PhamaSUG 2014 - Paper DS21-SAS

Round Trip Ticket – Using the Define.xml file to Send and Receive your Study Specifications

Julie Maddox, SAS Institute Inc., Cary, NC, USA

ABSTRACT

Why can't you always get want you want? All you need to do is ask for it! Now that CDISC data standards have matured, why not provide your project leads and external partners with the exact study specifications you require all wrapped up in a single CRT-DDS define.xml file. Code lists, domains, variable, computational algorithms, and value-level metadata information are all captured in the CRT-DDS define.xml file. SAS® Clinical Standards Toolkit 1.5 provides a variety of macros to extract this study information from the define file and produce domain templates, format catalogs and metadata tables ready to be populated. SAS® Drug Development provides a secure, web-based interface to a SAS programming environment with access to the SAS Clinical Standards Toolkit as well as the full power of the SAS analytics procedures and reporting features. SAS Drug Development also provides a secure repository to handle your organization's customized CDISC data standards and manage the study data in a compliant manner. This paper discusses how to round trip from a SDTM data standard, to a CRT-DDS define file and back out to a SDTM study using SAS Clinical Standards Toolkit within the SAS Drug Development SAS execution environment. Topics include:

- managing CDISC standards
- creating a CRT-DDS define file including controlled terminology, value-level metadata and computational algorithms
- extracting metadata from a CRT-DDS define file
- extracting code lists and domain templates from a CRT-DDS define file

This paper describes how to leverage the capabilities of SAS Drug Development with SAS Clinical Standards Toolkit to easily manage clinical data following industry standards. You can effectively and efficiently provide your project team or partners with precise data that is compliant with CDISC standards.
You can get what you want!

INTRODUCTION

The Case Report Tabulation Data Definition[1] specification, or define.xml, provides a machine-readable detailed description of the contents of each dataset in a submission. The metadata included in the define.xml can be processed programmatically to create the submission datasets, the code lists, computational algorithms, value-level metadata, and links to accompanying documentation. Study Administrators can use the define.xml to communicate their specifications of the datasets to the project leads. Once the data has been collected and mapped into the designated domains, a resulting define.xml can be created and sent back to the study administrator. The results of the outgoing and incoming define. xml files can be compared and discrepancies resolved.

This paper describes how to programmatically create a define.xml file and how to extract the metadata from a define.xml file. Also described are several SAS tools which facilitate the metadata gathering process and the programming techniques to produce the define.xml file.

CREATING A DEFINE.XML FILE

In order to create a define.xml file you must first identify and collect the metadata required for the study. Much of this information can be obtained from the statistical analysis plan. This metadata for a study includes:

- identifying which CDISC standard will be used
- identifying the domains/datasets that will be created
- selecting which code lists will be used
- determining the computational algorithms
- identifying value-level metadata

In general, when representing an XML-based standard in SAS, an XML element is mapped to a SAS data set and its associated attributes are mapped to the columns of the SAS data set. When the SAS Clinical Standards Toolkit creates a define.xml file, it converts the information from a SAS data set representation of the CRT-DDS model into XML. The SAS representation of the CRT-DDS standard can be derived in part from other standards (such as CDISC SDTM) and can include supporting metadata from other sources. The first step in creating a define.xml file using the SAS Clinical Standards Toolkit is creating the datasets which describe the source data in your study. The source_tables, source_columns, source_values, and source_documents datasets describe the SDTM domains while the source_study dataset describes metadata about the study itself. Execute the crtdds_sdtmtodefine macro which reads the metadata from the source tables and produces SAS datasets which represent the CRT-DDS standard. Once you have populated the metadata for your study into these tables, the define.xml file can be created using the crtdds_write macro.[2]

SAS® Clinical Data Integration provides a common metadata model for representing data standards that are based on CDISC models. The SAS Clinical Data Integration user interface allows the study administrator to easily create the necessary metadata for the define.xml file. The metadata managed by SAS Clinical Data Integration can be published to a define file using the CDISC-SOURCE to CRT-DDS transformation. This transformation extracts metadata on the domains and study, and passes it on to the SAS Clinical Standards Toolkit for define.xml creation (Figure 1). In addition, you can customize the define.xml file with value-level metadata and supplemental documentation (see reference 3 for an extensive discussion on this topic). [3]

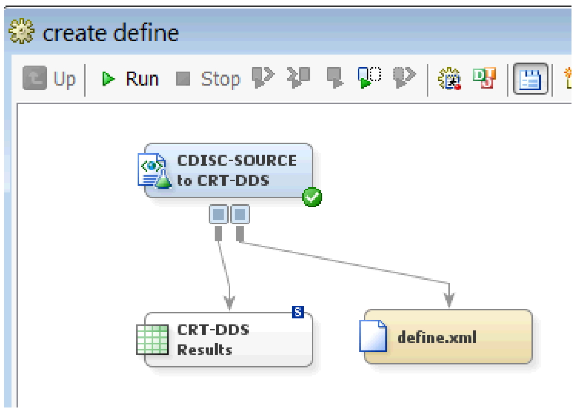


Figure 1 – SAS Clinical Data Integration CRT-DDS Define.xml Job

READING THE DEFINE.XML FILE

The metadata in a define.xml file can be programmatically processed to create the datasets and metadata for a study. We can use SAS Clinical Data Integration or SAS Clinical Standards Toolkit to read and process the metadata stored in the define.xml file.

READING THE DEFINE.XML METADATA USING SAS CLINICAL DATA INTEGRATION

In SAS Clinical Data Integration, you can select a define. xml file you are interested in and import the study definition, domains, analysis datasets and codelists directly into the SAS Clinical Data Integration component. Figure 2 below shows the results of importing a define file containing SDTM domains.

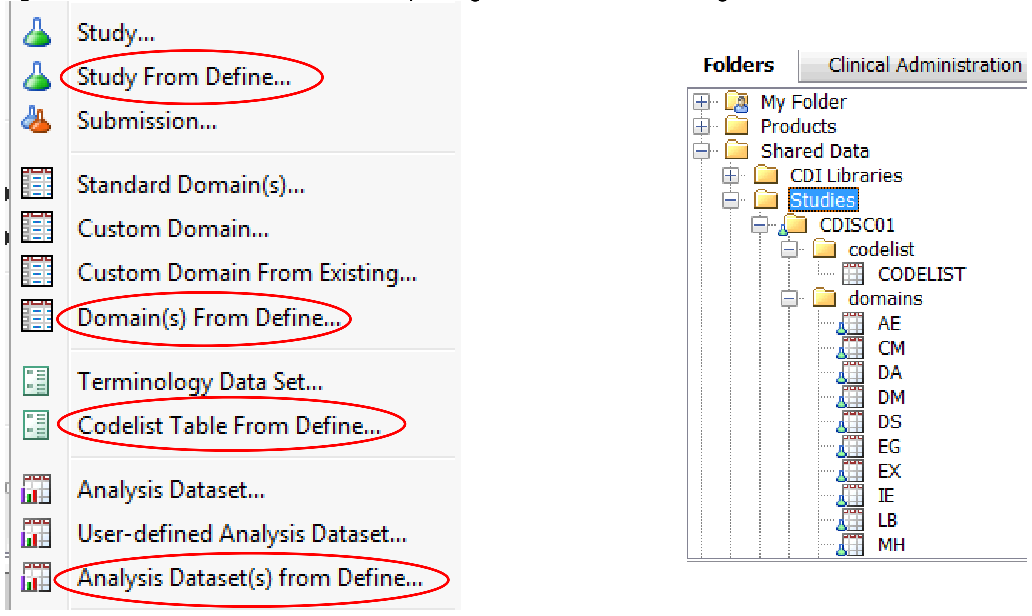
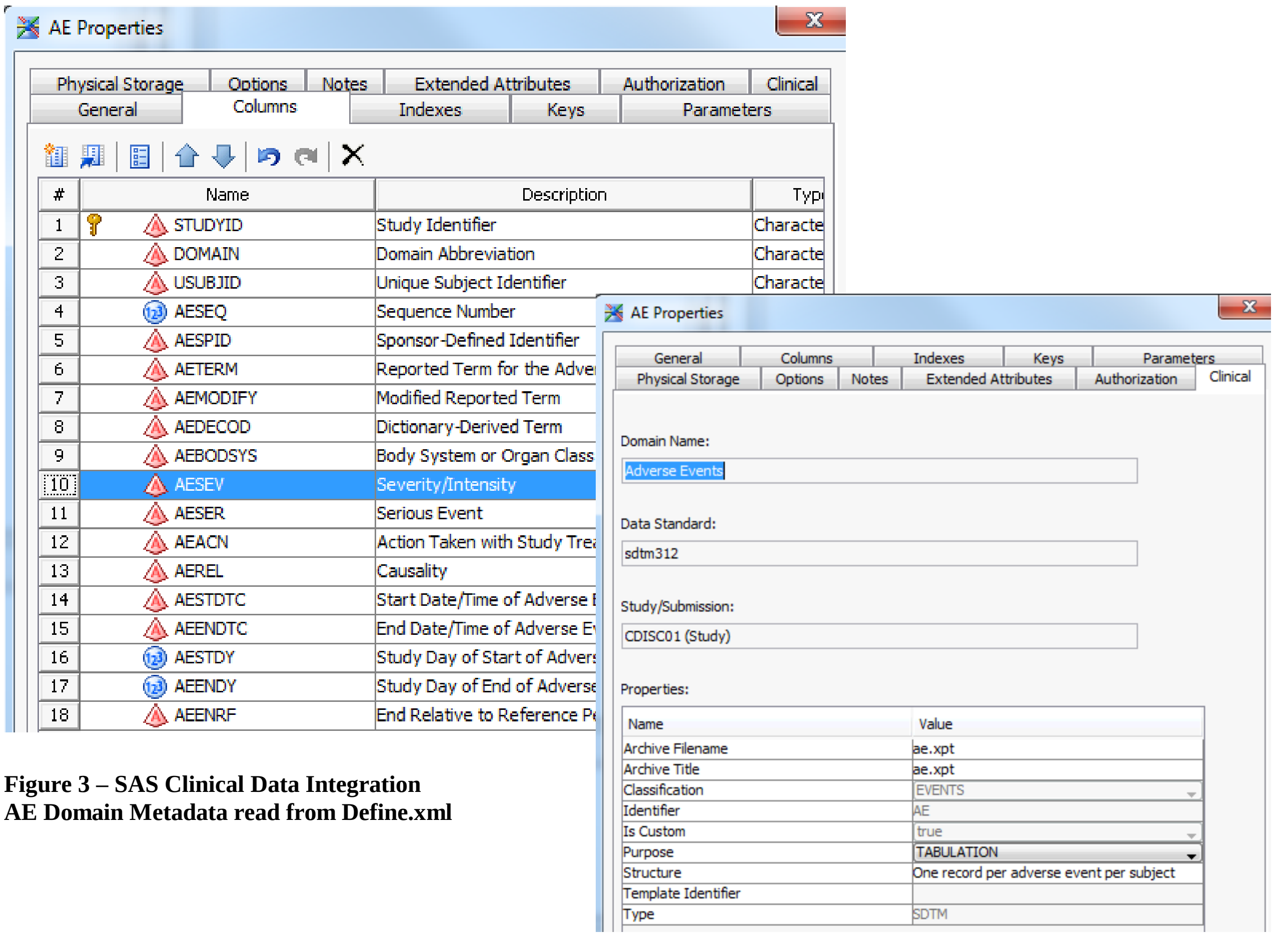


Figure 2 – SAS Clinical Data Integration Reading define.xml

Domain metadata is captured at the table and column level. Figure 3 shows the table metadata imported from the define.xml file for the AE domain.


AE Properties

Physical Storage | Options | Notes | Extended Attributes | Authorization | Clinical
General | Columns | Indexes | Keys | Parameters

#	Name	Description	Typ
1	STUDYID	Study Identifier	Characte
2	DOMAIN	Domain Abbreviation	Characte
3	USUBJID	Unique Subject Identifier	Characte
4	AESEQ	Sequence Number	
5	AESPID	Sponsor-Defined Identifier	
6	AETERM	Reported Term for the Adve	
7	AEMODIFY	Modified Reported Term	
8	AEDECOD	Dictionary-Derived Term	
9	AEBODSYS	Body System or Organ Class	
10	AESEV	Severity/Intensity	
11	AESER	Serious Event	
12	AEACN	Action Taken with Study Trea	
13	AEREL	Causality	
14	AESTDTC	Start Date/Time of Adverse I	
15	AEENDTC	End Date/Time of Adverse Ev	
16	AESTDY	Study Day of Start of Advers	
17	AEENDY	Study Day of End of Adverse	
18	AEENRF	End Relative to Reference Pe	

AE Properties

General | Columns | Indexes | Keys | Parameters
Physical Storage | Options | Notes | Extended Attributes | Authorization | Clinical

Domain Name: Adverse Events

Data Standard: sdtm312

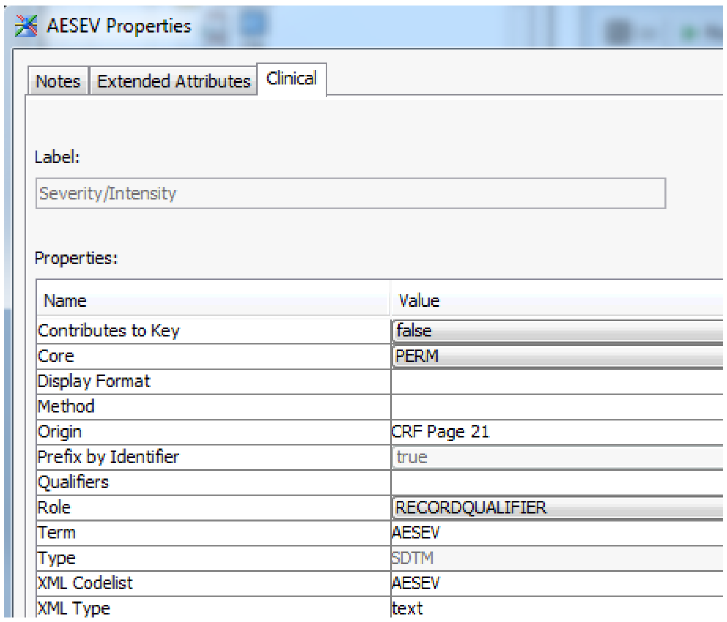
Study/Submission: CDISC01 (Study)

Properties:

Name	Value
Archive Filename	ae.xpt
Archive Title	ae.xpt
Classification	EVENTS
Identifier	AE
Is Custom	true
Purpose	TABULATION
Structure	One record per adverse event per subject
Template Identifier	
Type	SDTM

Figure 3 – SAS Clinical Data Integration
AE Domain Metadata read from Define.xml

For the domain columns, additional metadata is read from the define.xml file, including the origin, codelist, and computational method. Figure 4 displays the column clinical metadata read from the define.xml file for the AESEV domain.



Label: Severity/Intensity

Properties:

Name	Value
Contributes to Key	false
Core	PERM
Display Format	
Method	
Origin	CRF Page 21
Prefix by Identifier	true
Qualifiers	
Role	RECORDQUALIFIER
Term	AESEV
Type	SDTM
XML Codelist	AESEV
XML Type	text

Figure 4 – SAS Clinical Data Integration Domain Column Metadata read from Define.xml

Once the study and domain metadata is defined, the CRO and/or data management team can use the SAS Clinical Data Integration tools to visually transform the source data into the domains.

READING THE DEFINE.XML METADATA USING SAS CLINICAL STANDARDS TOOLKIT

We can also use the SAS Clinical Standards Toolkit macros to process the metadata in a define.xml file. The macro *crtdds_read* is available to read a define.xml file and translate the metadata into a SAS representation (data sets) of the CDISC CRT-DDS model. Once the CRT-DDS data sets are created, you can use the following SAS Clinical Standards Toolkit macros to further process the data:

sdtmutil_createsrcmetafromcrtdds
This macro will create the SOURCE_STUDY, SOURCE_TABLES, SOURCE_COLUMNS, SOURCE_DOCUMENTS, SOURCE_VALUES datasets from the CRT-DDS data sets. These tables define the metadata of the tables and columns derived from a define.xml file.

cstutilcreatetablesfrommetadata
This macro will create zero observation data sets representing the domains listed in the SOURCE_TABLES and SOURCE_COLUMNS datasets.

sdtmutil_createformatsfromcrtdds
This macro will create format catalogs from the CRT-DDS code list datasets (codelists, codelistitms, clitemdecodetranslatedtext)

SAS Clinical Standards Toolkit supplies several sample programs demonstrating the ability to read and write the CRT-DDS define.xml files. Refer to the SAS® Clinical Standards Toolkit 1.5: User's Guide for information on executing the sample programs.[2]

COMPARING STUDY DEFINITIONS USING THE CRT-DDS DEFINE.XML FILE

Once your data management team or CRO has completed their work they can generate a define.xml file which describes the domains and metadata as implemented. The study administrator can use SAS Clinical Standards Toolkit to read in the implemented define.xml file and compare the artifacts to the original distributed define.xml file artifacts. This will allow the sponsor and CRO to identify and discuss differences up front.

To compare the sponsor distributed define.xml with the one returned by the CRO, perform the following steps:

- create separate output folders for the sponsor define.xml and the CRO define.xml
- execute the SAS Clinical Standards Toolkit crtdds_read macro to read in the define.xml file , once for the Sponsor file and again for the CRO file.
- execute the SAS Clinical Standards Toolkit sdtmutil_createsrcmetafromcrtdds macro to create the

metadata tables for the study , once for the Sponsor file and again for the CRO file.
- use Proc Compare to compare the metadata sets for the study, and report any differences.

Appendix A provides sample code for the entire process.

CONCLUSION

The CRT-DDS define.xml file contains a complete representation of the metadata used to create the study submission datasets. Code lists, domains, computational algorithms, and value-level metadata information are all captured in the CRT-DDS define.xml file. SAS Clinical Standards Toolkit provides a variety of macros to extract this study information from the define.xml file and produce domain templates, format catalogs and metadata tables ready to be populated. SAS Clinical Data Integration provides a convenient user interface to manage the metadata for a study. Using their respective define.xml files, the sponsor and CRO can easily exchange study metadata and compare results.

REFERENCES

1. Case Report Tabulation Data Definition Specification (define.xml), Version 1.0, February 9, 2005 (http://www.cdisc.org/define-xml)
2. SAS Institute Inc. 2011. SAS® Clinical Standards Toolkit 1.5: User's Guide . Cary, NC: SAS Institute Inc. (http://support.sas.com/documentation/onlinedoc/clinical/index.html
3. Define.xml - Tips and Techniques for Creating CRT – DDS, Julie Maddox and Mark Lambrecht, PhUSE 2010, Berlin, Germany

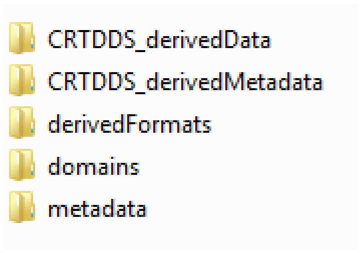
CONTACT INFORMATION

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

Julie Maddox
SAS Institute Inc.
SAS Campus Drive
Cary, NC, 27519, USA

APPENDIX 1 – SAMPLE CODE FOR COMPARING DEFINE.XML CONTENT

This sample code assumes you have created a folder named c:\phuse2013\studyRoot and created the following folders.


CRTDDS_derivedData
CRTDDS_derivedMetadata
derivedFormats
domains
metadata

It is also assumed that the define.xml file is located in the c:\phuse2013\studyRoot folder.
The sample code is broken down into 3 sections below. These programs should be run in the following order:

1. Reading in the define.xml file from the Sponsor (write to the Sponsor study root)
2. Creating the source metadata from the Sponsor SAS CRT-DDS datasets(write out to the Sponsor study root)
3. Reading in the define.xml file from the CRO (write to a separate CRO study root)
4. Creating the source metadata from the CRO SAS CRT-DDS datasets(write out to the CRO study root)
5. Comparing the results of the define files (output in the SAS Results viewer)

READING IN THE DEFINE.XML FILE SAMPLE CODE

```
*******************************************************************************;
* create_sascrtdds_fromxml.sas                                                *;
*                                                                             *;
* Sample program to read a CDISC-CRTDDS (define.xml) file                     *;
*                                                                             *;
* The following statements may require information from the user             *;
*******************************************************************************;

%let _cstSrcStandard=CDISC-SDTM;
%let _cstSrcStandardVersion=3.1.2;
%let _cstDefineName=define;
%let _cstDefinePath=C:\phuse2013\studyroot;
%let studyOutputPath=c:\phuse2013\studyroot;
%let _cstVersion=1.6;

*******************************************************************************;
*  The code below should not need modification                                *;
*******************************************************************************;


%cst_setStandardProperties(_cstStandard=CST-FRAMEWORK,_cstSubType=initialize);
%cstutil_setcstsroot;
data _null_;
  call symput('cstMapPath',cats("&_cstSRoot","/cdisc-crtdds-1.0-&_cstVersion"));
run;
%let workPath=%sysfunc(pathname(work));

%let _cstSetupSrc=SASREFERENCES;
%let _cstStandard=CDISC-CRTDDS;
%let _cstStandardVersion=1.0;


%cst_createdsfromtemplate(_cstStandard=CST-FRAMEWORK,
_cstType=control,_cstSubType=reference, _cstOutputDS=work.sasreferences);
%let _cstSASRefs=work.sasreferences;

proc sql;
  insert into work.sasreferences
  values("CST-FRAMEWORK"  "1.2"                   "messages"          ""
"messages" "libref"  "input"  "dataset"  "N"  "" ""                          1
""                         "")
```

```
  values("&_cstStandard"  "&_cstStandardVersion"  "messages"          ""
"crtmsg"   "libref"  "input"  "dataset"  "N"  "" ""                          2
""                         "")
  values("&_cstStandard"  "&_cstStandardVersion"  "autocall"          ""
"crtcode"  "fileref" "input"  "folder"   "N"  "" ""                          1
""                         "")

  values("&_cstStandard"  "&_cstStandardVersion"  "sourcedata"        ""
"srcdata"  "libref"  "output" "folder"   "Y"  "" "&studyOutputPath/CRTDDS_derivedData"
. ""                         "")
  values("&_cstStandard"  "&_cstStandardVersion"  "sourcemetadata"    "study"
"srcmeta"  "libref"  "output" "dataset"  "Y"  ""
"&studyOutputPath/CRTDDS_derivedmetadata" . "source_study.sas7bdat"     "")
  values("&_cstStandard"  "&_cstStandardVersion"  "sourcemetadata"    "table"
"srcmeta"  "libref"  "output" "dataset"  "Y"  ""
"&studyOutputPath/CRTDDS_derivedmetadata" . "source_tables.sas7bdat"    "")
  values("&_cstStandard"  "&_cstStandardVersion"  "sourcemetadata"    "column"
"srcmeta"  "libref"  "output" "dataset"  "Y"  ""
"&studyOutputPath/CRTDDS_derivedmetadata" . "source_columns.sas7bdat"   "")
  values("&_cstStandard"  "&_cstStandardVersion"  "referencemetadata" "table"
"refmeta"  "libref"  "input"  "dataset"  "N"  "" ""                          .
""                          "")
  values("&_cstStandard"  "&_cstStandardVersion"  "referencemetadata" "column"
"refmeta"  "libref"  "input"  "dataset"  "N"  "" ""                          .
""                          "")
  values("&_cstStandard"  "&_cstStandardVersion"  "results"           "results"
"results"  "libref"  "output" "dataset"  "Y"  "" "&workPath"         .
"results.sas7bdat"     "")
  values("&_cstStandard"  "&_cstStandardVersion"  "externalxml"       "xml"
"crtxml"   "fileref" "input"  "file"     "N"  "" "&_cstDefinePath"         .
"&_cstDefineName..xml"      "")
  values("&_cstStandard"  "&_cstStandardVersion"  "referencexml"      "map"
"crtmap"   "fileref" "input"  "file"     "N"  "" "&cstMapPath/referencexml"      .
"define.map"                "")
  values("&_cstStandard"  "&_cstStandardVersion"  "properties"        "initialize"
"inprop"   "fileref" "input"  "file"     "N"  "" ""                          1
""                          "")
  ;
quit;

%let _cstReallocateSASRefs=1;
%include "&_cstGRoot/standards/cst-framework-
&_cstVersion/programs/resetautocallpath.sas";

%cstutil_processsetup();

%crtdds_read();

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

CREATING SOURCE METADATA FOR SAS CRT-DDS DATA SETS SAMPLE CODE

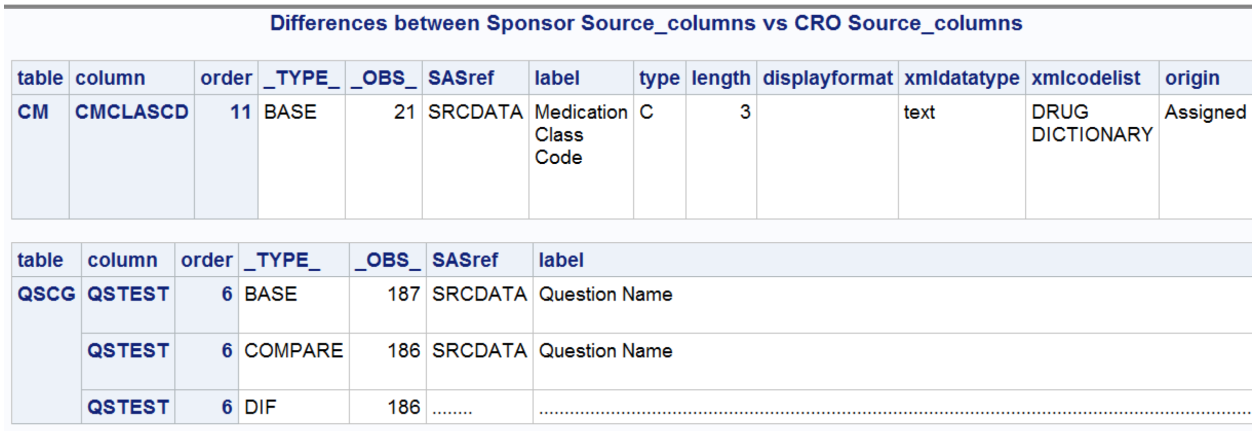
```
***********************************************************************************;
* create_sdtmSourceMetadata_from_sascrtdds.sas                                    *;
*                                                                                 *;
* Sample program to read the SAS data set representation of CDISC-CRTDDS          *;
* and produce metadata files source_study, source_tables, and source_columns      *;
*                                                                                 *;
* The following statements may require information from the user                 *;
***********************************************************************************;
%LET studyRootPath=C:\phuse2013\studyRoot;
%let _cstStandard=CDISC-SDTM;
%LET _cstStandardVersion=3.1.3;

***********************************************************************************;
*  The code below should not need modification                                    *;
***********************************************************************************;
%cstutil_setcstgroot;
%cst_setStandardProperties(_cstStandard=CST-FRAMEWORK,_cstSubType=initialize);

libname srcdata "&studyRootPath/CRTDDS_derivedData";
libname trgdata "&studyRootPath/domains";
libname trgmeta "&studyRootPath/metadata";

%let workPath=%sysfunc(pathname(work));
libname refmeta"&workpath";
%cst_createdsfromtemplate(_cstStandard=CDISC-SDTM,
_cstType=referencemetadata,_cstSubType=table, _cstOutputDS=work.reference_tables);
%cst_createdsfromtemplate(_cstStandard=CDISC-SDTM,
_cstType=referencemetadata,_cstSubType=column, _cstOutputDS=work.reference_columns);
%cst_createdsfromtemplate(_cstStandard=CST-FRAMEWORK,
_cstType=control,_cstSubType=reference, _cstOutputDS=work.sasreferences);
proc sql;
  insert into work.sasreferences
  values ("CST-FRAMEWORK" "1.2"                   "messages"           ""
"messages" "libref"  "input"  "dataset"  "N"  "" ""
1 ""                            "")
  values ("&_cstStandard" "&_cstStandardVersion"  "autocall"           ""
"sdtmauto" "fileref" "input"  "folder"   "N"  "" ""
1 ""                            "")
  values ("&_cstStandard" "&_cstStandardVersion"  "messages"           ""
"sdtmmsg"  "libref"  "input"  "dataset"  "N"  "" ""
2 ""                            "")
  values ("&_cstStandard" "&_cstStandardVersion"  "properties"         "initialize"
"inprop"   "fileref" "input"  "file"     "N"  "" ""
1 ""                            "")
  values ("&_cstStandard" "&_cstStandardVersion"  "results"            "results"
"results"  "libref"  "output" "dataset"  "N"  "" "&workPath"            .
"srcmeta_results.sas7bdat"   "")
  ;
quit;
%let _cstReallocateSASRefs=1;
%include "&_cstGRoot/standards/cst-framework-
&_cstVersion/programs/resetautocallpath.sas";
%cstutil_processsetup();
%let _cstClassColumnDS=class_columns;
%let _cstClassTableDS=class_tables;
%let _cstRefLib=refmeta;
%let _cstRefColumnDS=reference_columns;
%let _cstRefTableDS=reference_tables;
%let _cstCRTDataLib=srcdata;
%let _cstSDTMDataLib=srcdata;
%let _cstTrgMetaLibrary=trgmeta;
%let _cstTrgStudyDS=source_study;
%let _cstTrgDocumentDS=source_documents;
%let _cstTrgTableDS=source_tables;
%let _cstTrgColumnDS=source_columns;
%let _cstTrgValueDS=source_values;
%sdtmutil_createsrcmetafromcrtdds;
```

CREATING SOURCE METADATA FOR SAS CRT-DDS DATA SETS SAMPLE CODE

```
*********************************************************************************;
* compare_metadata_from__two_define_files.sas                                   *;
*                                                                               *;
* Sample program to compare metadata from two define.xml files                  *;
*                                                                               *;
* The following statements may require information from the user               *;
*********************************************************************************;
%let sponsorRoot=c:\phuse2013\studyRoot;
%let croRoot=c:\phuse2013\studyRootCRO;

*********************************************************************************;
*  The code below should not need modification                                  *;
*********************************************************************************;

libname sponsor "&sponsorRoot\metadata";
libname cro "&croRoot\metadata";

proc sort data=sponsor.source_tables out=sponsor_source_tables;
by table label class structure keys;
run;
proc sort data=cro.source_tables out=cro_source_tables;
by table label class structure keys;
run;
proc compare base=sponsor_source_tables compare=cro_source_tables out=result
outnoequal outbase outcomp outdif noprint;;
id table label class structure keys;
run;
title "Differences between Sponsor Source_tables vs CRO Source_tables";
proc print data=result;
by table ;
id table label class structure keys;
run;

proc sort data=sponsor.source_columns out=sponsor_source_columns;
by table column order;
run;
proc sort data=cro.source_columns out=cro_source_columns;
by table column order;
run;
proc compare base=sponsor_source_columns compare=cro_source_columns out=result
outnoequal outbase outcomp outdif noprint;
id table column order;
run;
title "Differences between Sponsor Source_columns vs CRO Source_columns";
proc print data=result;
by table  ;
id table column order;
run;

proc sort data=sponsor.source_values out=sponsor_source_values;
by table column value label origin type order;
run;
proc sort data=cro.source_values out=cro_source_values;
by table column value label origin type order;
run;
proc compare base=sponsor_source_values compare=cro_source_values out=result
outnoequal outbase outcomp outdif noprint;
id table column value label origin type order;
run;
title "Differences between Sponsor Source_values vs CRO Source_values";
proc print data=result;
id table column value label origin type order;
by table column value ;
run;

proc sort data=sponsor.source_documents out=sponsor_source_documents;
by doctype documentref href title;
run;
proc sort data=cro.source_documents out=cro_source_documents;
by doctype documentref href title;
```

```
run;
proc compare base=sponsor_source_documents compare=cro_source_documents out=result
outnoequal outbase outcomp outdif noprint;
id doctype documentref href title;
run;
title "Differences between Sponsor Source_documents vs CRO Source_documents";
proc print data=result;
id doctype documentref href title;
by doctype  ;
run;

proc sort data=sponsor.source_study out=sponsor_source_study;
by definedocumentname studyname studydescription protocolname;
run;
proc sort data=sponsor.source_study out=cro_source_study;
by definedocumentname studyname studydescription protocolname;
run;
proc compare base=sponsor.source_study compare=cro.source_study out=result outnoequal
outbase outcomp outdif noprint;
id definedocumentname studyname studydescription protocolname;
run;
title "Differences between Sponsor Source_study vs CRO Source_study";
proc print data=result;
id definedocumentname studyname studydescription protocolname;
by studyname  ;
run;
```

SAMPLE OUTPUT FROM REPORT



Differences between Sponsor Source_columns vs CRO Source_columns

table	column	order	_TYPE_	_OBS_	SASref	label	type	length	displayformat	xmldatatype	xmlcodelist	origin
CM	CMCLASCD	11	BASE	21	SRCDATA	Medication Class Code	C	3		text	DRUG DICTIONARY	Assigned

table	column	order	_TYPE_	_OBS_	SASref	label
QSCG	QSTEST	6	BASE	187	SRCDATA	Question Name
	QSTEST	6	COMPARE	186	SRCDATA	Question Name
	QSTEST	6	DIF	186	....	....