PharmaSUG China 2016 - Paper 24

Building Your Own CDISC Application with SAS® Clinical Standards Toolkit

Jing Gao, SAS R&D, Beijing, China

ABSTRACT

The SAS® Clinical Standards Toolkit (CST) is a framework that supports clinical research activities and focuses on the CDISC standards. The CST provides validation checks from WedSDM, OpenCDISC and SAS on the CDISC data, especially SDTM and ADaM, as well as the generation of XML-Based Standards files, and validation of XML files against an XML schema.

A nice plus is that the CST is a separately orderable component that is available at no additional charge to currently licensed SAS customers. However, the CST is a set of SAS macros, metadata and configuration files, interacting with it requires a technical background of Base SAS and the SAS macro language. So can we make the using of the CST more intuitive and easier? Of course, one solution is to add an easy to use graphical user interface (GUI) for the CST and invoke the CST macros that are provided as open source and are accessible to the user.

Compared with processing clinical data, it is simple to create a GUI. Many programming languages, such as Java and Python, are well suited for the GUI development. As we know, Python is an easy to learn, use and powerful programming language, thus in this paper, I will introduce how to use Python to build a CDISC GUI application based on the CST.

INTRODUCTION

The SAS Clinical Standards Toolkit provides support of multiple CDISC standards, including SDTM (3.1.2, 3.1.3, and 3.2), CRT-DDS (reading and creating define 1.0 XML files), Define-XML 2.0 (reading and creating define 2.0 XML files), Dataset-XML (creating Dataset-XML files from SAS data sets and creating SAS data sets from Dataset-XML files), ODM (reading and creating 1.3.0 and 1.3.1 XML files), ADaM 2.1, CDASH 1.1, SEND 3.0, and validating XML files against an XML schema file.

As mentioned above, the CST supports multiple CDISC standards. In this paper, focus will be on the validation checks of clinical data based on SDTM standard, and CDISC Dataset-XML creation from SAS data sets.

- CDISC SDTM Compliance Checks
- CDISC Dataset-XML Creation

Through these examples, this paper will try to give the user:

- An idea of how to get started with building a CDISC GUI application based on the CST
- Some of key points and useful tips

Then the user can build their GUI applications based on the CST for their own needs.

If you want to get the full benefit of the CST, SAS has a powerful product SAS Clinical Data Integration that provides an easy-to-use visual interface for transforming, managing and verifying the clinical research data and metadata.

OVERVIEW OF THE CDISC GUI APPLICATION BASED ON THE CST

The following screen shot is the sample GUI application that I created for implementing the CDISC standards with two available tools. The menu at the top provides quick access to the tools, configuration and help resource. The tool of compliance checks against SDTM data displays by default.

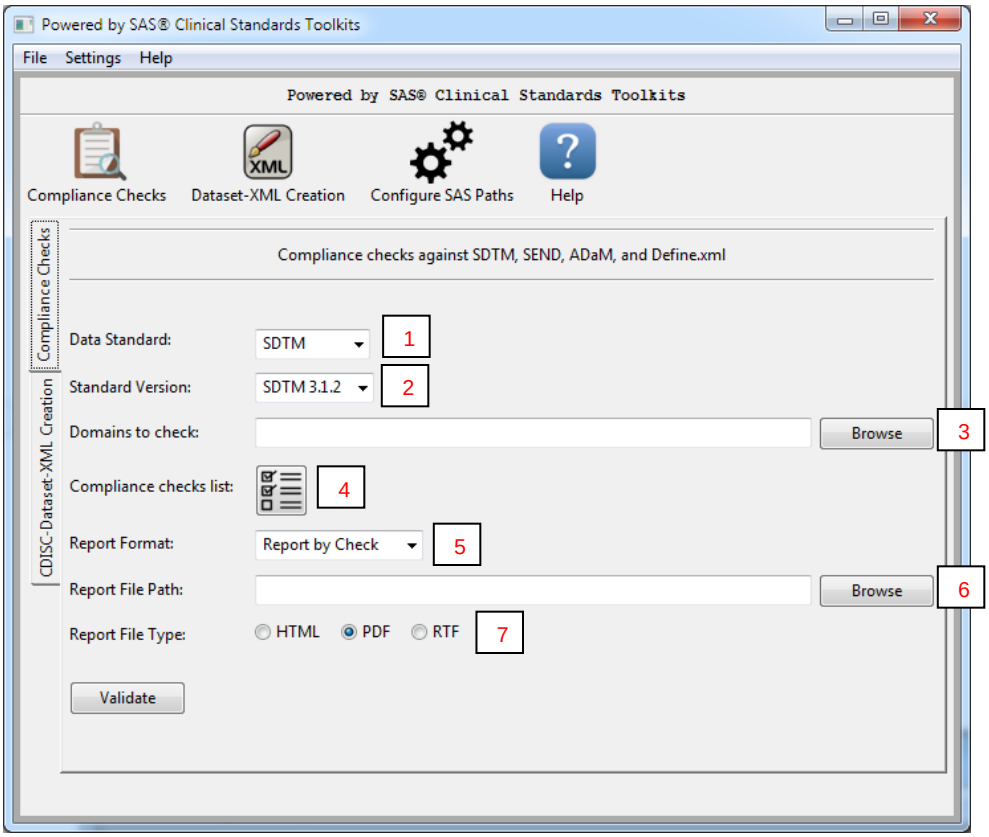


Figure 1. Sample-CDISC GUI application

CONFIGURE SAS PATHS

Before run this application, must specify the SAS installation directory and the CST global standards library directory in the following dialog. Then the GUI application get to know where the SAS Foundation and the CST are installed. In the background of the application, it invokes the CST macros that need to run on the SAS Foundation.

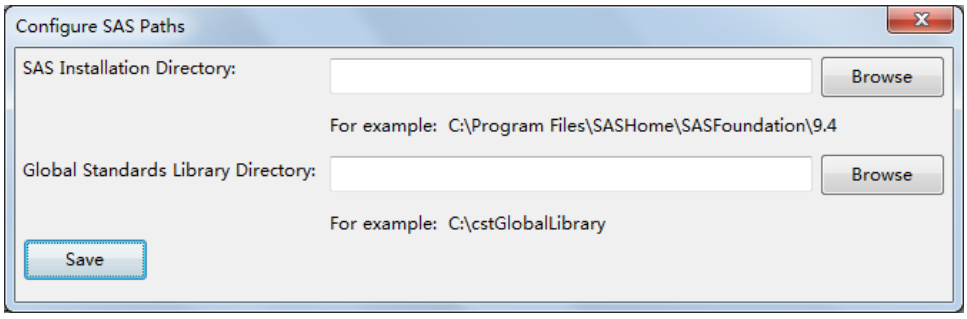


Figure 2. Configure SAS Paths

CDISC SDTM COMPLIANCE CHECKS

Validation Framework Overview

The SAS Clinical Standards Toolkit validation assesses the compliance of data, and the metadata describing the data, with an accepted reference standard. It assesses the consistency of values in a specific column, between columns, across records in a specific data set, and across data sets. The primary output is a results data set that itemizes the process findings, and an optional metrics data set that summarizes the results.

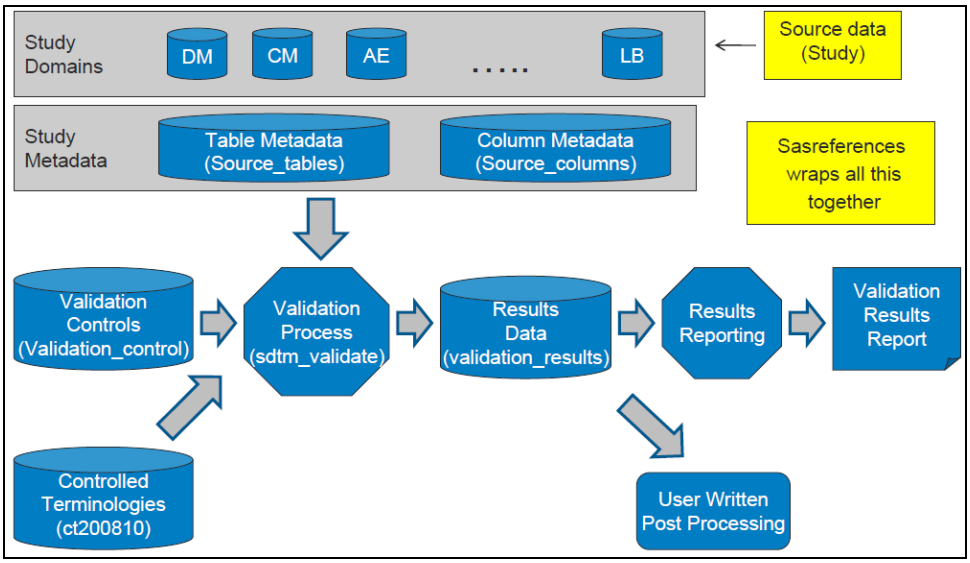


Figure 3. Components of the CST Validation Process

Running Validation checks from GUI

This tool assesses whether the metadata and data of the selected domains comply with the SDTM standard. The CST provides validation checks in support of CDISC SDTM 3.1.2, 3.1.3, and 3.2. These checks are derived from multiple sources that have evolved over time. Most checks in the CST are based on SAS data management and cleaning experiences building CDISC SDTM domains. Each version of the CDISC SDTM validation contains a different number of checks based on the rules that are in effect at the time of each version and the number and type of supported tabulation domains.

To run the validation checks:

1: As mentioned above, the CST supports compliance checks on clinical data with CDISC standards: SDTM, SEND, ADaM and define.xml. This paper will focus on SDTM data validation.

2: Select the standard version of SDTM, such as SDTM 3.1.2, 3.1.3, and 3.2.

3: Specifying the folder that contains data sets to be checked here.

4: It will show us a list of all available compliance checks that will be performed on the data and metadata. The checks list will change along with the choices of data standard and version.

5: Two report formats are available: ‘Report by Check’ and ‘Report by Domain’. The validation results can be generated by checks and by domains in the report.

6: Specifying the path in which the report will be generated.

7: The validation report can be generated in HTML, PDF and RTF.

When these options are set then the validation report can be generated by clicking the “Validate” button.

Validation Report

Example of validation report by checks

SAS Clinical Standards Toolkit 1.7
CDISC-SDTM 3.1.2 VALIDATION

Process Results, CheckID: SDTM0205

Description: Column value is not left-justified
Check scope: (Tables) _ALL_, (Columns) _ALL_
Source: SAS (SAS0012)
Validation check macro: cstcheck_column, using source metadata

Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
1	1	CSTCHECK_ COLUMN	CST0003	lib2016.AE could not be found	Warning: Check not run	Not Run		
1	2	lib2016.DM	CST0100	No errors detected in lib2016.DM	Info	No		

Figure 4. Validation Report by Checks

Example of validation report by domains

SAS Clinical Standards Toolkit 1.7
CDISC-SDTM 3.1.2 VALIDATION

Process Results, Table: AE

Check ID	Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
SDTM0205	1	1	CSTCHECK_ COLUMN	CST0003	lib2016.AE could not be found	Warning: Check not run	Not Run		
SDTM0606	1	1	CSTCHECK_ VIOLATESSTD	CST0003	lib2016.AE could not be found	Warning: Check not run	Not Run		

Figure 5. Validation Report by Domains

CDISC DATASET-XML CREATION

This tool creates a separate Dataset-XML file for each domain in the selected folder. Each Dataset-XML file is named based on the domain name. For example, the ae.xml file will be created for the AE domain.

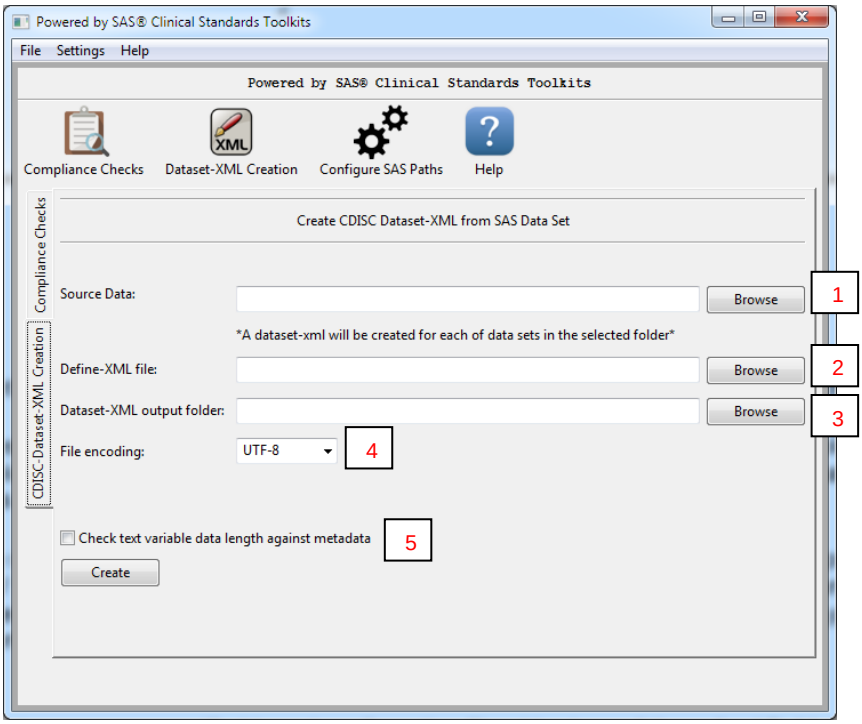


Figure 6. Dataset-XML Creation

To run the tool to create Dataset-XML from SAS data set.

1: Select the datasets for which to create Dataset-XML files.

2: A Define-XML file should be specified here. The Define-XML file contains metadata about all SAS data sets for which to create Dataset-XML files.

3: The Dataset-XML files will be created in the output folder.

4: Select a file encoding for the Dataset-XML file: UTF-8, Shift_JIS, ISO-8859-1 and US-ASCII can be selected.

5: Check the data lengths of text variables against the metadata in the define.xml input file.

Click Create button, then a Dataset-XML file will be created for each data sets in the selected folder.

SAS® CLINICAL STANDARDS TOOLKITS

ORDERING THE TOOLKIT

SAS Clinical Standards Toolkit 1.7 is supported with SAS 9.4 on the following operating environments:

- Windows x64
- Linux x64

The toolkit is a separately orderable component that is available at no additional charge to currently licensed SAS customers. Contact your SAS Account Representative to request the toolkit to be added to your Base SAS order.

Toolkit Version	Availability
1.7	Available with SAS 9.4 (TS1M2) or later
1.6	Available with SAS 9.4 (TS1M1) or later Available with SAS 9.3 (TS1M2) or later
1.5	Available with SAS 9.3 (TS1M2) or later
1.4	Available with SAS 9.3 for releases prior to TS1M2
1.3	Available only on SAS 9.2
1.2	Available only on SAS 9.1.3 Windows.

SUPPORT FOR CDISC STANDARDS

At present, version 1.7 of the SAS Clinical Standards Toolkit is the most current version. The SAS Clinical Standards Toolkit 1.7 provides supports for the following CDISC standards:

- SDTM 3.1.2, 3.1.3 and 3.2
- ADaM 2.1
- CRT-DDS 1.0 (define.xml 1.0)
- Define-XML 2.0
- Dataset-XML 1.0
- ODM 1.3.0 and 1.3.1
- CDASH 1.1
- SEND 3.0
- NCI CDISC Controlled Terminology

FRAMEWORK MODULE

The SAS Clinical Standards Toolkit files can be aggregated into three primary groups:

- Global Standards Library

- Sample Library
- Framework macros

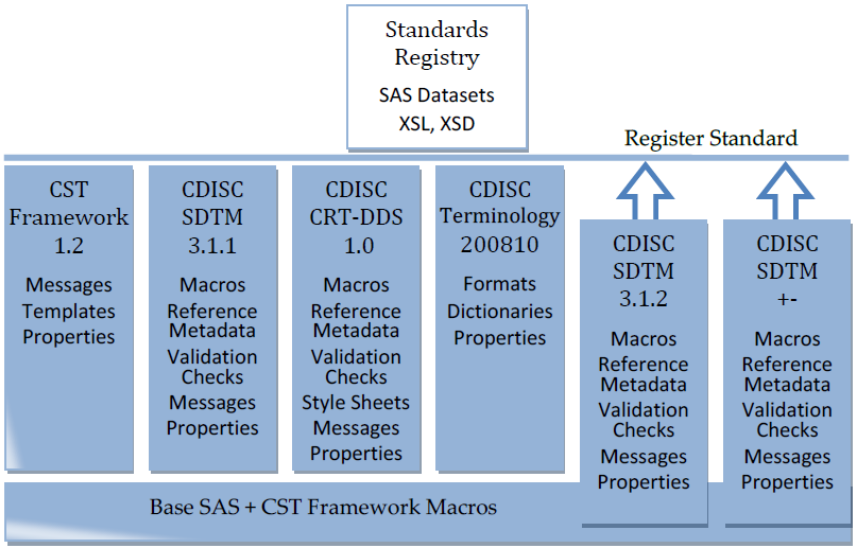


Figure 7. Architecture of the CST components

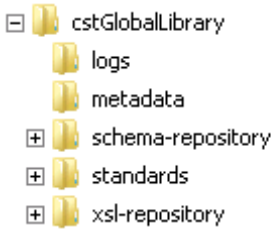
Figure 7 shows an overview of the components of the SAS Clinical Standards Toolkit.

The Global Standards Library

The global standards library is the metadata repository for the SAS Clinical Standards Toolkit. By default, the global standards library contains the metadata for the Framework module and the metadata for each data standard that is provided with the SAS Clinical Standards Toolkit.

During the installation and configuration of the SAS Clinical Standards Toolkit, you are prompted for the location where the global standards library should be installed. By default, on Microsoft Windows, this location is set to the c:\cstGlobalLibrary directory.

Here are the top-level subfolders of the global standards library:

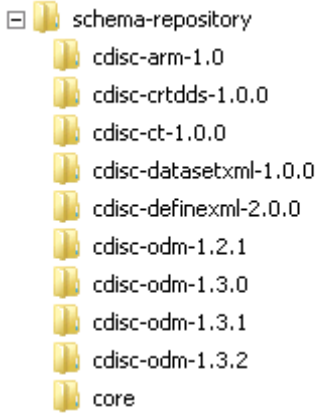


logs contains one data set: transactionlog. The data set contains metadata update information from all users.

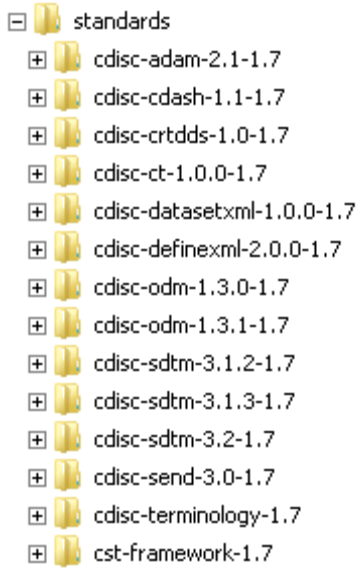
metadata contains three data sets and one XML file: Standards, Standardlookup, StandardSASReferences, and availabletransforms.xml.

- The Standards data set has a list of the registered standards and basic information relating to each standard.
- The StandardSASReferences data set defines the typical inputs and outputs of SAS processes that are associated with each standard.
- The Standardlookup data set contains discrete lookup values specific to a SAS Clinical Standards Toolkit registered standard. It provides specific information for column values and data set template names. In addition, this data set is used to perform internal validation of the SAS Clinical Standards Toolkit.
- The availabletransforms.xml file is for XML-based standards. It defines the location of the XML schema, the location of the XSL transformation style sheets, and the import and export locations of XML documents.

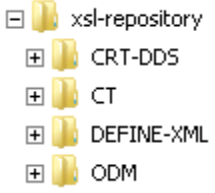
schema-repository contains XML schema definitions that are used to validate XML files. Standards that use XML should have their schemas in this directory so that they can be found.



standards contains subdirectories for each of the standard versions that is provided with the SAS Clinical Standards Toolkit.

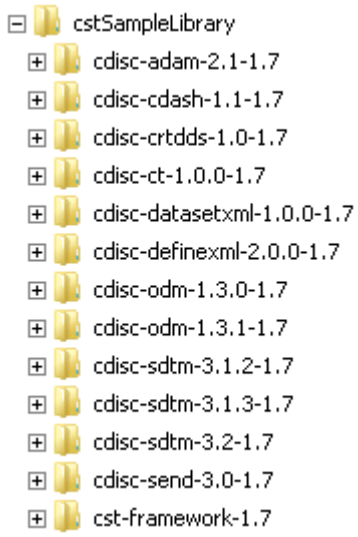


xsl-repository contains files that are used to transform XML files from one format to another.



The Sample Library

In the SAS Clinical Standards Toolkit sample library, there is a sample folder for each standard. A sample folder contains the files that represent a sample SAS implementation of the specific standard, sample study data sets, and sample programs and results that illustrate the SAS Clinical Standards Toolkit functionality.



Framework Macros

The framework enhances Base SAS by the framework macros, which provide auxiliary functions such as adding new standards or generating files from metadata.

The components that are installed as part of the SAS Foundation. The SAS Clinical Standards Toolkit framework macros can be found here for SAS 9.4:

- Microsoft Windows

  !SASROOT\cstframework\sasmacro, where !SASROOT is C:\Program Files\SASHome\SASFoundation\9.4

- UNIX

  !SASROOT/sasautos, where !SASROOT is /usr/local/SASHome/SASFoundation/9.4/

Macros for specific standards can be found in the global standards library directory/standards/standard/macros directory.

RUN THE CST INTERACTIVELY

Each SAS Clinical Standards Toolkit process uses a SAS driver program to processes parameters and executes the appropriate SAS macros. These driver programs show how to perform the necessary setup tasks for SAS Clinical Standards Toolkit processes, and how to reference and use sample data that is provided with the SAS Clinical Standards Toolkit. The users can follow the process flow of the driver program to create their driver program according to their own needs, which requires a technical background of Base SAS and the SAS macro language. To run the CST driver program interactively, start a SAS session, and load the driver program into the SAS editor, then submit the SAS code.

INTRODUCTION TO PYTHON AND PYTHON GUI PROGRAMMING TOOLKITS

The GUI of the CDISC sample application is implementing with Python, so in this section, I will introduce Python and its related GUI toolkits.

WHAT IS PYTHON?

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable clear

programs on both a small and large scale. Python is a programming language that lets you work more quickly and integrate your systems more effectively. Python can be easy to pick up whether you're a first time programmer or you're experienced with other languages.

PYTHON GUI PROGRAMMING TOOLKITS

Python provides various options for developing graphical user interfaces (GUIs). Most common used are listed below.

- Tkinter: Tkinter is the standard GUI library for Python. It provides a fast and easy way to create GUI applications.
- wxPython: wxPython is a GUI toolkit for Python. It provides a large variety of window types and controls, all implemented with a native look and feel on the platforms upon which it is supported.
- PyQt: PyQt is a cross-platform GUI toolkit for Python. It is able to create powerful and flexible GUI applications.

Tkinter is used for small applications with simple interface. For big applications, PyQt might be a better choice. It has a very good screen designer, but it requires a commercial license if it is used for commercial purposes. Finally, wxPython may be a good alternative, because it doesn't have these drawbacks.

wxPython

wxPython is a GUI toolkit for the Python programming language. It allows Python programmers to create programs with a robust, highly functional graphical user interface, simply and easily. It is implemented as a Python extension module (native code), since the language is Python, wxPython programs are simple, easy to write and easy to understand. Like Python, wxPython is Open Source which means that it is free for anyone to use.

wxPython is a cross-platform toolkit. This means that the same program will run on multiple platforms without modification. Currently supported platforms are 32-bit Microsoft Windows, most Unix or unix-like systems, and Macintosh OS X.

wxFormBuilder

wxFormBuilder is an open source GUI designer application, which allows creating corss-platform applications. A streamlined, easy to use interface enables faster development and easier maintenance of software.

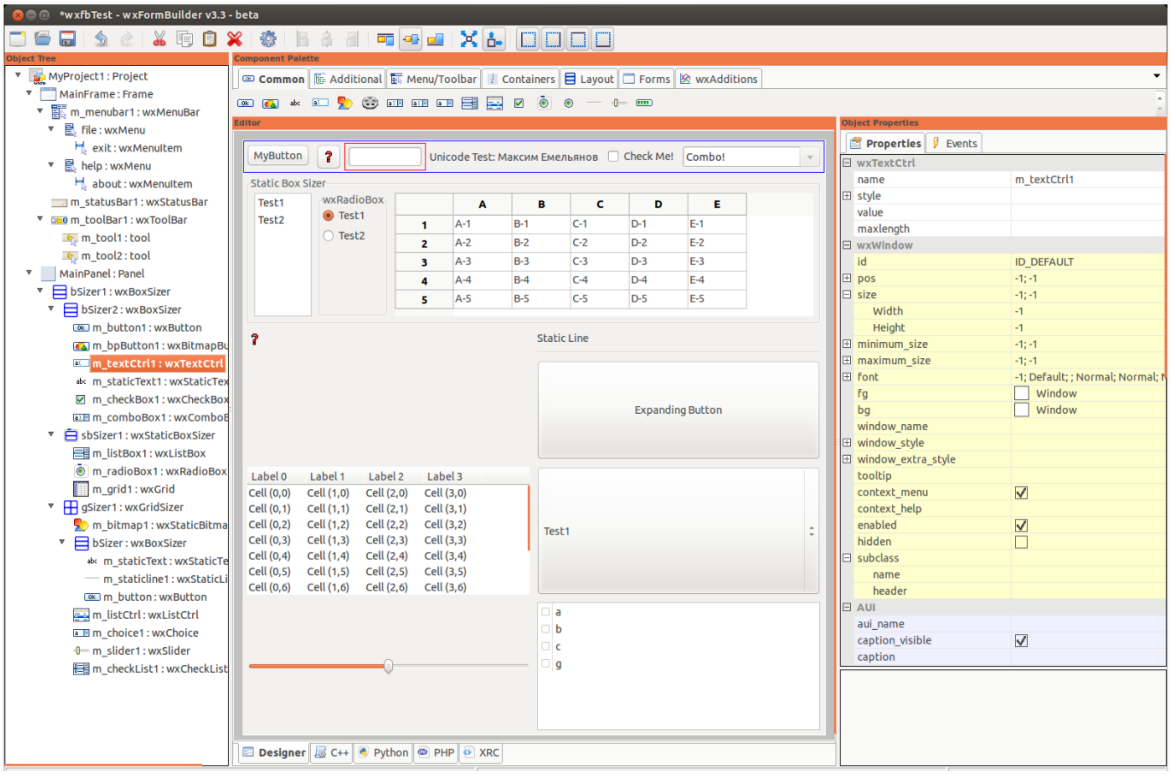


Figure 8. GUI of wxFormBuilder

HOW TO RUN SAS PROGRAM FROM PYTHON

In this section, I will introduce how Python interacts with SAS.

INVOKING SAS IN PYTHON

The followings are the Python codes to invoke a SAS program:

```
# import the subprocess module for spawning new processes

import subprocess

# execute a child program (SAS program) in a new process

subprocess.Popen(['C:\\Program Files\SASHome\SASFoundation\9.4\sas.exe', '-sysin',
'C:\\cstSampleLibrary\cdisc-datasetxml-1.0.0-1.7\programs\create_datasetxml.sas'])
```

We can also use another way to invoke SAS program:

```
# specify the path of SAS Foundation

sasexe_path="C:\\Program Files\SASHome\SASFoundation\9.4\sas.exe"

# specify the SAS program that will be invoked

sas_code="C:\\cstSampleLibrary\cdisc-datasetxml-1.0.0-
1.7\programs\create_datasetxml.sas"

# execute a child program (SAS program) in a new process

subprocess.Popen(args='"%s" -sysin "%s"'%(sasexe_path, sas_code))
```

HOW TO PASS PARAMETERS TO A SAS PROGRAM FROM PYTHON

The followings codes show how to send parameters into the SAS program by specifying the parameters when you run SAS from Python.

PART I: An example SAS program

The %SYSGET function gets the values that you specified by using the -SET option. Returns the value of the specified operating environment variable.

```
/* get values of environment variables that are set up in Python */

%let name=%sysget(userName);

%let id=%sysget(userId);

%put Name is &name;

%put Id is &id;
```

PART II: An example Python program

You can use the –SET option in Python to specify the parameter value and its corresponding parameter. Suppose that the SAS program above is stored in passParameters.sas.

```
# specify the paths of SAS Foundation

sasexe_path="C:\\Program Files\SASHome\SASFoundation\9.4\sas.exe"

# specify the example SAS program

sas_code="C:\\passParameters.sas"

# specify the parameter values

nameValue="Jing"

idValue="123"

# invoke the SAS program and pass the parameter values to the SAS program

subprocess.Popen(args='"%s" -sysin "%s" -set userName "%s" -set userId "%s"'%

(sasexe_path, sas_code, nameValue, idValue))
```

HOW TO INVOKE THE CST MACROS

Take the function as example: create Dataset-XML files from SAS data sets.

PART I: SAS PROGRAM

STEP 1: Set up macro variables

```
* Get the location of the Dataset-XML macros *;
%let DatasetXMLRoot=%sysget(DatasetXMLMacrosPath);
* Get Root paths for input and output *;
%let SourceDatasetPath=%sysget(SourceDatasetPath);
%let DatasetXMLOutputPath=%sysget(DatasetXMLOutputPath);
%let DefineXMLPath=%sysget(DefineXMLPath);
* Get the output encoding *;
%let FileEncoding=%sysget(FileEncoding);
/* Define libname statements for SAS data sets, define-xml file and output location */
libname srcdata  "&SourceDatasetPath";
libname xmldata  "&DatasetXMLOutputPath";
filename srcmeta "&DefineXMLPath";
```

STEP 2: Run the SDTM Dataset-XML creation macro

```
%datasetxml_write(
  _cstSourceLibrary=srcdata,
  _cstOutputLibrary=xmldata,
  _cstSourceMetadataDefineFileRef=srcmeta,
  _cstOutputEncoding=&FileEncoding,
  _cstCheckLengths=Y,
  _cstIndent=Y,
  _cstZip=N,
  _cstDeleteAfterZip=N
  );
```

PART II: PYTHON PROGRAM

```
# specify the path of SAS Foundation
SASEXEPATH="C:\\Program Files\SASHome\SASFoundation\9.4\sas.exe"
# specify the path of SAS program that will be invoked by Python
sascode="C:\\CST\demo\sascode\create_datasetxml.sas"
# specify the path of the SAS macro to create Dataset-XML files
DatasetXMLMacrosPath="C:\\cstGlobalLibrary\standards\cdisc-datasetxml-1.0.0-1.7"
# specify the parameters
arg1="DatasetXMLMacrosPath"  # Parameter for the path of Dataset-XML macro
arg2="SourceDatasetPath"     # Parameter for the path of source datasets
arg3="DefineXMLPath"         # Parameter for the path of Define-XML
arg4="DatasetXMLOutputPath"  # Parameter for the output folder of Dataset-XML files
arg5="FileEncoding"          # Parameter for the file encoding of Dataset-XML files
```

```
# invoke the SAS program to create Dataset-XML files from SAS data sets
self.childProcess=subprocess.Popen(args='"%s" -sysin "%s"
-set %s "%s" -set %s "%s" -set %s "%s" -set %s "%s" -set %s "%s"
-NOTERMINAL -NOSPLASH -NOSTATUSWIN -NOICON'%
(SASEXEPATH, sascode,
arg1, DatasetXMLMacrosPath,
arg2, self.SourceDatasetPath,
arg3, self.DefineXMLPath,
arg4, self.DatasetXMLOutputPath,
arg5, self.file_encoding))
```

BUILD A WEB APPLICATION WITH THE CST

This paper mainly focus on introducing how to build a desktop application based on the CST. Actually, the CST is not only for desktop application, but also works well with Web application. The way to invoke the CST macros is the same as the desktop application. The only difference is that the desktop application is Client/Server structure, and the web application is Browser/Server structure. SAS has integrated CST with a web application SAS Drug Development (SDD) with a new name Life Science Analysis Framework (LSAF).

INTEGRATED WITH YOUR OWN SAS PROGRAMS

Validation of the SDTM data and generation of Dataset-XML can also be done easily with other tools, for example, OpenCDISC. But to implement the whole process within SAS has its benefits: From the beginning of the clinical data process to the end, SAS is the only central software product, so the environment is a closed system. If you have your own SAS programs for processing the clinical data, you can integrate them together with the CST, and manage them in the same one GUI application.

INTEGRATED WITH SAS DATA INTEGRATION STUDIO

If you don't intend to develop a CDISC application by yourself, SAS also provides a powerful product SAS Clinical Data Integration (CDI) that provides an easy-to-use visual interface for transforming, managing and verifying the clinical research data and metadata. You can get the full benefit of the CST by the CDI.

SAS Clinical Data Integration relies on SAS Data Integration Studio to provide centralized metadata management using the SAS Metadata Server and the tools that it provides to visually transform data.

SAS Clinical Data Integration enhances its usability by adding new metadata types, plug-ins, and wizards. These enhancements help you perform clinically oriented tasks such as importing data standards, creating studies and submissions, and adding specialized transformations for mapping clinical data into a standard data model.

SAS Clinical Data Integration leverages the SAS Clinical Standards Toolkit to provide validation and conformance checking. SAS Clinical Data Integration enables you to accomplish these goals:

- Improve the consistency of submissions and studies
- Improve the long-term management and growth of data
- Use data standards effectively
- Use a centralized SAS Metadata Server
- Use the powerful and user-friendly features of SAS Data Integration Studio to manage metadata, generate and execute SAS Clinical Standards Toolkit code, and visualize the results.

CONCLUSION

The CST is a powerful and flexible toolkit supporting multiple CDISC standards. There is no additional charge to licensed SAS users. The CST macros are provided as open source and are accessible to the users, which allows users to customize the CST to meet their needs. As introduced in this paper, you can build a CDISC application based on the CST for your own needs. Maybe there are a lot of work to do at the beginning. After having done this, it is easy to use the CST.

REFERENCES

https://www.python.org/

https://wxpython.org/

https://sourceforge.net/projects/wxformbuilder/

SAS Clinical Standards Toolkit 1.7: User's Guide

http://support.sas.com/documentation

SAS Clinical Standards Toolkit - Macro API

http://support.sas.com/documentation/onlinedoc/clinical/1.7/cst1.7-macro-api/index.html

SAS Clinical Data Integration 2.6: User's Guide

http://support.sas.com/documentation

New Dataset-XML Standard v1.0

http://www.cdisc.org/dataset-xml

The SAS Clinical Standards Toolkit - A helpful tool for creating CDISC SDTM compliant study data

http://www.lexjansen.com/phuse/2015/pp/PP14.pdf

Using the SAS Clinical Standards Toolkit to Validate CDISC SDTM Data

http://www.lexjansen.com/pharmasug/2010/SAS/SAS-CD-SAS01.pdf

ACKNOWLEDGMENTS

I would like to thank all of my colleagues who reviewed this paper and gave me valuable comments.

Special thanks to Han Liu for support and guide for this paper.

CONTACT INFORMATION

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

Name: Jing Gao
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) 83193355-3462
Fax: (8610) 6310-9130
E-mail: Jing.gao@sas.com
Web: www.sas.com