value for [Serious Event (AESER)] is not found in Codelist [YESNO]</td>
<td>Invalid YESNO code</td>
</tr>
</tbody>
</table>

MORE COMPLETE EXAMPLE

```sas
%let studyRootPath=C:\projects\PhUSE\demo2;
%let c_initSASAutos='YESNO';
/*-- root location of the process input and output --*/
%let studyRootPath=C:\projects\PhUSE\demo2;
/*-- save session options and autocall path for later restore --*/
 proc optSave out=work._cstsessionoptions;
 run;
 data _null_; call symputx('_cstInitSASAutos',getoption('sasautos'));
 run;
/*-- load basic configuration to macro variables --*/
%cst_setStandardProperties(
   _cstStandard=CST-FRAMEWORK,
   _cstStandardVersion=1.2,
   _cstSubType=initialize
   );
%cst_setStandardProperties(
   _cstStandard=CDISC-SDTM,
   _cstStandardVersion=3.1.1,
   _cstSubType=initialize
   );
/*-- create empty SASReferences control data set --*/
%cst_createDS(
   _cstStandard=CST-FRAMEWORK,
   _cstStandardVersion=1.2,
   _cstType=control,
   _cstSubType=reference,
   _cstOutputDS=work.sasrefs
);
/*-- fill the SASReferences control data set --*/
 proc sql;
*/*-- inputs from the Global Standards Library, paths and member names will be filled in automatically --*/
insert into work.sasrefs ( standard , standardversion , type , subtype , saasref , reftype , order) 
```

values ('CDISC-SDTM' , '3.1.1' , 'referencecontrol' , 'validation' , 'refcntl' , 'libref' , .)
values ('CDISC-SDTM' , '3.1.1' , 'referencemetadata', 'table' , 'refmeta' , 'libref' , .)
values ('CDISC-SDTM' , '3.1.1' , 'referencemetadata', 'column' , 'refmeta' , 'libref' , .)
values ('CDISC-SDTM' , '3.1.1' , 'autocall' , ' '                , 'sdtmcode' , 'fileref' , 1)
values ('CDISC-SDTM' , '3.1.1' , 'messages'          , ' '                , 'sdtmmsg' , 'libref' , 1)
values ('CDISC-TERMINOLOGY', '200810', 'fmtsearch'        , ' '                , 'cstfmt'  , 'libref' , 2)
values ('CDISC-TERMINOLOGY', '200810', 'properties'       , 'initialize'       , 'initterm', 'fileref', .)
values ('CST-FRAMEWORK'    , '1.2'   , 'messages'         , ' '                , 'cstmsg'  , 'libref' , 2)
values ('CST-FRAMEWORK'    , '1.2'   , 'messages'         , ' '                , 'cstmsg'  , 'libref' , 2)

/*-- user defined inputs with explicit paths and member names --*/
insert into work.sasrefs (standard , standardversion, type , subtype , sasref , reftype , order ,
path, memname)
values ('CDISC-SDTM'       , '3.1.1' , 'control'          , 'reference'        , 'control' , 'libref' , .,
'&studyRootPath\data' , 'sasref.sas7bdat')
values ('CDISC-SDTM'       , '3.1.1' , 'control'          , 'validation'       , 'control' , 'libref' , .,
'&studyRootPath\data' , 'checks.sas7bdat')
values ('CDISC-SDTM'       , '3.1.1' , 'properties'       , 'initialize'       , 'valprop' , 'fileref', .,
'&studyRootPath\metadata' , 'validation.properties')
values ('CDISC-SDTM'       , '3.1.1' , 'properties'       , 'initialize'       , 'valprop' , 'fileref', .,
'&studyRootPath\metadata' , 'validation.properties')
values ('CDISC-SDTM'       , '3.1.1' , 'fmtsearch'        , ' '                , 'srcfmt'  , 'libref' , 1,
'&studyRootPath\terminology' , 'formats.sas7bcat')
values ('CUSTOM' , ' '     , 'referencecterm'   , ' '                , 'ctref'   , 'libref' , .,
'&studyRootPath\terminology' , 'meddra.sas7bdat')
values ('CDISC-SDTM'       , '3.1.1' , 'results'          , 'validationresults', 'results' , 'libref' , .,
'&studyRootPath\results' , 'results.sas7bdat')
values ('CDISC-SDTM'       , '3.1.1' , 'results'          , 'validationmetrics', 'results' , 'libref' , .,
'&studyRootPath\results' , 'metrics.sas7bdat')
quit;

/*-- process sasreferences: allocate librefs etc. --*/
%let _cstSASRefs=work.sasrefs;
%cstutil_allocatesasreferences;

/*-- select validation checks --*/
data work.checks;
set refcntl.validation_master; /* 246 checks */
where checkid='SDTM0452' and checksourc=Janus'
or checkid='SDTM0453' and checksourc=JanusFR'
or checkid='SDTM0011' and checksourc=Janus'
or checkid='SDTM0012' and checksourc=JanusFR'
or checkid='SDTM0013' and checksourc=Janus'
or checkid='SDTM0014' and checksourc=SAS'
or checkid='SDTM0015' and checksourc=Janus'
or checkid='SDTM0019' and checksourc=JanusFR'
or checkid='SDTM0020' and checksourc=SAS'
or checkid='SDTM0022' and checksourc=SAS'
or checkid='SDTM0023' and checksourc=SAS'
or checkid='SDTM0030' and checksourc=SAS'
or checkid='SDTM0031' and checksourc=SAS'
or checkid='SDTM0032' and checksourc=SAS'
;
run;

/*-- run validation, write results and metrics --*/
%let _cstSASRefs=work.sasrefs;
%sdtm_validate;

/*-- cleanup session using above saved options --*/
%cstutil_cleanupcstsession(
    _cstClearCompiledMacros=0,
    _cstClearLibRefs=1,
    _cstResetSASAutos=1,
    _cstResetFmtSearch=1,
    _cstResetSASOptions=1,
    _cstDeleteFiles=1,
    _cstDeleteGlobalMacroVars=1);

The above example shows the following additional functionality:

- Generating the SASReferences file: This file controls the allocation of many resources needed to run the process (see Figure 5): librefs, filerefs, autocall macros, formats, control datasets, input data, input metadata, reference metadata, messages, and properties. Study and run specific resources have to be specified completely with path. Standard specific allocation can be inserted without path and member name, those will be filled in by the allocatesasreferences macro call, which also allocates all the resources. Macro variables should be used as shown to make this process more flexible. Experience has shown that the SASReferences dataset must be prepared very carefully. Most problems can be traced back to the fact that errors were made here. Please refer to the user’s guide of the Clinical Standards Toolkit.

Comparing the two example programs, one can identify two strategies for usage of the SASReferences dataset:
  a) Standardize and reuse it, b) create it on the fly as part of the process.

- Selection of the validation checks from the validation master file: Never run all checks from the validation master. Many validation checks are duplicated in the validation master dataset because they are found in various sources (WebSDM™, Janus). The choice of validation checks has to be adapted to the respective project status and the purpose of the validation. Correct choice of validation checks and the interpretation of the output represent the main effort for the SDTM validation.

- Saving options when starting and restoring the environment at the end. Note that if you choose to clear compiled macros, you have to specify “options mrecall” in order to run the process again in the same SAS session.
GENERATION OF DEFINE.XML

OVERVIEW

The toolkit uses a special intermediate data model ("CRT-DDS datasets") which is a relational representation of the hierarchical structure of the define.xml (see Figure 6). It consists of 39 tables and its mapping to the CRT-DDS standard is described in the toolkit user’s guide. This approach enables the SAS programmer to fill in all the necessary information into SAS tables and then generate the define.xml with a simple macro call.

The toolkit provides a macro as an automated process which generates some of the CRT-DDS datasets from SDTM table and column metadata. Additional information (for instance value level metadata, computational methods, references to CRFs and to the transport files) can be added via SAS programs.

SIMPLE EXAMPLE

/*-- root location of the process input and output --*/
%let studyRootPath=C:\projects\PhUSE\demo3;

/*-- load basic configuration to macro variables --*/
%cst_setStandardProperties(
   _cstStandard=CST-FRAMEWORK
   ,_cstSubType=initialize
);
%cst_setStandardProperties(

The above program works in two steps.

- The first step (`%crtdds_sdtm311todefine10`) takes the study metadata and generates nine datasets in the intermediate CRT-DDS format.
- The second step (`%crtdds_write`) generates the define.xml from the intermediate CRT-DDS format. Figure 7 shows the
first page of the resulting define.xml.

Note that a SASReferences dataset is required; see above section about validation of SDTM data.

**EXTENDED EXAMPLE**

```sas
/*-- root location of the process input and output --*/
%let studyRootPath=C:\projects\PhUSE\demo4;

/*-- load basic configuration to macro variables --*/
%cst_setStandardProperties(
  _cstStandard=CST-FRAMEWORK,
  _cstSubType=initialize
);
%cst_setStandardProperties(
  _cstStandard=CDISC-CRTDDS,
  _cstStandardVersion=1.0,
  _cstSubType=initialize
);
%cst_setStandardProperties(
  _cstStandard=CDISC-TERMINOLOGY,
  _cstSubType=initialize
);

/*-- create SASReferences dataset --*/
%cst_createds(
  _cstStandard=CST-FRAMEWORK,
  _cstType=control,
  _cstSubType=reference,
  _cstOutputDS=work.sasrefs
);

proc sql;
  insert into work.sasrefs (standard, standardversion, type, subtype, sasref, reftype, order, path, memname)
  values (
    "CST-FRAMEWORK", "1.2", "messages", "", "messages", "libref", 1, "", ""
  ,
    "CDISC-CRTDDS", "1.0", "messages", "", "crtmsg", "libref", 2, "", ""
  ,
    "CDISC-CRTDDS", "1.0", "autocall", "", "auto1", "fileref", 1, "", ""
  ,
    "CDISC-CRTDDS", "1.0", "fmtsearch", "", "crtfmt", "libref", 1, "", ""
  ,
    "CDISC-CRTDDS", "1.0", "referencemetadata", "table", "refmeta", "libref", , "", ""
  ,
    "CDISC-CRTDDS", "1.0", "referencemetadata", "column", "refmeta", "libref", , "", ""
  ,
    "CDISC-CRTDDS", "1.0", "control", "validation", "work", "libref", , %sysfunc(pathname(work)), "validation.sas7bdat"
  ,
    "CDISC-CRTDDS", "1.0", "referencecontrol", "validation", "refcntl", "libref", , &cstgroot.standards.cdisc-crtdds-1.0\validation\control\validation_master.sas7bdat"
  ,
    "CDISC-CRTDDS", "1.0", "sourcemetadata", "table", "srcmeta", "libref", , (&studyrootpath/metadata\crtdds_tables)
  ,
    "CDISC-CRTDDS", "1.0", "sourcemetadata", "column", "srcmeta", "libref", , (&studyrootpath/metadata\crtdds_columns)
  ,
    "CDISC-CRTDDS", "1.0", "sourcedata", "", "srcdata", "libref", , (&studyRootPath/crtddsdata)
  ,
    "CDISC-CRTDDS", "1.0", "referencexml", "stylesheet", "xslt01", "fileref", , (&studyRootPath/results\define1-0-2.xsl)
  ,
    "CDISC-CRTDDS", "1.0", "externalxml", "xml", "extxml", "fileref", , (&studyRootPath/results\define.xml)
  ,
    "CDISC-CRTDDS", "1.0", "results", "validationresults", "results", "libref", , (&studyRootPath/results\results)
  ,
    "CDISC-TERMINOLOGY", "200810", "fmtsearch", "", ""
  );
```

13
"ctfmt" , "libref" , 2, "" , ");
quit;

/*-- process sasreferences: allocate librefs etc.--*/
%let _cstSASRefs=work.sasrefs;
%cstutil_allocatesasreferences;

/*-- create all 39 CRT-DDS data sets --*/
%cst_createTablesForDataStandard(
   _cstStandard=CDISC-CRTDDS,
   _cstOutputLibrary=srcdata
)

/*-- fill 9 of the 39 tables --*/
libname stdymeta "&studyRootPath/studymeta";
%crtdds_sdtm311todefine10(
   _cstOutLib = srcdata,
   _cstSourceTables  = stdymeta.source_tables,
   _cstSourceColumns = stdymeta.source_columns,
   _cstSourceStudy   = stdymeta.source_study
);

/*-- Add information about archive locations --*/
proc sql;
   update srcdata.itemgroupdefs
      set archivelocationid = 'ALID'!!oid;
   insert into srcdata.itemgroupleaf (id, href, fk_itemgroupdefs)
      select 'ALID'!!i.oid, s.xmlpath, i.oid
         from stdymeta.source_tables s join srcdata.itemgroupdefs i on
            s.table=i.name;
   delete from srcdata.itemgroupleaf where id=' ';
   insert into srcdata.itemgroupleaftitles (fk_itemgroupleaf, title)
      select 'ALID'!!i.oid, s.xmltitle
         from stdymeta.source_tables s join srcdata.itemgroupdefs i on
            s.table=i.name;
   delete from srcdata.itemgroupleaftitles where fk_itemgroupleaf=' ';
quit;
libname stdymeta;

/*-- select validation checks and validate CRT-DDS datasets --*/
data work.validation;
   set refcntl.validation_master;
   where lowcase(tablescope) contains ('itemgroupdefs')
      or lowcase(tablescope) contains ('itemgroupleaf')
      or lowcase(tablescope) contains ('itemgroupleaftitles');
run;
%crtdds_validate;

/*-- generate define.xml --*/
%crtdds_write(
   _cstCreateDisplayStyleSheet=1,
   _cstResultsOverrideDS=results.results
);

The following enhancements have been made to the above code:

- The SASReferences dataset is being created explicitly as part of the program.
- The record with type referencexml and subtype stylesheet references has been added in order to use an alternate stylesheet.
- The call to cst_createTablesForDataStandard creates all the 39 tables of the intermediate CRT-DDS data model.
There is a section which adds information about the location of the transport files to the tables itemgroupdef, itemgroupleaf and itemgroupleaf titles. The call to crtdds_write writes this information to the column “Location” of the table on the top of the define.xml, see Figure 8.

The intermediate CRT-DDS data model is being validated. The process is basically the same as for the SDTM validation explained above. The validation checks relevant for the three modified tables are being selected and the crtdds_validate macro runs the checks and writes results.

In order to be able to determine how to fill in additional information to the CRT-DDS data model, the following process has to be followed:

- Look at the CDISC "Case Report Tabulation Data Definition Specification" (see recommended reading below) and determine which (sub-)elements and attributes have to be supplied to address the metadata in question.
- Follow the section about the CRT-DDS data model in the toolkit user’s guide in order to identify the data sets and columns of interest and to sort out how tables have to be linked together by foreign keys.
- Write a program which fills the data sets accordingly.

Of course one has to occupy with the details of the CRT-DDS specification and the CRT-DDS data model, which are beyond the scope of this tutorial.

Of course one has to occupy with the details of the CRT-DDS specification and the CRT-DDS data model, which are beyond the scope of this tutorial.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
<th>Class</th>
<th>Structure</th>
<th>Purpose</th>
<th>Keys</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>Adverse Events</td>
<td>Events</td>
<td>One record per event per subject</td>
<td>Tabulation</td>
<td>STUDYID USUBJID AETERM ABSTDDTC</td>
<td>Adverse Events SAS transport file</td>
</tr>
<tr>
<td>CM</td>
<td>Concomitant Medications</td>
<td>Interventions</td>
<td>One record per medication intervention episode per subject</td>
<td>Tabulation</td>
<td>STUDYID USUBJID CMTRT CMSTDDTC</td>
<td>Concomitant Medications SAS transport file</td>
</tr>
<tr>
<td>CO</td>
<td>Comments</td>
<td>Special Purpose</td>
<td>One record per comment per subject</td>
<td>Tabulation</td>
<td>STUDYID USUBJID COSEQ</td>
<td>Comments SAS transport file</td>
</tr>
<tr>
<td>DM</td>
<td>Demographics</td>
<td>Special Purpose</td>
<td>One record per subject</td>
<td>Tabulation</td>
<td>STUDYID USUBJID</td>
<td>Demographics SAS transport file</td>
</tr>
<tr>
<td>DS</td>
<td>Disposition</td>
<td>Events</td>
<td>One record per disposition status or protocol milestone per subject</td>
<td>Tabulation</td>
<td>STUDYID USUBJID DSTDTC</td>
<td>Disposition SAS transport file</td>
</tr>
<tr>
<td>DV</td>
<td>Protocol Deviations</td>
<td>Events</td>
<td>One record per protocol deviation per subject</td>
<td>Tabulation</td>
<td>STUDYID USUBJID DVSEQ</td>
<td>Protocol Deviations SAS transport file</td>
</tr>
<tr>
<td>EG</td>
<td>ECG Test Results</td>
<td>Findings</td>
<td>One record per ECG observation per time point per risk per subject</td>
<td>Tabulation</td>
<td>STUDYID USUBJID EGTESTCD VISITNUM</td>
<td>ECG Test Results SAS transport file</td>
</tr>
</tbody>
</table>

Figure 8: First page of resulting define.xml with links to archives

FURTHER STEPS
This tutorial focuses on programming aspects of the Clinical Standards Toolkit. The following topics must also be considered:

- Administration of standards: installing new versions of standards (SDTM 3.1.2), modification of existing standards, bringing in of new domains and development of company specific (variants of) standards.
- Implementation and processes: There will be different kinds of toolkit users, users who administer metadata and standards and users who merely use them. Who has which responsibilities? Which users need which access rights? Who needs which kind of training?
- Validation: Procedures for the installation qualification are included. There are no complete operational qualification procedures (e.g. test scenarios for every validation check) and risk assessment is necessary in order to determine required work in this area. Process validation has to be done as always.

CONCLUSION
With this tutorial we have tried to make clear the basic architecture of the Clinical Standards Toolkit and to show some basic and some more advanced usage scenarios of the Clinical Standards Toolkit.

If you want to try out the above examples by yourself, send an e-mail to the authors and request the sample data and programs.
REFERENCES
All links to the internet were valid on October 13th 2010.


RECOMMENDED READING


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

Nicole Wächter, Andreas Mangold
HMS Analytical Software GmbH
Rohrbacher Strasse 26
69115 Heidelberg
Work Phone: +49 (6221) 6051-0
Fax: +49 (6221) 6051-99
Email: nicole.waechter@analytical-software.de
Web: www.analytical-software.de

Brand and product names are trademarks of their respective companies.