SAS®
Clinical Standards Toolkit 1.3
User's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2010. *SAS® Clinical Standards Toolkit 1.3: User's Guide* . Cary, NC: SAS Institute Inc.

SAS® Clinical Standards Toolkit 1.3: User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, November 2010

Contents

What's New

Overview

The following are some of the new capabilities in SAS Clinical Standards Toolkit 1.3:

- The CDISC SDTM 3.1.1 validation checks are updated to reflect WebSDM 3.0 updates revised June 29, 2009. There are 10 new validation checks, eight modified validation checks, and five deprecated validation checks in SAS Clinical Standards Toolkit 1.3.
- The SAS implementation of the CDISC SDTM 3.1.2 reference standard is published. This reference standard now includes 32 domains (with seven new domains). SDTM 3.1.2 domain validation includes the WebSDM 3.0 updates revised June 29, 2009, and applicable checks published by the open source community OpenCDISC.
- A new snapshot of the National Cancer Institute (NCI) Enterprise Vocabulary Services (EVS) Thesaurus controlled terminology for CDISC is captured as the CDISC-Terminology-201003 reference standard. This cumulative snapshot includes 60 distinct code lists, many of which support CDISC SDTM 3.1.2 validation.
- There is an initial implementation of the CDISC ODM 1.3.0 reference standard, which is targeted specifically at translating the (study) metadata and ClinicalData sections of an odm.xml file into the SAS representation of the CDISC ODM standard.
- Updated sample reports that offer more user-friendly views of SAS Clinical Standards Toolkit process results are provided. Comprehensive views of validation check metadata for those standards that support validation are available.
- Sample driver programs that streamline and standardize setup tasks associated with SAS Clinical Standards Toolkit processes are updated. These driver programs support 17 distinct tasks across the CDISC SDTM, CDISC CRT-DDS, and CDISC ODM models.
- The SAS Clinical Standards Toolkit metadata and code base are significantly updated. These changes are discussed in a following section.

Changes to Metadata and Code Base

Framework Changes

The following autocall macros are new. These macros can be found in the **`!sasroot/cstframework/sasmacro`** directory:

- **cstcheck_notimplemented** is a placeholder validation check macro that documents in the Results data set that a specific check has not yet been implemented in SAS Clinical Standards Toolkit.
- **cstcheckutil_formatlookup** is used in the CDISC SDTM 3.1.2 Validation Master data set in the check metadata codelogic field to evaluate value-level compliance with CDISC terminology.
- **cstutil_createmetadatareport** is a driver macro that generates report output based on all available check metadata information.
- **cstutil_createreport** is a driver macro that generates report output summarizing SAS Clinical Standards Toolkit process results.
- **cstutil_createunixsubdir** is used in sample SAS Clinical Standards Toolkit driver modules for UNIX users to create work subdirectories to support process write operations.
- **cstutil_processsetup** is used in sample SAS Clinical Standards Toolkit driver modules to perform necessary setup operations, including library and file allocations.
- **cstutil_reportgeneralprocess** is called by the cstutil_createreport macro to create the General Process Reporting panel.
- **cstutil_reportinputsoutputs** is called by the cstutil_createreport macro to create the Process Inputs/Outputs panel.
- **cstutil_reportprocessmetrics** is called by the cstutil_createreport macro to create the Process Metrics panel.
- **cstutil_reportprocessresults** is called by the cstutil_createreport macro to create the Process Results panel.
- **cstutil_reportprocesssummary** is called by the cstutil_createreport macro to create the Process Summary panel.
- **cstutil_reportsetup** interprets information from either a SAS Clinical Standards Toolkit Results data set or a SASReferences data set to perform setup functions for SAS Clinical Standards Toolkit reporting.
- **cstutil_reporttabledata** is called by the cstutil_reportprocessmetrics and cstutil_reportprocessresults macros to facilitate reporting by domain.
- **cstutil_saveresults** performs the common task of saving process results based on SASReferences data set settings.

The following autocall macros have been modified. These macros are located in the `!sasroot/cstframework/sasmacro` directory:

- cstcheck_metamismatch
- cstutil_getsasreference

The following framework properties have been modified. The properties can be found in ***<global standards library directory>*****/standards/cst-framework-1.3/programs/initialize.properties**:

- The _cstSASRefsLoc=, _cstSASRefsName=, and _cstSASRefs=work._cstsasrefs properties were moved from the CDISC SDTM initialize.properties file. The default value for _cstSASRefsLoc was changed to &workpath.
- The default value for the _cstFMTLibraries properties was changed from WORK to <blank>.

Eight type/subtype combinations were added and five type/subtype combinations were removed from the Standardlookup data set. The data set can be found in ***<global***

***standards library directory*>/standards/cst-framework-1.3/ control/standardlookup.sas7bdat**.

The CST0009, CST0024, and CST0025 messages have been added to the Messages data set.

The **productRevision** column has been added to standards.sas7bdat. The column is blank for standards installed with SAS Clinical Standards Toolkit 1.2. The column value is 1.3 for standards added with SAS Clinical Standards Toolkit 1.3 or when SAS Clinical Standards Toolkit 1.3 is allowed to overwrite any standard installed with SAS Clinical Standards Toolkit 1.2. It can be found in **<*global standards library directory*>/metadata/standards.sas7bdat**.

The codetype, uniqueid, and comment columns were added to the Validation Master data set. The checkno column was removed. The data set can be found in **<*global standards library directory*>/standards/cst-framework-1.3/templates/validation_ master.sas7bdat**.

The SAS Note 37164, "An error can occur if you run the macro %CSTUTIL_ALLOCATESASREFERENCES in SAS® Clinical Standards Toolkit more than once in the same SAS® session," is available on **http://support.sas.com**.

The SAS Note 38421, "SAS® Clinical Standards Toolkit folder is missing in SAS® 9.2," was added. The **<*global standards library directory*>/standards/ <*standard-version*>/programs/development** directory for each standard that is provided by SAS should no longer be available for any SAS version.

CDISC SDTM Changes

CDISC SDTM 3.1.2 is the default for new SAS Clinical Standards Toolkit 1.3 installations. For more information about upgrades from SAS Clinical Standards Toolkit 1.2, see Chapter 2, "Framework," on page 5.

The following CDISC SDTM 3.1.1 macros are new. These macros are also available with the CDISC SDTM 3.1.2 standard. These macros can be found in the **<*global standards library directory*>/standards/cdisc-sdtm-3.1.1-1.3/ macros** directory:

- **sdtmutil_createformatsfromcrtdds** is a utility macro that creates a SAS format catalog from CDISC CRT-DDS define.xml codelists.
- **sdtmutil_createsasdatafromxpt** is utility macro that creates SAS (SDTM) data sets from reachable CDISC CRT-DDS define.xml referenced SAS transport files.
- **sdtmutil_createsrcmetafromcrtdds** is a utility macro that creates SAS (SDTM) source metadata (study, table, column) from CDISC CRT-DDS define.xml metadata.
- **sdtmutil_createsrcmetafromsaslib** is utility macro that creates SAS (SDTM) source metadata (study, table, column) from a library of CDISC SDTM domains.

The CDISC SDTM 3.1.1 validation checks have been updated to reflect WebSDM 3.0 updates revised June 29, 2009. This includes 10 new, eight modified, and five deprecated validation checks. The modified checks have a UNIQUEID value that includes an effective date substring that is not equal to '2009-05-13'.

The SAS Note 36343, "SAS Clinical Standards Toolkit sample data set needs to be updated," has been added. This note provides corrected paths in the sample SASReferences data set.

The SAS Note 37853, "SAS Clinical Standards Toolkit validation master list contains an error," has been added. This note provides the corrected value for the

CHECKID=SDTM0651 record in the Validation Master data set. The column value for SOURCEID should be IR4140.

CDISC CRT-DDS Changes

The following CDISC CRT-DDS macros are new. These macros can be found in the **`<global standards library directory>/standards/cdisc-crtdds-1.0-1.3/macros`** directory:

- **crtdds_computationmethods** is called by the crtdds_sdtm311todefine10 macro to populate the CDISC CRT-DDS ComputationMethods data set from the CDISC SDTM column metadata.
- **crtdds_itemgroupleaf** is called by the crtdds_sdtm311todefine10 macro to populate the CDISC CRT-DDS ItemGroupLeaf data set from the CDISC SDTM table metadata.
- **crtdds_itemgroupleaftitles** is called by the crtdds_sdtm311todefine10 macro to populate the CDISC CRT-DDS ItemGroupLeafTitles data set from the CDISC SDTM table metadata.
- **crtdds_read** translates a define.xml file into the SAS representation of CDISC CRT-DDS 1.0. The representation included 39 tables, table metadata, and column metadata.
- **crtdds_buildchecktablelist** is a utility macro that identifies the domains to be validated by each check. The check is based on the contents of the tableScope and columnScope check metadata columns in the Validation Master data set.

CDISC CRT-DDS column length and label metadata have been updated. This metadata can be found in **`<global standards library directory>/standards/cdisc-crtdds-1.0-1.3/metadata/reference_columns.sas7bdat`**, **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/data`**, and **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/metadata/source_columns.sas7bdat`**. These metadata updates include:

- All *OID and FK* that reference *OIDs have been reset to a length of 128.
- Column labels have been added to some files.

SAS Note 38149, “%CRTDDS_XMLVALIDATE() macro might generate errors with logging levels above 'Info',” has been added.

CDISC-Terminology Changes

201003 is the default CDISC-Terminology standard version for new SAS Clinical Standards Toolkit 1.3 installations. For more information about upgrades from SAS Clinical Standards Toolkit 1.2, see Chapter 2, “Framework,” on page 5. This version can be found in the **`<global standards library directory>/metadata`** Standard data set. It fully supports CDISC SDTM 3.1.2.

CDISC SDTM, CDISC ODM, and CDISC CRT-DDS Changes

The use of cstutil_saveresults was added to standardize the persistence of process results to honor SASReferences type=results records. If there are no such records (by default), results are written to work._cstresults.

Changes between SAS Clinical Standards Toolkit 1.2 and SAS Clinical Standards Toolkit 1.3

Global Changes

The validation check data set structure was modified. The following are the new column details:

- **codetype** defines whether to use codelogic and what type of codelogic can be used in the validation code. Valid values include:
  - 0—No codelogic used.
  - 1—DATA step statement level. (For example, if &_cstColumn <0 then _cstError=1.)
  - 2—Full DATA step, PROC SQL step, or multiple steps.
  - 3—Calls a SAS macro or %include that can contain only DATA step statement level code. (For example, codetype=1.)
  - 4—Calls a SAS macro or %include that can contain only full DATA step or PROC SQL step code. (For example, codetype=2.)

  This column is required.
- **uniqueid** ensures uniqueness in the data set and in SAS Clinical Standards Toolkit. This column allows any shipped or derived check to be uniquely identifiable over time. This column is required. An example is SDTM000100CST120SDTM3112009-05-12T12:00:00CDI. In the legend, characters 1 through 8 are checkid, characters 9 through 10 are checkid repeat indicator, characters 11 through 16 are the SAS Clinical Standards Toolkit version, characters 17 through 23 are standard version, characters 24 through 42 are implementation datetime, and characters 43 through 48 are assigning authority.
- **comment** contains additional information for the check. This column is optional.
- **checkno** column was removed.
- All template, master (Validation Master), and run-specific (Validation Control) data sets for all standards that support validation have been changed to include or exclude these columns. For more information, see Table 6.3 on page 90.

The Results data set output for most primary SAS Clinical Standards Toolkit tasks was modified. Results data sets now include records that report on the following process metadata:

- PROCESS STANDARD (example: CDISC SDTM)
- PROCESS STANDARDVERSION (example: 3.1.2)

- PROCESS DRIVER (example: SDTM_VALIDATE)
- PROCESS DATE (example: 2010-07-16T13:25:11)
- PROCESS TYPE (example: VALIDATION)
- PROCESS SASREFERENCES (example: c:/…/_cstsasrefs.sas7bdat)
- PROCESS STUDYROOTPATH (example: c:/myStudy)
- PROCESS GLOBALLIBRARY (example: c:/cstGlobalLibrary)
- PROCESS CSTVERSION (example: 1.3)

Framework Changes

The following properties were modified:

- The installed location for properties is ***<global standards library directory>*****`/standards/cst-framework-1.3/programs/initialize.properties`**.
- You should reset the default value for _cstFMTLibraries from WORK to <blank>. For any SAS Clinical Standards Toolkit process, to include formats built in the SAS Work directory in the format search path, you should:
  1. Reset the property to WORK.<*catalog*>.
  2. Issue %let _cstFMTLibraries=WORK.<*catalog*> after the call to cst_setStandardProperties and before the call to cstutil_processsetup.
  3. Issue the OPTIONS FMTSEARCH statement to include WORK.<*catalog*> at the right point in the job stream.

The Standardlookup data set contents were modified.

- The installed location for the data set is ***<global standards library directory>*****`/standards/cst-framework-1.3/control/standardlookup.sas7bdat`**.
- This data set serves as the primary lookup to perform basic validation of each SASReferences data set. Any type and subtype values found in a SASReferences data set must be registered in the Standardlookup data set.
  - type=**referencecontrol**, subtype=**validation** These values point to the standard-specific Validation Master data set.
  - type=**referencexml**, subtype=**map** For XML-based standards, these values reference a SAS XML map file that interprets an interim XML file that was generated by SAS for any process reading from or writing to XML.
  - type=**report**
  - type=**report**, subtype=**library**
  - type=**report**, subtype=**outputfile** For reporting processes, subtype=library points to a destination library for report output, and subtype=outputfile points to a file location for a specific report output file.
  - type=**sourcemetadata**, subtype=**study** These values facilitate conversions to and from CDISC CRT-DDS define.xml files, which contain metadata information that is associated with the <Study> element.
  - type=**targetdata** This value points to an output library where derived or target data is to be written. For example, reading a define.xml file generates data sets comprising the SAS representation of CDISC CRT-DDS files.

  - type=**transport** This value points to a library of SAS transport files as referenced in CDISC CRT-DDS define.xml files.

The autocall macros were modified.

- The installed location for autocall macros is **`!sasroot/cstframework/sasmacro`**.
- **cstcheck_metamismatch**: In codeLogic, you can produce two possible data sets that are processed in the check macro. These data sets are work._cstnonmatch (mismatches that prevent assessment) and work._cstmatch (reportable problems found).
- **cstutil_getsasreference**: A new parameter is available: _cstAllowZeroObs. If set to 1, then it allows SASReferences to operate without warnings when a row that is requested is not found and returns zero observations. The default value is 0. This default value creates a warning when zero observations are encountered.

The Messages data set was modified. The following messages were added:

- **CST0009**: &_cstparm1 macro variable was not defined (error).
- **CST0024**: The column &_cstparm1 was not found in &_cstparm2. Compliance was not assessed (warning: check incomplete).
- **CST0025**: Data set was not found in reference standard. Compliance was not assessed (warning: check incomplete).

CDISC CRT-DDS Changes

A parameter was added to the crtdds_write macro.

- The installed location for the autocall macros is ***`<global standards library directory>`*****`/standards/cdisc-crtdds-1.0-1.3/macros`**.
- The new parameter is **_cstLogLevel**. This parameter is optional. It identifies the level of error reporting. Valid values are Info, Warning, Error, and Fatal Error.

Chapter 1
Introduction to SAS Clinical Standards Toolkit

How to Use This Document

The following list provides suggestions for using this document:

- For an introduction to the software, see Chapter 1, "Introduction to SAS Clinical Standards Toolkit," on page 1.
- For an overview of the Toolkit framework including how standards are defined, registered, and managed, see Chapter 2, "Framework," on page 5.
- For a summary of the Toolkit metadata supporting key framework functions and common tasks across multiple standards, see Chapter 3, "Metadata File Descriptions," on page 21.
- For an overview of the standards supported in Toolkit 1.3, see Chapter 4, "Supported Standards," on page 35.
- For a description of a key metadata file—SASReferences—which itemizes all inputs and outputs of a Toolkit process, see Chapter 5, "SASReferences File," on page 69.
- For information about one key feature of Toolkit 1.3, the validation of user metadata and data against a registered Toolkit standard, see Chapter 6, "Validation," on page 83.
- For information about another key feature of Toolkit 1.3, the creation of a CDISC CRT-DDS define.xml file, see Chapter 7, "XML-Based Standards," on page 165.
- For a list of all global macro variables, see Appendix A1, "Global Macro Variables," on page 211.
- For framework messages, see Appendix A2, "Framework Messages," on page 219.
- For the macro application programming interface (API) reference, see Appendix A3, "Macro Application Programming Interface," on page 225.
- For the CDISC SDTM 3.1.1 validation checks, see Appendix A4, "CDISC SDTM Validation Checks," on page 281.
- For the CDISC CRT-DDS 1.0 validation checks, see Appendix A5, "CDISC CRT-DDS 1.0 Validation Checks," on page 335.

What Is the SAS Clinical Standards Toolkit?

The purpose and scope of the SAS Clinical Standards Toolkit can best be described by considering the product name.

Clinical
SAS Clinical Standards Toolkit focuses primarily on supporting clinical research activities. These activities involve the discovery and development of new pharmaceutical and biotechnology products and medical devices. These activities occur from project initiation through product submission and throughout the full product lifecycle. They do not include non-research patient records or health-care, pharmacy, hospital, and insurance electronic records.

Standards
SAS Clinical Standards Toolkit initially focuses on standards defined by the Clinical Data Interchange Standards Consortium (CDISC). CDISC is a global, open, multidisciplinary, nonprofit organization that has established standards to support the acquisition, exchange, submission, and archival of clinical research data and metadata. The CDISC mission is to develop and support global, platform-independent data standards that enable information-system interoperability, which, in turn, improves medical research and related areas of health care. The Toolkit is not limited to supporting CDISC standards. In time, the SAS Clinical Standards Toolkit will support other evolving industry-standard data models. The Toolkit framework is designed to support the specification and use of any user-defined standard.

Toolkit
The term “toolkit” connotes a collection of tools, products, and solutions. SAS Clinical Standards Toolkit provides a set of standards and functionality that will evolve and grow with future product updates and releases. Customer requirements and expectations of Toolkit will play a key role in the deciding what functionality to provide in future releases.

References

Table 1.1 *References*

Reference	Web Address	Description
CDISC SDTM 3.1.1	`http://www.cdisc.org/content1605`	Provides access to *CDISC SDTM Implementation Guide V3.1.1 Final* and *CDISC Study Data Tabulation Model Version 1.1 Final*.
CDISC SDTM 3.1.2	`http://www.cdisc.org/extranet/index.php?a=1209`	Provides access to the *Study Data Tabulation Model, Version 1.2 (SDTM v1.2)* and the *SDTM Implementation Guide for Human Clinical Trials (SDTMIG v.3.1.2)*.

Reference	Web Address	Description
CDISC CRT-DDS 1.0	`http://www.cdisc.org/define-xml`	Provides access to *Case Report Tabulation Data Definition Specification (CRT-DDS, also called define.xml) Final Version 1.0*.
CDISC ODM 1.3.0	`http://www.cdisc.org/contentmgr/showdetails.php/id/2347`	Provides access to ODMFinal Version 1.3.0 files and documentation.
CDISC Controlled Terminology	`http://www.cancer.gov/cancertopics/terminologyresources/page6`	Provides access to an FTP directory of supported CDISC terminology. *Note:* The site **`http://evs.nci.nih.gov/ftp1/CDISC/SDTM/`** offers a current, cumulative set of terminology that supports CDISC SDTM.
CDISC ADaM 2.1	`http://www.cdisc.org/extranet/index.php?a=1067`	Provides access to the *Analysis Data Model Version 2.1* and the *Analysis Data Model Implementation Guide Version 1.0*.
Download Form for Validation Checks Performed by WebSDM Version 2.6 on SDTM Version 3.1.1 Data Sets	`https://www.phaseforward.com/resource/whitepapers/Validation%20Checks%202.6/default.aspx`	WebSDM Version 2.6 validation checks.
Download Form for Validation Checks Performed by WebSDM Version 3.0 on SDTM Version 3.1.2 Data Sets	`https://www.phaseforward.com/resource/whitepapers/Validation%20Checks%203.0/default.aspx`	WebSDM Version 3.0 validation checks.
OpenCDISC SDTM Validation Rules	`http://www.opencdisc.org/projects/validator/cdisc-validation-rules-repository`	OpenCDISC CDISC validation rules repository.
Janus Operational Pilot	`http://www.fda.gov/ForIndustry/DataStandards/StudyDataStandards/ucm155327.htm`	Provides information about operational pilots to date, including error checks.
ISO 8601:2004 Data Elements and Interchange Formats—Information Interchange—Representation of Dates and Times	`http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40874`	The official ISO 8601 standard.

Reference	Web Address	Description
SAS Knowledge Base for SAS Clinical Standards Toolkit	`http://support.sas.com/rnd/base/cdisc/cst/index.html`	Find current information and documentation about SAS Clinical Standards Toolkit.
SAS Clinical Standards Toolkit 1.3: User's Guide	`http://support.sas.com/documentation/onlinedoc/clinical/index.html`	Link to this document.
SAS Knowledge Base for SAS Clinical Standards Toolkit Samples and SAS Notes	`http://support.sas.com/notes/index.html`	Provides access to SAS installation problems, usage problems, and SAS Notes that are associated with SAS Clinical Standards Toolkit. (Enter “Clinical Standards Toolkit” in the search field.)
SAS and Clinical Trials Forum	`http://support.sas.com/forums/forum.jspa?forumID=9`	Primary public discussion forum for SAS Clinical Standards Toolkit.

Chapter 2
Framework

Overview

The Framework module of SAS Clinical Standards Toolkit enables you to manage the registration of standards, and provides the metadata and API infrastructure to interact with those standards.

To understand the Framework module, you must understand the fundamentals of how the files are structured and used. The Framework module has two distinct pieces:

- the components that are installed as part of the SAS Foundation and shared files (SAS macros, JAR files, and so on)
- the global standards library

The following sections describe the structure of the global standards library. The sections use some of the framework macros to show how the shared files are used.

Global Standards Library

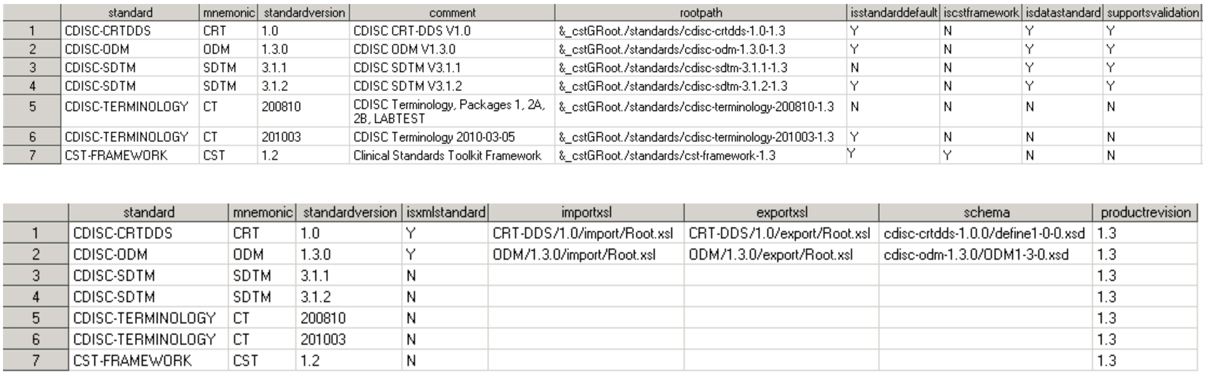
The global standards library is the metadata repository for the SAS Clinical Standards Toolkit. By default, the global standards library contains the metadata for the Framework module and the metadata for each data standard that is provided by SAS (such as the CDISC SDTM 3.1.2 standard).

During the installation and configuration of the SAS Clinical Standards Toolkit, the user is prompted for the location where the global standards library should be installed. The configuration process creates a series of directories in this location.

- **`metadata`** contains data sets that have information about the registered standards. For more information, see "Common Framework Metadata" on page 8.
- **`schema-repository`** contains the schemas for XML-based standards that are supported.
- **`standards`** contains a standard-specific directory hierarchy for each of the supported standards.
- **`xsl-repository`** contains directories and XSL files used in reading and writing XML files.

The **`metadata`** directory contains two data sets—Standards and StandardSASReferences. The Standards data set has a list of the registered standards and basic information relating to each standard. The following display provides the full content of the global standards library Standards data set included with the SAS Clinical Standards Toolkit.

Display 2.1 *Global Standards Library: Metadata Standards Data Set*



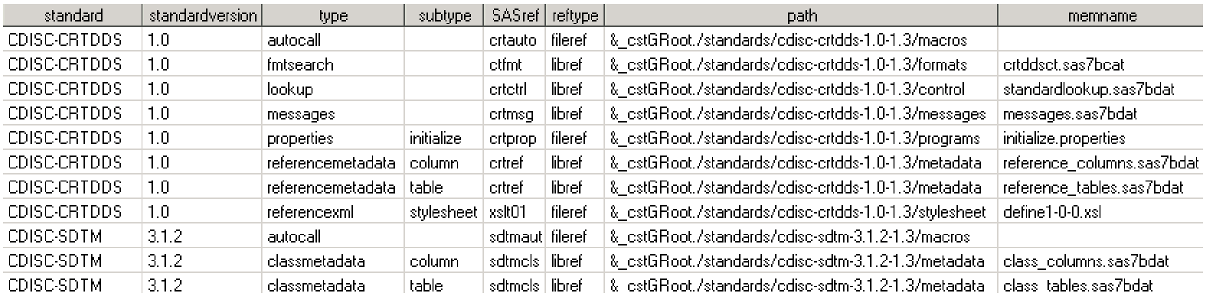
	standard	mnemonic	standardversion	comment	rootpath	isstandarddefault	iscstframework	isdatastandard	supportsvalidation
1	CDISC-CRTDDS	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3	Y	N	Y	Y
2	CDISC-ODM	ODM	1.3.0	CDISC ODM V1.3.0	&_cstGRoot./standards/cdisc-odm-1.3.0-1.3	Y	N	Y	Y
3	CDISC-SDTM	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.3	N	N	Y	Y
4	CDISC-SDTM	SDTM	3.1.2	CDISC SDTM V3.1.2	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3	Y	N	Y	Y
5	CDISC-TERMINOLOGY	CT	200810	CDISC Terminology, Packages 1, 2A, 2B, LABTEST	&_cstGRoot./standards/cdisc-terminology-200810-1.3	N	N	N	N
6	CDISC-TERMINOLOGY	CT	201003	CDISC Terminology 2010-03-05	&_cstGRoot./standards/cdisc-terminology-201003-1.3	Y	N	N	N
7	CST-FRAMEWORK	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework-1.3	Y	Y	N	N

	standard	mnemonic	standardversion	isxmlstandard	importxsl	exportxsl	schema	productrevision
1	CDISC-CRTDDS	CRT	1.0	Y	CRT-DDS/1.0/import/Root.xsl	CRT-DDS/1.0/export/Root.xsl	cdisc-crtdds-1.0.0/define1-0-0.xsd	1.3
2	CDISC-ODM	ODM	1.3.0	Y	ODM/1.3.0/import/Root.xsl	ODM/1.3.0/export/Root.xsl	cdisc-odm-1.3.0/ODM1-3-0.xsd	1.3
3	CDISC-SDTM	SDTM	3.1.1	N				1.3
4	CDISC-SDTM	SDTM	3.1.2	N				1.3
5	CDISC-TERMINOLOGY	CT	200810	N				1.3
6	CDISC-TERMINOLOGY	CT	201003	N				1.3
7	CST-FRAMEWORK	CST	1.2	N				1.3

Note: The **`&_cstGRoot`** directory in the **rootpath** column maps to the <*global standards library directory*>.

The StandardSASReferences data set defines the typical inputs and outputs of SAS processes that are associated with each standard. The following display shows some rows and columns.

Display 2.2 *Global Standards Library: Metadata StandardSASReferences Data Set*

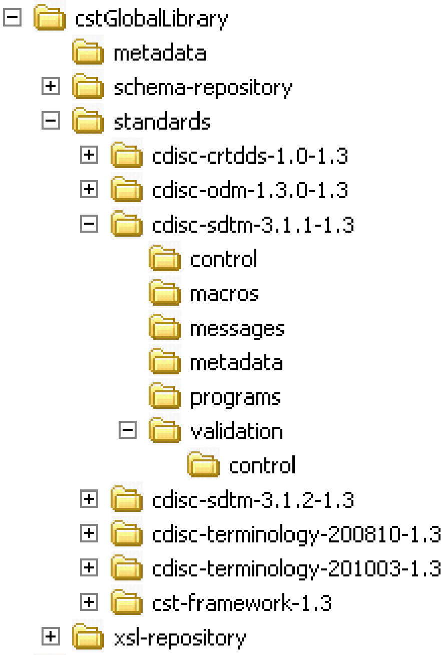


standard	standardversion	type	subtype	SASref	reftype	path	memname
CDISC-CRTDDS	1.0	autocall		crtauto	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3/macros	
CDISC-CRTDDS	1.0	fmtsearch		ctfmt	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3/formats	crtddsct.sas7bcat
CDISC-CRTDDS	1.0	lookup		crtctrl	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3/control	standardlookup.sas7bdat
CDISC-CRTDDS	1.0	messages		crtmsg	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3/messages	messages.sas7bdat
CDISC-CRTDDS	1.0	properties	initialize	crtprop	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3/programs	initialize.properties
CDISC-CRTDDS	1.0	referencemetadata	column	crtref	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3/metadata	reference_columns.sas7bdat
CDISC-CRTDDS	1.0	referencemetadata	table	crtref	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3/metadata	reference_tables.sas7bdat
CDISC-CRTDDS	1.0	referencexml	stylesheet	xslt01	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3/stylesheet	define1-0-0.xsl
CDISC-SDTM	3.1.2	autocall		sdtmaut	fileref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/macros	
CDISC-SDTM	3.1.2	classmetadata	column	sdtmcls	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	class_columns.sas7bdat
CDISC-SDTM	3.1.2	classmetadata	table	sdtmcls	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	class_tables.sas7bdat

The **type** and **subtype** columns can be used to reference information that SAS Clinical Standards Toolkit needs. This information is in the directory structures and file naming standards used by the customer. A full list of valid types and subtypes are provided in this document.

The **`standards`** directory contains subdirectories for each of the standard versions that is provided by SAS. In addition, there are subdirectories for user-customized versions of these standards and any new user-defined standards. Each subdirectory should be considered a stand-alone module. This is how the SAS Clinical Standards Toolkit can keep parallel standards and reduce the need for revalidation. Within each subdirectory, there might be directories that group the files, data sets, and housekeeping programs. The following display shows the directory structure for a Microsoft Windows global standards library with **cdisc-sdtm-3.1.1-1.3** expanded.

Display 2.3 *Directory Structure for a Microsoft Windows Global Standards Library*



The **`schema-repository`** directory 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. For example, the **`schema-repository`** directory for CDISC CRT-DDS 1.0 as defined in the Standards data set maps to:

`<global standards library directory>`**`/schema-repository/cdisc-crtdds-1.0.0`**

See Display 2.1 on page 6, row 1, **schema** column.

The **`xsl-repository`** directory contains files that are used to transform XML files from one format to another. For example, the default style sheet directory for CDISC CRT-DDS 1.0 define.xml files created by the SAS Clinical Standards Toolkit as defined in the Standards data set maps to:

***<global standards library directory>*/xsl-repository/CRT-DDS/1.0/export**

See Display 2.1 on page 6, row 1, **exportxsl** column.

What Is a Standard?

The answer to this question depends on what the standard is supposed to do. In the case of terminology, it might be a format catalog and a data set. In the case of an XML-based standard, it might be metadata that describes the SAS representation of the XML. It might be data sets that control validating the SAS representation of the XML. It might be routines to convert the SAS representation to the actual XML files. Or, it might be initialization files for standard-specific properties.

The minimum number of items that are needed to register a standard to the framework are the data sets that define the standard, as well as the standard's SASReferences data set. The macro to register a standard is described in "Registering a New Version of a Standard" on page 16.

For more information about what a SAS Clinical Standards Toolkit standard is, see Chapter 4, "Supported Standards," on page 35.

Common Framework Metadata

The following SAS Clinical Standards Toolkit metadata files support the functions and common tasks across multiple standards.

File structure and content for each of these metadata files are fully described in Chapter 3, "Metadata File Descriptions," on page 21. Use of these metadata files is documented in sections that use the SAS Clinical Standards Toolkit metadata.

Standards
: This data set has a list of the registered standards (for example, CDISC SDTM 3.1.1) and basic information relating to each standard. The Standards data set can be found in the global standards library metadata folder and within each registered standard folder hierarchy at:

  ***<global standards library directory>*/standards/*<standard>*/control**

StandardSASReferences
: This data set defines the typical inputs and outputs of SAS processes that are associated with each standard. The StandardSASReferences data set can be found in the global standards library metadata folder and within each registered standard folder hierarchy at:

  ***<global standards library directory>*/standards/*<standard>*/control**

Standardlookup

This data set contains valid values for discrete variables in SAS Clinical Standards Toolkit metadata files. The Standardlookup data set can be found within each registered standard folder hierarchy at:

***<global standards library directory>*/standards/*<standard>*/control**

SASReferences

This data set defines generic system and study-specific input and output files that are required by each SAS Clinical Standards Toolkit process. A sample SASReferences data set is provided with each supported standard.

Properties

These files provide the set of name-value pairs that are required to establish the environment for each SAS Clinical Standards Toolkit process. Properties are translated into SAS global macro variables at the start of each process. Properties can be found within each registered standard folder hierarchy at:

***<global standards library directory>*/standards/*<standard>*/programs**

Messages

This data set contains a list of codes and associated text that are specific to each standard. It can contain specific actions (such as validation) that are used to report process results. The Messages data set can be found within each registered standard folder hierarchy at:

***<global standards library directory>*/standards/*<standard>*/messages**

Results

This data set summarizes each SAS Clinical Standards Toolkit process. It captures the outcome of specific actions and uses the Messages data set to standardize output.

Other SAS Clinical Standards Toolkit metadata files specific to supported standards or specific to actions (such as validation) are described in Chapter 3, “Metadata File Descriptions,” on page 21. They are also discussed elsewhere in this document.

Common Usage Scenarios for the Framework

Overview

The following sections describe usage scenarios that the framework accommodates. Code that is required to complete the usage scenario is included in each section. All macros that are provided in the usage scenarios can be found in the primary SAS Clinical Data Standards Toolkit autocall path:

!sasroot/cstframework/sasmacro

For complete macro documentation, see Appendix A3, “Macro Application Programming Interface,” on page 225.

Initializing the Framework's Global Macro Variables

The framework requires certain global macro variables to execute properly. A user should initialize these global macro variables at the start of each SAS Clinical Standards Toolkit session. The same requirement might exist for a standard. The standard might need global macro variables to call its macros. The framework provides a macro to help with this requirement.

```
/*
initialize the global macro variables needed by the framework
*/
%cst_setStandardProperties(
_cstStandard=CST-FRAMEWORK
,_cstSubType=initialize
);
```

This code looks at the global SASReferences data set for a properties entry with a **SubType** of **initialize**. By default, this entry is located at:

***<global standards library directory>*/standards/cst-framework-1.3/programs/initialize.properties**

Global macro variables are initialized based on the name-value pairs in this properties file. After this macro has been called once, a user does not need to call it again during the SAS session, unless the user wants to override macro variables or reset them.

Referencing the Default Version of a Standard

If a version must be specified, then the specification can usually be omitted if the default version is to be used. The default version is specified in the global standards library metadata Standards data set. For example, the code to initialize CDISC SDTM 3.1.2 properties can be written as:

```
/*
initialize the global macro variables needed by CDISC SDTM
*/
%cst_setStandardProperties(
_cstStandard=CDISC-SDTM
,_cstSubType=initialize
);
```

In this example, the initialization properties for the default version of the CDISC SDTM standard (currently 3.1.2) are used without needing to specify a version.

Getting a List of the Standards That Are Installed

It is programmatically possible to get a list of the current standards that are registered to the framework. The following code can be used:

```
/*
get a list of the registered standards
*/
%cst_getRegisteredStandards(
_cstOutputDS=work.regStds
);
```

The data set work.regStds contains the information from the global standards library metadata Standards data set. The work.regStds data set's content matches the information provided in Display 2.1 on page 6.

Determining Which Revision (Release) of a Standard Version Is Installed

It is programmatically possible to determine which revision of a standard version is installed. The following code can be used:

```
/*
initialize the global macro variables needed by the framework
*/
%cst_setStandardProperties(
    _cstStandard=CST-FRAMEWORK
    ,_cstSubType=initialize
    );
/*
get a list of the registered standards
*/
%cst_getRegisteredStandards(
    _cstOutputDS=work.regStds
    );
```

The data set work.regStds contains the information from the global standards library metadata Standards data set. The last column is **productRevision**. This column contains the revision of each standard version. If the **productRevision** column is blank, then the standard was originally registered with SAS Clinical Standards Toolkit 1.2.

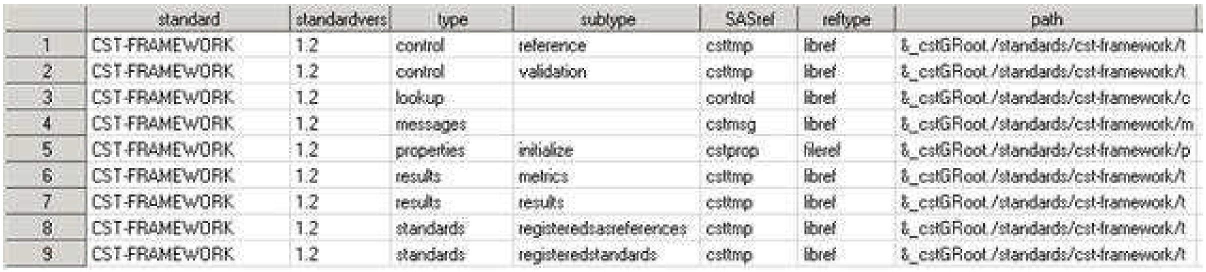
Getting a List of the Files and Data Sets That Are Associated with a Registered Standard

When standards are registered, information about the files and data sets that comprise the standard is registered also. The following macro call returns records from the StandardSASReferences data set that are associated with the specified standard. It returns records for standardversion if applicable.

```
%cst_getStandardSASReferences(
_cstStandard=CST-FRAMEWORK
,_cstOutputDS=sasrefs
);
```

The parameters that are used in this macro call specify the standard **`CST-FRAMEWORK`** and the data set to create to contain the information. Because the standard version is omitted, the default standard version is used. The data set that is returned is a SASReferences data set. For the macro call, the following shows the first few columns of data that are returned:

Display 2.4 *StandardSASReferences Returned in work.sasrefs Data Set (Column Subset)*



	standard	standardvers	type	subtype	SASref	reftype	path
1	CST-FRAMEWORK	1.2	control	reference	csttmp	libref	&_cstGRoot./standards/cst-framework/t
2	CST-FRAMEWORK	1.2	control	validation	csttmp	libref	&_cstGRoot./standards/cst-framework/t
3	CST-FRAMEWORK	1.2	lookup		control	libref	&_cstGRoot./standards/cst-framework/c
4	CST-FRAMEWORK	1.2	messages		cstmsg	libref	&_cstGRoot./standards/cst-framework/m
5	CST-FRAMEWORK	1.2	properties	initialize	cstprop	fileref	&_cstGRoot./standards/cst-framework/p
6	CST-FRAMEWORK	1.2	results	metrics	csttmp	libref	&_cstGRoot./standards/cst-framework/t
7	CST-FRAMEWORK	1.2	results	results	csttmp	libref	&_cstGRoot./standards/cst-framework/t
8	CST-FRAMEWORK	1.2	standards	registeredsasreferences	csttmp	libref	&_cstGRoot./standards/cst-framework/t
9	CST-FRAMEWORK	1.2	standards	registeredstandards	csttmp	libref	&_cstGRoot./standards/cst-framework/t

Note: If the **cst_setStandardProperties** macro has not been called before invoking the **cst_getStandardSASReferences** macro, then the following errors are reported in the SAS log:

```
WARNING: Apparent symbolic reference _CSTDEBUG not resolved.
ERROR: A character operand was found in the %EVAL function or %IF condition where
a numeric operand is required. The condition was: (&_cstDebug))
ERROR: The macro CST_GETSTANDARDSASREFERENCES will stop executing.
```

Calling cst_setStandardProperties to create global macro variables for the SAS Clinical Standards Toolkit session is a prerequisite for most SAS Clinical Standards Toolkit tasks.

Creating Data Sets Used by the Framework

Many macro calls to the framework require tables to be passed in or referenced. The structure of these tables can be difficult to build manually, so the SAS Clinical Standards Toolkit provides functionality to create table shells that can be filled in. The following is an example of the macro call:

```
/*
Create the empty SASReferences data set used in the next
step
 */
%cst_createDS(
    _cstStandard=CST-FRAMEWORK,
    _cstType=control,
    _cstSubType=reference,
    _cstOutputDS=work.sasrefs
    );
```

The **Type** and **SubType** identify that it is a SASReferences table. The **Standard** identifies the module to be used. If the standard version is not specified, then the default for standard version is used. The output is a data set named work.sasrefs that contains 0 observations and 10 columns.

Creating Table Shells Based on a Data Standard

Data standards like CDISC SDTM have reference metadata that describes the tables and columns that comprise the data standard. Creating table shells using this metadata is useful and saves time. The following is the code to do this:

```
/*
Create the table shells for CDISC SDTM 3.1.1 in the work library
*/
%cst_createTablesForDataStandard(
    _cstStandard=CDISC-SDTM
    ,_cststandardVersion=3.1.1
    ,_cstOutputLibrary=work
    );
```

This code creates the 25 domains described by CDISC SDTM version 3.1.1 in the Work library. Each domain contains 0 observations.

Getting a Copy of the Reference Metadata for a Data Standard

The SAS representation of many standards (such as CDISC SDTM) includes table and column metadata for all domains that are specific to each standard. The SAS Clinical Standards Toolkit framework provides a way to create and populate the metadata files.

```
/*
Step 1. Create the empty SASReferences data set used in
the next step
 */
%cst_createDS(
    _cstStandard=CST-FRAMEWORK,
    _cstType=control,
    _cstSubType=reference,
    _cstOutputDS=work.sasrefs);
/*
Step 2. Prep the type of information to be returned.
 */
data work.sasrefs;
    if 0 then set work.sasrefs;
    standard='CDISC-SDTM';
    standardVersion='3.1.2';
    * ----- REFERENCE METADATA -----;
    * tables metadata;
    type='referencemetadata';
    subType='table';
    sasRef='work';
    refType='libref';
    memname='refTables';
    output;
    * columns metadata;
    type='referencemetadata';
    subType='column';
    sasRef='work';
    refType='libref';
    memname='refColumns';
    output;
run;
/*
Step 3. Call the macro to get the metadata.
 */
%cst_getStandardMetadata(
    _cstSASReferences=work.sasrefs
    );
```

Step 1 uses one macro to create an empty SASReferences data set named **`work.sasrefs`**.

Step 2 determines the information to be returned. The standard and version is CDISC SDTM 3.1.2. The **`type`** and **`subType`** identify the types of metadata to be returned. The **`sasRef`** and **`memname`** identify the target library and name for each data set.

Step 3 is the actual macro call that does the processing. The data set **`work.sasrefs`** is read, and the global metadata is used to fulfill the request.

The outcome of these steps is two data sets. The data set **`work.refTables`** contains metadata about the 32 CDISC SDTM 3.1.2 domains. The data set **`work.refColumns`** contains metadata about each of the 723 columns defined in the 32 domains.

Inserting Information from Registered Standards into a SASReferences File

When a standard is registered, information about the data sets and files that comprise the standard is registered. These data sets and files are in a default folder hierarchy within the global standards library. The SAS Clinical Standards Toolkit provides a mechanism to reference the location of, and metadata about, these data sets and files. As a result, users do not have to specify paths and member names in each SASReferences file they create. When a SAS Clinical Standards Toolkit process encounters an incomplete file reference in a SASReferences file, it looks in the standard-specific folder hierarchy for the information. This mechanism is useful for a number of reasons:

- Programmers do not need to know all of the locations.
- If the global standards library needs to move, it can without having to change all of the SASReferences files that use a standard.
- To change standard versions, you only need to change the contents of the **standardversion** column.

The following code creates a partial SASReferences file:

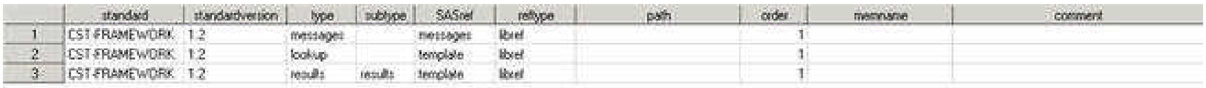
```
/*
Step 1. Initialize the global macro variables needed by the
framework.
*/
%cst_setStandardProperties(
    _cstStandard=CST-FRAMEWORK
    ,_cstSubType=initialize
    );
/*
Step 2. Create the empty SASReferences data set.
*/
%cst_createDS(
    _cstStandard=CST-FRAMEWORK,
    _cstType=control,
    _cstSubType=reference,
    _cstOutputDS=sasrefs
    );
/*
Step 3. Fill in the minimal information for a series of
records
*/
data sasrefs;
    if 0 then set sasrefs;

    standard='CST-FRAMEWORK';
    standardversion='1.2';
    type='messages';
    subtype='';
    sasref='messages';
    reftype='libref';
    order=1;
    output;
```

```
        standard='CST-FRAMEWORK';
        standardversion='1.2';
        type='lookup';
        subtype='';
        sasref='template';
        reftype='libref';
        order=1;
        output;
        standard='CST-FRAMEWORK';
        standardversion='1.2';
        type='results';
        subtype='results';
        sasref='template';
        reftype='libref';
        order=1;
        output;
    run;
```

Here is what the data set looks like:

Display 2.5 *Example SASReferences Data Set*



	standard	standardversion	type	subtype	SASref	reftype	path	order	memname	comment
1	CST-FRAMEWORK	1.2	messages		messages	libref		1		
2	CST-FRAMEWORK	1.2	lookup		template	libref		1		
3	CST-FRAMEWORK	1.2	results	results	template	libref		1		

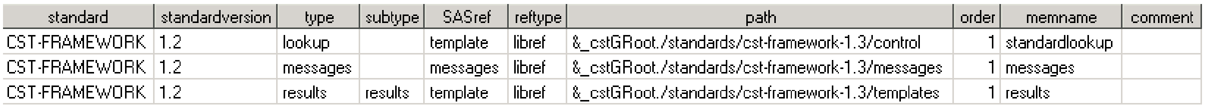
The **path** and **memname** columns are missing. The user has specified the standard, standardversion, type, subtype, SASref, and reftype. This information is sufficient. The rest of the information is available from the registered standard's metadata.

The following macro call attempts to insert the missing information if it is found in a registered standard's metadata:

```
/*
Step 4. Insert the missing information from registered
standard.
*/
%cst_insertStandardSASRefs(
    _cstSASReferences=sasrefs
    ,_cstOutputDS=outSASRefs
    );
```

Here is what the output data set looks like:

Display 2.6 *work.outSASRefs Data Set with Added Content*



standard	standardversion	type	subtype	SASref	reftype	path	order	memname	comment
CST-FRAMEWORK	1.2	lookup		template	libref	&_cstGRoot./standards/cst-framework-1.3/control	1	standardlookup	
CST-FRAMEWORK	1.2	messages		messages	libref	&_cstGRoot./standards/cst-framework-1.3/messages	1	messages	
CST-FRAMEWORK	1.2	results	results	template	libref	&_cstGRoot./standards/cst-framework-1.3/templates	1	results	

Maintenance Usage Scenarios

Overview

The following sections describe usage scenarios that the framework accommodates. Code that is required to complete the usage scenario included in each section. All macros that are provided in the usage scenarios can be found in the primary SAS Clinical Data Standards Toolkit autocall path:

!sasroot/cstframework/sasmacro

For complete macro documentation, see Appendix A3, “Macro Application Programming Interface,” on page 225.

Registering a New Version of a Standard

The following code defines and registers a new standard. The code can also be used to register a new version of an existing standard.

```
/*
Step 1. Ensure that the macro variable pointing to the global standards
library exists.
*/
%cstutil_setcstgroot;
/*
Step 2. Register the standard with the Toolkit global standards
library
*/
%cst_registerStandard(
    _cstRootPath=%nrstr(&_cstGRoot./standards/myStandard),
    _cstControlSubPath=control,
    _cstStdDSName=standards,
    _cstStdSASRefsDSName=StandardSASReferences);
```

Step 1 ensures that the macro variable that contains the global standards library path is set. Step 2 registers the standard by passing the following information:

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

The **_cstRootPath** parameter uses **%nrstr(&_cstGroot)** so that the **&_cstGroot** is registered as a macro variable. This specification allows the global standards library to be moved or copied without reregistering the full path of the new standard.

When defining and registering a new standard, you should evaluate which of the metadata files described in "Common Framework Metadata" on page 8 should be provided to support new standard functionality. For example:

- Should a sample SASReferences file be created to perform some task?
- Should a Messages data set be added to provide standard-specific informational messages?
- Should properties files be provided to set standard-specific global macro variables?

For more information about the metadata files that support SAS Clinical Standards Toolkit, see Chapter 3, "Metadata File Descriptions," on page 21.You can define new metadata types. These new metadata types should be documented in the standard-specific StandardSASReferences and Standardlookup data sets, and in the SAS Clinical Standards Toolkit framework Standardlookup data set.

Setting the Default Version for a Standard

When multiple versions of a standard exist, the first version that is installed is set as the default. The default version is used when multiple versions of a standard have been registered, and a specific version is not provided in a macro call or in a SASReferences file. The following code modifies the default version of a specific standard:

```
%cst_setStandardVersionDefault(
    _cstStandard=CDISC-SDTM
    ,_cstStandardVersion=3.1.1
    );
```

The version **`3.1.1`** is set as the default version for the CDISC SDTM standard.

Unregistering a Standard Version

If a standard becomes obsolete and needs to be unregistered, then use the framework to do this. Unregistering a standard might be needed during the development of a custom standard. The following macro call unregisters the CDISC SDTM 3.1.1 standard, removes it from the global standards library metadata Standards data set, and removes all records for 3.1.1 from the StandardSASReferences data set:

```
%cst_unregisterStandard(
    _cstStandard=CDISC-SDTM
    ,_cstStandardVersion=3.1.1
    );
```

Unregistering an Old Version of a Standard, and then Registering a New Version of a Standard

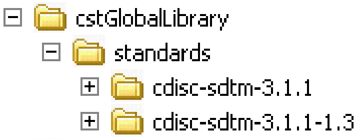
Suppose SAS Clinical Standards Toolkit 1.2 is currently installed and used. SAS Clinical Standards Toolkit 1.3 is released. You want the product updates for a standard version. In the following steps, the CDISC SDTM standard is used as an example. However, the steps apply to all other standard versions. You want to set version 3.1.2 as the default version for the CDISC SDTM standard. The SAS Clinical Standards Toolkit installation process does not do this automatically because you might have made updates to the SAS Clinical Standards Toolkit 1.2 code base or metadata that you want to preserve. Or, you might want to test the SAS Clinical Standards Toolkit 1.3 CDISC SDTM 3.1.2 implementation before declaring it the new default version.

Step 1: Confirm that multiple versions of the standard are available. Confirm that registration of a new version is needed.

1. Navigate to the global standards library Standards directory ***<global standards library directory>*/standards**.

2. Confirm that multiple libraries exist for the same standard version. In the following example, two subdirectories exist for CDISC SDTM 3.1.1:

   Display 2.7 *Multiple Versions per Standard in the Global Standards Library*

   cstGlobalLibrary
   standards
   cdisc-sdtm-3.1.1
   cdisc-sdtm-3.1.1-1.3

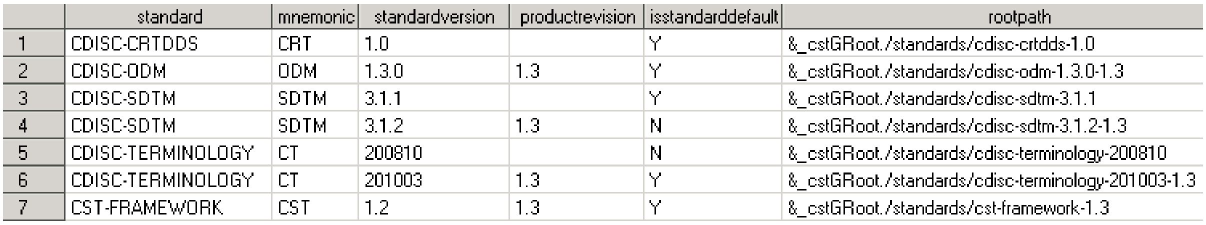
   The **cdisc-sdtm-3.1.1** directory contains files installed with SAS Clinical Standards Toolkit 1.2. The **cdisc-sdtm-3.1.1-1.3** directory contains files installed with SAS Clinical Standards Toolkit 1.3.

3. Confirm which revision of the standard-version is currently in use.

   - Assign a LIBNAME to the **metadata** subdirectory in the global standards library.

   - Open the Standards data set in the library, and confirm that the older version is the one being used. The following display shows that the registered version CDISC SDTM 3.1.1 has no product revision value that indicates that it is the original version that was shipped with SAS Clinical Standards Toolkit 1.2. It is defined as the default version for the CDISC SDTM standard.

Display 2.8 *Global Standards Library Metadata Standards Data Set Before Updates*



	standard	mnemonic	standardversion	productrevision	isstandarddefault	rootpath
1	CDISC-CRTDDS	CRT	1.0		Y	&_cstGRoot./standards/cdisc-crtdds-1.0
2	CDISC-ODM	ODM	1.3.0	1.3	Y	&_cstGRoot./standards/cdisc-odm-1.3.0-1.3
3	CDISC-SDTM	SDTM	3.1.1		Y	&_cstGRoot./standards/cdisc-sdtm-3.1.1
4	CDISC-SDTM	SDTM	3.1.2	1.3	N	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3
5	CDISC-TERMINOLOGY	CT	200810		N	&_cstGRoot./standards/cdisc-terminology-200810
6	CDISC-TERMINOLOGY	CT	201003	1.3	Y	&_cstGRoot./standards/cdisc-terminology-201003-1.3
7	CST-FRAMEWORK	CST	1.2	1.3	Y	&_cstGRoot./standards/cst-framework-1.3

Step 2: Register the updated CDISC SDTM 3.1.1 metadata in the global standards library to use the SAS Clinical Standards Toolkit 1.3.

1. Navigate to the Standards directory in the global standards library. Go to the **programs** directory of the revision of the standard version that needs to be registered. For example, go to ***<global standards library directory>*/standards/cdisc-sdtm-3.1.1-1.3/programs**.

2. Start a SAS session. Make sure that the current directory is the **programs** directory.

3. To unregister the currently installed revision and version, submit the following code:

```
%cstutil_setcstgroot;
/*
Set the framework properties used for the uninstall
*/
%cst_setStandardProperties(
    _cstStandard=CST-FRAMEWORK,
    _cstSubType=initialize
    );

/*
```

```
If the version to be replaced is the default, you must
make another version the default.
In this case, this is the desired final outcome anyway.
*/
%cst_setStandardVersionDefault(
    _cstStandard=CDISC-SDTM
    ,_cstStandardVersion=3.1.2
    );

/*
Unregister the standard
*/
%cst_unregisterStandard(
    _cstStandard=CDISC-SDTM
    ,_cstStandardVersion=3.1.1
    );
```

Note: The **`cst_setStandardVersionDefault`** macro call needs to be used only if the version being updated is the default version of the standard.

4. Check the Results data set. By default, the data set is work._cstResults. The final line in the data set should report that the standard version is no longer registered as a standard.

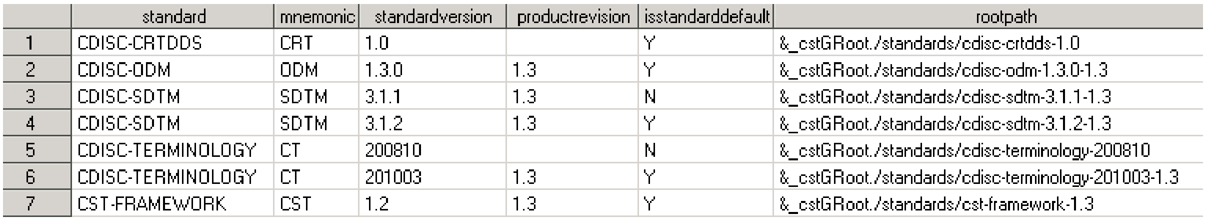
5. Open and submit the registerstandard.sas file from the **`programs`** directory into the Program Editor.

6. Confirm that the new revision was registered.

   - Assign a LIBNAME to the **`metadata`** subdirectory in the global standards library.

   - Open the Standards data set in the library, and confirm that the newer revision is the one being used. The following display shows that the CDISC SDTM 3.1.1 standard is now reregistered and the product revision in use is 1.3.

Display 2.9 *Global Standards Library Metadata Standards Data Set After Updates*



	standard	mnemonic	standardversion	productrevision	isstandarddefault	rootpath
1	CDISC-CRTDDS	CRT	1.0		Y	&_cstGRoot./standards/cdisc-crtdds-1.0
2	CDISC-ODM	ODM	1.3.0	1.3	Y	&_cstGRoot./standards/cdisc-odm-1.3.0-1.3
3	CDISC-SDTM	SDTM	3.1.1	1.3	N	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.3
4	CDISC-SDTM	SDTM	3.1.2	1.3	Y	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3
5	CDISC-TERMINOLOGY	CT	200810		N	&_cstGRoot./standards/cdisc-terminology-200810
6	CDISC-TERMINOLOGY	CT	201003	1.3	Y	&_cstGRoot./standards/cdisc-terminology-201003-1.3
7	CST-FRAMEWORK	CST	1.2	1.3	Y	&_cstGRoot./standards/cst-framework-1.3

Chapter 3
Metadata File Descriptions

Overview

SAS Clinical Standards Toolkit provides and uses metadata files to support its basic core functions, and to support specific functionality within the SAS Clinical Standards Toolkit. The file content and structure are described in the following sections. The usage of each of these metadata files is described in the document.

Standards

The Standards data set is used by the SAS Clinical Standards Toolkit framework to store information about a standard version. All standards that are provided by SAS, and standards that you might want to add are defined in the global standards library in the metadata/standards data set. All calls to the %cst_registerStandard macro that are described in Chapter 2 interact directly with the metadata/standards data set.

Table 3.1 *Metadata/Standards Data Set Structure in the Global Standards Library*

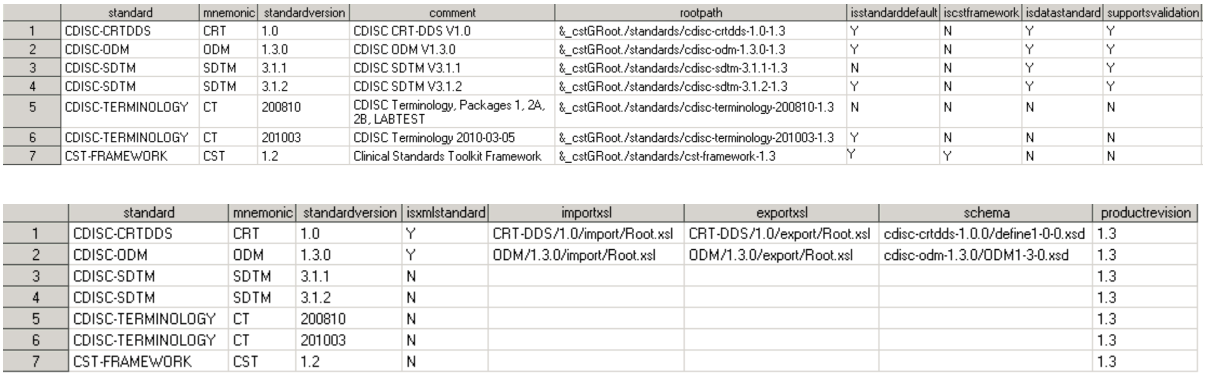
Column Name	Column Length	Description
standard	($20)	The name of the registered standard. Must be unique within the data set.
mnemonic	($4)	A short mnemonic for the standard.
standardversion	($20)	The version number of the registered standard. Must be unique within the standard.
comment	($200)	A description of the registered standard version.
rootpath	($200)	The root path for the standard version's directory in the global standards library.
isstandarddefault	($1)	A value that identifies whether the version is the default for the standard. More than one version can be registered and you can still have a default version. Valid values are Y and N.
iscstframework	($1)	A value that identifies whether the standard version is part of the framework. This column can be used to subset the list of registered standards. Valid values are Y and N.
isdatastandard	($1)	A value that identifies whether the standard version is a data standard. For example, CDISC SDTM versions are data standards, and CDISC Terminology is not. Valid values are Y and N.
supportsvalidation	($1)	A value that identifies whether the standard version supports validation. Valid values are Y and N.
isxmlstandard	($1)	A value that identifies whether the standard version is based on XML. CDISC SDTM is not, and CDISC CRT-DDS is. Valid values are Y and N.
importxsl	($200)	If the standard version is based on XML, then this is the path to the XSL file to import the XML into the SAS representation.
exportxsl	($200)	If the standard version is based on XML, then this is the path to the XSL file to export the XML file.
schema	($200)	If the standard version is based on XML, then this is the path to the XML schema document that can be used to validate the XML.
productrevision	($10)	The revision of the standard and standardversion that is currently installed.

The global standards library data set provided with the SAS Clinical Standards Toolkit can be found at:

```
<global standards library directory>/metadata/
standards.sas7bdat
```

The global standards library data set contains the following records. These records are provided with SAS Clinical Standards Toolkit 1.3:

Display 3.1 *Metadata/Standards Data Set Content in the Global Standards Library*



	standard	mnemonic	standardversion	comment	rootpath	isstandarddefault	iscstframework	isdatastandard	supportsvalidation
1	CDISC-CRTDDS	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1.0-1.3	Y	N	Y	Y
2	CDISC-ODM	ODM	1.3.0	CDISC ODM V1.3.0	&_cstGRoot./standards/cdisc-odm-1.3.0-1.3	Y	N	Y	Y
3	CDISC-SDTM	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.3	N	N	Y	Y
4	CDISC-SDTM	SDTM	3.1.2	CDISC SDTM V3.1.2	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3	Y	N	Y	Y
5	CDISC-TERMINOLOGY	CT	200810	CDISC Terminology, Packages 1, 2A, 2B, LABTEST	&_cstGRoot./standards/cdisc-terminology-200810-1.3	N	N	N	N
6	CDISC-TERMINOLOGY	CT	201003	CDISC Terminology 2010-03-05	&_cstGRoot./standards/cdisc-terminology-201003-1.3	Y	N	N	N
7	CST-FRAMEWORK	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework-1.3	Y	Y	N	N

	standard	mnemonic	standardversion	isxmlstandard	importxsl	exportxsl	schema	productrevision
1	CDISC-CRTDDS	CRT	1.0	Y	CRT-DDS/1.0/import/Root.xsl	CRT-DDS/1.0/export/Root.xsl	cdisc-crtdds-1.0.0/define1-0-0.xsd	1.3
2	CDISC-ODM	ODM	1.3.0	Y	ODM/1.3.0/import/Root.xsl	ODM/1.3.0/export/Root.xsl	cdisc-odm-1.3.0/ODM1-3-0.xsd	1.3
3	CDISC-SDTM	SDTM	3.1.1	N				1.3
4	CDISC-SDTM	SDTM	3.1.2	N				1.3
5	CDISC-TERMINOLOGY	CT	200810	N				1.3
6	CDISC-TERMINOLOGY	CT	201003	N				1.3
7	CST-FRAMEWORK	CST	1.2	N				1.3

The `&_cstGRoot` in the **rootpath** column maps to the ***`<global standards library directory>`*** that is set by calling the cstutil_setcstgroot macro.

An example of the global standards library data set that is used to register a specific standard can be found at:

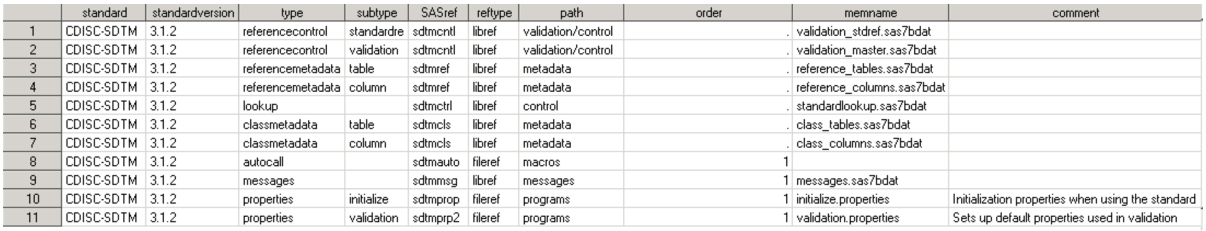
`<global standards library directory>`**`/standards/cdisc-sdtm-3.1.2-1.3/control/standards.sas7bdat`**

StandardSASReferences

The StandardSASReferences metadata data set specifies a set of library and file records that are used by most processes that are provided with the SAS Clinical Standards Toolkit implementation of each standard. It contains references to those libraries and files that are installed with each standard that SAS provides. A standard-specific StandardSASReferences data set exists for each SAS Clinical Standards Toolkit data standard that is supported by SAS. For example, the CDISC SDTM 3.1.2 StandardSASReferences data set can be found at:

`<global standards library directory>`**`/standards/cdisc-sdtm-3.1.2-1.3/control/standardsasreferences.sas7bdat`**

Display 3.2 *Metadata/StandardSASReferences Data Set Content in the Global Standards Library*



	standard	standardversion	type	subtype	SASref	reftype	path	order	memname	comment
1	CDISC-SDTM	3.1.2	referencecontrol	standardre	sdtmcntl	libref	validation/control	.	validation_stdref.sas7bdat	
2	CDISC-SDTM	3.1.2	referencecontrol	validation	sdtmcntl	libref	validation/control	.	validation_master.sas7bdat	
3	CDISC-SDTM	3.1.2	referencemetadata	table	sdtmref	libref	metadata	.	reference_tables.sas7bdat	
4	CDISC-SDTM	3.1.2	referencemetadata	column	sdtmref	libref	metadata	.	reference_columns.sas7bdat	
5	CDISC-SDTM	3.1.2	lookup		sdtmctrl	libref	control	.	standardlookup.sas7bdat	
6	CDISC-SDTM	3.1.2	classmetadata	table	sdtmcls	libref	metadata	.	class_tables.sas7bdat	
7	CDISC-SDTM	3.1.2	classmetadata	column	sdtmcls	libref	metadata	.	class_columns.sas7bdat	
8	CDISC-SDTM	3.1.2	autocall		sdtmauto	fileref	macros	1		
9	CDISC-SDTM	3.1.2	messages		sdtmmsg	libref	messages	1	messages.sas7bdat	
10	CDISC-SDTM	3.1.2	properties	initialize	sdtmprop	fileref	programs	1	initialize.properties	Initialization properties when using the standard
11	CDISC-SDTM	3.1.2	properties	validation	sdtmprp2	fileref	programs	1	validation.properties	Sets up default properties used in validation

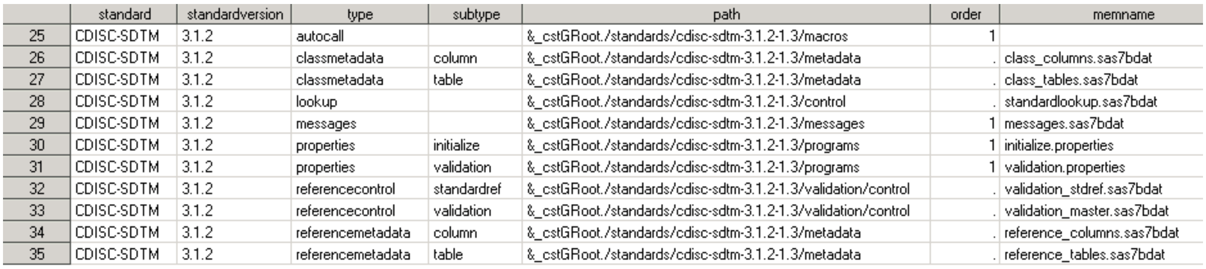
The **type** and **subtype** values are discussed in the following section. The **SASref** value is the default value that is used in the library and filename allocation process. This value can be overwritten by the user. The **path** value represents the global standards library subdirectory, which is relative to the rootpath location that is specified in the standard-specific Standards data set.

The cross-standard global standards library StandardSASReferences data set that is provided with the SAS Clinical Standards Toolkit can be found at:

***<global standards library directory>*/metadata/ standardsasreferences.sas7bdat**

This data set contains the concatenation of each StandardSASReferences data set that is provided for each supported standard in the SAS Clinical Standards Toolkit. The only enhancement to the data set during concatenation is that the **path** column is populated with the full global standards library path for each record. The following display shows the content for the CDISC SDTM StandardSASReferences data set that is described in Display 3.2 on page 23.In the display, **&_cstGRoot** maps to the ***<global standards library directory>*** that is set by calling the cstutil_setcstgroot macro.

Display 3.3 *Metadata/StandardSASReferences Data Set in the Global Standards Library (CDISC SDTM 3.1.2 Excerpt)*



	standard	standardversion	type	subtype	path	order	memname
25	CDISC-SDTM	3.1.2	autocall		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/macros	1	
26	CDISC-SDTM	3.1.2	classmetadata	column	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	.	class_columns.sas7bdat
27	CDISC-SDTM	3.1.2	classmetadata	table	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	.	class_tables.sas7bdat
28	CDISC-SDTM	3.1.2	lookup		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/control	.	standardlookup.sas7bdat
29	CDISC-SDTM	3.1.2	messages		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/messages	1	messages.sas7bdat
30	CDISC-SDTM	3.1.2	properties	initialize	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/programs	1	initialize.properties
31	CDISC-SDTM	3.1.2	properties	validation	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/programs	1	validation.properties
32	CDISC-SDTM	3.1.2	referencecontrol	standardref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/validation/control	.	validation_stdref.sas7bdat
33	CDISC-SDTM	3.1.2	referencecontrol	validation	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/validation/control	.	validation_master.sas7bdat
34	CDISC-SDTM	3.1.2	referencemetadata	column	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	.	reference_columns.sas7bdat
35	CDISC-SDTM	3.1.2	referencemetadata	table	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	.	reference_tables.sas7bdat

The structure of the StandardSASReferences data set is the same structure for all SASReferences data sets that are provided and used by the SAS Clinical Standards Toolkit. This structure is described in “SASReferences” on page 25.

Standardlookup

The Standardlookup data set provides a mechanism to capture valid values for discrete variables in the SAS Clinical Standards Toolkit metadata files. This data set supports such tasks as validating the content of SAS Clinical Standards Toolkit metadata files and providing selectable values in the user interfaces of other tools and solutions.

Table 3.2 *Standardlookup Data Set Structure in the Global Standards Library*

Column Name	Column Length	Description
sasref	($8)	SAS libref
table	($32)	A SAS Clinical Standards Toolkit table name
column	($32)	A SAS Clinical Standards Toolkit column name
refcolumn	($32)	Associated SAS Clinical Standards Toolkit column name
refvalue	($200)	Associated SAS Clinical Standards Toolkit column value

Column Name	Column Length	Description
value	($200)	Unique SAS Clinical Standards Toolkit column value
default	($200)	Default SAS Clinical Standards Toolkit column value
nonnull	($1)	Value that specifies whether a SAS Clinical Standards Toolkit column value must be non-null
order	(8.)	A SAS Clinical Standards Toolkit column value order
comment	($200)	Explanatory comments

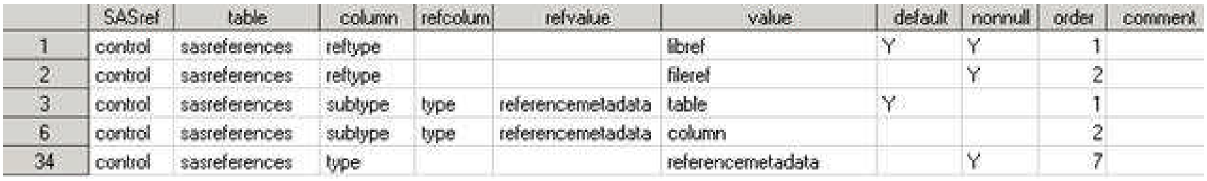
A Standardlookup data set is provided for most standards with the SAS Clinical Standards Toolkit. This data set can be used to define and register custom standards in the SAS Clinical Standards Toolkit.

An example of the Standardlookup data set can be found at:

`<global standards library directory>`**`/standards/cst-framework/control/standardlookup.sas7bdat`**

An example of the records in a Standardlookup data set is provided in the following figure:

Display 3.4 *Standardlookup Data Set Content in the Global Standards Library*



	SASref	table	column	refcolum	refvalue	value	default	nonnull	order	comment
1	control	sasreferences	reftype			libref	Y	Y	1	
2	control	sasreferences	reftype			fileref		Y	2	
3	control	sasreferences	subtype	type	referencemetadata	table	Y		1	
6	control	sasreferences	subtype	type	referencemetadata	column			2	
34	control	sasreferences	type			referencemetadata		Y	7	

These records show that the SASReferences data set allows a value of **`referencemetadata`** for the **type** column. The type value in a SASReferences data set must always be non-null. Two **subtype** values (**table** and **column**) are allowed when **type** is **referencemetadata**. For more information about the columns and values in the SASReferences data set, see the following section.

SASReferences

Each SAS Clinical Standards Toolkit process (for example, a primary task or action such as validating source data against a SAS Clinical Standards Toolkit standard) requires using a SASReferences data set. The SASReferences data set identifies all of the inputs required and the outputs that are created by the process. Each process might have its own unique SASReferences data set.

Chapter 5, "SASReferences File," on page 69 describes the content and usage of SASReferences data sets. The following table identifies and describes each column within a SASReferences data set.

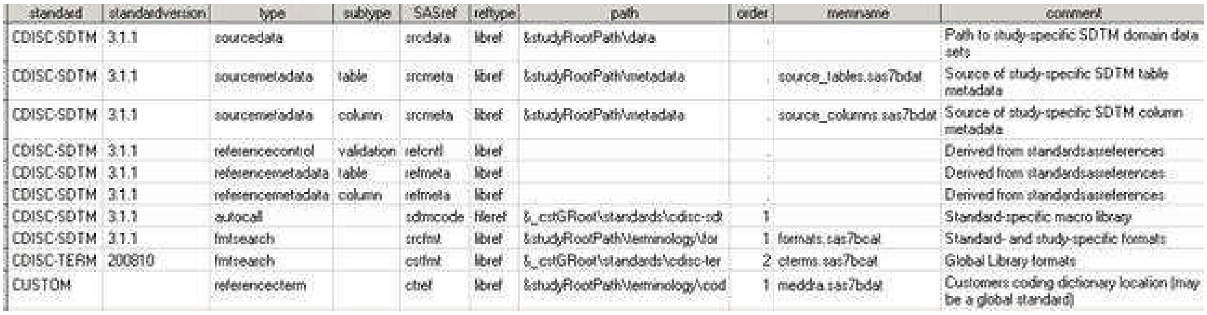
Table 3.3 SASReferences Data Set Structure

Column Name	Column Length	Description
standard	($20)	Standard name. This value should match the **standard** field in the Standards data set in ***<global standards library directory>*/metadata** and in other metadata files referenced in SASReferences (for example, CDISC SDTM and CDISC CRT-DDS). This column is required.
standardversion	($20)	Specific version of a standard. This value should match one of the standardversion values associated with the **standard** field in the Standards data set in ***<global standards library directory>*/metadata** and in other metadata files referenced in SASReferences (for example, 3.1.1 or 1.0). This column is required.
type	($40)	The type of input and output data or metadata. This is a predefined set of values that are documented in the ***<global standards library directory>*/standards/cst-framework-1.3/control/standardlookup** data set. These values are also itemized in Table 5.1. This column is required.
subtype	($40)	The specific subtype within type of input and output data or metadata. This is a predefined set of values that are documented in the ***<global standards library directory>*/standards/cst-framework-1.3/control/standardlookup** data set. These values are also itemized in Table 5.1. This column is optional, depending on type.
SASref	($8)	The SAS libref or fileref that references the library or file in the SAS Clinical Standards Toolkit SAS process. This value should match the value of sasref that is used in any other associated metadata files (for example, in the Source Columns data set, the value is type=srcmeta). This column is required. It must conform to SAS libref or fileref naming conventions.
reftype	($8)	Reference type. This column is required. Valid values are libref and fileref.
path	($200)	The path of the library or the path portion of the file reference. If you want to use the default value for a standard, standardversion, type, or subtype, then leave the path blank. The value is added to the &_cstSASRefs working version of the SASReferences data set from the standard-specific StandardSASReferences data set. Specific paths should be provided for any type or subtype that is study- or run-specific. Paths might be relative to an environment variable (for example, !sasroot) or to a SAS macro variable (for example, &studyrootpath).

Column Name	Column Length	Description
order	(8.)	Processing or concatenation order within type. If this value exists, then it should be a positive integer with no duplicates within type. This column is optional, depending on type. The order should be specified if one of the following is true: 1. Multiple records exist within these types—autocall, fmtsearch, messages. 2. Library concatenation is wanted (multiple librefs are within the same value of SASref for a type). 3. There is a need to establish precedence within a type (for example, look first in this library, then look in another library).
memname	($48)	The name of a specific SAS file (data set or catalog) or file that is not created by SAS (for example, properties or an XML file). The memname column should be blank for library references. This column is optional, depending on type. As a general rule, memname should be provided if the path is provided, except where individual file references are not appropriate (for example, type=autocall and type=sourcedata). If you want to use the default value for a standard, standardversion, type, or subtype, then leave memname blank. The value is added to the &_cstSASRefs working version of the SASReferences data set from the standard-specific StandardSASReferences data set. The file suffix for SAS files is optional.
comment	($200)	Explanatory comments. This column is optional.

The following display shows some information in a typical SAS Clinical Standards Toolkit SASReferences data set.

Display 3.5 *A Sample SASReferences Data Set*



standard	standardversion	type	subtype	SASref	reftype	path	order	memname	comment
CDISC-SDTM	3.1.1	sourcedata		srcdata	libref	&studyRootPath\data	.		Path to study-specific SDTM domain data sets
CDISC-SDTM	3.1.1	sourcemetadata	table	srcmeta	libref	&studyRootPath\metadata	.	source_tables.sas7bdat	Source of study-specific SDTM table metadata
CDISC-SDTM	3.1.1	sourcemetadata	column	srcmeta	libref	&studyRootPath\metadata	.	source_columns.sas7bdat	Source of study-specific SDTM column metadata
CDISC-SDTM	3.1.1	referencecontrol	validation	refcntl	libref		.		Derived from standardsasreferences
CDISC-SDTM	3.1.1	referencemetadata	table	refmeta	libref		.		Derived from standardsasreferences
CDISC-SDTM	3.1.1	referencemetadata	column	refmeta	libref		.		Derived from standardsasreferences
CDISC-SDTM	3.1.1	autocall		sdtmcode	fileref	&_cstGRoot\standards\cdisc-sdt	1		Standard-specific macro library
CDISC-SDTM	3.1.1	fmtsearch		srcfmt	libref	&studyRootPath\terminology\for	1	formats.sas7bcat	Standard- and study-specific formats
CDISC-TERM	200810	fmtsearch		cstfmt	libref	&_cstGRoot\standards\cdisc-ter	2	cterms.sas7bcat	Global Library formats
CUSTOM		referencecterm		ctref	libref	&studyRootPath\terminology\cod	1	meddra.sas7bdat	Customers coding dictionary location (may be a global standard)

From this display, you can see that the data set contains information about types of data and metadata and where they are located. SAS Clinical Standards Toolkit imposes a rigid SASReferences file structure. No additional or fewer columns are allowed. No changes to column attributes are allowed (for example, changing column length).

Properties

SAS Clinical Standards Toolkit uses properties files to set default preferences for each process. Properties are name-value pairs that are translated into SAS global macro variables. These macro variables are available for the duration of a SAS Clinical Standards Toolkit process. Properties can be defined in any number of files. Both text file and SAS data set formats are supported. All SAS Clinical Standards Toolkit global macro variables are documented in Appendix A1, "Global Macro Variables," on page 211. These macro variables are derived from properties files provided by SAS.

The following table describes the contents of a sample properties file in ***<global standards library directory*>/standards/cst-framework/programs/initialize.properties**. Each property (global macro variable) is described in Appendix 1.

Table 3.4 Properties File Structure

Name (Global Macro Variable)	Default Value
_cstDebug	0
_cstDebugOptions	mprint mlogic symbolgen mautolocdisplay
_cst_rc	0
_cst_MsgID	
_cst_MsgParm1	
_cst_MsgParm2	
_cstResultSeq	0
_cstSeqCnt	0
_cstSrcData	
_cstResultFlag	0
_cstResultsDS	work._cstresults
_cstMessages	work._cstmessages
_cstReallocateSASRefs	0
_cstFMTLibraries	
_cstMessageOrder	APPEND
_cstSASRefsLoc	

Name (Global Macro Variable)	Default Value
_cstSASRefsName	
_cstSASRefs	work._cstsasrefs

Messages

By default, SAS Clinical Standards Toolkit provides a Messages data set for all SAS Clinical Standards Toolkit framework standards and for each data standard provided by SAS. Each Messages data set includes a list of codes and associated text that are specific to each standard. In some cases, actions such as validation are used to report process results. The structure of all the message files is described in the following table.

Table 3.5 *Messages Data Set Structure*

Column Name	Column Length	Description	Optional or Required
resultid	($8)	The message ID. SAS Clinical Standards Toolkit has adopted a naming convention matching each standard. The resultid values are prefixed with an up to 4-character prefix (CST for framework messaging; CDISC examples: ODM, SDTM, ADAM, and CRT). By convention, the prefix matches the **mnemonic** field in the Standards data set in ***<global standards library directory>*/metadata**. This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). Users can use any naming convention limited to 8 characters. For CDISC standards supporting validation, the resultid should match the checkid from the Validation Master data set for standard records that support validation.	Required
standardversion	($20)	A specific version of a standard. This value must match one of the standard versions that is associated with a registered standard. This value must also match the **standardversion** field in the SASReferences data set. The only exception to this rule is that *** can be used to signify that the check applies to all supported versions of the standard (for example, 3.1.1, 1.0, ***). If a subsequent version of the standard is released, then *** would be applicable if the check is valid for the new version.	Required

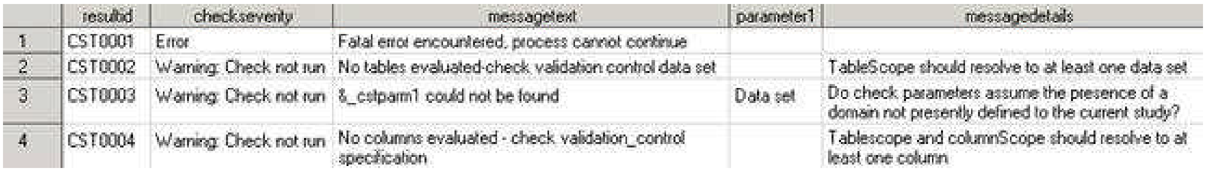
Column Name	Column Length	Description	Optional or Required
checksource	($40)	A string that identifies the source of the message. This string is used to provide source-specific messages generated within the SAS Clinical Standards Toolkit. CDISC examples include Janus, OpenCDISC, SAS, and WebSDM. This field can contain any user-defined value.	Required
sourceid	($8)	A reference identifier for this message from the **checksource**.	Optional
checkseverity	($40)	The severity as assigned by **checksource**. This value is mapped to the following standardized values: Note (Low), Warning (Medium), Error (High). A value is expected, although it is not technically required. It is used in reporting.	Optional
sourcedescription	($500)	A full description of the validation check that is associated with **checksource** if the source is external to SAS Clinical Standards Toolkit. If **checksource** is set to **CST**, then this field is null.	Optional
messagetext	($500)	The default message text to be written to the Results data set. This field can contain 0, 1, or 2 parameters. By convention, parameters are _cstParm1 and _cstParm2, but any _cst prefix parameter is recognized. The fully resolved **messagetext** that includes substituted parameter values is written to the Results data set.	Required
parameter1	($100)	The message parameter1 (_cstParm1) default value. If the code using the message does not provide a parameter value, then this default value is used. This column can be null.	Optional
parameter2	($100)	The message parameter2 (_cstParm2) default value. If the code using the message does not provide a parameter value, then this default value is used. This column can be null.	Optional
messagedetails	($200)	Any additional information that explains the message.	Optional

The Messages data set that supports the SAS Clinical Standards Toolkit framework can be found at:

***<global standards library directory>*/standards/cst-framework-1.3/messages/messages.sas7bdat**

The following display provides an excerpt of records and columns from the SAS Clinical Standards Toolkit framework Messages data set.

Display 3.6 *Framework Messages Data Set*



	resultid	checkseverity	messagetext	parameter1	messagedetails
1	CST0001	Error	Fatal error encountered, process cannot continue		
2	CST0002	Warning: Check not run	No tables evaluated-check validation control data set		TableScope should resolve to at least one data set
3	CST0003	Warning: Check not run	&_cstparm1 could not be found	Data set	Do check parameters assume the presence of a domain not presently defined to the current study?
4	CST0004	Warning: Check not run	No columns evaluated - check validation_control specification		Tablescope and columnScope should resolve to at least one column

For more information about messages supporting the SAS Clinical Standards Toolkit framework, see Appendix A2, “Framework Messages,” on page 219. Other message-type data sets that support non-framework standards are described in this document.

Results

Each SAS Clinical Standards Toolkit process generates a Results data set. The Results data set can be persisted beyond the SAS session based on SASReferences data set settings. Each Results data set captures the outcome of specific process actions. Each Results data set uses the Messages data set to standardize output.

The structure of each SAS Clinical Standards Toolkit Results data set is described in the following table:

Table 3.6 *Results Data Set Structure*

Column Name	Column Length	Description
resultid	($8)	Result ID. The resultid is a message ID from the standard Messages data set (for example, framework or CDISC SDTM). SAS Clinical Standards Toolkit has adopted a naming convention matching a resultid with each standard. The resultid values are prefixed with an up to 4-character prefix (CST for framework messaging; CDISC examples: ODM, SDTM, ADAM, and CRT). By convention, the prefix matches the **`mnemonic`** field in the Standards data set in ***`<global standards library directory>/`*** **`metadata`**. This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). Users can use any naming convention limited to 8 characters. Value should be non-null.
checkid	($8)	Validation check ID. SAS Clinical Standards Toolkit has adopted a naming convention matching each standard to be validated. The checkid values are prefixed with an up to 4-character prefix (CDISC examples: ODM, SDTM, ADAM, and CRT). By convention, the prefix matches the **`mnemonic`** field in the Standards data set in ***`<global standards library directory>/`*** **`metadata`**. This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). Users can use any naming convention limited to 8 characters. Value should be non-null for validation processes. Otherwise, this column is optional.

Column Name	Column Length	Description
resultseq	(8.)	Unique invocation of resultid. For validation processes, a sequence number to indicate the record number relative to checkid in the Validation Control run-time set of checks. If set to 1, then this is incremented only with each repeat invocation of a check. For non-validation processes, this value is generally a constant 1, but is reset to 1 with each new invocation of the SAS Clinical Standards Toolkit macro that is being run when the Results record is generated. Value should be non-null positive integer.
seqno	(8.)	Sequence number relative to resultseq. This value is a unique sequence number for the Results record in each unique value of resultseq. Value should be non-null positive integer.
srcdata	($200)	Source data. This string generally specifies: • (for validation) the domains evaluated or the check macro used • (otherwise) the SAS Clinical Standards Toolkit macro that is being run when the Results record is generated Value should be non-null.
message	($500)	Resolved message text from Messages data set. The message value includes up to two run-time parameter values in message text. Value should be non-null.
resultseverity	($40)	Result severity (for example, warning or error). Info — Informational note Note — Problem detected, low severity Warning — Problem detected, medium severity Warning: Check not run — No assessment able to be made Warning: Check not completed — Full compliance assessment could not be made Error — Problem detected, high severity Value should be non-null.
resultflag	(8.)	A value that determines whether a problem has been detected. The values are 0=no, otherwise, yes. -1 — Validation check not run 0 — No problem detected (value always 0 when resultseverity=Info) 1 — Validation check run, error detected Value should be non-null.
_cst_rc	(8.)	Process status. Values are nonzero and aborted. A nonzero value typically indicates that the process ended abnormally. Value should be non-null.

Column Name	Column Length	Description
actual	($240)	Actual value observed. This value is generally used for validation reporting. It provides the actual column values that are in error. This column is optional.
keyvalues	($2000)	Record-level keys and values. This value is generally used for validation reporting. It provides domain key values for records that are in error. This column is optional.
resultdetails	($200)	Basis or explanation for result. This column is optional.

For an example of a SAS Clinical Standards Toolkit Results data set, see Display 6.9 on page 116 and Display 6.10 on page 117.

Additional Metadata Files

The following metadata files can be used for specific tasks. In some cases, the file structures might be unique to the supported or referenced standard. These metadata files are provided by SAS Clinical Standards Toolkit.

Validation Master (Validation Control)

Each standard that supports validation has a Validation Master data set that provides the full set of validation checks defined for that standard. (For a description of the standards.supportsvalidation field, see Table 3.1 on page 22 .) This data set should have the columns as defined in Table 6.3 on page 90, though additional columns are permitted for user customizations. For each SAS Clinical Standards Toolkit validation process, the set of run-specific checks is captured in a Validation Control data set. The Validation Control data set is identical in structure to the Validation Master data set, but can be different only in the number of records (checks) included.

Reference_Tables(Source_Tables)

Part of the definition of each standard is the itemization of the data tables that define the SAS representation of that standard and version. The Reference_Tables data set captures table-level metadata about each reference standard data set. The structure of this data set can be standard specific. For example, Table 6.1 on page 86 describes the table metadata for the CDISC SDTM standard. For selected actions, SAS Clinical Standards Toolkit requires a similarly structured Source_Tables data set that defines study-specific tables. For example, a SAS Clinical Standards Toolkit validation process compares the study metadata in the Source_Tables data set with the reference standard metadata in the Reference_Tables data set.

Reference_Columns(Source_Columns)

Part of the definition of each standard is the itemization of the columns in each data table that defines the SAS representation of that standard and version. The Reference_Columns data set captures column-level metadata about each reference standard column. The structure of this data set can be standard specific. For example, Table 6.2 on page 88 describes the column metadata for the CDISC SDTM standard. For selected actions, SAS Clinical Standards Toolkit requires a similarly structured Source_Columns data set that defines study-specific columns. For example, a SAS Clinical Standards Toolkit validation process compares the study metadata in the Source_Columns data set with the reference standard metadata in the Reference_Columns data set.

Validation Metrics

Each SAS Clinical Standards Toolkit validation process can generate a Summary data set that provides a meaningful denominator for most validation checks. The Summary data set enables you to more accurately assess the relative scope of errors that are detected. The generation of this data set is based on validation property settings. This data set can be persisted beyond the SAS session based on SASReferences data set settings. For example, Table 6.9 on page 102 describes the metrics metadata for the CDISC SDTM standard, and Display 6.2 on page 103 provides sample content for the CDISC SDTM standard.

CDISC CRT-DDS Style Sheet

A sample XML style sheet (define1-0-0.xsl) is provided with the CDISC CRT-DDS standard. The style sheet is copied from **`http://www.cdisc.org/stuff/contentmgr/files/0/464923b10ea16b477151fcaa9f465166/misc/define1_0_0.xsl`**. A define.xml file can be rendered in a human-readable form if it contains an explicit XML style sheet reference, such as a reference to the default style sheet. Alternative style sheets can be used to provide metadata support for CDISC CRT-DDS.

Chapter 4
Supported Standards

SAS Representation of Standards

Overview

SAS Clinical Standards Toolkit is designed to support various clinical standards. SAS Clinical Standards Toolkit was initially built to support the Clinical Data Interchange Standards Consortium (CDISC) standards. However, the generic framework enables you to define any type of standard, including Health Level 7 (HL7) messages.

Each SAS Clinical Standards Toolkit standard provides a SAS representation of the published source guidelines or source specification. The SAS representation is designed to serve as a model or template of the source specification.

Two key design requirements shaped the implementation of SAS Clinical Standards Toolkit standards.

- Each supported standard is represented in one or more SAS files. This facilitates the following:
  - It provides SAS users with an implementation of data models and standards that are based on SAS.
  - It enables you to use SAS routines to assess how well any user-defined set of data and metadata conforms to the standard.
  - It enables you to use SAS code to read and derive files in other formats (for example, XML).

    Each SAS Clinical Standards Toolkit standard is an optimized reference standard from a SAS perspective.
- Users are able to define their own customized standards, or they are able to modify existing SAS standards. For more information about how new standards are registered in SAS Clinical Standards Toolkit, see "Registering a New Version of a Standard" on page 16.

SAS anticipates providing new standards and updates to existing SAS Clinical Standards Toolkit standards periodically. New standards and updates would be based on customer requirements and changes to source guidelines and source specifications.

This document uses the term "reference standard" to refer to the SAS representation of each source specification.

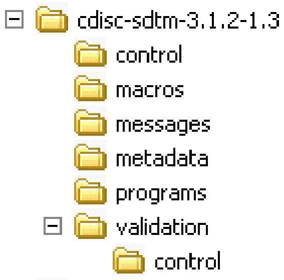
The definition of reference standard depends on several factors, including the complexity of the external source standard, the intended use of the standard, and the user's preferred implementation methodology. Here are three ways to define reference standard:

- A limited SAS representation of an external standard, defined as one or more SAS files.

  For example, consider two of the CDISC standards supported in SAS Clinical Standards Toolkit. Each CDISC Terminology standard can be represented in its simplest form as either a SAS data set or SAS format catalog of acceptable values. Each CDISC SDTM standard can be represented as a set of domains (SAS data sets), and as an associated set of data sets that describe the data set and column metadata for those domains. For some users, this might be the only information about the standards needed from SAS Clinical Standards Toolkit.
- A distinct folder hierarchy within the global standards library, comprising the previous definition and any supporting files required by SAS Clinical Standards Toolkit.

By default, reference standards are specified in the global standards library that is created when SAS Clinical Standards Toolkit is deployed. Each reference standard can be unique in regard to the folder hierarchy and supporting files. Consider the CDISC SDTM standard. The following global standards library folder hierarchy is provided for CDISC SDTM:

Display 4.1 *Global Standards Library Folder Hierarchy*



```
⊟ cdisc-sdtm-3.1.2-1.3
    control
    macros
    messages
    metadata
    programs
  ⊟ validation
      control
```

The **`metadata`** folder contains the data set and column metadata for each supported domain. SAS Clinical Standards Toolkit provides a utility macro (cst_createTablesForDataStandard) that reads this metadata, and builds an empty data set for each supported SDTM domain. All supporting files required by SAS Clinical Standards Toolkit to support the specific CDISC SDTM standard are provided in the remaining folders.

- The **`control`** folder provides these data sets:

  Standards
  is a single-record file that provides metadata about the standard.

  Standardlookup
  provides acceptable values for many discrete-value columns for a number of standard metadata files.

  StandardSASReferences
  is a sample or template specification of records that describes input or output files relevant to using the standard.

- The **`macros`** folder contains any SAS code specific to the CDISC SDTM standard.

- The **`messages`** folder contains messages that are associated with tasks (such as validation) that are supported by SAS Clinical Standards Toolkit.

- The **`metadata`** folder provides these data sets:

  Class_tables
  identifies a limited set of column collections specific to one or more SDTM domains. This data set is unique to CDISC SDTM.

  Class_columns
  identifies the full set of unique column definitions used in the SDTM domains. This data set is unique to SDTM.

  Reference_tables
  provides metadata for the specific data sets (domains) that are supported for CDISC SDTM. This information is different for CDISC SDTM 3.1.1 and CDISC SDTM 3.1.2.

  Reference_columns
  provides metadata for the specific columns in the domains that are supported for CDISC SDTM. This information is different for CDISC

SDTM 3.1.1 and CDISC SDTM 3.1.2.

  - The **programs** folder contains several properties files that specify generic SAS Clinical Standards Toolkit and specific CDISC SDTM properties translated into SAS global macro variables for a SAS Clinical Standards Toolkit process.
  - The **validation/control** folder provides check metadata that is associated with the primary CDISC SDTM task supported by SAS Clinical Standards Toolkit.

  Each of these folders is discussed in greater detail in this document.

- A logical set of files from multiple SAS libraries and multiple standards as defined in the previous two definitions. These are all collated within a single SASReferences data set.

  Each reference standard can be defined by the files itemized in a SASReferences data set and used to perform a standard task. The SASReferences data set documents all of the input and output files that are associated with a SAS Clinical Standards Toolkit process. These files do not need to be limited to a single standard or be resident in a single standard folder hierarchy. Consider a SASReferences data set that supports a process that builds a CDISC CRT-DDS define.xml file. That SASReferences data set might point to CDISC SDTM source data and metadata, a CDISC Terminology SAS format catalog, a set of reference table and column metadata documenting the SAS data sets used to build the define.xml file, and a default style sheet for the generated define.xml file. A broader view of what comprises the CDISC CRT-DDS reference standard must recognize that the standard also references data and metadata from other standards.

Best Practice Recommendation: Instead of changing an existing SAS standard, you should define a new standard. This allows seamless updates to SAS standards, which facilitates operational qualification, demo scripts, and Technical Support debugging a fixed standard. There is a way for you to request a change to an existing standard if there are errors. To define a new standard, which can be just changing an existing standard and saving it as a new standard, see Chapter 2, “Framework,” on page 5.

CDISC SDTM 3.1.1

Purpose

CDISC SDTM defines a standard structure for data tabulations that are submitted as part of a product application to a regulatory authority such as the FDA. The data sets and columns required for a regulatory application are not prescribed by the standard. Instead, these requirements are based on the trial protocol and discussions with the regulatory authority in charge of reviewing the submission. Therefore, any SAS Clinical Standards Toolkit standard, including any CDISC SDTM standard, is only a representative sample or template.

Release Dates

CDISC SDTM 3.1.1

- CDISC SDTM Model, Final Version 1.1, May 4, 2005
- CDISC SDTM Implementation Guide, Final Version 3.1.1, September 8, 2005

CDISC SDTM 3.1.2

- CDISC SDTM Model, Final Version 1.2, November 12, 2008
- CDISC SDTM Implementation Guide, Final Version 3.1.2, November 12, 2008

CDISC SDTM 3.1.1 Reference Standard

The CDISC SDTM 3.1.1 SAS Clinical Standards Toolkit reference standard includes the following 25 domains:

Table 4.1 *CDISC SDTM 3.1.1 Reference Standard Domains*

CDISC SDTM 3.1.1 Domains	
Special Purpose Domains	**Events**
Demographics-DM	Adverse Events-AE
Comments-CO	Disposition-DS
	Medical History-MH
Interventions	Protocol Deviations-DV
Concomitant Medications-CM	
Exposure-EX	**Trial Design Domains**
Substance Use-SU	Trial Elements-TE
	Trial Arms-TA
	Trial Visits-TV
Findings	Subject Elements-SE
ECG Tests-EG	Subject Visits-SV
Inclusion/Exclusion Exceptions-IE	Trial Inclusion/Exclusion Criteria-TI
Laboratory Tests-LB	Trial Summary-TS
Questionnaires-QS	
Physical Examinations-PE	**Special Purpose Relationship Data Sets**
Subject Characteristics-SC	Supplemental Qualifiers-SUPPQUAL
Vital Signs-VS	Related Records-RELREC

Within these 25 domains, 495 columns have been defined.

Description

CDISC standards allow for the inclusion and exclusion of some columns. (For example, timing variables can be included or excluded.) In addition, CDISC standards do not specify a length for most columns. Therefore, any implementation of a CDISC standard requires interpretation of that standard. This interpretation might lead to differences in the implementation of that standard. Reference standards are derived based on internal conventions and experiences, and discussions with regulatory authorities.

The domain and column metadata that constitute the SAS representation of CDISC SDTM 3.1.1 are derived from the global standards library in these formats:

- as empty data sets (using the utility macro cst_createTablesForDataStandard)
- as table metadata (reference_tables in the standard metadata folder)
- as column metadata for each domain (reference_columns in the standard metadata folder)

The SAS Clinical Standards Toolkit CDISC SDTM reference standard provides metadata and code to validate the structure and content of the SDTM domains. To enable validation, supplemental files supporting SDTM validation processes include the following global standards library files:

- The Validation Master data set in the **validation/control** folder contains the super-set of checks validating domain structure and content.
- The Messages data set in the **messages** folder provides error messaging for all Validation Master checks.
- SAS code in the **macros** folder provides code specific to SDTM that augments code that is provided in the primary SAS Clinical Standards Toolkit autocall library (**!sasroot/cstframework/sasmacro**).

It is this set of files, in whole or in part, that defines the CDISC SDTM reference standard.

CDISC SDTM 3.1.2 Reference Standard

The CDISC SDTM 3.1.2 SAS Clinical Standards Toolkit reference standard includes the following 32 domains:

Table 4.2 CDISC SDTM 3.1.2 Reference Standard Domains

CDISC SDTM 3.1.2 Domains	
Special Purpose Domains	**Events**
Demographics-DM	Adverse Events-AE
Comments-CO	Clinical Events-CE
Subject Elements-SE	Disposition-DS
Subject Visits-SV	Protocol Deviations-DV
	Medical History-MH

CDISC SDTM 3.1.2 Domains	
Findings	
Drug Accountability-DA	**Interventions**
ECG Tests-EG	Concomitant Medications-CM
Inclusion/Exclusion Criterion Not Met-IE	Exposure-EX
Laboratory Test Results-LB	Substance Use-SU
Microbiology Specimen-MB	
Microbiology Susceptibility Test-MS	**Trial Design Domains**
PK Concentrations-PC	Trial Arms-TA
Physical Examinations-PE	Trial Elements-TE
PK Parameters-PP	Trial Inclusion/Exclusion Criteria-TI
Questionnaires-QS	Trial Summary-TS
Subject Characteristics-SC	Trial Visits-TV
Vital Signs-VS	
	Relationship Data Sets
Findings About	Supplemental Qualifiers-SUPPQUAL
Findings About-FA	Related Records-RELREC

Within these 32 domains, 723 columns have been defined.

Description

CDISC standards allow for the inclusion and exclusion of some columns. (For example, timing variables can be included or excluded.) CDISC standards do not specify a length for most columns. Therefore, any implementation of a CDISC standard requires interpretation of that standard. This interpretation might lead to differences in the implementation of that standard. Reference standards are derived based on internal conventions and experiences, and discussions with regulatory authorities.

The domain and column metadata that constitute the SAS representation of CDISC SDTM 3.1.2 are derived from the global standards library in the following formats:

- as empty data sets (using the utility macro cst_createTablesForDataStandard)
- as table metadata (reference_tables in the standard metadata folder (see the example in the following table))
- as column metadata for each domain (reference_columns in the standard metadata folder (see the example in Table 4.4 on page 42))

Table 4.3 *Sample Reference_Tables Record (CDISC SDTM 3.1.2)*

Column Name	Column Value
sasref	REFDATA
table	CE
label	Clinical Events
class	Events
xmlpath	.../transport/ce.xpt
xmltitle	Clinical Events SAS transport file
structure	One record per event per subject
purpose	Tabulation
keys	STUDYID USUBJID CETERM CESTDTC
state	Final
date	November 12, 2008
standard	CDISC-SDTM
standardversion	3.1.2
standardref	
comment	

Table 4.4 *Sample Reference_Columns Record (CDISC SDTM 3.1.2)*

Column Name	Column Value
sasref	REFDATA
table	SU
column	SUSTRF
label	Start Relative to Reference Period
order	32
type	C
length	20
displayformat	

Column Name	Column Value
xmldatatype	text
xmlcodelist	STENRF
core	Perm
origin	Derived
role	Timing
term	** BEFORE, DURING, AFTER
algorithm	
qualifiers	UPPERCASE
standard	CDISC-SDTM
standardversion	3.1.2
standardref	SDTMIG4.1.4.7
comment	Identifies the start of the substance use period with respect to the sponsor-defined reference period. Sponsors should define the reference period in the study metadata. SUSTRF should be populated when a start date is not collected. If information such as PRIOR, ONGOING, or CONTINUING was collected, then this information should be translated into SUSTRF.

The SAS Clinical Standards Toolkit CDISC SDTM reference standard provides metadata and code to validate the structure and content of the SDTM domains. To enable validation, supplemental files supporting SDTM validation processes include the following global standards library files:

- The Validation Master data set in the **`validation/control`** folder contains the super-set of checks validating domain structure and content.
- The Messages data set in the **`messages`** folder provides error messaging for all Validation Master checks.
- SAS code in the **`macros`** folder provides code specific to SDTM that augments code that is provided in the primary SAS Clinical Standards Toolkit autocall library (**`!sasroot/cstframework/sasmacro`**).

It is this set of files, in whole or in part, that defines each CDISC SDTM reference standard.

CDISC CRT-DDS 1.0

Purpose

The CDISC CRT-DDS standard defines the metadata structures in a machine-readable XML format. These metadata structures are used to describe the CRT data sets and variables for regulatory submissions. The XML schema that is used to define the metadata structures in an XML format is based on an extension to the CDISC Operational Data Model (ODM).

Release Date

CDISC CRT-DDS, Final Version 1.0, February 10, 2005

Regulatory Basis

(Source: CDISC Case Report Tabulation Data Definition Specification)

In 1999, the FDA standardized the submission of clinical and non-clinical data and metadata in a set of eSubmission guidelines to include metadata descriptions of the data sets and columns within a Data Definition Document (define.pdf). In 2003, the FDA published a set of guidance documents on receiving electronic product applications per the International Conference on Harmonisation (ICH) electronic Common Technical Document (eCTD) specifications. In these specifications, the FDA expanded the acceptable file types to include the XML format.

CDISC CRT-DDS 1.0 Reference Standard

The domain and column metadata that constitute the SAS representation of CDISC CRT-DDS 1.0 are derived from the global standards library in these formats:

- as empty data sets (using the utility macro cst_createTablesForDataStandard)
- as table metadata for 39 data sets (reference_tables in the standard metadata folder (see the example in the following table))
- as column metadata for 176 columns in the 39 data sets (reference_columns in the standard metadata folder (see the example in Table 4.6 on page 45))

Table 4.5 *Sample Reference_Tables Record (CDISC CRT-DDS 1.0)*

Column Name	Column Value
sasref	REFDATA
table	ItemGroupDefs
label	
keys	OID
standard	CDISC-CRTDDS

Column Name	Column Value
standardversion	1.0
standardref	
comment	
xmlelementname	ItemGroupDefs
class	ItemGroupDefs
qualifiers	

Table 4.6 *Sample Reference_Columns Record (CDISC CRT-DDS 1.0)*

Column Name	Column Value
sasref	REFDATA
table	DefineDocument
column	FileType
label	File type (Snapshot | Transactional)
order	5
type	C
length	13
displayformat	$13.
standard	CDISC-CRTDDS
standardversion	1.0
standardref	
comment	
core	Req
xmlcodelist	FILETYPE
qualifiers	

As a general rule, the SAS representation of the CDISC CRT-DDS standard is patterned to match the XML element (data set) and attribute (column) structure of define.xml. For example, for CDISC SDTM, domain-level metadata is represented by a define.xml

ItemGroupDef element. This metadata is captured in the ItemGroupDefs SAS data set. This is shown in the following code and table that represent the **TE** domain metadata:

```
<ItemGroupDef OID="docroot.IG.TE"
    Name="TE"
    Repeating="No"
    IsReferenceData="Yes"
    Purpose="Tabulation"
    def:Label="Trial Elements"
    def:Structure="One record per planned element"
    def:DomainKeys="STUDYID,ETCD"
    def:Class="Trial Design"
    def:ArchiveLocationID="ArchiveLocation.te">
    !-- All ItemRefs would be listed here -->
    <def:leaf ID="ArchiveLocation.te"
    xlink:href="te.xpt"> <def:title>te.xpt</def:title>
  </def:leaf>
</ItemGroupDef>
```

Table 4.7 *Sample Data Set Representation: ItemGroupDefs.sas7bdat*

Column	Value
OID	docroot.IG.TE
Name	TE
Repeating	No
IsReferenceData	Yes
SASDatasetName	
Domain	
Origin	
Role	
Purpose	Tabulation
Comment	
Label	Trial Elements
Class	Trial Design
Structure	One record per planned element
DomainKeys	STUDYID, ETCD
ArchiveLocationID	ArchiveLocation.te
FK_MetaDataVersion	

Note: Empty or null attributes are not typically included in the XML file.

The following table lists the complete set of 39 tables that form the SAS Clinical Standards Toolkit SAS representation of the CDISC CRT-DDS 1.0 standard.

***Table 4.8** Data Sets in the SAS Representation of the CDISC CRT-DDS 1.0 Standard*

Table	Table
AnnotatedCRFs	ItemQuestionTranslatedText
CLItemDecodeTranslatedText	ItemRangeCheckValues
CodeListItems	ItemRangeChecks
CodeLists	ItemRole
ComputationMethods	ItemValueListRefs
DefineDocument	MDVLeaf
ExternalCodeLists	MDVLeafTitles
FormDefArchLayouts	MUTranslatedText
FormDefItemGroupRefs	MeasurementUnits
FormDefs	MetaDataVersion
ImputationMethods	Presentation
ItemAliases	ProtocolEventRefs
ItemDefs	RCErrorTranslatedText
ItemGroupAliases	Study
ItemGroupDefItemRefs	StudyEventDefs
ItemGroupDefs	StudyEventFormRefs
ItemGroupLeaf	SupplementalDocs
ItemGroupLeafTitles	ValueListItemRefs
ItemMURefs	ValueLists
ItemQuestionExternal	

The highly structured nature of CDISC CRT-DDS data requires that any mapping to a relational format include a large number of data sets, with foreign key relationships to help preserve the intended non-relational object structure. In the SAS Clinical Standards Toolkit, foreign key relationships are enforced when validating the CDISC CRT-DDS data sets.

Field lengths in the CDISC CRT-DDS data sets are consistent by core data type. CDISC has not specified any limit to the length of most character fields. Arbitrary lengths have been chosen by data type. These lengths are listed in the following table. In the table, standard data types are distilled into core data types. To be safe, larger lengths have been chosen to ensure that no data loss occurs in the SAS Clinical Standards Toolkit pre-installed data sets. Production tables might be compressed using SAS mechanisms to preserve disk space.

Table 4.9 *CDISC CRT-DDS Default Lengths by Data Type*

Type Name	Length	Description
oid	128	A unique object identifier or a reference
text	2000	A character field that can accommodate a large number of characters
name	128	A descriptive identifier
value	512	An item of collected or reference data
path	512	An absolute or relative file system path or URL

The following table lists the data sets with member columns that form the CDISC CRT-DDS 1.0 data in the SAS Clinical Standards Toolkit.

No data set has more than one variable that acts as the key or index for that data set. The names of key variables are prepended with two asterisks (**). Some data sets do not have a key.

Foreign key variable names are prepended with two carat characters (^^). Foreign key variable names reference, in brackets [ ], the name of the data set for which it is a foreign key.

Required fields are marked with an X between brackets [X]. Required fields are fields for which a non-nil and non-whitespace-only value must be supplied in any observation for that data set.

Only the DefineDocument data set, which contains valid values for the FileOID and FileType variables, is needed to create a minimal, but valid CDISC CRT-DDS-compliant XML document. This is based on the CDISC CRT-DDS standard, which is very flexible.

All table and column names are case sensitive. They must be specified exactly as shown.

Table 4.10 *CDISC CRT-DDS SAS Table Construction*

Data Set Name	Variable Name	SAS Data Type	Length (if char)
DefineDocument			
	**FileOID [X]	character	128 (oid)
	Archival	character	3
	AsOfDateTime	character	24

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	Description	character	2000 (text)
	FileType [X]	character	13
	Granularity	character	15
	Id	character	128 (oid)
	ODMVersion	character	2000 (text)
	Originator	character	2000 (text)
	PriorFileOID	character	128 (oid)
	SourceSystem	character	2000 (text)
	SourceSystemVersion	character	2000 (text)
Study			
	**OID [X]	character	128 (oid)
	StudyName [X]	character	128 (name)
	StudyDescription [X]	character	2000 (text)
	ProtocolName [X]	character	128 (name)
	^^FK_DefineDocument [DefineDocument] [X]	character	128 (oid)
MeasurementUnits			
	**OID [X]	character	128 (oid)
	Name [X]	character	128 (name)
	^^FK_Study [Study] [X]	character	128 (oid)
MUTranslatedText			

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	TranslatedText	character	2000 (text)
	lang	character	128 (name)
	^^FK_MeasurementUnits [MeasurementUnits][X]	character	128 (oid)
MetaDataVersion			
	**OID [X]	character	128 (oid)
	Name [X]	character	128 (name)
	Description	character	2000 (text)
	IncludedOID	character	128 (oid)
	IncludedStudyOID	character	128 (oid)
	DefineVersion [X]	character	2000 (text)
	StandardName [X]	character	2000 (text)
	StandardVersion [X]	character	2000 (text)
	^^FK_Study [Study] [X]	character	128 (oid)
AnnotatedCRFs			
	DocumentRef	character	2000 (text)
	^^leafID [MDVLeaf] [X]	character	128 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
SupplementalDocs			
	DocumentRef	character	2000 (text)
	^^leafID [MDVLeaf] [X]	character	128 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
MDVLeaf			
	**ID [X]	character	128 (oid)
	href	character	512 (path)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
MDVLeafTitles			
	title	character	2000 (text)
	^^FK_MDVLeaf [MDVLeaf] [X]	character	128 (oid)
ComputationMethods			
	**OID [X]	character	128 (oid)
	method	character	2000 (text)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
ValueLists			
	**OID [X]	character	128 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
ValueListItemRefs			
	^^ItemOID [ItemDefs] [X]	character	128 (oid)
	OrderNumber	numeric	8
	Mandatory [X]	character	3
	KeySequence	numeric	8
	^^ImputationMethodOID [ImputationMethods]	character	128 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	Role	character	128 (name)
	^^RoleCodeListOID [CodeLists]	character	128 (oid)
	^^FK_ValueLists [ValueLists] [X]	character	128 (oid)
ProtocolEventRefs			
	Mandatory [X]	character	3
	OrderNumber	numeric	8
	^^StudyEventOID [StudyEventDefs] [X]	character	128 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
StudyEventDefs			
	**OID [X]	character	128 (oid)
	Category	character	2000 (text)
	Name [X]	character	128 (name)
	Repeating [X]	character	3
	Type [X]	character	11
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
StudyEventFormRefs			
	^^FormOID [FormDefs] [X]	character	128 (oid)
	Mandatory [X]	character	3
	OrderNumber	numeric	8
	^^FK_StudyEventDefs [StudyEventDefs] [X]	character	128 (oid)
FormDefs			
	**OID [X]	character	128 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	Name [X]	character	128 (name)
	Repeating [X]	character	3
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
FormDefItemGroupRefs			
	^^ItemGroupOID [ItemGroupDefs] [X]	character	128 (oid)
	Mandatory [X]	character	3
	OrderNumber	numeric	8
	^^FK_FormDefs [FormDefs] [X]	character	128 (oid)
FormDefArchLayouts			
	**OID [X]	character	128 (oid)
	PdfFileName [X]	character	512 (path)
	^^PresentationOID [Presentation]	character	128 (oid)
	^^FK_FormDefs [FormDefs] [X]	character	128 (oid)
ItemGroupDefs			
	**OID [X]	character	128 (oid)
	Name [X]	character	128 (name)
	Repeating [X]	character	3
	IsReferenceData	character	3
	SASDatasetName	character	8
	Domain	character	2000 (text)
	Origin	character	2000 (text)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	Role	character	128 (name)
	Purpose	character	2000 (text)
	Comment	character	2000 (text)
	Label [X]	character	2000 (text)
	Class	character	2000 (text)
	Structure	character	2000 (text)
	DomainKeys	character	2000 (text)
	^^ArchiveLocationID [ItemGroupLeaf] [X]	character	128 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
ItemGroupDefItemRefs			
	^^ItemOID [ItemDefs] [X]	character	128 (oid)
	Mandatory [X]	character	3
	OrderNumber	numeric	8
	KeySequence	numeric	8
	^^ImputationMethodOID [ImputationMethods]	character	128 (oid)
	Role [X]	character	128 (name)
	^^RoleCodeListOID [CodeLists]	character	128 (oid)
	^^FK_ItemGroupDefs [ItemGroupDefs][X]	character	128 (oid)
ItemGroupAliases			
	Context [X]	character	2000 (text)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	Name [X]	character	2000 (text)
	^^FK_ItemGroupDefs [ItemGroupDefs] [X]	character	128 (oid)
ItemGroupLeaf			
	**ID [X]	character	128 (oid)
	href	character	512 (path)
	^^FK_ItemGroupDefs [ItemGroupDefs] [X]	character	128 (oid)
ItemGroupLeafTitles			
	title	character	2000 (text)
	^^FK_ItemGroupLeaf [ItemGroupLeaf] [X]	character	128 (oid)
ItemDefs			
	**OID [X]	character	128 (oid)
	Name [X]	character	128 (name)
	DataType [X]	character	8
	Length	numeric	8
	SignificantDigits	numeric	8
	SASFieldName	character	8
	SDSVarName	character	8
	Origin	character	2000 (text)
	Comment	character	2000 (text)
	^^CodeListRef [CodeLists]	character	128 (oid)
	Label	character	2000 (text)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	DisplayFormat	character	2000 (text)
	^^ComputationMethodOID[ComputationMethods]	character	128 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
ItemQuestionTranslatedText			
	TranslatedText	character	2000 (text)
	lang	character	17
	^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)
ItemQuestionExternal			
	Dictionary	character	2000 (text)
	Version	character	2000 (text)
	Code	character	2000 (text)
	^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)
ItemMURefs			
	^^MeasurementUnitOID [MeasurementUnits] [X]	character	128 (oid)
	^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)
ItemRangeChecks			
	**OID [X]	character	128 (oid)
	Comparator [X]	character	5
	SoftHard [X]	character	4
	^^MURefOID [MeasurementUnits]	character	128 (oid)
	^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)

Data Set Name	**Variable Name**	**SAS Data Type**	**Length (if char)**
ItemRangeCheckValues			
	CheckValue	character	512 (value)
	^^FK_ItemRangeChecks [ItemRangeChecks] [X]	character	128 (oid)
RCErrorTranslatedText			
	TranslatedText	character	2000 (text)
	lang	character	17
	^^FK_ItemRangeChecks [ItemRangeChecks] [X]	character	128 (oid)
ItemRole			
	Name	character	2000 (text)
	^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)
ItemAliases			
	Context [X]	character	2000 (text)
	Name [X]	character	2000 (text)
	^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)
ItemValueListRefs			
	^^ValueListOID [ValueLists] [X]	character	128 (oid)
	^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)
CodeLists			
	**OID [X]	character	128 (oid)
	Name [X]	character	128 (name)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	DataType [X]	character	7
	SASFormatName	character	8
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
ExternalCodeLists			
	Dictionary	character	2000 (text)
	Version	character	2000 (text)
	^^FK_CodeLists [CodeLists] [X]	character	128 (oid)
CodeListItems			
	**OID [X]	character	128 (oid)
	CodedValue	character	512 (value)
	^^FK_CodeLists [CodeLists] [X]	character	128 (oid)
	Rank	numeric	8
CLItemDecodeTranslatedText			
	TranslatedText	character	2000 (text)
	lang	character	17
	^^FK_CodeListItems [CodeListItems] [X]	character	128 (oid)
ImputationMethods			
	**OID [X]	character	128 (oid)
	method	character	2000 (text)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)
Presentation			

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	**OID [X]	character	128 (oid)
	presentation	character	2000 (text)
	lang	character	17
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

The SAS Clinical Standards Toolkit CDISC CRT-DDS reference standard supports reading and representing in SAS a define.xml file, building a define.xml file, and validating the structure and content of the SAS representation of a define.xml file. In addition, it validates the structural integrity of the define.xml file. To support this functionality, supplemental files include the following global standards library files:

- A SAS format catalog (crtddsct.sas7bcat) in the **formats** folder provides valid values for selected columns in the 39 tables of the SAS representation.
- The Validation Master data set in the **validation/control** folder contains the super-set of checks validating the structure and content of the 39 tables.
- The Messages data set in the **messages** folder provides error messaging for all Validation Master checks.
- SAS code in the **macros** folder provides CDISC CRT-DDS-specific code that augments code that is provided in the primary SAS Clinical Standards Toolkit autocall library (**!sasroot/cstframework/sasmacro**).
- The **style sheet** folder contains the define1-0-0.xsl file. The style sheet is copied from **http://www.cdisc.org/stuff/contentmgr/files/0/464923b10ea16b477151fcaa9f465166/misc/define1_0_0.xsl**. A define.xml file can be rendered in a human-readable form if it contains an explicit XML style sheet reference, such as a reference to the default style sheet.

It is this set of files, in whole or in part, that defines the CDISC CRT-DDS reference standard.

CDISC ODM 1.3.0

Purpose

(Source: CDISC Web site **http://www.cdisc.org/odm**)

The CDISC ODM standard facilitates the archival and interchange of the metadata and data for clinical research. ODM is a vendor-neutral, platform-independent format for the interchange and archival of clinical study data. ODM includes the clinical data and its associated metadata, administrative data, reference data, and audit information. All of the information that needs to be shared during setup, operation, analysis, and submission, as well as for long-term retention as part of an archive, is included in ODM.

Release Date

CDISC ODM, Version 1.3.0, December 15, 2006

CDISC ODM 1.3.0 Reference Standard

SAS Clinical Standards Toolkit 1.3 provides only partial support of CDISC ODM 1.3.0. The current release of SAS Clinical Standards Toolkit supports reading an odm.xml file, and translating the metadata (<Study>) and clinical data (<ClinicalData>) sections of the odm.xml file into a SAS representation. The domain and column metadata that constitute the SAS representation of CDISC ODM 1.3.0 are derived from the global standards library in these formats:

- as empty data sets (using the utility macro cst_createTablesForDataStandard)
- as table metadata for 52 data sets (reference_tables in the standard metadata folder (see the example in the following table))
- as column metadata for 241 columns in the 52 data sets (reference_columns in the standard metadata folder (see the example in Table 4.12 on page 60))

Table 4.11 *Sample Reference_Tables Record (CDISC ODM 1.3.0)*

Column Name	Column Value
sasref	REFDATA
table	ItemGroupData
label	Item group-level data information
keys	OID
standard	CDISC-ODM
standardversion	1.3.0
standardref	
comment	
xmlelementname	ItemGroupData
class	ItemGroupData

Table 4.12 *Sample Reference_Columns Record (CDISC ODM 1.3.0)*

Column Name	Column Value
sasref	REFDATA
table	SubjectData

Column Name	Column Value
column	SubjectKey
label	Uniquely identifies a subject in a study
order	2
type	C
length	2000
displayformat	$2000.
standard	CDISC-ODM
standardversion	1.3.0
standardref	
comment	
core	Req
xmlcodelist	
qualifier	

As a general rule, the SAS representation of the CDISC ODM standard is patterned to match the XML element (data set) and attribute (column) structure of odm.xml. For example, consider the following XML extract:

```
<ClinicalData StudyOID="P2006-101" MetadataVersionOID="101.01">
 <SubjectData SubjectKey="1000" TransactionType="Insert">
  <StudyEventData StudyEventOID="101.Screen">
   <FormData FormOID="101.DEMOG">
    <ItemGroupData ItemGroupOID="101.DM">
     <ItemDataString ItemOID="101.USUBJID">101-01-01</ItemDataString>
     <ItemDataString ItemOID="101.SEX">F</ItemDataString>
    </ItemGroupData>
   </FormData>
  </StudyEventData>
 </SubjectData>
</ClinicalData>
```

The following table describes how the XML element and attribute information maps to the SAS representation.

Table 4.13 Sample Mapping of odm.xml File to SAS Representation

XML Element or Attribute	SAS Data Set	SAS Column	SAS Column Value
<ClinicalData StudyOID="P2006-101" MetadataVersionOID="101.01">	ClinicalData	StudyOID MetaDataVersionOID	"P2006-101" "101.01"
<SubjectData SubjectKey="1000" TransactionType="Insert">	SubjectData	SubjectKey TransactionType	"1000" "Insert"
<StudyEventData StudyEventOID="101.Screen">	StudyEventData	StudyEventOID	"101.Screen"
<FormData FormOID="101.DEMOG">	FormData	FormOID	"101.DEMOG"
<ItemGroupData ItemGroupOID="101.DM">	ItemGroupData	ItemGroupOID	"101.DM"
<ItemDataString ItemOID="101.USUBJID">101-01-01 </ItemDataString>	ItemData	ItemOID ItemDataType Value	"101.USUBJID" "ItemDataString" "101-01-01"
<ItemDataString ItemOID="101.SEX">F</ ItemDataString>	ItemData	ItemOID ItemDataType Value	"101.SEX" "ItemDataString" "F"

The following table lists the complete set of 52 tables that form the SAS Clinical Standards Toolkit SAS representation of the CDISC ODM 1.3.0 standard.

Table 4.14 Data Sets in the SAS Representation of the CDISC ODM 1.3.0 Standard

Table	Table
Annotations	ItemMURefs
AuditRecords	ItemQuestionExternal
CLItemDecodeTranslatedText	ItemQuestionTranslatedText
ClinicalData	ItemRCFormalExpression
CodeListItems	ItemRangeCheckValues
CodeLists	ItemRangeChecks
ConditionDefFormalExpression	ItemRole
ConditionDefTranslatedText	MUTranslatedText
ConditionDefs	MeasurementUnits
EnumeratedItems	MetaDataVersion

Table	**Table**
ExternalCodeLists	MethodDefFormalExpression
FormData	MethodDefTranslatedText
FormDefArchLayouts	MethodDefs
FormDefItemGroupRefs	ODM
FormDefTranslatedText	Presentation
FormDefs	ProtocolEventRefs
ImputationMethods	ProtocolTranslatedText
ItemAliases	RCErrorTranslatedText
ItemData	ReferenceData
ItemDefTranslatedText	SignatureDefs
ItemDefs	Signatures
ItemGroupAliases	Study
ItemGroupData	StudyEventData
ItemGroupDefItemRefs	StudyEventDefs
ItemGroupDefTranslatedText	StudyEventFormRefs
ItemGroupDefs	SubjectData

The highly structured nature of CDISC ODM data requires that any mapping to a relational format include a large number of data sets, with foreign key relationships to help preserve the intended non-relational object structure. In the SAS Clinical Standards Toolkit, foreign key relationships are enforced when validating the CDISC ODM data sets.

Field lengths in the CDISC ODM data sets are consistent by core data type. CDISC has not specified any limit to the length of most character fields. Arbitrary lengths have been chosen by data type. These lengths are listed in the following table. In the table, standard data types are distilled into core data types. To be safe, larger lengths have been chosen to ensure that no data loss occurs in the SAS Clinical Standards Toolkit pre-installed data sets. Production tables might be compressed using SAS mechanisms to preserve disk space.

Table 4.15 *CDISC ODM Default Lengths by Data Type*

Type Name	**Length**	**Description**
oid	128	A unique object identifier or a reference
text	2000	A character field that can accommodate a large number of characters

Type Name	Length	Description
name	128	A descriptive identifier
value	512	An item of collected or reference data
path	512	An absolute or relative file system path or URL

The table metadata for the 52 data sets and the column metadata for the 241 columns in those data sets that comprise the SAS representation of the CDISC ODM 1.3.0 standard are in the following folder:

***<global standards library directory>*/standards/cdisc-odm-1.3.0-1.3/metadata**.

Table metadata is in reference_tables.sas7bdat, and column metadata is in reference_columns.sas7bdat.

In the future, the CDISC ODM reference standard will support reading and representing in SAS a complete odm.xml file, building an odm.xml file, and validating the structure and content of the SAS representation of an odm.xml file. In addition, it will validate the structural integrity of the odm.xml file. To support this functionality, supplemental files include the following global standards library files:

- The Messages data set in the **messages** folder provides error messaging for all Validation Master checks.
- SAS code in the **macros** folder provides CDISC ODM-specific code that augments code that is provided in the primary SAS Clinical Standards Toolkit autocall library (**!sasroot/cstframework/sasmacro**).

It is this set of files, in whole or in part, that defines the CDISC ODM reference standard.

CDISC Terminology

Purpose

The CDISC Terminology standard supports standardizing values for columns in data submitted to the regulatory authorities. Standardization facilitates loads into regulatory databases, data review, and analysis. The initial standardization of values has primarily been in support of SDTM submission data and the CDISC CDASH (Clinical Data Acquisition Standards Harmonization) development of standardized data collection instruments.

Release Dates

CDISC-Terminology-200810	Comprised of SDTM Packages 1, 2A, and 2B, and Labtest Packages 1 and 2. September 24, 2008.
CDISC-Terminology-201003	All SDTM controlled terminology developed and in production as of March 8, 2010, comprised of SDTM Packages 1, 2A, 2B, 3, and 4, and Labtest Packages 1,

2, 3, and 4. Also contains commonly used controlled terminology in the CDASH 1.0 standard.

CDISC Terminology Reference Standard

CDISC Terminology is maintained by and distributed as part of the National Cancer Institute (NCI) Enterprise Vocabulary Services (EVS) Thesaurus. For more information, see "References" on page 2. Periodically, CDISC Terminology is updated to include the work of numerous terminology project teams. Updates are in the form of new packages or sets of terminology.

SAS Clinical Standards Toolkit offers snapshots of the NCI EVS Thesaurus, These snapshots are typically coordinated with the release of other CDISC standards that use the thesaurus. Two snapshots are currently supported:

- The CDISC-Terminology-200810 snapshot was taken in October 2008 in support of SAS Clinical Standards Toolkit 1.2. This snapshot supports CDISC SDTM 3.1.1.
- The CDISC-Terminology-201003 snapshot was taken in March 2010 in support of SAS Clinical Standards Toolkit 1.3. This snapshot supports CDISC SDTM 3.1.2.

Each CDISC Terminology standard includes a SAS format catalog (**cterms.sas7bcat**) and a SAS data set (**cterms.sas7bdat**). The catalog and data set are found in the following global standards library folder (where ***xxxxxx*** is the specific snapshot (200810 or 201003):

***<global standards library directory>*/standards/cdisc-terminology-*xxxxxx*-1.3/formats**.

The following 60 code lists (SAS formats) are in the cumulative CDISC-Teminology-201003 snapshot:

Table 4.16 *Supported CDISC Terminology Code Lists/Formats*

Code List/Format Name	Description	Unique Values
ACN	Action Taken with Study Treatment	7
AESEV	Severity/Intensity Scale for Adverse Events	3
AGESPAN	Age Span	8
AGEU	Age Unit	5
COUNTRY	Country	246
DATEST	Drug Accountability Test Name	2
DATESTCD	Drug Accountability Test Code	2
DICTNAM	Dictionary Name	7
DOMAIN	Domain Abbreviation	45
DSCAT	Category for Disposition Event	3

Code List/Format Name	Description	Unique Values
EGMETHOD	ECG Test Method	22
EGSTRESC	ECG Result	109
EGTEST	ECG Test Name	46
EGTESTCD	ECG Test Code	46
ETHNIC	Ethnic Group	4
EVAL	Evaluator	15
FREQ	Frequency	50
FRM	Pharmaceutical Dosage Form	168
IECAT	Category for Inclusion/Exclusion	2
LBTEST	Laboratory Test Name	580
LBTESTCD	Laboratory Test Code	580
LOC	Anatomical Location	303
MARISTAT	Marital Status	9
METHOD	Method	65
MICROORG	Microorganism	868
MSRESCAT	Microbiology Susceptibility Testing Result Category	7
NCF	Never/Current/Former Classification	3
NCOMPLT	Completion/Reason for Non-Completion	16
ND	Not Done	1
NRIND	Reference Range Indicator	4
NY	No Yes Response	4
OUT	Outcome of Event	6
PKUNIT	PK Parameter Units of Measure	208
POSITION	Position	10
RACE	Race	5
RELTYPE	Relationship Type	2

Code List/Format Name	Description	Unique Values
ROUTE	Route of Administration	112
SCCD	Subject Characteristic Code	7
SEX	Sex	4
SEXPOP	Sex of Participants	3
SIZE	Size	3
SKINCLAS	Skin Classification	6
SKINTYP	Skin Type	3
SOC	CDISC System Organ Class	26
SPECCOND	Specimen Condition	8
SPECTYPE	Specimen Type	41
STENRF	Relation to Reference Period	7
TBLIND	Trial Blinding Schema	3
TCNTRL	Control Type	3
TDIGRP	Diagnosis Group	1
TINDTP	Trial Indication Type	5
TOXGR	Common Terminology Criteria for Adverse Events	5
TPHASE	Trial Phase	12
TSPARM	Trial Summary Parameter Test Name	24
TSPARMCD	Trial Summary Parameter Test Code	24
TTYPE	Trial Type	8
UNIT	Unit	310
VSRESU	Units for Vital Signs Results	14
VSTEST	Vital Signs Test Name	14
VSTESTCD	Vital Signs Test Code	14

Support for Upcoming Standards

From a CDISC perspective, the following standards are candidates for future SAS Clinical Standards Toolkit support. For more information, check with your on-site SAS support personnel and SAS product management. Other CDISC standards might be considered as candidates based on user requests.

CDISC ODM 1.3.0 and CDISC ODM 1.3.1

Completion of the CDISC ODM 1.3.0 standard. For example, support will include handling the AdminData and ReferenceData sections, and validating the odm.xml file and the SAS representation of CDISC ODM 1.3.1 is also a candidate for support.

CDISC ADaM 2.1

The CDISC ADaM standard defines a standard for analysis data sets that are to be submitted in support of the statistical analyses performed by the sponsor. CDISC ADaM 2.1 and its Implementation Guide were released in December 2009. CDISC ADaM 2.1 is a candidate for support.

CDISC Terminology

Updates to the NCI EVS Thesaurus for CDISC Terminology after March 8, 2010 will be packaged as a CDISC Terminology snapshot in a future SAS Clinical Standards Toolkit release. These updates are expected to support CDISC ADaM 2.1.

Chapter 5
SASReferences File

Overview

SAS Clinical Standards Toolkit supports the submission of SAS processes using predefined metadata files. These files are introduced and described in Chapter 3, “Metadata File Descriptions,” on page 21. The key metadata file that supports this functionality is the SASReferences file. This SAS data set essentially identifies all of the key inputs and outputs for any SAS Clinical Standards Toolkit process. Each unique process can have an associated, unique SASReferences file. However, SAS Clinical Standards Toolkit offers many standardization aids, so more generic SASReferences files are preferable.

The required SASReferences file structure is provided in Table 3.3 on page 26 and example content is provided in Display 3.5 on page 27.

Building a SASReferences File

Each SASReferences file requires content that is specific to its planned use. For example, a SAS Clinical Standards Toolkit process that creates a define.xml file requires the specification of XML and recommends the specification of style sheet information. A SAS Clinical Standards Toolkit process that validates data against a standard requires the specification of the validation checks to be run.

The SAS Clinical Standards Toolkit offers several ways to create a SASReferences file for use in subsequent processes.

1. Use sample SASReferences files that are provided with the SAS Clinical Standards Toolkit. These sample SASReferences files contain the required and optional contents for specific tasks. For example, the task of validating the functionality of CDISC SDTM 3.1.2 uses the SASReferences file found at the following location in SAS 9.2:

!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata/control

An excerpt of this sample SASReferences file is provided in Display 3.5 on page 27.

2. The SAS Clinical Standards Toolkit provides SASReferences templates for use. These templates are either zero-observation data sets or data sets containing records that must be modified. A SASReferences data set template can be found in:

   ***<global standards library directory>*/standards/cst-framework-1.3/templatesSAS**.

   The SAS Clinical Standards Toolkit provides default SASReferences data sets for each supported standard. These default SASReferences data sets contain records that are commonly required for certain SAS Clinical Standards Toolkit tasks (such as validation). However, all records that are required might not be included. Or, all records that are included might not be required for certain tasks. And, SAS librefs, filerefs, paths, and memname values might require modification. For example, see the StandardSASReferences data set found in:

   ***<global standards library directory>*/standards/cdisc-sdtm-3.1.2-1.3/control**.

3. The SAS Clinical Standards Toolkit provides the utility macros to build and return many SAS Clinical Standards Toolkit metadata data sets.

   - The %cst_getStandardSASReferences macro returns the StandardSASReferences data set. (See the file description in Chapter 3, “Metadata File Descriptions,” on page 21 for the specified standard.)
   - The %cst_createds macro can be used to return an empty SASReferences data set.

     Use of these utility macros is illustrated later in this chapter.

The primary function of the SASReferences file is to define the SAS Clinical Standards Toolkit process inputs and outputs. What information does the process need to reference? What does the process produce? Where does the information come from and go? The “what” information is determined by the use of two SASReferences fields—type and subtype. The “where” information is determined by path and memname. The values for all of these fields are restricted for SAS Clinical Standards Toolkit to values itemized in the framework Standardlookup data set found in:

***<global standards library directory>*/standards/cst-framework/control/standardlookup.sas7bdat**.

Customizing the type and subtype values in the Standardlookup data set is allowed. Customization is a prerequisite if you want to use the field values in any SASReferences data set that is used by the SAS Clinical Standards Toolkit.

The following table lists and describes the acceptable type and subtype values in the framework Standardlookup data set.

Table 5.1 *SAS Clinical Standards Toolkit SASReferences Type and Subtype Values*

Type	Subtype	Comments
autocall		One record for each library that contains macros to be included in the SAS autocall path. Typically, this includes one record for each standard that is referenced in the SASReferences file, excluding the SAS Clinical Standards Toolkit framework. The framework and cross-standard macros are already included in the autocall path at product deployment. User-written macros, as referenced in one or more additional code libraries, require an autocall record for each library.
classmetadata	column or table	Identifies the SAS data sets (sasref.memname) that contain the column and table metadata for specific CDISC SDTM template data sets that are used to build standard SDTM-compliant data sets. This type is provided by default in StandardSASReferences and is optional.
control	validation or reference	Identifies any run-time process control file, including the SASReferences data set itself. (In other words, it is a self-documentation record). For SAS Clinical Standards Toolkit validation processes, the Validation Control data set that specifies the validation checks to be run is identified with subtype=validation.
externalxml	xml	Identifies an external XML file. Depending on the standard version and the subsequent macro that is called, this file can be read or written. Using CDISC CRT-DDS as an example, this type specifies the define.xml file that is created when the %crtdds_write() macro is called. When the %crtdds_read() macro is supported, this type identifies the XML file to be read.

Type	Subtype	Comments
fmtsearch		Provides a way to build the format search path for a validation process. SAS Clinical Standards Toolkit sets the SAS fmtsearch type based on each record, specifying a SAS catalog that uses the order=n sequence. This type is not provided by default in StandardSASReferences, so user specification is required. The type=fmtsearch value is optional unless one or more checks are to be run that assess value compliance against a SAS format.
lookup		Identifies a data set (Standardlookup) that is associated with each SAS Clinical Standards Toolkit standard that contains valid values for discrete metadata fields. This type is provided by default in StandardSASReferences and is required for each standard. For example, the valid values for type and subtype that are documented in this table have been defined in one or more SAS Clinical Standards Toolkit Standardlookup data sets.
messages		Identifies one or more Messages data sets that are associated with each SAS Clinical Standards Toolkit standard. This type is provided by default in StandardSASReferences. User specification is necessary only with user customizations that require new or modified messages. SAS Clinical Standards Toolkit populates the data set that is referenced by the global macro variable &_cstMessages with all Messages data sets that are included in SASReferences. This type is required for each standard.
properties	validation or initialize	Initializes a standard version's required macro variables. Specification in SASReferences is optional. (These macro variables can be defined with calls to %cst_setstandardproperties or %cst_setproperties instead.) Each standard should have at least one properties (initialize) file. Each standard can have any additional files that are needed. A subtype=validation value is specific to SAS Clinical Standards Toolkit validation processes.

Type	Subtype	Comments
referencecontrol	validation or standardref	If subtype=validation, then the value identifies the standard-supplied master super-set of supported validation checks. While this is key metadata, it is not typically referenced at run time and does not need to be included. It is the Validation Control file that is identified with type=control and subtype=validation that must be included. If subtype=standardref, then the value identifies an optional data set that contains a list of references that provide the basis for each validation check that is included in the subtype=validation data set.
referencecterm		Identifies a SAS data set (sasref.memname) that most often contains controlled terminology, as opposed to a SAS format containing controlled terminology (for example, medDRA). The type=referencecterm value is optional unless one or more checks are to be run that assess value compliance against a SAS data set.
referencemetadata	column or table	Identifies the SAS data sets (sasref.memname) that contain the column and table metadata for a standard version. This type is provided by default in StandardSASReferences, so user specification is required only to override the default for the standard. Records for both subtypes are required.
referencexml	stylesheet or map	If subtype=stylesheet, then this value identifies the directory and filename of an XML style sheet. In the production of CDISC CRT-DDS XML files, this value should point to the style sheet to be copied into the directory with the XML file. If subtype=map, then this value identifies the persisted location of a SAS XML map file. The SAS XML map file reads the Work cube.xml file generated by SAS Clinical Standards Toolkit that translates an XML file into the SAS representation of the XML-based standard (such as CDISC CRT-DDS and CDISC ODM).

Type	Subtype	Comments
report	library or outputfile	Specifies the storage location of the SAS Clinical Standards Toolkit process reports. If a single, specific report is referenced, then it can be specified with a subtype of outputfile, a valid path, and valid memname values. If the process produces multiple reports, then a subtype of library is used with a valid path to the directory or folder. In the latter case, default report names as defined in the code are used.
results	results or validationresults, metrics or validationmetrics	Specifies the storage location of the Results and Metrics data sets that are generated by the SAS Clinical Standards Toolkit process. The Metrics data set is specific to SAS Clinical Standards Toolkit validation processes and is optional depending on property settings. A **`results/ validationresults`** record is required.
resultspackage	xml or log	This type is not used in SAS Clinical Standards Toolkit 1.3. This type bundles a set of process inputs and outputs together for later access.
sourcedata		Defines the folder location of the data for a specific study. This type is required for validation processes if one or more checks are to be run that access a specific source data domain.
sourcemetadata	column, table, or study	Identifies the SAS data sets (sasref.memname) that contain the column and table metadata for a study or set of source data. This type is not provided by default in StandardSASReferences so user specification is required. Records for both subtypes are required.
standards	registeredstandards or registeredsasreferences	Identifies the template for the registered Standards and SASReferences data sets, respectively. This value is used by the framework when the global metadata library is created. This type is not used in post-deployment processes.

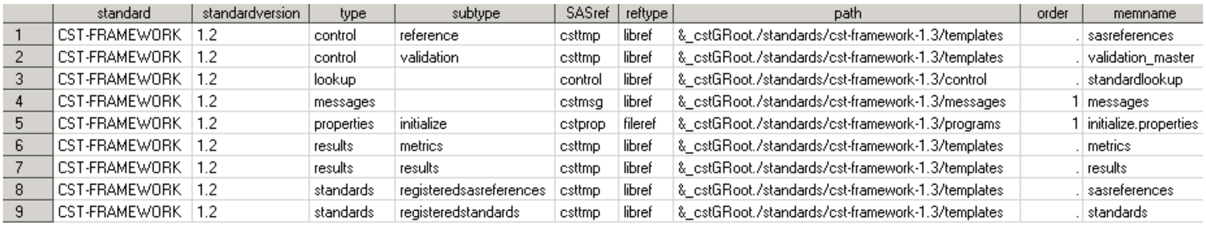
Type	Subtype	Comments
targetdata		Defines the location of the data to be derived for a specific standard. For example, for CDISC CTR-DDS, the crtdds_read macro derives a set of CRT-DDS data sets from the referenced define.xml file. This type is optional.
targetmetadata	column, table, or study	Identifies the SAS data sets (sasref.memname) that contain the column, table, and study metadata to be derived for a specific standard. For example, for CDISC CRT-DDS, the crtdds_read macro derives files that describe metadata about the targetdata data sets that are derived from the referenced define.xml file. If this type is used, then a record for each subtype is required.
transport		This type is not used in SAS Clinical Standards Toolkit 1.3. This type identifies a library of SAS transport files that are optionally referenced by a define.xml file.

Every instance of the SASReferences file does not require a specific path and filename. At the beginning of this section, a call to the following macro was described:

```
%cst_getStandardSASReferences(_cstStandard=CST-FRAMEWORK,_cstStandardVersion=1.2,
_cstOutputDS=sasreferences);
```

This macro call produces the following SASReferences file:

Display 5.1 *Standard SASReferences File for CST-FRAMEWORK*



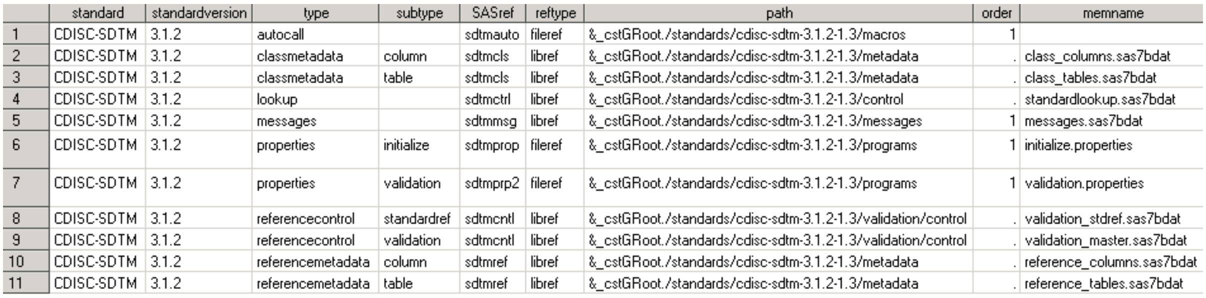
	standard	standardversion	type	subtype	SASref	reftype	path	order	memname
1	CST-FRAMEWORK	1.2	control	reference	csttmp	libref	&_cstGRoot./standards/cst-framework-1.3/templates	.	sasreferences
2	CST-FRAMEWORK	1.2	control	validation	csttmp	libref	&_cstGRoot./standards/cst-framework-1.3/templates	.	validation_master
3	CST-FRAMEWORK	1.2	lookup		control	libref	&_cstGRoot./standards/cst-framework-1.3/control	.	standardlookup
4	CST-FRAMEWORK	1.2	messages		cstmsg	libref	&_cstGRoot./standards/cst-framework-1.3/messages	1	messages
5	CST-FRAMEWORK	1.2	properties	initialize	cstprop	fileref	&_cstGRoot./standards/cst-framework-1.3/programs	1	initialize.properties
6	CST-FRAMEWORK	1.2	results	metrics	csttmp	libref	&_cstGRoot./standards/cst-framework-1.3/templates	.	metrics
7	CST-FRAMEWORK	1.2	results	results	csttmp	libref	&_cstGRoot./standards/cst-framework-1.3/templates	.	results
8	CST-FRAMEWORK	1.2	standards	registeredsasreferences	csttmp	libref	&_cstGRoot./standards/cst-framework-1.3/templates	.	sasreferences
9	CST-FRAMEWORK	1.2	standards	registeredstandards	csttmp	libref	&_cstGRoot./standards/cst-framework-1.3/templates	.	standards

Note the **SASref** and **path** fields. For most rows, SASref is set to **`csttmp`** and path is set to **`&_cstGRoot/standards/cst-framework/templates`**. The **memname** field points to empty examples of each file type. From a generic SAS Clinical Standards Toolkit framework perspective, these are the best available file references. All SAS Clinical Standards Toolkit processes require specification of some of these data and metadata sources (for example, generic properties, messages, and process results).

Here is the information returned by the following call to %cst_getStandardSASReferences for the CDISC SDTM standard: Display 5.2 on page 76.

```
%cst_getStandardSASReferences(_cstStandard=CDISC-SDTM, _cstOutputDS=sasreferences);
```

Display 5.2 Standard SASReferences for CDISC SDTM



	standard	standardversion	type	subtype	SASref	reftype	path	order	memname
1	CDISC-SDTM	3.1.2	autocall		sdtmauto	fileref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/macros	1	
2	CDISC-SDTM	3.1.2	classmetadata	column	sdtmcls	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	.	class_columns.sas7bdat
3	CDISC-SDTM	3.1.2	classmetadata	table	sdtmcls	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	.	class_tables.sas7bdat
4	CDISC-SDTM	3.1.2	lookup		sdtmctrl	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/control	.	standardlookup.sas7bdat
5	CDISC-SDTM	3.1.2	messages		sdtmmsg	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/messages	1	messages.sas7bdat
6	CDISC-SDTM	3.1.2	properties	initialize	sdtmprop	fileref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/programs	1	initialize.properties
7	CDISC-SDTM	3.1.2	properties	validation	sdtmprp2	fileref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/programs	1	validation.properties
8	CDISC-SDTM	3.1.2	referencecontrol	standardref	sdtmcntl	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/validation/control	.	validation_stdref.sas7bdat
9	CDISC-SDTM	3.1.2	referencecontrol	validation	sdtmcntl	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/validation/control	.	validation_master.sas7bdat
10	CDISC-SDTM	3.1.2	referencemetadata	column	sdtmref	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	.	reference_columns.sas7bdat
11	CDISC-SDTM	3.1.2	referencemetadata	table	sdtmref	libref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.3/metadata	.	reference_tables.sas7bdat

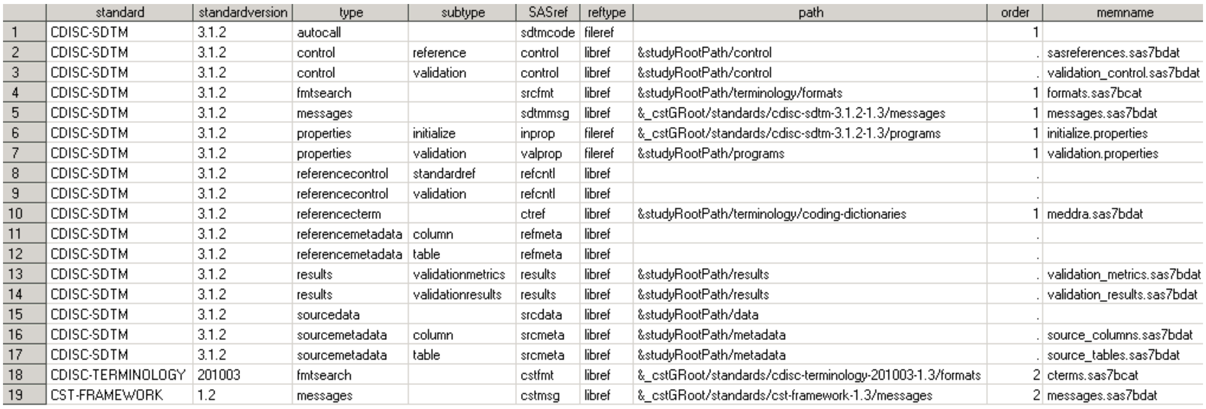
A comparison of Display 5.1 on page 75 and Display 5.2 on page 76 shows little similarity in the record types and no overlap in references to specific files. The target inputs and outputs for CDISC SDTM are more focused on the task (for example, validating SDTM domains). SAS Clinical Standards Toolkit validation processes require specification of a comparative reference standard. Here, there are references to a standard-specific macro library (autocall), Messages data set, and properties files. Unique SASref values by type are provided, pointing to distinct files and folders in the global standards library.

Consider an actual SASReferences file built to support CDISC SDTM 3.1.2 validation. The task of validating the functionality of CDISC SDTM 3.1.2 uses the SASReferences file found at the following location in SAS 9.2:

```
!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/
cdisc-sdtm-3.1.2/sascstdemodata/control
```

The following figure shows the complete contents of the SASReferences file.

Display 5.3 Sample SASReferences File for CDISC SDTM Validation



	standard	standardversion	type	subtype	SASref	reftype	path	order	memname
1	CDISC-SDTM	3.1.2	autocall		sdtmcode	fileref		1	
2	CDISC-SDTM	3.1.2	control	reference	control	libref	&studyRootPath/control	.	sasreferences.sas7bdat
3	CDISC-SDTM	3.1.2	control	validation	control	libref	&studyRootPath/control	.	validation_control.sas7bdat
4	CDISC-SDTM	3.1.2	fmtsearch		srcfmt	libref	&studyRootPath/terminology/formats	1	formats.sas7bcat
5	CDISC-SDTM	3.1.2	messages		sdtmmsg	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.3/messages	1	messages.sas7bdat
6	CDISC-SDTM	3.1.2	properties	initialize	inprop	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.3/programs	1	initialize.properties
7	CDISC-SDTM	3.1.2	properties	validation	valprop	fileref	&studyRootPath/programs	1	validation.properties
8	CDISC-SDTM	3.1.2	referencecontrol	standardref	refcntl	libref		.	
9	CDISC-SDTM	3.1.2	referencecontrol	validation	refcntl	libref		.	
10	CDISC-SDTM	3.1.2	referencecterm		ctref	libref	&studyRootPath/terminology/coding-dictionaries	1	meddra.sas7bdat
11	CDISC-SDTM	3.1.2	referencemetadata	column	refmeta	libref		.	
12	CDISC-SDTM	3.1.2	referencemetadata	table	refmeta	libref		.	
13	CDISC-SDTM	3.1.2	results	validationmetrics	results	libref	&studyRootPath/results	.	validation_metrics.sas7bdat
14	CDISC-SDTM	3.1.2	results	validationresults	results	libref	&studyRootPath/results	.	validation_results.sas7bdat
15	CDISC-SDTM	3.1.2	sourcedata		srcdata	libref	&studyRootPath/data	.	
16	CDISC-SDTM	3.1.2	sourcemetadata	column	srcmeta	libref	&studyRootPath/metadata	.	source_columns.sas7bdat
17	CDISC-SDTM	3.1.2	sourcemetadata	table	srcmeta	libref	&studyRootPath/metadata	.	source_tables.sas7bdat
18	CDISC-TERMINOLOGY	201003	fmtsearch		cstfmt	libref	&_cstGRoot/standards/cdisc-terminology-201003-1.3/formats	2	cterms.sas7bcat
19	CST-FRAMEWORK	1.2	messages		cstmsg	libref	&_cstGRoot/standards/cst-framework-1.3/messages	2	messages.sas7bdat

Table 5.2 Explanation of Sample SASReferences File for CDISC SDTM Validation

Lines	Comment
1	Instructs the SAS Clinical Standards Toolkit to add any SDTM-specific macros to the autocall path.
2	Documents the name and location of this file. This information is used in the sample reports that are discussed in this document.

Lines	Comment
3	Points to the set of validation checks to be run in this validation assessment. The framework default values for SASref, path, and memname have been overridden.
4, 18	Two standards are referenced to create a format search path. Line 4 references the SDTM study-specific formats catalog. Line 18 references the more general CDISC Terminology cterms catalog. The precedence is set by the order column.
5, 19	These records are identical to the CST-FRAMEWORK and CDISC-SDTM StandardSASReferences records.
6	Illustrates the call to a standard-specific properties file that is used to initialize a global macro variable that is specific to that standard. Referencing a standard-specific properties files in the SASReferences data set is recommended. The call to the CST-FRAMEWORK initialize.properties file is a prerequisite setup step outside of SASReferences and performed before processing SASReferences.
7	The validation properties path has been modified to point to a location in the study hierarchy, rather than to the global standards library that is defined in the StandardSASReferences file.
8–9 11–12	Points to the reference standard for CDISC SDTM 3.1.2, but unlike the template defaults in Display 5.2 on page 76, path and memname are blank. Leaving them blank tells SAS Clinical Standards Toolkit to look in the CDISC SDTM 3.1.2 StandardSASReferences file and use the defaults for that standard and version. This convention facilitates portability of the data set by doing a run-time lookup for the current information. The lookup results in the inclusion of the path and memname values as defined in Display 5.2 on page 76.
10	References a medDRA data set that is maintained in the study-specific hierarchy. A more common implementation might reference a non-study-specific coding dictionary.
13-14	Specifies that process results are to be stored in a location in the study hierarchy.
15	This is a new type not in the template files (StandardSASReferences). It defines the location of the study (source) data. The use of &studyRootPath, coupled with the assumption of a fixed-folder hierarchy, enables portability across studies. The memname value is not relevant for a library of SAS data sets.
16-17	These source metadata references are new. These values follow the style used in line 15 for source data. The same SASref is used for multiple subtypes in a single type because the subtypes reference two differently named SAS data sets from the same folder.

An alternative way to build the SASReferences file is to use the %cst_createds utility macro.

```
%cst_createds(_cstStandard=CST-FRAMEWORK,_cstType=control,_cstSubType=reference,
_cstOutputDS=work.sasreferences);
proc sql;
insert into work.sasreferences
values(CST-FRAMEWORK 1.2 messages messages libref 1 );
.
.
.
quit;
```

This macro copies the template. New records can be added various ways, including the previous PROC SQL technique. There is no requirement that the SASReferences file has to live outside the SAS Work area and be kept beyond the SAS Clinical Standards Toolkit process. However, these are best practices that enable future capabilities such as process reruns and reporting.

How Is a SASReferences File Used?

Overview

After a SASReferences file has been created for a task, three key steps occur.

1. The name and location of the file must be communicated to the SAS Clinical Standards Toolkit.
2. The structural integrity and content of the file are assessed.
3. The file content is translated into allocated SAS libraries and filenames, system options are set, and required work files are created.

After these steps are completed, a SAS environment has been properly established to support subsequent SAS Clinical Standards Toolkit tasks.

Communicating the Filename and Location to the SAS Clinical Standards Toolkit

Three global macro variables are used to define the name and location of the SASReferences file:

- The _cstSASRefsLoc macro provides the path to the SAS library that contains the file.
- The _cstSASRefsName macro provides the SASReferences filename in _cstSASRefsLoc.
- The _cstSASRefs macro provides libref.dset for the SASReferences file that is returned from the call to the cst_insertstandardsasrefs macro. The libref.dset is used in SAS Clinical Standards Toolkit code for the remainder of the process.

Sample driver modules are provided with the SAS Clinical Standards Toolkit. These driver modules 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 key macro cstutil_processsetup is called in all sample driver modules. This macro interprets information about the location and name of the SASReferences file, and calls the cstutil_allocatesasreferences macro to allocate SAS librefs and filerefs based on SASReferences content.

Here is the macro code:

```
%macro cstutil_processsetup( _cstSASReferencesSource=SASREFERENCES,
           _cstSASReferencesName=sasreferences,
           _cstSASReferencesLocation=)  /des='CST: Setup Process Metadata';
```

The following table lists the parameters that are supported by the cstutil_processsetup macro:

Parameter	Description
_cstSASReferencesSource	Specifies the initial source that setup should be based on. Valid values are SASReferences (default) or Results. If Results, then no other parameters are required, setup responsibility is passed to the cstutil_reportsetup macro, and the Results data set name must be passed to cstutil_reportsetup as libref.memname.
_cstSASReferencesLocation	Specifies the path (folder location) of the SASReferences data set. The default is the path to the Work library. This is the value of the global macro variable.
_cstSASReferencesName	Specifies the name of the SASReferences data set. The default is SASReferences. The value of the global macro variable _cstSASRefsName is set to this parameter value.

Excluding SAS Clinical Standards Toolkit reporting processes, to communicate with a SASReferences file, use one of the following two methods.

Note: SAS Clinical Standards Toolkit reporting processes might use the _cstSASReferencesSource=RESULTS parameter.

1. Create and reference the SASReferences file in the SAS Work library.

```
%* The following call assumes the existence of work.sasreferences;
%cstutil_processsetup();
```

2. Reference an existing SASReferences file.

```
data _null_;
  select("&sysver");
   when("9.1") call symput('studyRootPath',
        '!sasroot/../SASClinicalStandardsToolkitSDTM312/
         1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata');
   otherwise   call symput('studyRootPath',
         '!sasroot/../../SASClinicalStandardsToolkitSDTM312/
         1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata');
  end;
```

```
run;
%* Look for the data set named sasreferences in the specified folder ;
%cstutil_processsetup(_cstSASReferencesLocation=&studyrootpath/control);
```

Assessing Structural Integrity and Content

Two SAS Clinical Standards Toolkit framework utility macros perform key functions in assessing whether the SASReferences file is valid.

The cst_insertstandardsasrefs macro looks up missing paths and memnames in the constructed SASReferences file from each StandardSASReferences data set. For example, this macro sets the path and memname values for lines 8 and 9 and 11 and 12 in the example in Display 5.3 on page 76. This macro attempts to update only records for supported standards (and standardversions) that have missing path and memname information. It does not update records with non-null values, and it does not add any records from the StandardSASReferences data set. If this macro runs successfully, then the resulting data set has paths for all records and memnames for all records that require them. This does not include autocall and sourcedata records. By default, the resulting data set is referenced by the &_cstSASRefs global macro variable.

The cstutil_checkds macro checks the structure and content of the data sets used by SAS Clinical Standards Toolkit, including SASReferences. This macro validates that SASReferences has the structure and content defined by the StandardSASReferences and Standardlookup data sets.

The following is the syntax of this macro:

%cstutil_checkDS(_cstDSname=, _cstType=, _cstSubType=, _cstStandard, _cstStandardVersion);

_cstDSname specifies a two-level name of the data set to be validated. This value is required.

_cstType specifies the type of the data set to be validated. This value is required. This value comes from the Type column in the registered SASReferences for the standard-version combination.

_cstSubType specifies the subtype for the corresponding type. This value comes from the Subtype column in the registered SASReferences for the standard-version combination. If the type has no subtypes registered, then this option can be omitted. Otherwise, this value is required.

_cstStandard specifies the name of the data standard to validate against. This value is optional. By default, all standards are included.

_cstStandardVersion specifies the version of the data standard to validate against. This value is optional. By default, all standard versions are included.

Results are written to the Results data set defined by the &_cstResultsDS global macro variable.

The following table describes the most common errors detected by the cstutil_checkds macro. It suggests solutions as well.

Table 5.3 Common Errors and Solutions

Error	Location Where It Is Reported	Possible Cause and Solution
Input parameters to macro insufficient for cstutil_checkds macro to run.	Results Data Set	One of the required macro variable options is missing.
Location for Results data set is undefined.	SAS Log	Define the Results data set in the macro variable _cstResultsDS.
Data set could not be found.	Results Data Set	The data set that is passed in via the _cstdsname parameter cannot be found. Verify that the data set exists in the location specified.
Data set could not be opened.	Results Data Set	The data set that is passed in via the _cstdsname parameter cannot be opened. Make sure that you do not have the data set open in another window. Verify you have read access to the data set.
Differences found between data set and the template data set.	Results Data Set	The data set that is passed in via the _cstdsname parameter has a different structure than the template data set. Use the cst_createds macro to create a valid empty version of the data set, and then populate this data set with your data.
Null values are not permitted for column.	Results Data Set	Some columns are required to be non-null. If you receive this error, then you are also informed which column must contain a value. Enter a non-null value for this column.
Invalid value for column column_name, row ## in data set.	Results Data Set	Some columns are limited to a set of values. This error indicates that the value for column_name, listed in row ##, has an invalid value. The list of valid values can be found in the Standardlookup data set that is registered with each data standard. Review the list of valid values, and update the column value.

Translating Content for a SAS Session

After the SASReferences file has been built, its content must be translated for use by a SAS Clinical Standards Toolkit process. A call to the SAS Clinical Standards Toolkit framework utility macro %cstutil_processsetup performs the translation. If this macro runs successfully, then the SAS session is properly configured for any tasks (such as validation) that follow.

When the %cstutil_processsetup macro is called, the following happens:

1. The cstutil_allocatesasreferences macro is called.
2. The cst_insertstandardsasrefs macro is called to insert paths into any records that are missing that information. The information is retrieved from the StandardSASReferences data set for each standard.
3. The cstutil_checkds macro is called to perform internal validation on the SASReferences data set updated in step 2.
4. All filerefs and librefs are allocated.
5. Any property files are passed to %cst_setProperties to create global macro variables.
6. The format search path is set if any type=fmtsearch records are found, based on the order that is specified.
7. The autocall path is set if any type=autocall records are found, based on the order that is specified. By default, the framework macro library was added to the autocall path when SAS Clinical Standards Toolkit was deployed.
8. A Messages data set is created to contain records from each standard, based on the properties or global macro variables _cstMessages and _cstMessageOrder. The Messages data set is used for the duration of the process to add fully resolved messages to the Results data set.

After all of these steps have been performed, all libraries should be allocated, all paths and global macros should be set, and the global status macro variable _cst_rc should be set to 0. The process is ready to proceed.

This is a common process failure point because of the importance of the SASReferences file, and the strict structural and content expectations of the file. SASReferences is key to the process, and any errors will cause the process to fail. For tips on debugging problems with the SASReferences file, see Table 5.3 on page 81.

Best Practice Recommendation: Each SASReferences file is customized for the specific task to be completed. Later sections describe SASReferences implementations required by these specific tasks.

Chapter 6
Validation

Validation Framework Overview

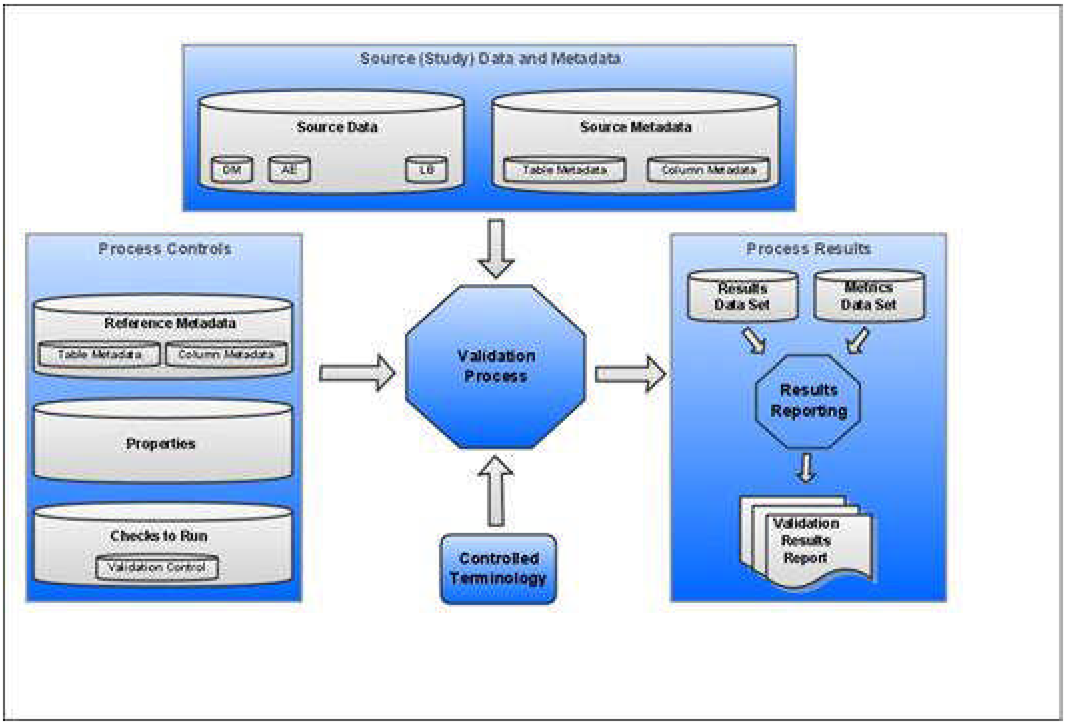
SAS Clinical Standards Toolkit validation assesses the compliance of data, and the metadata describing the data, with an accepted reference standard. It assesses the consistency of values in a specific column, between columns, across records in a specific data set, and across data sets. The primary output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

The SAS Clinical Standards Toolkit provides a framework to build a process. The process uses inputs or process controls to evaluate the compliance of source data with a reference standard. Each SAS Clinical Standards Toolkit process uses a SAS program file to point to a SASReferences control data set, and to execute a primary action SAS macro (such as sdtm_validate). This SAS program file is referred to as a driver module in this document.

Generally, validation is performed by running SAS macros against the standard, which is represented by SAS files. Validation of some standards, such as CDISC CRT-DDS, might include validating files that are not SAS files (such as define.xml).

The following display shows a SAS Clinical Standards Toolkit validation process. Each component is fully described in the following sections.

Display 6.1 *Components of a SAS Clinical Standards Toolkit Validation Process*



- *Source Data* is a set of SAS data sets in one or more libraries that collectively represents a clinical study. These SAS data sets are referred to as study domains or study data sets. One or more source data sets are required by a typical SAS Clinical Standards Toolkit validation process. However, it is possible to test only the structural compliance of source metadata by limiting validation to a subset of validation checks.

- *Source Metadata* is a set of SAS data sets in one or more libraries that provide metadata about the source data. The source metadata is typically in a format specific to a standard. For example, metadata about source data sets might be captured in a source_tables data set. Metadata about columns in those source data sets might be captured in a source_columns data set.

- *Process Controls* is the set of instructions that each SAS Clinical Standards Toolkit process uses to perform a specific action. These instructions might be provided in a varied number and in various type of files. For a SAS Clinical Standards Toolkit validation process, these files include the following:

  - *Reference Metadata* is a set of SAS data sets that provide metadata. This metadata defines a specific standard and is typically in a format specific to a standard. For example, metadata about data sets might be captured in a reference_tables data set. Metadata about columns in those data sets might be captured in a reference_columns data set. For an example, see Table 4.3 on page 42 and Table 4.4 on page 42.

  - *Properties* are a series of name-value pairs that are translated into SAS global macro variables. These macro variables are available for the duration of the SAS Clinical Standards Toolkit process. Properties might be defined in a varied number of files. Both text file format and SAS data set format are supported. For information about a sample validation.properties file, see "Validation Check of Metadata: Validation Master" on page 90. For information about the SAS Clinical Standards Toolkit global macro variables, see Appendix A1, "Global Macro Variables," on page 211.

  - *Set of Checks to Run* is a set of checks that represent all or some of the checks defined for a standard. Each check provides metadata that is used by the validation code to perform a specific compliance assessment.

- *Controlled Terminology* is an optional set of lookup values against which source data columns can be evaluated. These values can be in the form of SAS format catalogs or SAS data sets.

- *Results* are presented in a Results data set that itemizes the process findings, and in a Metrics data set that summarizes the results. The Results data set usually contains a record indicating that each check was run successfully without error, or it contains a record that itemizes the errors detected. Information about the process also might be included. The generation of a Metrics data set is conditional based on property file settings.

The SAS Clinical Standards Toolkit validation makes the following basic assumptions:

1. There is some combination of source data and metadata available as SAS files that the user wants to validate.

2. A reference standard has been defined with which the source data and metadata are to be compared. The SAS Clinical Standards Toolkit provides representative reference metadata for each supported standard.

3. The source data can be in a varied number of SAS files, and those SAS files can have any form. However, the metadata describing the source data must accurately represent the source data. The metadata must be in a form specific to a supported standard and defined by the SAS Clinical Standards Toolkit.

4. A set of validation checks must be defined, and the validation checks must conform to a generic SAS Clinical Standards Toolkit SAS data set structure. The SAS Clinical Standards Toolkit provides a representative set of validation checks for each supported standard.

Metadata Requirements

Overview

As noted in Chapter 4, “Supported Standards,” on page 35, a standard consists of properties, messages, and metadata files that collectively represent the standard in the SAS Clinical Standards Toolkit. Each SAS Clinical Standards Toolkit registered standard can support validation if the standards.supportsvalidation flag is set to Y. This setting indicates that the required set of validation files defining the standard exist. By default, the set of validation files that supports the standards that are supplied by SAS can be found in the cstGlobalLibrary folder hierarchy.

For example, validation files defining the CDISC SDTM 3.1.1 standard can be found in the folder hierarchy at:

***<global standards library directory>*/standards/cdisc-sdtm-3.1.1** .

The following sections describe each type of file that defines metadata. The file type is either entirely unique to a SAS Clinical Standards Toolkit validation process, it has validation-specific elements. For information about metadata files that are common to all SAS Clinical Standards Toolkit processes, see Chapter 3, “Metadata File Descriptions,” on page 21.

Reference Metadata

For CDISC standards, reference metadata refers to metadata about data sets. Reference metadata is defined in a reference_tables data set, and metadata about columns is defined in a reference_columns data set. An example of a reference_tables record is provided in Table 4.3 on page 42 and an example of a reference_columns record is provided in Table 4.4 on page 42. The reference metadata in these examples is required, and it serves as the gold standard specifically describing the tables and columns of CDISC SDTM. As noted in Chapter 4, each standard that is supplied by SAS provides a SAS interpretation of the published source guidelines or specification of that standard. Each standard is designed to serve as a representative model or template of the source specification. Each model or template can be modified to establish your own gold standard.

Table 6.1 *Reference_Tables Data Set*

Column Name	Column Length	Description
sasref	$8	The SAS libref that refers to the table in the SAS Clinical Standards Toolkit process. This value should match the value of the SASReferences.sasref field, where type=referencemetadata and subtype=table. This column is required.
table	$32	The name of the domain being defined in the standard. The value must conform to SAS naming conventions. This column is required.

Column Name	Column Length	Description
label	$40	The label of the domain being defined in the standard. The value must conform to SAS naming conventions. This column is optional.
class	$40	The observation class in the standard. Example CDISC SDTM values are Events, Findings, Interventions, Relates, Special Purpose, and Trial Design. This column is optional and not relevant for all standards.
xmlpath	$200	The path to the SAS transport file. This path can be specified as a relative path. The value can be used when creating define.xml to populate the value for the def:leaf xlink:href link to the domain file. The value should be the pathname and filename of the SAS transport file relative to the location of define.xml file. This column is optional and not relevant for all standards.
xmltitle	$200	The title of the SAS transport file. The value can be used when creating a define.xml file to populate the value for the def:leaf def:title value. It can provide a meaningful description, label, or location of the domain leaf (for example, crt/datasets/Protocol 1234/AE.xpt). This column is optional and not relevant for all standards.
structure	$200	The description of the general structure of the table. An example value is one record per event per subject. This column is optional and not relevant for all standards.
purpose	$20	The description of the general purpose of the table. Examples are Tabulation (required for CDISC SDTM) and Analysis (required for CDISC ADaM). This column is optional and not relevant for all standards.
keys	$200	A space-delimited string of keys that captures the table columns that uniquely define records in the table. This set of keys can also define the sort order of records in the table. Example is STUDYID USUBJID. This column is required.
state	$20	A description of the table state, such as Draft or Final. This column is optional.
date	$20	A meaningful, distinguishing date that describes the table. For example, release date, creation date, or modified date. This column is optional.
standard	$20	This value captures the standard name. This value must match the name of a registered standard in the SAS Clinical Standards Toolkit framework. For a discussion of registered standards, see "Framework" on page 5. This value must match the **standard** field in the SASReferences data set. Examples are CDISC SDTM and CDISC CRT-DDS. This column is required.
standardversion	$20	This value captures a specific version of a standard. This value must match one of the standard versions associated with a registered standard. This value must match the **standardversion** field in the SASReferences data set. Examples are 3.1.1 and 1.0. This column is required.
standardref	$200	Any reference to an associated standard definition, implementation guide, schema, and so on, that provides additional information about the table or describes the table in greater detail. This column is optional.
comment	$200	Any character string that provides comments relevant to the table. This column is optional.

Table 6.2 *Reference_Columns Data Set*

Column Name	Column Length	Description
sasref	$8	The SAS libref that refers to the table containing the column in the SAS Clinical Standards Toolkit process. This value should match the value of the SASReferences.sasref field, where type=referencemetadata and subtype=column. This column is required.
table	$32	The name of the domain being defined in the standard. The value must conform to SAS naming conventions. This column is required.
column	$32	The name of the column in the table. The value must conform to SAS naming conventions. This column is required.
label	$200	The label of the column. The value must conform to SAS naming conventions. This column is optional.
order	8.	The order of the columns in each table. Values must be integers >0 and unique in each table. This column is required.
type	$1	The SAS type, N for numeric, C for character. This column is required.
length	8.	The length of the column. Numeric columns have a length of 8. This column is required.
displayformat	$32	The display format for numeric variables. For example, 8.2 indicates that floating-point variable values should be displayed to the second decimal place. This value is optional and not relevant for all standards.
xmldatatype	$8	The data type of the column as it is defined in the define.xml file. Values are integer | float | date | datetime | time | text. This column is optional and not relevant for all standards.
xmlcodelist	$32	A SAS format name that is used to assess conformance to controlled terminology. This value does not have a $ prefix for character formats and does not have the trailing period. This value is also the codelist name in the define.xml file. The SAS format name must be in the format search path for successful column-value validation. This record is optional and not relevant for all standards.
core	$10	The value indicates whether the column is required. Sample CDISC SDTM values are Req (required), Exp (expected), Perm (permissible), and Dep (deprecated). This column is optional and not relevant for all standards.
origin	$40	Information about the source of the column. Values can include CRF page numbers and derived or variable references. Values are user extensible. This column is optional and not relevant for all standards.

Column Name	Column Length	Description
role	$200	Space-delimited column classification. Examples are Identifier, Topic, Qualifier, Timing, Selection, and Analysis. Columns can have multiple roles. This column is optional and not relevant for all standards.
term	$80	The value indicates whether the column is subject to controlled terminology as defined in each standard source specification. This column is optional and not relevant for all standards.
algorithm	$1000	Imputation or computation method to derive the column value. This column is optional and not be relevant for all standards.
qualifiers	$200	Space-delimited string containing supplemental column attributes. Example CDISC SDTM values are MIXEDCASE, UPPERCASE, DATETIME, and DURATION. This column is optional and not relevant for all standards.
standard	$20	This value captures the standard name. This value must match the name of a registered standard in the SAS Clinical Standards Toolkit framework. For a discussion of registered standards, see "Framework" on page 5. This value must match the **standard** field in the SASReferences data set. Examples are CDISC SDTM and CDISC CRT-DDS. This column is required.
standardversion	$20	This value captures a specific version of a standard. This value must match one of the standard versions associated with a registered standard. This value must match the **standardversion** field in the SASReferences data set. Examples are 3.1.1 and 1.0. This column is required.
standardref	$200	Any reference to an associated standard definition, implementation guide, schema, and so on, that provides additional information about the column or describes the column in greater detail. This column is optional.
comment	$1000	Any character string that provides comments relevant to the column. This column is optional.

The standard reference metadata provided by SAS can be found in the SAS Clinical Standards Toolkit global standards library. By default, this library can be found at:

```
<global standards library directory>/standards/<specific standard>/metadata
```

For example, for the CDISC SDTM 3.1.1 standard, the location is:

```
<global standards library directory>/standards/cdisc-sdtm-3.1.1-1.3/metadata
```

This global standards library metadata folder can contain other standard-specific metadata. For example, CDISC SDTM includes class_tables and class_columns data sets. These data sets have more generic metadata than specific domain instances like DM or AE, and they are most useful when deriving new, custom domains. For example, if a new CDISC SDTM events domain is required, users can initialize table metadata based on the EVENTS record in class_tables data set, and can initialize column metadata based on the EVENTS, IDENTIFIERS, and TIMING records in the class_columns data set.

Source Metadata

The SAS Clinical Standards Toolkit validation processes require source metadata that describes source (study) domains and columns. This is the study data that is to be validated. The SAS Clinical Standards Toolkit assumes that the reference metadata (that is, reference_tables and reference_columns) for a standard serves as a model or template for the source metadata (that is, source_tables and source_columns). It is recommended that these two sets of metadata be structurally equivalent. However, additional metadata attributes might exist if they are used for other purposes or for custom extensions to the SAS Clinical Standards Toolkit.

The SAS Clinical Standards Toolkit assumes that source_tables and source_columns data sets accurately reflect and are consistent with the source data that they describe. While some standard-specific validation checks might look for discrepancies and report them in detail, failure to accurately reflect and be consistent with the source data can lead to errors in the SAS Clinical Standards Toolkit validation process. It can even halt the execution of the process.

Validation Check of Metadata: Validation Master

The Validation Master data set contains all validation checks defined for a standard. By default, this data set is deployed to the following directory in each supported standard:

***<global standards library directory>*/standards/*<standard>*/ validation/control**

By default, the Validation Master SAS data set's actual name is validation_master.sas7bdat.

The SAS Clinical Standards Toolkit requires that this data set have a fixed structure. The following table lists the columns in the Validation Master data set. These columns are described and examples are reviewed in the following sections.

Table 6.3 Column Descriptions of the Validation Master Data Set

Column Name	Column Length	Description
checkid	$8	Validation check ID. The SAS Clinical Standards Toolkit has adopted a naming convention matching each standard to be validated. The checkid values are prefixed with an up to 4-character prefix (CDISC examples: ODM, SDTM, ADAM, and CRT). By convention, the prefix matches the **mnemonic** field in the Standards data set in ***<global standards library directory>*/metadata**. This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). Users can use any naming convention limited to 8 characters. By default, the checkid column is the first (primary) sort field in the Validation Master data set provided by SAS. Sorting by checkid is not required. This column is required.
standard	$20	This value captures the standard name. This value must match the name of a registered standard in the SAS Clinical Standards Toolkit framework. For a discussion of registered standards, see "Framework" on page 5. This value must match the **standard** field in the SASReferences data set. Examples are CDISC SDTM and CDISC CRT-DDS. This column is required.

Column Name	Column Length	Description
standardversion	$20	This value captures a specific version of a standard. This value must match one of the standard versions associated with a registered standard. This value must match the **standardversion** field in the SASReferences data set. The only exception to this rule is that *** can be used to signify that the check applies to all supported versions of the standard. For example, 3.1.1, 1.0, ***. If a subsequent version of the standard is released, then *** would be applicable if the check is valid for the new version. This column is required.
checksource	$40	A string that identifies the source of the check. CDISC examples include Janus, JanusFR (FAIL-REJECT), SAS, WebSDM, and OpenCDISC. This field can contain any user-defined value. A primary use of this field is to subset the full set of checks in the run-time Validation Control data set. This column is required.
sourceid	$8	A reference identifier for this check from the checksource. In the Validation Master data set, a SAS identifier (for example, SAS0001) is used for checks provided by SAS with no external source. An example is IR4000 (WebSDM identifier). This column is optional.
checkseverity	$40	The severity as assigned by checksource. This value is mapped to the following standardized values: Note (Low), Warning (Medium), Error (High). A value is expected, although it is not technically required. It is used in messages and reporting.
checktype	$20	General type of check. This value categorizes checks and helps register customized checks. Values are user extensible and can be standard specific. A primary use of this field is to subset the full set of checks in the run-time Validation Control data set. Example CDISC SDTM values are: Metadata-structural—Checks some metadata-only property (no data access required). ColumnValue-content—Checks a column value or compares two column values. Date-content—Checks ISO 8601 compliance or compares two date values. Multirecord-content—Looks across multiple records in a single domain. Multitable-content—Looks across multiple domains. Controlterm-content—Assesses whether column value is consistent with controlled terminology. This column is optional.
codesource	$32	The name of the check macro. The name must conform to SAS naming conventions. The value must be in the SAS autocall path. An example is cstcheck_notunique. This column is required.
usesourcemetadata	$1	The value indicates whether to use source metadata rather than reference metadata. The metadata controls the derivation of domains and column lists to be validated, program flow, and looping. Values are Y and N (default). This column is optional.

Column Name	Column Length	Description
tablescope	$200	The value specifies the domains to be validated by the check. The domains must exist in either or both of the reference metadata or source metadata. The value can be in the form: _ALL_-DM-DS—Multiple domains that exclude one or more specific domains that are delimited with a -. DM—Any single domain; can be specified as libref.domain. DM+AE—Multiple domains delimited with a +. _ALL_—Multiple DM domains that exclude specific domains delimited with a -. SUPP**—Wildcard to include multiple domains. CLASS:EVENTS—All domains capturing event results. (This syntax specifies to use table metadata column CLASS for EVENTS as the value-similar syntax for all other fields and values.) [_ALL_-DM][DM]—Bracket syntax to define sublists for comparative purposes. In this example, all non-DM domains are compared with the DM domain. See the Validation Master data set for a full set of values. This column is required.
columnscope	$200	The value specifies one or more space-delimited columns identified for inclusion or exclusion in the specified check. The value can be in the form: _ALL_—All columns (equivalent to ** or a null value). _NA_—Not applicable (that is, domain-level check). AGE—Any single column. This value can be specified as libref.domain.column or domain.column. ARM+ARMCD—Multiple columns delimited with a +. **BLFL-LBBLFL—Multiple columns that exclude specific columns delimited with a -. **DTC—Wildcard to include multiple columns with ** representing the domain name. xxx**—(For example, AE**, where ** is a column wildcard). [**STDTC][**ENDTC]—Bracket syntax to define sublists for comparative purposes. In this example, all start dates are compared with all end dates. The number of columns in each sublist must be equivalent. See the Validation Master data set for a full set of values. This column is optional. (If null, the value is equivalent to _ALL_.)

Column Name	Column Length	Description

codelogic	$2000	Check-specific code segment that is inserted into the check macro defined in codesource and consistent with codetype. The codelogic value enables check-level customization and allows the reuse of more general check macros. The field length of $2000 limits the code to short code segments, although referencing another macro or using %include expands this capability. The codelogic value can use global and local macro variables (for example, variables provided as macro input parameters and variables set within the calling code). Examples include:
		`If ( . < &_cstColumn1 <`
		`&_cstColumn2), then _cstError=1;`
		`%include <fileref>`
		`/* where <fileref> can be set outside of the SAS Clinical Standards Toolkit`
		`or in the SASReferences control data set */`
		The previous code is limited to filerefs set outside of the SAS Clinical Standards Toolkit or in the SASReferences control data set.
		%sdtmcheckutil_recordlookup
		`data _cstProblems;`
		`set&_cstDSName;`
		`if <some condition>;`
		`run;`
		This column is optional.

codetype	8.	This value defines whether to use codelogic and what type of codelogic can be used in the validation code. Values include: 0—No codelogic used. 1—DATA step statement level. (For example, if &_cstColumn <0 then _cstError=1.) 2—Full DATA step, PROC SQL step, or multiple steps. 3—Calls a SAS macro or %include that can contain only DATA step statement level code. (For example, codetype=1.) 4—Calls a SAS macro or %include that can contain only full DATA step or PROC SQL step code. (For example, codetype=2.) This column is required.

Column Name	Column Length	Description
lookuptype	$20	This value defines the type of information to use for value comparison to some standard. Values include: Metadata—Use the SAS Clinical Standards Toolkit metadata. Specifically, use the value of the column metadata field xmlcodelist to identify the codelist (rendered as a SAS format). Format—Use a SAS format from the SAS format search path. Dataset—Use a reference SAS data set (for example, medDRA). There are no SAS Clinical Standards Toolkit requirements for the structure and content of the reference SAS data set. <extensible>—Other user-defined values can be used if there are explicitly referenced in user-written code. This column in optional.
lookupsource	$32	The specific SAS format or file associated with lookuptype. For example: If lookuptype is metadata, then lookupsource should be blank. The code gets the value from the source_columns.xmlcodelist field. If lookuptype is format, then lookupsource should be the SAS format and must be in the format search path if it is specified. This value should generally match any value in source_columns.xmlcodelist for the columns specified in **columnscope**. This field allows a run-time validation check against another format. If lookuptype is dataset, then lookupsource should be the name of a SAS data set. This value is specified as the data set name (for example, meddra) or libref.dataset. If a value is provided without a libref, then the SAS Clinical Standards Toolkit looks for any SASReferences type=referencecterm records for the sasref value. This column is optional.
standardref	$200	Any reference to an associated standard definition, implementation guide, schema, and so on, that provides additional information about the check or describes the basis for the check in greater detail. This column is optional.
reportingcolumns	$200	This value includes columns not included in columnscope for code-processing purposes and to help resolve errors. If this value is specified, then it should be a space-delimited list of columns in the domains specified in the **tablescope** field. The values of these columns can be reported in the Results data set. This column is optional.
checkstatus	8.	This value determines whether the check is ready to be used and included in any Validation Control run-time data set. If the check is ready, then the value should be set to any positive integer. Values include: 0—(inactive, default) >0—(active) -1—(deprecated, archived) -2—(not implemented in this SAS Clinical Standards Toolkit release) This column is optional, although it is expected.

Column Name	Column Length	Description
reportall	$1	This value enables more concise reporting of errors. Values include: Y—(yes, report all records, default) N—(no) This column is required although not all check macro modules support abbreviated (N) reporting.
uniqueid	$48	This value provides a unique ID for the check. It ensures uniqueness in the data set and in the SAS Clinical Standards Toolkit. This value allows any provided or derived check to be uniquely identifiable over time. An example is SDTM000100CST120SDTM3112009-05-12T12:00:00CDI. Legend: characters 1-8—checkid characters 9-10—checkid repeat indicator (00 unless multiple invocations of checkid are included) characters 11-16—the version of the SAS Clinical Standards Toolkit where the check metadata was last materially modified characters 17-23—standard version characters 24-42—implementation datetime of the last metadata update characters 43-48—assigning authority This column is optional, although it is expected.
comment	$200	Any character string that provides comments relevant to the check. This column is optional.

The content of the Validation Master data set is based on a combination of compliance requirements and the SAS representation of the standard.

The following table describes a sample Validation Master data set record for the CDISC SDTM 3.1.2 standard.

Table 6.4 *Sample CDISC SDTM 3.1.2 Validation Master Data Set Record*

Column Name	Column Value	Comment
checkid	SDTM0207	The SAS Clinical Standards Toolkit check identifier used in validation results and reports.
standard	CDISC-SDTM	The registered standard.
standardversion	***	The standard version. A value of *** indicates that the check is applicable to all versions of the standard.
checksource	WebSDM	This check originated as a WebSDM check.
sourceid	IR5010	WebSDM check IR5010.

Column Name	Column Value	Comment
checkseverity	Warning	
checktype	ColumnValue	
codesource	cstcheck_column	This check uses the cstcheck_column check macro in the SAS Clinical Standards Toolkit autocall library.
usesourcemetadata	Y	This check is run on source data domains.
tablescope	_ALL_	This check is run on all domains.
columnscope	VISITNUM	This check evaluates VISITNUM values from each domain.
codelogic	_vnum=strip(put(&_cstColumn,best.));_dot=indexc(_vnum,"."); if _dot then if length(substr(_vnum,_dot+1))>3 then _cstError=1;	This logic is used in cstcheck_column. Errors are documented in a work._cstProblems data set.
lookuptype		
lookupsource		
standardref		
reportingcolumns		
checkstatus	1	
reportall	Y	This check reports all errors that are identified.
uniqueid	SDTM020700CST120SDTM3112009-05-13T15:57:59CST	
codetype	1	This code logic is used in the DATA step.
comment		

While the Validation Master data set contains all validation checks for a standard, the Validation Control data set is the run-time equivalent and contains just the validation checks to be run in a validation process. The Validation Control data set is structurally equivalent to the Validation Master data set. For additional information about how the validation check metadata in the Validation Control data set is used in the SAS Clinical Standards Toolkit validation processes, see "Special Topic: How SAS Clinical Standards Toolkit Interprets Validation Check Metadata" on page 142.

Supplemental Validation Check Metadata: Validation Standard References

The validation standard references data set contains additional information about each of the checks in the validation master data set. This data set is used in the validation metadata reporting process to provide additional information to the user about the origin of the check. It also provides any supporting documentation about the check. By default, this data set is deployed to the following directory in each supported standard:

```
<global standards library directory>/standards/<standard>/
validation/control
```

Table 6.5 *Column Descriptions of the Validation_StdRef Data Set*

Column Name	Column Length	Description
checkid	$8	The validation check ID, as specified in the validation master data set (see Table 6.3 on page 90).
standard	$20	This value captures the standard name. This value must match the standard in the associated validation master data set. This column is required.
standardversion	$20	This value captures a specific version of a standard. This value should be the version for which the supplemental reference information is applicable. This column is required.
informationsource	$80	This value captures the origin of the reference information. The value can be an implementation guide, Web site, harmonization document, and so on. It can be any source that can be referenced that provides insight into the check.
sourcelocation	$200	This value contains the location in the information source, such as a page number or a section number.
seqno	8.	This value provides a sequence number for checkid if multiple sources of information are available for a check. This column is required.
sourcetext	$2000	This value captures descriptive information from the source that supports the check. This information attempts to provide a basis for inclusion of the check.

The content of the Validation_StdRef data set is based on information from any source that supports the check.

The following table describes information about a specific check in the Validation_StdRef data set for the CDISC SDTM 3.1.2 standard.

Table 6.6 *Sample CDISC SDTM 3.1.2 Validation_StdRef Data Set for Check SDTM0207*

Column Name	Column Value	Comment
Record 1		
checkid	SDTM0207	The SAS Clinical Standards Toolkit check identifier used in results and reports.

Column Name	Column Value	Comment
standard	CDISC-SDTM	The registered standard.
standardversion	3.1.2	The standard version.
informationsource	*SDTM 3.1.2 Implementation Guide*	This reference information originated from the *SDTM 3.1.2 Implementation Guide*.
sourcelocation	5.3.2, page 72	Section 5.3.2, page 72 of the *SDTM 3.1.2 Implementation Guide*.
seqno	1	The first record for this checkid.
sourcetext	Clinical encounter number. (Decimal numbering might be useful for inserting unplanned visits.)	The text of the information retrieved from section 5.3.2, page 72 of the *SDTM 3.1.2 Implementation Guide*.
Record 2		
checkid	SDTM0207	The SAS Clinical Standards Toolkit check identifier used in results and reports.
standard	CDISC-SDTM	The registered standard.
standardversion	3.1.2	The standard version.
informationsource	WebSDM	This reference information originated from the WebSDM validation checks.
sourcelocation	Convention	Compliance convention set by WebSDM.
seqno	2	The second record for this checkid.
sourcetext	Compliance convention set by WebSDM. No supporting implementation guide found.	Representative text for an accepted convention.

Supplemental Validation Check Metadata: Domains by Check

The SAS Clinical Standards Toolkit validation metadata, as specified in the Validation Master data set, uses the tablescope and columnscope columns to define the scope of the check. The scope being what domains (tables) and what columns will be validated when the check is run. The SAS Clinical Standards Toolkit uses a shorthand syntax in these columns that is interpreted by the SAS Clinical Standards Toolkit framework macros to build a list of target tables and columns. For more information, see “Special Topic: How SAS Clinical Standards Toolkit Interprets Validation Check Metadata” on page 142. The Validation_DomainsByCheck data set is supplied in ***<global standards library directory>*/standards/*<standard>*/validation/control**. It contains records for each domain that is to-be-validated by each check in the Validation Master data set. This data set is used by reporting tools that are provided with the SAS Clinical Standards Toolkit to report domain-specific errors. For more information, see Chapter 8, “Reporting,” on page 195. It is also available to other programs and applications that might need to subset checks that are applicable to specific domains.

The version of the Validation_DomainsByCheck data set that is supplied by SAS is built from the version of the Validation Master data set that is also supplied by SAS. If the tableScope and columnScope columns are modified, then the Validation_DomainsByCheck data set must also be modified or rebuilt.

***Table 6.7** Column Descriptions of the Validation_DomainsByCheck Data Set*

Column Name	Column Length	Description
checkid	$8	The validation check ID, as specified in the validation master data set (see Table 6.3 on page 90).
table	$32	This value captures the domain or table name. This column is required.
standardversio n	$20	This value captures a specific version of a standard. This value must match standardversion in the associated validation master data set.
checksource	$40	A string that identifies the source of the check. This value must match checksource in the associated validation master data set.
resultseq	8.	The unique invocation of a check within the validation master data set. This value is incremented if multiple record or domain combinations exist.

For CDISC SDTM 3.1.2 validation check SDTM0207, the Validation_DomainsByCheck data set contains records for 14 domains. These 14 domains are DA, EG, FA, IE, LB, MB, MS, PC, PE, PP, QS, SV, TV, and VS. The target domains and columns for check SDTM0207 are defined as tableScope=_ALL_ and columnScope=VISITNUM. This means there are 14 domains in the sample study metadata provided for CDISC SDTM 3.1.2 that contain the column VISITNUM.

Validation.Properties

Properties specific to validation processes are provided with the SAS Clinical Standards Toolkit. These properties enable you to specify how validation checks are to be processed and whether metrics are to be reported.

As with all SAS Clinical Standards Toolkit properties files, a call to the %cst_setProperties macro is required to translate the properties into SAS global macro variables. This call can be explicitly made as a driver module setup task, or it can be made by including the Validation.Properties file as a record in the SASReferences data set. For all standards that support validation, the Validation.Properties file is required, even if no metrics are wanted because the SAS Clinical Standards Toolkit validation process does expect and use the metrics global macro variables.

The following table describes the properties in the Validation.Properties file:

Table 6.8 Properties in the Validation.Properties File

Property Name	Description
_cstCheckSortOrder	This property determines the order in which validation checks are processed. If no value is provided, or the default value _DATA_ is used, then the data set order is assumed. Or, _cstCheckSortOrder can be set to sort the Validation Control data set at run time by any fields in that data set. For example, CHECKSOURCE CHECKID.
_cstMetrics	This property determines whether to calculate and report metrics. An example value is 1=Yes.
_cstMetricsDS	This property sets the SAS data set name to use to accumulate metrics during the process. The default value is work._cstmetrics.
_cstMetricsNumSubj _cstMetricsCntNumSubj	This property determines whether to calculate and report subject-level counts. An example value is 1=Yes, initialize _cstMetricsCntNumSubj to 0. The calculation of subject-level counts might not be appropriate for all check macros.
_cstMetricsNumRecs _cstMetricsCntNumRecs	This property determines whether to calculate and report record-level counts. An example value is 1=Yes, initialize cstMetricsCntNumRecs to 0.
_cstMetricsNumChecks _cstMetricsCntNumChecks	This property determines whether to summarize and report the number of checks run. An example value is 1=Yes, initialize cstMetricsCntNumChecks to 0.
_cstMetricsNumBadChecks _cstMetricsCntNumBadChecks	This property determines whether to summarize and report the number of check invocations that failed. An example is 1=Yes, initialize cstMetricsCntNumBadChecks to 0.
_cstMetricsNumErrors _cstMetricsCntNumErrors	This property determines whether to summarize and report the total number of errors (resultseverity=Error) found. An example is 1=Yes, initialize cstMetricsCntNumErrors to 0.
_cstMetricsNumWarnings _cstMetricsCntNumWarnings	This property determines whether to summarize and report the total number of warnings (resultseverity=Warning) found. An example is 1=Yes, initialize cstMetricsCntNumWarnings to 0.
_cstMetricsNumNotes _cstMetricsCntNumNotes	This property determines whether to summarize and report the total number of notes (resultseverity=Note) found. An example value is 1=Yes, initialize cstMetricsCntNumNotes to 0.

Property Name	Description
_cstMetricsNumStructural _cstMetricsCntNumStructural	This property determines whether to summarize and report the total number of structural (metadata) errors found. An example value is 1=Yes, initialize cstMetricsCntNumStructural to 0.
_cstMetricsNumContent _cstMetricsCntNumContent	This property determines whether to summarize and report the total number of content (data) errors found. An example value is 1=Yes, initialize cstMetricsCntNumContent to 0.
_cstMetricsTimer	This property determines whether to report the elapsed time for each check invocation. An example value is 1=Yes.

By default, for all standards that support validation, Validation.Properties can be found at:

***<global standards library directory>*/standards/*<standard>*/programs**

Properties can logically be associated with each study. Using the CDISC SDTM 3.1.1 sample study provided with the SAS Clinical Standards Toolkit as an example, a study-specific instance of the Validation.Properties file can be found in a **!sasroot** subdirectory similar to **/sample/cdisc-sdtm-3.1.1/sascstdemodata/programs**.

Messages

Each SAS Clinical Standards Toolkit registered standard that supports validation has a Validation Master data set, and an associated Messages data set. The Validation Master data set provides the super-set of checks defined for that standard. The Messages data set provides messages to be generated during the execution of each validation process. A distinct Messages data set record is expected for each set of checkid and checksource values in the Validation Master data set. Messages can be parameterized and internationalized.

By default, the standard-specific Messages data set is deployed to the following directory in each supported standard:

***<global standards library directory>*/standards/*<standard>*/messages**

All Messages data sets in the SAS Clinical Standards Toolkit should have the same structure. The structure is defined in Chapter 3, “Metadata File Descriptions,” on page 21.

During a process, the SAS Clinical Standards Toolkit appends any standard-specific messages that are required by the process to any generic SAS Clinical Standards Toolkit framework messages that are available to all processes. This appended Messages data set follows the naming convention that is defined within the global macro variable _cstMessages.

For complete message lists supporting the SAS Clinical Standards Toolkit standards, see the following appendices:

- Appendix A2, “Framework Messages,” on page 219
- Appendix A4, “CDISC SDTM Validation Checks,” on page 281

- Appendix A5, “CDISC CRT-DDS 1.0 Validation Checks,” on page 335

Validation Metrics

Generating SAS Clinical Standards Toolkit validation metrics provides a meaningful denominator for most validation checks. This enables you to more accurately assess the relative scope of errors that are detected. Generally, the calculated denominator is a count of the number of records processed in a domain.

The following code segment, which is extracted from a validation check macro, shows a typical calculation of the number of records in a domain. It also shows the macro call to add the count to the Metrics data set:

```
data _null_;
if 0 then set &_cstDSName nobs=_numobs;
call symputx('_cstMetricsCntNumRecs',_numobs);
stop;
run;

* Write applicable metrics *;
%if &_cstMetrics %then %do;
%if &_cstMetricsNumRecs %then
   %cstutil_writemetric(
     _cstMetricParameter=# of records tested,
     _cstResultID=&_cstCheckID,
     _cstResultSeqParm=&_cstResultSeq,
     _cstMetricCnt=&_cstMetricsCntNumRecs,
     _cstSrcDataParm=&_cstDSname
   );
%end;
```

Because a check can evaluate multiple columns in a domain, the count will be greater. In addition, a metadata-level check that does not access the domain data directly might report the number of metadata records instead.

Metrics processing is enabled based on settings in the Validation.Properties file. See Table 6.8 on page 100.

The following table provides a description of the Validation Metrics data set, including the meaning of each field.

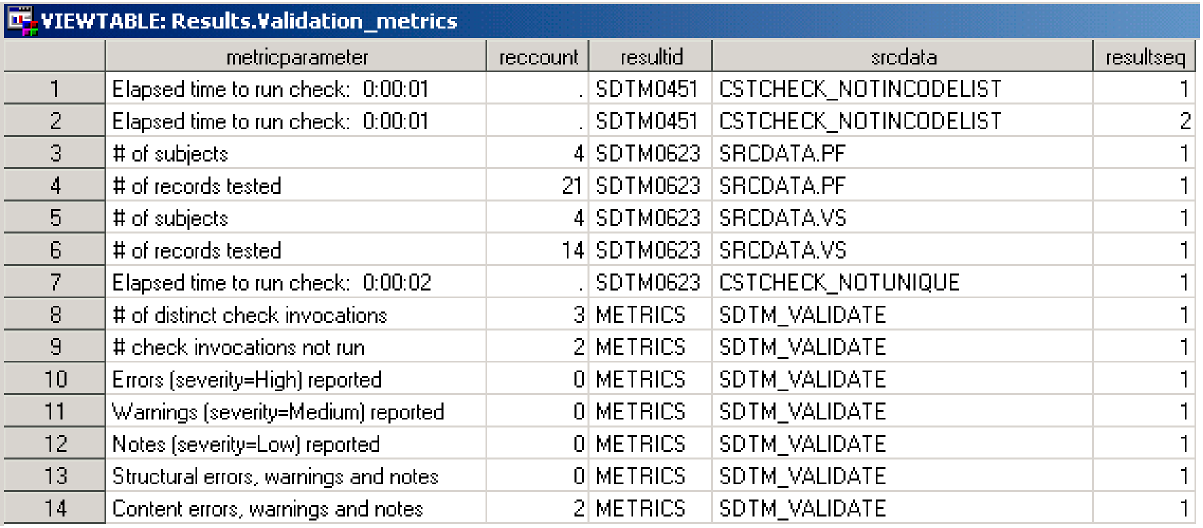
Table 6.9 *Column Descriptions of the Validation Metrics Data Set*

Column Name	Column Length	Description
metricparameter	$40	A descriptive text string that specifies the metric of interest. This string is hardcoded in the check macro and cannot be modified without code changes. Values should be non-null.
reccount	8.	A count of the number of records specific to the combination of metricparameter and resultid. This number is derived in the check macro and cannot be modified without code changes. This column can contain a summary count of records written to the Results data set (resultid=METRICS). Reccount can be null for selected metricparameters, such as the assessment of elapsed time for each check.

Column Name	Column Length	Description
resultid	$8	The resultid is either the checkid or a hardcoded constant such as METRICS. The SAS Clinical Standards Toolkit has adopted a naming convention matching each standard. The checkid (resultid) values are prefixed with an up to 4-character prefix (CST for framework messaging; CDISC examples: ODM, SDTM, ADAM, and CRT). By convention, the prefix matches the **mnemonic** field in the Standards data set in ***<global standards library directory>*/metadata**. This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). Users can use any naming convention limited to 8 characters. Values should be non-null.
srcdata	$200	The string that specifies the domain or check macro to which the metricparameter applies. Values should be non-null.
resultseq	8.	A counter that indicates the record number in checkid in the Validation Control run-time set of checks. If set to 1, then this counter is incremented only with each repeat invocation of a check. This value enables you to link to the Validation Control and Results data sets. Values should be non-null.

The following display illustrates Validation Metrics output from a SAS Clinical Standards Toolkit validation process running CDISC SDTM 3.1.1 validation. The Validation Control data set contains three records: two SDTM0451 checks and one SDTM0623 check.

Display 6.2 *Sample Validation Metrics Data Set*



	metricparameter	reccount	resultid	srcdata	resultseq
1	Elapsed time to run check: 0:00:01	.	SDTM0451	CSTCHECK_NOTINCODELIST	1
2	Elapsed time to run check: 0:00:01	.	SDTM0451	CSTCHECK_NOTINCODELIST	2
3	# of subjects	4	SDTM0623	SRCDATA.PF	1
4	# of records tested	21	SDTM0623	SRCDATA.PF	1
5	# of subjects	4	SDTM0623	SRCDATA.VS	1
6	# of records tested	14	SDTM0623	SRCDATA.VS	1
7	Elapsed time to run check: 0:00:02	.	SDTM0623	CSTCHECK_NOTUNIQUE	1
8	# of distinct check invocations	3	METRICS	SDTM_VALIDATE	1
9	# check invocations not run	2	METRICS	SDTM_VALIDATE	1
10	Errors (severity=High) reported	0	METRICS	SDTM_VALIDATE	1
11	Warnings (severity=Medium) reported	0	METRICS	SDTM_VALIDATE	1
12	Notes (severity=Low) reported	0	METRICS	SDTM_VALIDATE	1
13	Structural errors, warnings and notes	0	METRICS	SDTM_VALIDATE	1
14	Content errors, warnings and notes	2	METRICS	SDTM_VALIDATE	1

Lines 1 through 2 document that the SDTM0451 check was invoked twice. The missing recount value and the absence of other metrics indicate that the two check invocations failed. This should be reported in the Results data set.

Lines 3 through 7 provide metrics information about the SDTM0623 check. SDTM0623 checks that multiple standard units do not exist for any test in the findings domains. The SDTM0623 check was run on two domains using the cstcheck_notunique check macro. The number of subjects and records tested, and the elapsed time to run the check are reported.

Lines 8 through 14 are summary metrics reported at the end of the SDTM validation process in the sdtm_validate macro. There are no errors. It is noted that two checks could not be run (lines 9 and 14).

For more information about the Validation Metrics data set, see Table 6.9 on page 102.

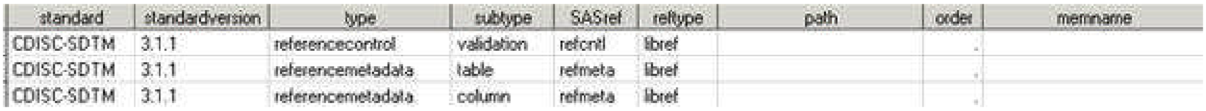
Building a Validation Process

Building a SAS Clinical Standards Toolkit validation process is similar to building any SAS Clinical Standards Toolkit process. The differences are the validation process inputs and outputs, as defined in the SASReferences data set, can differ, a standard-specific validate macro is called, and process output can include an optional Metrics data set.

SASReferences Customizations

A SAS Clinical Standards Toolkit validation process requires that you specify a reference standard with which the source data and metadata can be compared. The following three records, specific to the standard and standardversion of interest, should be included in the SASReferences data set:

Display 6.3 *Defining the Reference Standard in the SASReferences Data Set*



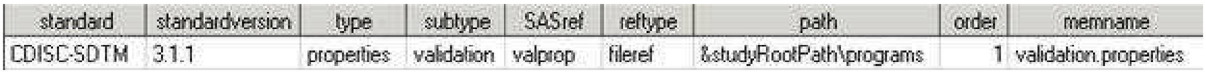
standard	standardversion	type	subtype	SASref	reftype	path	order	memname
CDISC-SDTM	3.1.1	referencecontrol	validation	refcntl	libref		.	
CDISC-SDTM	3.1.1	referencemetadata	table	refmeta	libref		.	
CDISC-SDTM	3.1.1	referencemetadata	column	refmeta	libref		.	

The empty **path** field signals that the path and memname information should be derived from the StandardSASReferences data set associated with the standard and standardversion. Including the referencecontrol and referencemetadata records is unique to validation process in the SAS Clinical Standards Toolkit.

SAS Clinical Standards Toolkit validation can include references to the following files:

1. A validation-specific properties file.

Display 6.4 *Defining the Validation-Specific Properties File in the SASReferences Data Set*



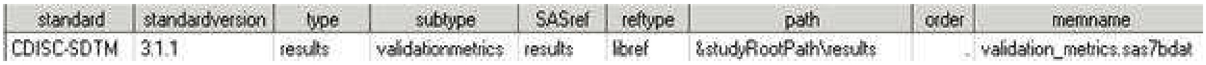
standard	standardversion	type	subtype	SASref	reftype	path	order	memname
CDISC-SDTM	3.1.1	properties	validation	valprop	fileref	&studyRootPath\programs	1	validation.properties

The Validation.Properties file sets process global macro variables specific to validation, such as metrics. For a complete discussion of these properties, see "Validation.Properties" on page 99. For information about the derived global macro variables, see Appendix A1, "Global Macro Variables," on page 211. The Validation.Properties file is a required file to support SAS Clinical Standards Toolkit validation.

For CDISC CRT-DDS, validation properties have been included in the standard-specific Initialize.Properties file. Validation properties do not need to be separately referenced in SASReferences.

2. The output location of any process-generated Metrics data set.

Display 6.5 *Defining the Metrics Output Location in the SASReferences Data Set*

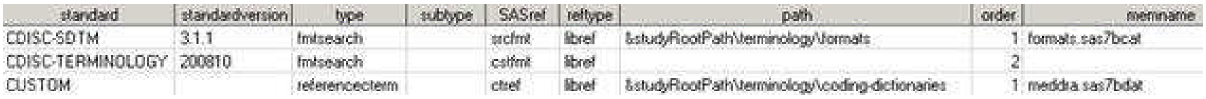


standard	standardversion	type	subtype	SASref	reftype	path	order	memname
CDISC-SDTM	3.1.1	results	validationmetrics	results	libref	&studyRootPath\results	.	validation_metrics.sas7bdat

The Metrics data set provides a summary of the validation process, including error counts, processing time, and denominators for specific checks. For a complete discussion of validation metrics, see "Validation Metrics" on page 102 and "Validation Results and Metrics" on page 116. For information about the global macro variables that govern metrics output, see Appendix A1, "Global Macro Variables," on page 211. The Metrics data set is typically output to the same location as the validation Results data set. This location is common to all SAS Clinical Standards Toolkit processes.

3. The location of any libraries containing controlled terminology, format catalogs, and coding dictionary data sets.

***Display 6.6** Defining Controlled Terminology in the SASReferences Data Set*

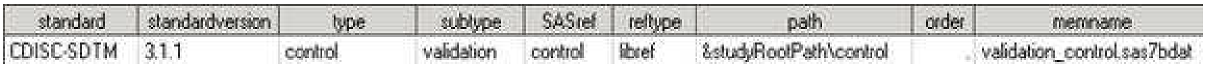


standard	standardversion	type	subtype	SASref	reftype	path	order	memname
CDISC-SDTM	3.1.1	fmtsearch		srcfmt	libref	&studyRootPath\terminology\formats	1	formats.sas7bcat
CDISC-TERMINOLOGY	200810	fmtsearch		cstfmt	libref		2	
CUSTOM		referencecterm		ctref	libref	&studyRootPath\terminology\coding-dictionaries	1	meddra.sas7bdat

The type=fmtsearch records enable you to specify multiple format catalogs (for example, company-wide, compound, group-level, and study-level). Order in the format search path is set by the **order** field. The type=referencecterm record enables you to specify one or more lookup data sets (such as dictionary lookups like LOINC and MedDRA). These lookup data sets do not need to conform to a specific structure, and they do not need to be in a structure that can be read into a SAS format. Customized code (typically in the Validation Master **codelogic** field) is required to join domain data with each associated lookup data set.

4. The location of the run-time Validation Control data set.

***Display 6.7** Defining the Run-Time Validation Control Location in the SASReferences Data Set*



standard	standardversion	type	subtype	SASref	reftype	path	order	memname
CDISC-SDTM	3.1.1	control	validation	control	libref	&studyRootPath\control	.	validation_control.sas7bdat

The Validation Control data set is required and discussed in the following section.

Validation Control: Specification of Run-Time Checks

Each SAS Clinical Standards Toolkit validation process requires you to specify the validation checks to be run. This is accomplished by cloning, subsetting, or building a set of validation checks based on the Validation Master data set. (See "Validation Check of Metadata: Validation Master" on page 90.) The SAS Clinical Standards Toolkit assumes that each Validation Control data set is structurally equivalent to the Validation Master data set.

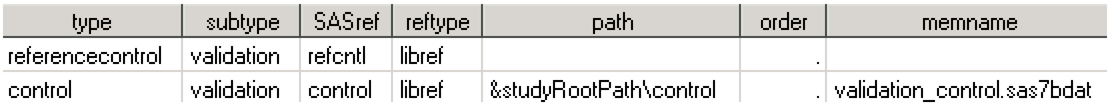
A sample CDISC SDTM 3.1.1 Validation Control data set is deployed to the following SAS 9.1.3 directory. (The deployed location for SAS 9.2 is different, but similar.)

```
!sasroot/../SASClinicalStandardsToolkitSDTM311/1.3/sample/
cdisc-sdtm-3.1.1/sascstdemodata/control
```

By default, the Validation Control data set name is validation_control.sas7bdat.

As a required input to a validation process, the Validation Control data set must be referenced in the run-time SASReferences file. The following display shows how the SASReferences file and the Validation Control data set are defined in the sample CDISC SDTM 3.1.1 SASReferences data set:

Display 6.8 *Defining Validation Control and SASReferences Data Set Locations*



type	subtype	SASref	reftype	path	order	memname
referencecontrol	validation	refcntl	libref		.	
control	validation	control	libref	&studyRootPath\control	.	validation_control.sas7bdat

The &studyRootPath value is assumed to have been set to `!sasroot/../SASClinicalStandardsToolkitSDTM311/1.3/sample/cdisc-sdtm-3.1.1/sascstdemodata`.

The following table provides examples of how to create a Validation Control data set from the Validation Master data set. The sample code is written assuming that the code will be submitted in a context where libraries have been allocated and the format search and autocall paths have been set.

Table 6.10 *Sample Code to Create Validation Control Data Set*

Check Subset	Sample Code

All checks provided with the SAS Clinical Standards Toolkit.	`data control.validation_control;`
	`set refcntl.validation_master;`
	`run;`

Structural checks (metadata-only checks that do not require access to the domain data).	`data control.validation_control;`
	`set refcntl.validation_master (where=(upcase(checktype)="METADATA"));run;`

Content checks (checks that require access to the domain data).	`data control.validation_control;`
	`set refcntl.validation_master (where=(upcase(checktype) ne "METADATA"));`
	`run;`

Checks with a production status.	`data control.validation_control;`
	`set refcntl.validation_master (where=(checkstatus>0));`
	`run;`

WebSDM checks (CDISC SDTM only).	`data control.validation_control;`
	`set refcntl.validation_master (where=(upcase(checksource)= "WEBSDM"));`
	`run;`

Check Subset	Sample Code

Sampling of checks, one for each check macro.	`proc sort data=refcntl.validation_master out=work.control;`
	`by codesource checkid;`
	`run;`
	`data work.control;`
	`set work.control;`
	`by codesource;`
	`if first.codesource;`
	`run;`
	`proc sort data=work.control out=control.validation_control (label="Check sampler");`
	`by checkid;`
	`run;`

CDISC SDTM 3.1.1 checks.	`data control.validation_control;`
	`set refcntl.validation_master (where=(standardVersion = "3.1.1" or standardVersion = "***"));`
	`run;`

All codelist-related checks (checks that use the cstcheck_notincodelist macro).	`data control.validation_control;`
	`set refcntl.validation_master(where=(upcase(checksource)="CSTCHECK_NOTINCODELIST"));`
	`run;`

Check Subset	Sample Code

All checks applicable to a specific domain.	%macro buildcheckdomainlist
	(_cstCheckDS=,_cstOutputDS=work._cstcheckdomains);
	%let _cstOldCheckID=;
	%let _cstCheckSeqCount=0;
	data _null_;
	if 0 then set &_cstCheckDSnobs=_numobs;
	call symputx('_cstCheckCnt',_numobs);
	stop;
	run;
	data &_cstOutputDS;
	attrib checkid format=$8. label="Validation check identifier"
	table format=$32. label="Table Name"
	standardversion format=$20. label="Standard version"
	checksource format=$40. label="Source of check"
	resultseq format=8. label="Unique invocation of check";
	stop;
	run;
	%do check=1 %to &_cstCheckCnt;
	data _null_;
	set &_cstCheckDS (keep=checkid standardversion checksource
	tablescope columnscope usesourcemetadata
	firstObs=&check);
	call symputx('_cstCheckID',checkid);
	call symputx('_cstStandardVersion',standardversion);
	call symputx('_cstChecksource',checksource);
	call symputx('_cstTableScope',tablescope);
	call symputx('_cstColumnScope',columnscope);
	call symputx('_cstUseSourceMetadata',usesourcemetadata);

Check Subset	Sample Code

	stop;
	run;
	%if &_cstCheckID=&_cstOldCheckID %then %do;
	%let _cstCheckSeqCount=%eval(&_cstCheckSeqCount+1) ;
	%end;
	%else %let _cstCheckSeqCount=1;
	%* Call macro to interpret tableScope and columnScope to build work._cstcolumnmetadata for each check *;
	%* _cstDomSubOverride=Y parameter allows us to also look at check records with unequal sublist lengths *;
	%cstutil_buildcollist(_cstFormatType=DATASET,_cstDomSubOverride=Y);
	proc sql noprint;
	create table work._csttempds as
	select distinct table, "&_cstCheckID" as checkid length=8,
	&_cstCheckSeqCount as resultseq,
	"&_cstStandardVersion" as standardversion length=20,
	"&_cstChecksource" as checksource length=40
	from work._cstcolumnmetadata;
	quit;
	proc append base=&_cstOutputDSdata=work._csttempds force;
	run;
	%let _cstOldCheckID=&_cstCheckID;
	* Clear contents for next loop, in case of problems *;
	data work._csttempds;
	set work._csttempds;
	if _n_=1 then stop;
	run;
	%end;
	%mend;

Check Subset	Sample Code

	%* Run this only once per stable reference validation_master - it takes a while... ;
	%buildcheckdomainlist(_cstCheckDS=refcntl.validation_master);
	
	%* The libname for validation_control is assigned in sasreferences. In the sample study it is cntl_v. This might need to be changed either in this macro or the call to it.;
	%macro
	subsetdomainlist(_cstInputDS=work._cstcheckdomains,_
	cstOutputDS=cntl_v.validation_control,
	_cstDomain=);
	proc sql noprint;
	create table &_cstOutputDS as
	select vm.* from refcntl.validation_master vm
	right join &_cstInputDS dom
	on vm.checkid=dom.checkid and
	vm.standardversion=dom.standardversion and
	vm.checksource=dom.checksource
	where table="&_cstDomain";
	quit;
	%mend;
	
	%* Example call: subset validation data set just to those checks for the specified domain ;
	%* Returns all records for checkid/standardversion/checksource if any matches domain - needs tweaking... ;
	%subsetdomainlist(_cstDomain=AE);

Generally, the SAS Clinical Standards Toolkit processes validation checks in the order in which they appear in the Validation Control data set. Each validation process honors the default validation property _cstCheckSortOrder. If this property is not set, then the data set order is assumed. As a part of the Validation Control derivation, checks can be sorted in any user-defined order. Or, _cstCheckSortOrder can be set to sort the Validation Control data set at run time by any fields in that data set.

Best Practice Recommendation: Users might find the prioritization of checks to be helpful in identifying problems early in the process, or for using as prerequisites for checks that follow.

Setting Properties for the Validation Process

Across all standards, the set of properties that are available for a validation process is extensive. (For the full list of properties, see Appendix A1, “Global Macro Variables,” on page 211.) However, only a few properties are modified on a regular basis. These include:

- _cstSASRefsLoc, If you want to point to another location for the SASReferences file.
- _cstSASRefsName, which points to another SASReferences filename.

- _cstSASRefs, which points to a specific libref.sasreferences file to use. (This file is typically in Work.)
- _cstSubjectColumns, which resets the columns that identify a subject.
- _cstReallocateSASRefs, which reallocates SAS librefs and filerefs in the same SAS session, typically when changing studies or standards.
- _cstFMTLibraries, which modifies the format search path built from SASReferences. This change is most often used to add a reference to a Work format catalog.
- _cstCheckSortOrder, which provides a set of Validation Control columns to resort the check processing order.
- _cstMetrics, set to 1 to enable metrics calculations and reporting.
- _cstDebug, which turns on or off debugging for the session.
- _cstDebugOptions, which alters the SAS options when debugging.

These changes should be made before the process setup begins (as changes to the properties file), or after the process setup ends (as a series of %let statements in the code stream).

Best Practice Recommendation: Centralizing property changes in properties files, rather than distributing them in code segments, offers advantages for debugging and documenting processes. Properties are translated to global macro variables by calls to the cst_setstandardproperties or cst_setproperties framework utility macros during process setup. They are reported in the SAS log, and are generally documented in the process SASReferences file.

Running a Validation Process

Sample CDISC SDTM 3.1.1 Driver Program: validate_data.sas

Each SAS Clinical Standards Toolkit process uses a SAS driver module to set up the program execution flow. The following steps show the execution flow in a typical SAS driver module to perform SAS Clinical Standards Toolkit validation. For example, in a SAS 9.2 deployment, the CDISC SDTM 3.1.2 validation driver module can be found in: **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata/programs/validate_data.sas`**

Step 1: Define the study data and metadata locations.

```
/* There are several ways to define the study data and metadata
locations. These include (but are not limited to):
    - Pre-allocation of libraries through some user-defined set-up mechanism
    - Definition within a user-defined driver program such as this one
    - Full explicit definition within a work sasreferences control data set
    - Use of a global macro variable referenced within each sasreferences file

This driver program illustrates use of the last mechanism, setting the
global macro variables studyRootPath and studyOutputPath, which are referenced
within the sample study sasreferences data set path column.

Note this example is dependent on the SAS version and installation folder structure. */
data _null_;
 select("&sysver");
```

```
    when("9.1")
    do;
     call symput('studyRootPath','!sasroot/../SASClinicalStandardsToolkitSDTM312
      /1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata');
     call symput('studyOutputPath','!sasroot/../SASClinicalStandardsToolkitSDTM312
      /1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata');
    end;
    otherwise do;
     call symput('studyRootPath','!sasroot/../../SASClinicalStandardsToolkitSDTM312
      /1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata');
     call symput('studyOutputPath','!sasroot/../../SASClinicalStandardsToolkitSDTM312
      /1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata');
     end;
    end;
  run;
```

The &studyRootPath and &studyOutputPath variables facilitate code standardization and portability. They are not required.

```
%let workPath=%sysfunc(pathname(work));
```

The workPath value provides the path for the Work directory. This directory is referenced within the sample study SASReferences data set path column. It is not required.

```
* Note the number of calls should match the unique
studyOutputPath subdirectories in sasreferences  *;
%****cstutil_createunixsubdir(_cstSubDir=results);      *   <---
example UNIX override  *;
```

The SAS Clinical Standards Toolkit processes normally create one or more output files. These files might reside in the Work directory or point to some external location. The &studyRootPath variable points to read-only locations in the !sasroot folder hierarchy. The &studyOutputPath variable points to writable locations for process output, often in the !sasroot folder heirarchy. UNIX users (or any users) might find it necessary to reset &studyOutputPath to some write-enabled location since the !sasroot directories are typically write protected. For these users, calls to the %cstutil_createunixsubdir macro create any workpath subdirectories that are expected by SASReferences records and set &studyOutputPath to workpath.

```
%let _cstSetupSrc=SASREFERENCES;
%let _cstStandard=CDISC-SDTM;
%let _cstStandardVersion=3.1.2;
```

These convenience macro variables are used primarily for reporting purposes later in the validation process.

Step 2: Set the SAS Clinical Standards Toolkit framework properties and global macro variables. Create an empty work.sasreferences data set to be populated in the validation process.

```
* Set properties provided as part of the CST-FRAMEWORK standard. ;
%cst_setStandardProperties(
_cstStandard=CST-FRAMEWORK,_cstSubType=initialize);
%cst_createds(_cstStandard=CST-FRAMEWORK,
_cstType=control,_cstSubType=reference,
_cstOutputDS=work.sasreferences);
```

Each registered standard should have its own initialize.properties. For each standard that is included in a specific process, the %cst_setStandardProperties macro can be called at this point. Alternatively, type=properties records can be added to the SASReferences data set, and properties are processed when the %cstutil_allocatesasreferences macro is called. This latter approach is followed in the SDTM validate_data.sas driver module.

Step 3: Build the work.sasreferences data set.

The validate_data.sas module initializes the SASReferences data set that is required for SDTM validation. The SASReference data set defines the location and name of the Validation Control data set. The Validation Control data set contains the set of checks to be included in the validation process. The sample validate_data.sas driver progra, sets the path of the Validation Control data set to **`&studyRootPath/control`** and name to **`validation_control.sas7bdat`**. In SAS 9.2, this translates to **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata/control/validation_control.sas7bdat`**. For an explanation of the purpose and content of each SASReferences file, see Chapter 5, "SASReferences File," on page 69. For a fully initialized SASReferences data set for SDTM validation, see Display 5.3 on page 76.

Step 4: Call the %cstutil_processsetup macro.

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

The %cstutil_processsetup macro uses the following parameters.

cstSASReferencesSource
: This parameter determines what initial source setup should be based on. Valid values are SASREFERENCES (default) or RESULTS. If RESULTS is specified, then no other parameters are required, and setup responsibility is passed to the cstutil_reportsetup macro. The Results data set name must be passed to cstutil_reportsetup as libref.memname.

cstSASReferencesLocation
: This parameter specifies the folder location of the SASReferences data set. (The default value is the path to the Work library.)

cstSASReferencesName
: This parameter specifies the name of the SASReferences data set. (The default value is SASREFERENCES.)

The %cstutil_processsetup macro call:

```
%cstutil_processsetup();
```

in the validate_data.sas driver reflects the acceptance of the macro parameter defaults listed above.

The %cstutil_processsetup macro parameter values tell the process where to find the SASReferences data set.

```
    ***********************************************************************;
    * Set global macro variables for the location of the sasreferences  *;
    * file (overrides default properties initialized above              *;
    ***********************************************************************;

    %let _cstSASRefsName=&_cstSASReferencesName;
    %let _cstSASRefsLoc=&_cstSASReferencesLocation;
```

The final setup step for the %cstutil_processsetup macro is a call to the %cstutil_allocatesasreferences utility macro. The SASReferences data set is now interpreted by the SAS Clinical Standards Toolkit. The following actions complete the process:

1. The %cst_insertstandardsasrefs macro is called to insert paths into any records that are missing path information. The information is captured from the StandardSASReferences data set for each standard. For more information about how this works, see "Inserting Information from Registered Standards into a SASReferences File" on page 14.
2. The %cstutil_checkds macro is called to perform internal validation on the SASReferences data set updated in the %cst_insertstandardsasrefs macro.
3. All filerefs and librefs are allocated. (This action is contingent on the _cstReallocateSASRefs property or global macro variable value).
4. Any property files are passed to the %cst_setProperties macro to create global macro variables.
5. The format search path is set if any type=fmtsearch records are found. This is based on the order specified.
6. The autocall path is set if any type=autocall records are found. This is based on the order specified.
7. A Messages data set is created to contain records from each referenced standard. This data set is based on the _cstMessages and _cstMessageOrder properties or global macro variable values. This data set is used for the duration of the process to add fully resolved messages to the Results data set.

At this point, all libraries should be allocated, all paths and global macros should be set, and the global status macro variable _cst_rc should be set to 0. The process is ready to proceed.

This is a common process failure point because of the importance of the SASReferences data set. The SASReferences data set is key to the process, and any errors will cause the process to fail. For tips on debugging problems with the SASReferences data set, see "Special Topic: Debugging a Validation Process" on page 153.

Step 5: Run validation tasks.

```
* Run the standard-specific validation macro. ;
%sdtm_validate;
```

The %sdtm_validate macro performs the following tasks:

1. The macro looks up the Validation Control data set reference from SASReferences.
2. The macro resorts the Validation Control data set based on the _cstCheckSortOrder property or global macro variable value. This step is optional.
3. For each check in the Validation Control data set, this macro calls the check macro specified in the Validation Control **codesource** field. It passes all of the check metadata to the check macro.
4. After all of the checks are run, the following happens:
   - The results are saved to the file specified in SASReferences (type=results, subtype=validationresults).
   - Any process results are summarized in the Metrics data set if specified.
   - The metrics are saved to the file specified in SASReferences (type=results, subtype=validationmetrics).
   - Various SAS Work files are cleaned up if needed.

For tips on debugging if unexpected errors occur, see "Special Topic: Debugging a Validation Process" on page 153.

Step 6: Clean up the session.

```
* Clean up the SAS Clinical Standards Toolkit process
files, macro variables and macros.;
%*cstutil_cleanupcstsession(
     cstClearCompiledMacros=0
    ,cstClearLibRefs=0
    ,cstResetSASAutos=0
    ,cstResetFmtSearch=0
    ,cstResetSASOptions=1
    ,cstDeleteFiles=1
  ,cstDeleteGlobalMacroVars=0);
```

Step 6 is optional, and it is unnecessary with batch processing. You should not clean up prematurely or aggressively if additional SAS Clinical Standards Toolkit processes are to be run in the same interactive SAS session.

The following table summarizes what the SAS Clinical Standards Toolkit attempts to do when each of the %cstutil_cleanupcstsession macro parameters is enabled:

Table 6.11 *Parameter Details for the %cstutil_cleanupcstsession Macro*

Macro Parameter	Action Attempted
_cstClearCompiledMacros	Delete all macros from the work.sasmacr catalog.
_cstResetSASAutos	Reset the SASAutos path based on the value of the macro variable cstInitSASAutos. This macro parameter is typically set in the driver module to capture the SASAutos value at the start of the SAS Clinical Standards Toolkit process (before calling %cstutil_allocatesasreferences). This parameter is ignored if _cstInitSASAutos does not exist.
_cstClearLibRefs	Clear all filerefs and librefs included in SASReferences, except any autocall filerefs.
_cstResetFmtSearch	Reset the fmtsearch path based on the fmtsearch value at the start of the SAS Clinical Standards Toolkit process. This macro parameter is ignored if the work._cstsessionoptions data set does not exist. To support this functionality, this data set is created in the %cstutil_processsetup macro before calling the %cstutil_allocatesasreferences macro.
_cstResetSASOptions	Reset all SAS options back to their status at the start of the SAS Clinical Standards Toolkit process. This macro parameter is ignored if the work._cstsessionoptions data set does not exist. To support this functionality, this data set is created in the %cstutil_processsetup macro before calling the %cstutil_allocatesasreferences macro.

Macro Parameter	Action Attempted
_cstDeleteFiles	Delete files if the global macro variable _cstDebug=0. Files are &_cstsasrefs, &_cstmessages, and work._cstsessionoptions.
_cstDeleteGlobalMacroVars	Call %symdel for all macro variables found in sashelp.vmacro (where=(lowcase(name) =:"_cst" and scope="GLOBAL")).

Validation Results and Metrics

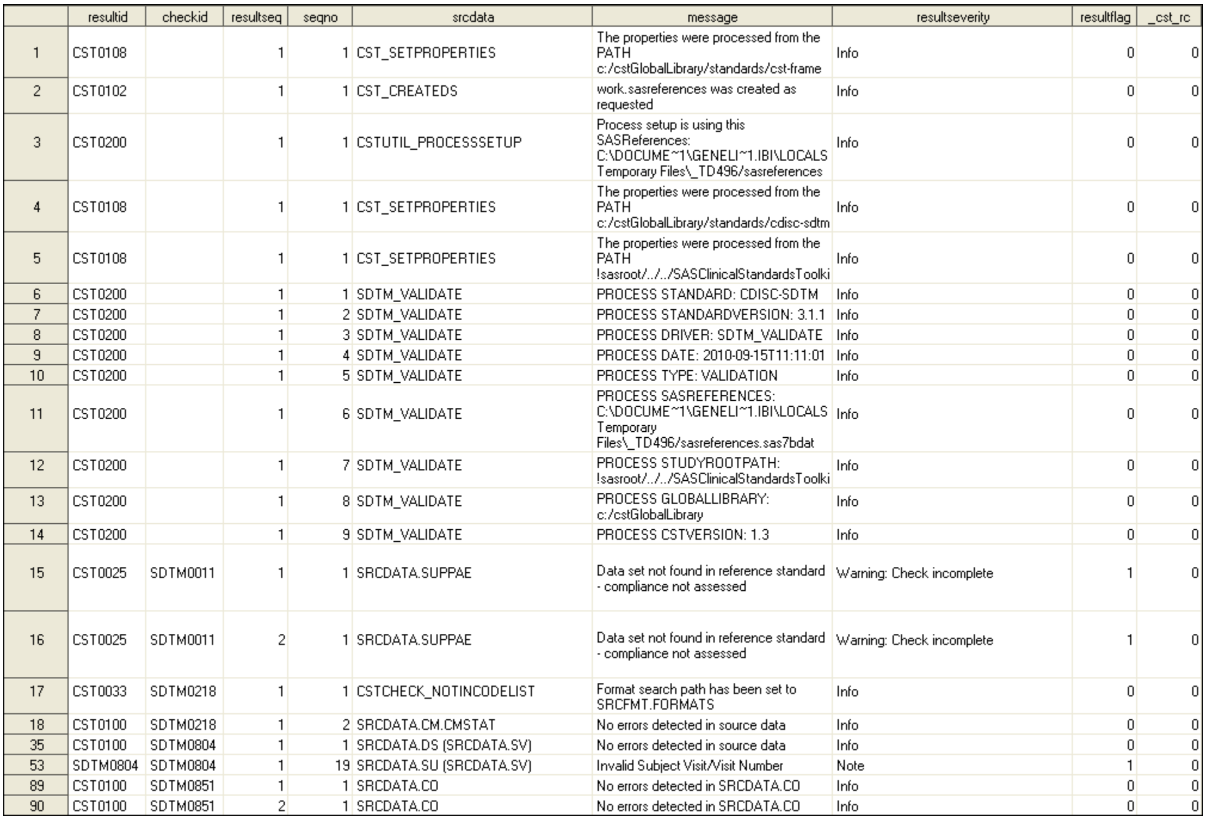
For SAS Clinical Standards Toolkit validation processes, the primary products of each validation process are the Results data set and the Metrics data set. These data sets itemize and summarize the findings of the validation process.

The following displays summarize a sample validation process. A few facts about the sample validation process follow:

1. The validation process was run on CDISC SDTM 3.1.1 source data.
2. It referenced a Validation Control data set that contained metadata for four checks.
3. It included SASReferences records to persist the results as results.validation_results and results.validation_metrics.

   Note: In the following displays, some rows have been hidden to reduce redundant examples.

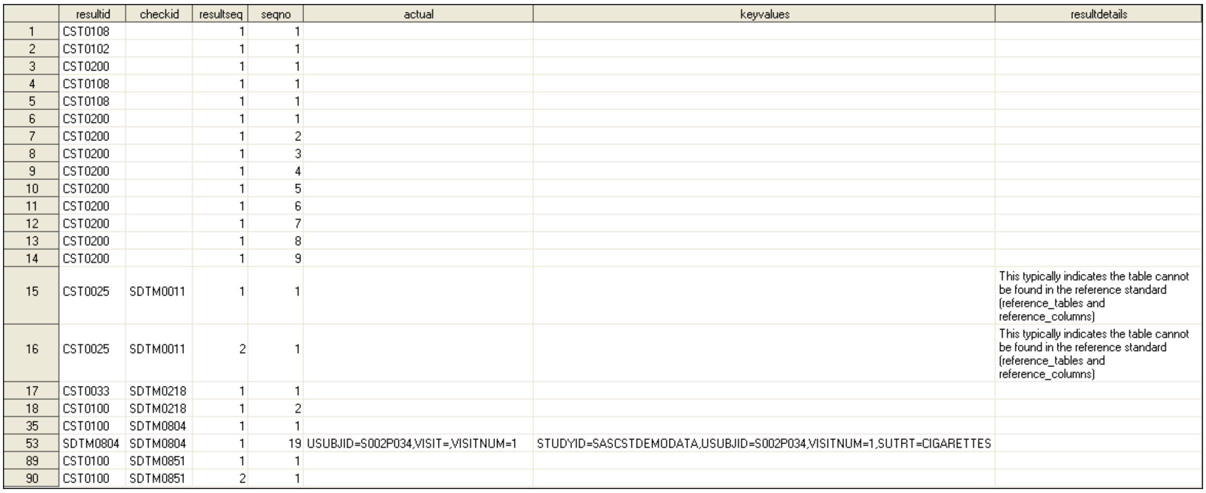
Display 6.9 *Validation Results Data Set (#1)*



	resultid	checkid	resultseq	seqno	srcdata	message	resultseverity	resultflag	_cst_rc
1	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cst-frame	Info	0	0
2	CST0102		1	1	CST_CREATEDS	work.sasreferences was created as requested	Info	0	0
3	CST0200		1	1	CSTUTIL_PROCESSSETUP	Process setup is using this SASReferences: C:\DOCUME~1\GENELI~1.IBI\LOCALS Temporary Files_TD496/sasreferences	Info	0	0
4	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-sdtm	Info	0	0
5	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH !sasroot/../../SASClinicalStandardsToolki	Info	0	0
6	CST0200		1	1	SDTM_VALIDATE	PROCESS STANDARD: CDISC-SDTM	Info	0	0
7	CST0200		1	2	SDTM_VALIDATE	PROCESS STANDARDVERSION: 3.1.1	Info	0	0
8	CST0200		1	3	SDTM_VALIDATE	PROCESS DRIVER: SDTM_VALIDATE	Info	0	0
9	CST0200		1	4	SDTM_VALIDATE	PROCESS DATE: 2010-09-15T11:11:01	Info	0	0
10	CST0200		1	5	SDTM_VALIDATE	PROCESS TYPE: VALIDATION	Info	0	0
11	CST0200		1	6	SDTM_VALIDATE	PROCESS SASREFERENCES: C:\DOCUME~1\GENELI~1.IBI\LOCALS Temporary Files_TD496/sasreferences.sas7bdat	Info	0	0
12	CST0200		1	7	SDTM_VALIDATE	PROCESS STUDYROOTPATH: !sasroot/../../SASClinicalStandardsToolki	Info	0	0
13	CST0200		1	8	SDTM_VALIDATE	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info	0	0
14	CST0200		1	9	SDTM_VALIDATE	PROCESS CSTVERSION: 1.3	Info	0	0
15	CST0025	SDTM0011	1	1	SRCDATA.SUPPAE	Data set not found in reference standard - compliance not assessed	Warning: Check incomplete	1	0
16	CST0025	SDTM0011	2	1	SRCDATA.SUPPAE	Data set not found in reference standard - compliance not assessed	Warning: Check incomplete	1	0
17	CST0033	SDTM0218	1	1	CSTCHECK_NOTINCODELIST	Format search path has been set to SRCFMT.FORMATS	Info	0	0
18	CST0100	SDTM0218	1	2	SRCDATA.CM.CMSTAT	No errors detected in source data	Info	0	0
35	CST0100	SDTM0804	1	1	SRCDATA.DS (SRCDATA.SV)	No errors detected in source data	Info	0	0
53	SDTM0804	SDTM0804	1	19	SRCDATA.SU (SRCDATA.SV)	Invalid Subject Visit/Visit Number	Note	1	0
89	CST0100	SDTM0851	1	1	SRCDATA.CO	No errors detected in SRCDATA.CO	Info	0	0
90	CST0100	SDTM0851	2	1	SRCDATA.CO	No errors detected in SRCDATA.CO	Info	0	0

Note: In the following display, some rows have been hidden to reduce redundant examples.

Display 6.10 *Validation Results Data Set (#2)*



	resultid	checkid	resultseq	seqno	actual	keyvalues	resultdetails
1	CST0108		1	1			
2	CST0102		1	1			
3	CST0200		1	1			
4	CST0108		1	1			
5	CST0108		1	1			
6	CST0200		1	1			
7	CST0200		1	2			
8	CST0200		1	3			
9	CST0200		1	4			
10	CST0200		1	5			
11	CST0200		1	6			
12	CST0200		1	7			
13	CST0200		1	8			
14	CST0200		1	9			
15	CST0025	SDTM0011	1	1			This typically indicates the table cannot be found in the reference standard (reference_tables and reference_columns)
16	CST0025	SDTM0011	2	1			This typically indicates the table cannot be found in the reference standard (reference_tables and reference_columns)
17	CST0033	SDTM0218	1	1			
18	CST0100	SDTM0218	1	2			
35	CST0100	SDTM0804	1	1			
53	SDTM0804	SDTM0804	1	19	USUBJID=S002P034,VISIT=,VISITNUM=1	STUDYID=SASCSTDEMODATA,USUBJID=S002P034,VISITNUM=1,SUTRT=CIGARETTES	
89	CST0100	SDTM0851	1	1			
90	CST0100	SDTM0851	2	1			

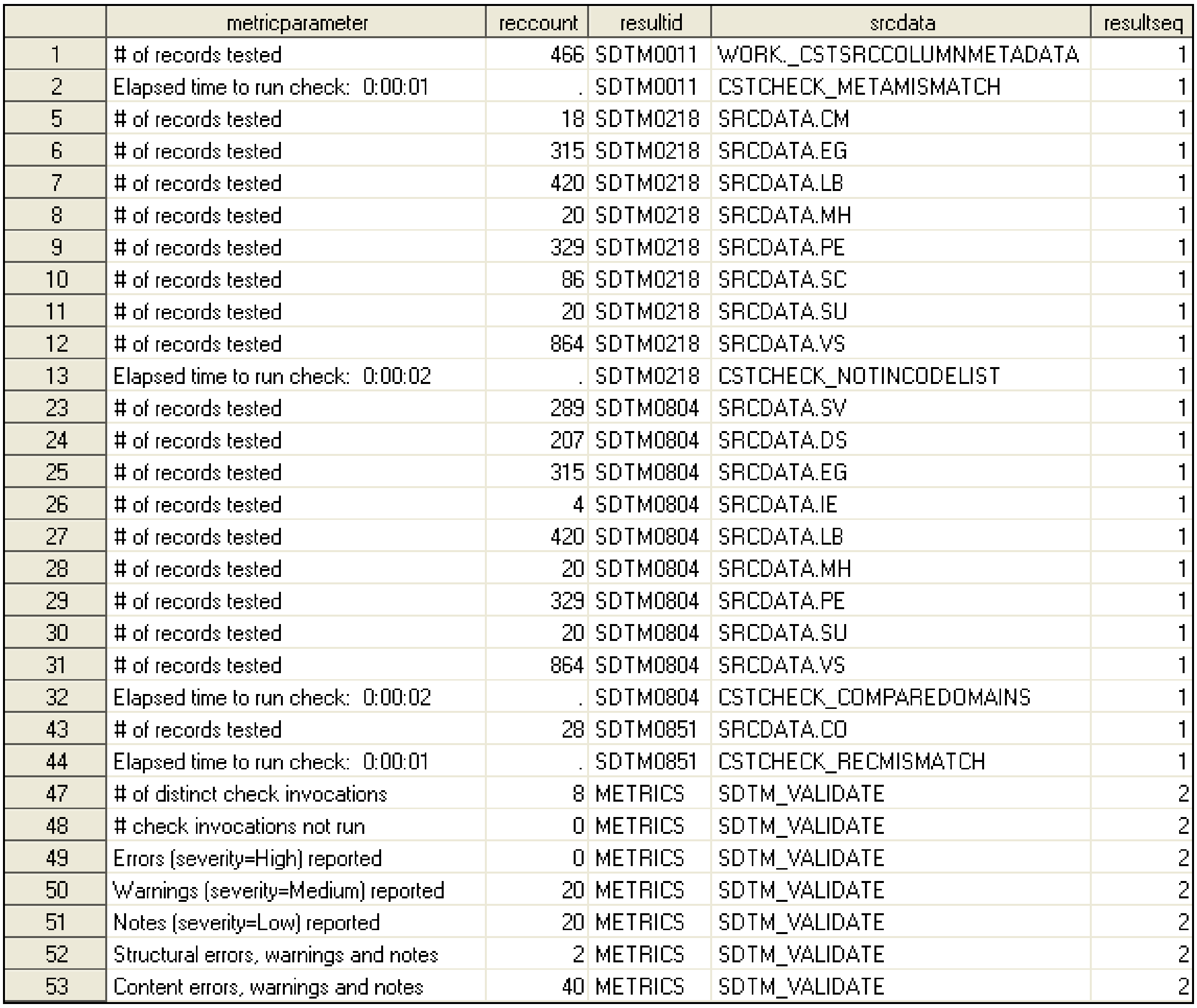
Table 6.12 *Comments about the Validation Results Data Sets in Displays 6.9 and 6.10*

Lines	**Comment**
1,4,5	Informational notes about processing theproperties files.
2	Informational note saying that the creation of work.sasreferences was successful.
3	Informational note from cstutil_processsetup that informs the user of the location of the SASReferences data set.
6-14	Informational summary that provides internal documentation about the process.
15-16	Check SDTM0011 detected an error. SRCDATA.SUPPAE exists in the source metadata (source_tables), but does not exist in the reference metadata (reference_tables). This is a metadata-only check that runs against the source_columns metadata (WORK._ CSTSRCCOLUMNMETADATA). A warning message is displayed informing the user that the check processing was incomplete.
17	Check SDTM0218 informational note that is produced by the check macro cstcheck_notincodelist. Note is about the availability of fmtsearch format catalogs.
18	Check SDTM0218 completed successfully. No errors were detected.

Lines	Comment
35	Check SDTM0804 completed successfully. No problems were found in the comparison of SRCDATA.DS with SRCDATA.SV.
53	In check SDTM0804, one SRCDATA.SU record was found that had invalid VISIT and VISITNUM values, which are relative to records in the SV domain. The actual value in error is listed in the actual column. The keyvalues column identifies the specific record in error.
89-90	Check SDTM0851 completed successfully. No errors were detected. Two records (1 and 2 in the **resultseq** column) are listed because the check was run twice because there is a record for each of two checksources in the Validation Control data set.

Note: In the following display, some rows have been hidden to reduce redundant examples.

Display 6.11 *Validation Metrics Data Set*



	metricparameter	reccount	resultid	srcdata	resultseq
1	# of records tested	466	SDTM0011	WORK._CSTSRCCOLUMNMETADATA	1
2	Elapsed time to run check: 0:00:01	.	SDTM0011	CSTCHECK_METAMISMATCH	1
5	# of records tested	18	SDTM0218	SRCDATA.CM	1
6	# of records tested	315	SDTM0218	SRCDATA.EG	1
7	# of records tested	420	SDTM0218	SRCDATA.LB	1
8	# of records tested	20	SDTM0218	SRCDATA.MH	1
9	# of records tested	329	SDTM0218	SRCDATA.PE	1
10	# of records tested	86	SDTM0218	SRCDATA.SC	1
11	# of records tested	20	SDTM0218	SRCDATA.SU	1
12	# of records tested	864	SDTM0218	SRCDATA.VS	1
13	Elapsed time to run check: 0:00:02	.	SDTM0218	CSTCHECK_NOTINCODELIST	1
23	# of records tested	289	SDTM0804	SRCDATA.SV	1
24	# of records tested	207	SDTM0804	SRCDATA.DS	1
25	# of records tested	315	SDTM0804	SRCDATA.EG	1
26	# of records tested	4	SDTM0804	SRCDATA.IE	1
27	# of records tested	420	SDTM0804	SRCDATA.LB	1
28	# of records tested	20	SDTM0804	SRCDATA.MH	1
29	# of records tested	329	SDTM0804	SRCDATA.PE	1
30	# of records tested	20	SDTM0804	SRCDATA.SU	1
31	# of records tested	864	SDTM0804	SRCDATA.VS	1
32	Elapsed time to run check: 0:00:02	.	SDTM0804	CSTCHECK_COMPAREDOMAINS	1
43	# of records tested	28	SDTM0851	SRCDATA.CO	1
44	Elapsed time to run check: 0:00:01	.	SDTM0851	CSTCHECK_RECMISMATCH	1
47	# of distinct check invocations	8	METRICS	SDTM_VALIDATE	2
48	# check invocations not run	0	METRICS	SDTM_VALIDATE	2
49	Errors (severity=High) reported	0	METRICS	SDTM_VALIDATE	2
50	Warnings (severity=Medium) reported	20	METRICS	SDTM_VALIDATE	2
51	Notes (severity=Low) reported	20	METRICS	SDTM_VALIDATE	2
52	Structural errors, warnings and notes	2	METRICS	SDTM_VALIDATE	2
53	Content errors, warnings and notes	40	METRICS	SDTM_VALIDATE	2

Table 6.13 *Comments About the Validation Metrics Data Set*

Lines	**Comment**
1	In check SDTM0011, 466 columns were evaluated.
2	Check SDTM0011 took one second to run using cstcheck_metamismatch.
5-13	Check SDTM0218 ran against eight domains. Record counts were provided for each domain. The check took two seconds to run using cstcheck_notincodelist.
23-32	Check SDTM0804 ran against nine domains. Record counts were provided for each domain. The check took two seconds to run using cstcheck_comparedomains.

Lines	Comment
43-44	Check SDTM0851 evaluated 28 records in the SRCDATA.CO domain. The check took one second to run using cstcheck_recmismatch.
47	A summary metric of unique check invocations. Display 6.11 on page 119 does not itemize all eight checks.
48	A summary metric of the number of checks that failed to run. (These metrics are defined as distinct checkid and resultseq combinations in the Results data set where resultflag=-1).
49-53	Summary metric counts of the number of records, by type of metric, in the Results data set.

Note: In Display 6.9 on page 116 and Display 6.10 on page 117, some records in the validation Results data set have been deleted for brevity. This creates an inconsistency with the metrics listed in Display 6.11 on page 119.

The following are some general observations:

- The absence of a value in the results.checkid field can be used as an indicator of whether messaging has been set up. If the **checkid** field is nonmissing in a Results record, then messaging related to a specific validation check is available.
- A resultseq value > 1 indicates a repeat invocation of a specific validation check. There should be differences in the Validation Control metadata for the specific validation check.
- The **seqno** field is intended to be a record (message) counter in each specific check invocation. Generally, this value starts with 1 on the first record, and increments by 1 until the last record for each checkid and resultseq combination. One exception is with the Validation Control column reportAll=N. This signals the code to not write a record to the Results data set for each record in error. However, seqno continues to increment in this case, resulting in a gap in seqno values, with the last seqno approximating the total number of records in error.

A set of sample validation reports is available to summarize the SAS Clinical Standards Toolkit validation process results and metrics. For more information, see Chapter 8, “Reporting,” on page 195.

Sample CDISC CRT-DDS 1.0 Driver Program: validate_crtdds_data.sas

SAS Clinical Standards Toolkit validation of the SAS representation of the CDISC CRT-DDS standard follows the same pattern used for CDISC SDTM validation. A sample driver module—validate_crtdds_data.sas—is provided to perform process setup steps and to call the crtdds_validate.sas macro. For a more complete description of the validation of the SAS representation of the CDISC CRT-DDS standard, see Chapter 7, “XML-Based Standards,” on page 165. In this chapter, the use of the validate_crtdds_data driver module is described.

Validation Checks by Standard

CDISC SDTM 3.1.1

The SAS Clinical Standards Toolkit 1.3 provides 150 unique SDTM 3.1.1 validation checks. These checks are derived from four sources.

- The SAS interpretation of the CDISC SDTM WebSDM 2.6 documented checks. See the white paper at:

  `http://phaseforward.com/resource/whitepapers/Validation Checks 2.6/WebSDM V2.6 Validation Checks FINAL.pdf`

- An update to the WebSDM validation checks (Version 3.0, revised June 2009) available at:

  `http://www.phaseforward.com/products/cdisc/`

- Checks supporting loads into the Janus study data repository being developed by the FDA and the NCI. This information is documented in the *SDTM Validation Specification, v.1.0, November 2007* available at:

  `http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM190628.pdf`

- SAS checks based on SAS data management and cleaning experiences building CDISC SDTM domains.

The CDISC SDTM 3.1.1 Validation Master data set, as defined in the SAS Clinical Standards Toolkit 1.3, contains 257 records. Even though the SAS Clinical Standards Toolkit provides 150 unique CDISC SDTM 3.1.1 checks, there are 257 records in the Validation Master data set. The Validation Master data set is built with multiple instances of the checks. This better supports check selection by version or checksource (that is, WebSDM, Janus, or customer-defined checks), and enables unique check logic and messaging by version or checksource. Of these 257 checks, three are inactive, and 12 are deprecated. Deprecated CDISC SDTM checks generally reflect changes in the WebSDM specifications over time.

Note: The validation check data set column checkstatus is designed to provide an indication of the “state” of each check. It says whether the check is ready to be run in its current defined state, or should it be run based on some external criteria. Valid values are 1 (active), 0 (inactive), -1 (deprecated), and -2 (not yet implemented). Values are extensible to meet the user's given requirements. No SAS Clinical Standards Toolkit code requires specific values. You can elect to use other values such as 0 (draft), 1 (test), and 2 (production). If a check is included in the run-time validation control data set, then SAS Clinical Standards Toolkit attempts to run the check as defined, regardless of the value of the checkstatus column.

The following table provides the distribution of all 257 CDISC SDTM validation checks by the original source of the check (the Validation Master **checksource** field).

Table 6.14 *Distribution of CDISC SDTM 3.1.1 Validation Checks*

Check Source	Count	Deprecated	Inactive
WebSDM	114	5	1

Check Source	Count	Deprecated	Inactive
Janus	53	2	0
JanusFR	58	3	1
SAS	32	2	1
Total	257	12	3

This does not mean that the SAS Clinical Standards Toolkit 1.3 supports 114 different WebSDM checks or 32 unique SAS checks. There are multiple instances of specific checks to handle different sets of SDTM domains. For example, check SDTM0604 assesses whether the sequence numbers (**SEQ) are consecutively numbered. For most domains, this is assessed within each patient (USUBJID). However, the trial summary (TS) domain does not contain patient-level data, so the check logic differs. The Validation Master metadata differs for these two instances of the SDTM0604 check, but reports the same error message for the check.

Information about the 257 records in the CDISC SDTM 3.1.1 Validation Master data set is itemized in Appendix A4, "CDISC SDTM Validation Checks," on page 281. Only selected columns are listed in the appendix. For a full description of a sample Validation Master data set for the CDISC SDTM standard, see Table 6.4 on page 95.

Consider the interrelationships among SAS Clinical Standards Toolkit validation check metadata. All run-time Validation Control data sets, any programs that build or derive from these data sets, corresponding Messages data sets, and the Validation_StdRef data set are examples of how interconnected many SAS Clinical Standards Toolkit metadata files are. For more information about the Messages data set, see "Messages" on page 101. By default, the Validation_StdRef data set is found in the `*<global standards library directory>*/standards/cdisc-sdtm-3.1.1-1.3/validation/control` folder.

Note: Cuurently, the SAS Clinical Standards Toolkit does not fully support all WebSDM checks. Checks that are not supported require a comparison between SDTM metadata and an associated define.xml file. Loads into the Janus repository require the existence and use of a define.xml file. However, the SAS Clinical Standards Toolkit 1.3 does not require an associated define.xml file for SDTM validation. For more information, see the SAS site `support.sas.com` for SAS Notes, other usage notes, and their current status.

CDISC SDTM 3.1.2

SAS Clinical Standards Toolkit 1.3 provides 243 unique SDTM 3.1.2 validation checks. These checks are derived from four sources.

- The SAS interpretation of the CDISC SDTM WebSDM 3.0 documented checks. Documentation is available at `http://www.phaseforward.com/products/cdisc/`.
- The SAS interpretation of OpenCDISC CDISC SDTM 3.1.2 validation rules. The validation rules are available at `http://www.opencdisc.org/projects/validator/cdisc-sdtm-3.1.2-validation-rules`.
- Checks supporting loads into the Janus study data repository being developed by the FDA and the NCI. This information is documented in the *SDTM Validation*

Specification, v1.0, November 2007 available at **`http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM190628.pdf`**.

- SAS checks based on SAS data management and cleaning experiences building CDISC SDTM domains.

The CDISC SDTM 3.1.2 Validation Master data set, as defined in the SAS Clinical Standards Toolkit 1.3, contains 247 records. Even though the SAS Clinical Standards Toolkit provides 243 unique CDISC SDTM 3.1.1 checks, there are 247 records in the Validation Master data set. Of these 247 checks, one is inactive, eight are deprecated, and 18 are not implemented. Deprecated CDISC SDTM checks generally reflect changes in the WebSDM specifications over time. Checks that are not implemented generally involve a comparison of the CDISC SDTM data, metadata, or both with an associated define.xml file. Such cross-standard validation is not supported in the current release of SAS Clinical Standards Toolkit. In the SAS Clinical Standards Toolkit 1.3, the Janus and JanusFR checks were dropped for SDTM 3.1.2.

The following table provides the distribution of all 247 CDISC SDTM validation checks by the original source of the check (the Validation Master **checksource** field).

Table 6.15 *Distribution of CDISC SDTM 3.1.2 Validation Checks*

Check Source	Count	Inactive	Deprecated	Not Implemented
WebSDM	164	0	5	12
Janus	1	0	1	0
OpenCDISC	44	0	0	6
SAS	38	1	2	0
Total	247	1	8	18

Note: The SAS Clinical Standards Toolkit allows multiple invocations of the same validation check. Multiple invocations for four checks account for the difference between the 243 unique checks and 247 records in the validation master data set. For example, check SDTM0604 assesses whether the sequence numbers (**SEQ) are consecutively numbered. For most domains, this is assessed within each patient (USUBJID). However, the trial summary (TS) domain does not contain patient-level data, so the check logic differs. The Validation Master metadata differs for these two instances of the SDTM0604 check, but reports the same error message for the check.

Information about the 247 records in the CDISC SDTM 3.1.2 Validation Master data set is itemized in Appendix A4, “CDISC SDTM Validation Checks,” on page 281. Only selected columns are listed in the appendix.

Consider the interrelationships among SAS Clinical Standards Toolkit validation check metadata. All run-time Validation Control data sets, any programs that build or derive from these data sets, corresponding Messages data sets, and the validation_stdref data set are examples of how interconnected many SAS Clinical Standards Toolkit metadata files are. For more information about the Messages data set, see “Messages” on page 101. By default, the Validation_StdRef data set is found in the **`<`*`global standards library directory`*`>/standards/cdisc-sdtm-3.1.2-13/validation/control`** folder.

Note: Currently, the SAS Clinical Standards Toolkit does not fully support all WebSDM checks. Checks that are not supported require a comparison between SDTM metadata and an associated define.xml file. Loads into the Janus repository require the existence and use of a define.xml file. However, the SAS Clinical Standards Toolkit 1.3 does not require an associated define.xml file for SDTM validation. For more information, see the SAS site at `support.sas.com` for SAS Notes, other usage notes, and their current status.

CDISC CRT-DDS 1.0

The SAS Clinical Standards Toolkit provides check macros that validate the data in the SAS data sets representing CDISC CRT-DDS data. The goal of these check macros is to ensure that all data is correctly specified and that referential integrity is maintained. As a result, a standards-compliant CDISC define.xml file can be produced from these data sets.

The validity of CRT-DDS data is determined by the standard in the form of XML schema definitions. These XML schema definitions must be translated into checks appropriate for the relational and tabular format.

Checks fall into these general categories:

- Ensures that all cross-table references are satisfied and that the referenced item actually exists (referential integrity).
- Ensures that required variables are not missing or empty for an observation or row.
- Ensures that character data conforms to a particular format.

Formats are specified in the standard in one of two ways:

- an enumeration
- a regular expression

The following table lists the types of checks for CRT-DDS data.

Each check type is assumed to operate on data that exists in a source column in a source data set. A check type can reference one or more parameters that validate the source column data. A parameter can be a character string or a representation of some column other than the source column against which the source column data must be compared.

All character comparisons are case sensitive. Character data is assumed to have been trimmed of leading or trailing white space.

Table 6.16 *CRT-DDS Validation Check Types*

Check Type	Check ID	Category	Description
Unique in data set	CRT0100	Structural	No two values for the source column can be the same in the same source data set.
Required character value	CRT0101	Data	The trimmed (white space removed) value of the character data must consist of one or more characters.
Required numeric value	CRT0101	Data	The numeric value of the column cannot be missing.
Enumeration(s0,s1,...)	CRT0114	Data	If character data exists, its value must match one of the enumerated character strings. All string comparisons are case sensitive.

Check Type	Check ID	Category	Description
Foreign key(targetColumn)	CRT0110	Structural	Each existing value in this column must have an equivalent value in the target column.
Foreign key required(targetColumn)	(1)	Structural	A value is required for this column in every row. Each value must have an equivalent value in the target column. This check is the equivalent of running the required character value check, and this check failing if that check fails. If the required character value passes, the foreign key() check is run.
Character format: language	CRT0106	Data	The character data must consist of 1 to 8 alphabetical characters of any case. It can be followed by a hyphen and any sequence of 1 to 8 alphabetical characters in any case or numeric digits after that hyphen. For example, e is a legal value, as is en-us, english, and english-d842. Illegal values include 1en, mumblespeak, and en_us. The hyphen character sequence can be repeated, making a value such as english-mumbly-growly-47 a legal value. Regular expression: [a-zA-Z]{1,8}(-[a-zA-Z0-9]{1,8})*.
Character format: fileName	CRT0107	Data	The character data must not contain any characters other than uppercase and lowercase letters of the alphabet, numeric digits, an underscore (_), or a period. Regular expression: [A-Za-z0-9_.]+.
Character format: sasFormat	CRT0109	Data	The first character must be either a lowercase or uppercase letter, an underscore (_), or the dollar sign ($). Any subsequent character must be either an uppercase or lowercase letter, a numeric digit, an underscore (_), or a period. Regular expression: [A-Za-z_$][A-Za-z0-9_.]*.
Character format: sasName	CRT0108	Data	The first character must be either a lowercase or uppercase letter or an underscore (_). Any subsequent character must be either an uppercase or lowercase letter, a numeric digit, or an underscore (_). Regular expression: [A-Za-z_][A-Za-z0-9_]*.
Unique across data sets(targetcolumn0,...)	CRT0112	Structural	No value in this column can be the same as any value in any of the data set columns.
Primary key	(2)	Data	Must be unique in data set check type and the required character value check type.
Must Have Corresponding Value(targetColumn)	CRT0111	Structural	For each distinct value in this column, there must be at least one equivalent value in the target column.
No Duplicates Per Unique Value(targetColumn)	CRT0113	Structural	For each distinct value in the target column, each value in the source column must be unique. That is, the same value cannot appear more than once in the source column for each distinct value in the target column.

(1) This validation is a combination of checks CRT0101 and CRT0110.

(2) This validation is a combination of checks CRT0100 and CRT0101.

Each check type belongs to one of two categories.

1. Data checks have no dependencies on data outside of the source table. An example is ensuring that a value exists in a column in which values cannot be missing.
2. Structural checks deal with relationships and data integrity between tables. Foreign key enforcement is an example of a structural check. Structural conditions must be met for the successful generation of a define.xml file. A user might want to defer structural checks until later in the process of populating the CRT-DDS data sets. This is because foreign key relationships require that the data be made available in a particular order (that is, a referenced key must be available before the foreign key to it can exist).

Table 6.17 *CRT-DDS Validation Checks*

CRT-DDS Validation Number	Source Data Set	Check ID	Variable Being Checked	Check
0000	DefineDocument	CRT0100, CRT0101	FileOID	Primary key
0001	DefineDocument	CRT0101	FileType	Required character value
0002	DefineDocument	CRT0100	ID	Unique in data set
0003	DefineDocument	CRT0112	ID	Unique across data sets (MDVLeaf.ID, ItemGroupLeaf.ID)
0005	DefineDocument	CRT0114	FileType	Enumeration ("Snapshot", "Transactional")
0006	DefineDocument	CRT0114	Archival	Enumeration ("Yes")
0007	DefineDocument	CRT0114	Granularity	Enumeration ("All", "Metadata", "AdminData", "ReferenceData", "AllClinicalData", "SingleSite", "SingleSubject")
0008	Study	CRT0101, CRT0110	FK_DefineDocument	Foreign key required (DefineDocument.FileOID)
0009	Study	CRT0100, CRT0101	OID	Primary key
0147	Study	CRT0101	StudyName	Required character value
0148	Study	CRT0101	StudyDescription	Required character value
0149	Study	CRT0101	ProtocolName	Required character value
0010	MeasurementUnits	CRT0100, CRT0101	OID	Primary key
0011	MeasurementUnits	CRT0101	Name	Required character value

CRT-DDS Validation Number	Source Data Set	Check ID	Variable Being Checked	Check
0012	MeasurementUnits	CRT0101, CRT0110	FK_Study	Foreign key required (Study.OID)
0013	MUTranslatedText	CRT0106	lang	Character format: language
0014	MUTranslatedText	CRT0101, CRT0110	FK_ MeasurementUnits	Foreign key required (MeasurementUnits.OID)
0015	MetaDataVersion	CRT0100, CRT0101	OID	Primary key
0016	MetaDataVersion	CRT0101	Name	Required character value
0017	MetaDataVersion	CRT0101, CRT0110	FK_Study	Foreign key required (Study.OID)
0150	MetaDataVersion	CRT0101	DefineVersion	Required character value
0151	MetaDataVersion	CRT0101	StandardName	Required character value
0152	MetaDataVersion	CRT0101	StandardVersion	Required character value
0018	AnnotatedCRFs	CRT0101, CRT0110	leafID	Foreign key required (MDVLeaf.ID)
0019	AnnotatedCRFs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0020	SupplementalDocs	CRT0101, CRT0110	leafID	Foreign key required (MDVLeaf.ID)
0021	SupplementalDocs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0022	MDVLeaf	CRT0100, CRT0101	ID	Primary key
0023	MDVLeaf	CRT0111	ID	Must have corresponding value (MDVLeafTitles.FK_MDVLeaf)
0024	MDVLeaf	CRT0112	ID	Unique across data sets (DefineDocument.ID, ItemGroupLeaf.ID)
0025	MDVLeaf	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)

CRT-DDS Validation Number	Source Data Set	Check ID	Variable Being Checked	Check
0026	MDVLeafTitles	CRT0101, CRT0110	FK_MDVLeaf	Foreign key required (MDVLeaf.ID)
0027	ComputationMethods	CRT0100, CRT0101	OID	Primary key
0028	ComputationMethods	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0029	ValueLists	CRT0100, CRT0101	OID	Primary key
0030	ValueLists	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0031	ValueListItemRefs	CRT0101, CRT0110	ItemOID	Foreign key required (ItemDefs.OID)
0032	ValueListItemRefs	CRT0114	Mandatory	Enumeration ("Yes", "No")
0033	ValueListItemRefs	CRT0110	ImputationMethodOID	Foreign key (ImputationMethods.OID)
0034	ValueListItemRefs	CRT0110	RoleCodeListOID	Foreign key (CodeLists.OID)
0035	ValueListItemRefs	CRT0101, CRT0110	FK_ValueLists	Foreign key required (ValueLists.OID)
0036	ValueListItemRefs	CRT0101	Mandatory	Required character value
0037	ProtocolEventRefs	CRT0101, CRT0110	StudyEventOID	Foreign key required (StudyEventDefs.OID)
0038	ProtocolEventRefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0039	ProtocolEventRefs	CRT0114	Mandatory	Enumeration ("Yes", "No")
0040	ProtocolEventRefs	CRT0101	Mandatory	Required character value
0041	ProtocolEventRefs	CRT0113	StudyEventOID	No duplicates per unique value (FK_MetaDataVersion)
0042	ProtocolEventRefs	CRT0113	OrderNumber	No duplicates per unique value (FK_MetaDataVersion)
0043	StudyEventDefs	CRT0100, CRT0101	OID	Primary key

CRT-DDS Validation Number	**Source Data Set**	**Check ID**	**Variable Being Checked**	**Check**
0044	StudyEventDefs	CRT0101	Name	Required character value
0045	StudyEventDefs	CRT0114	Repeating	Enumeration ("Yes", "No")
0046	StudyEventDefs	CRT0101	Repeating	Required character value
0047	StudyEventDefs	CRT0114	Type	Enumeration ("Scheduled", "Unscheduled", "Common")
0048	StudyEventDefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0153	StudyEventDefs	CRT0101	Type	Required character value
0049	StudyEventFormRefs	CRT0101, CRT0110	FormOID	Foreign key required (FormDefs.OID)
0050	StudyEventFormRefs	CRT0114	Mandatory	Enumeration ("Yes", "No")
0051	StudyEventFormRefs	CRT0101	Mandatory	Required character value
0052	StudyEventFormRefs	CRT0101, CRT0110	FK_StudyEventDefs	Foreign key required (StudyEventDefs.OID)
0053	StudyEventFormRefs	CRT0113	FormOID	No duplicates per unique value (StudyEventFormRefs.FK_ StudyEventDefs)
0054	StudyEventFormRefs	CRT0113	OrderNumber	No duplicates per unique value (StudyEventFormRefs.FK_ StudyEventDefs)
0055	FormDefs	CRT0100, CRT0101	OID	Primary key
0056	FormDefs	CRT0101	Name	Required character value
0057	FormDefs	CRT0114	Repeating	Enumeration ("Yes", "No")
0058	FormDefs	CRT0101	Repeating	Required character value
0059	FormDefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0060	FormDefItemGroupRefs	CRT0101, CRT0110	ItemGroupOID	Foreign key required (ItemGroupDefs.OID)
0061	FormDefItemGroupRefs	CRT0114	Mandatory	Enumeration ("Yes", "No")

CRT-DDS Validation Number	Source Data Set	Check ID	Variable Being Checked	Check
0062	FormDefItemGroupRefs	CRT0101	Mandatory	Required character value
0063	FormDefItemGroupRefs	CRT0101, CRT0110	FK_FormDefs	Foreign key required (FormDefs.OID)
0064	FormDefItemGroupRefs	CRT0113	OrderNumber	No duplicates per unique value (FormDefItemGroupRefs.FK_ FormDefs)
0065	FormDefItemGroupRefs	CRT0113	ItemGroupOID	No duplicates per unique value (FormDefItemGroupRefs.FK_ FormDefs)
0066	FormDefArchLayouts	CRT0100, CRT0101	OID	Primary key
0067	FormDefArchLayouts	CRT0101	PdfFileName	Required character value
0068	FormDefArchLayouts	CRT0107	PdfFileName	Character format: filename
0069	FormDefArchLayouts	CRT0110	PresentationOID	Foreign key (Presentation.OID)
0070	FormDefArchLayouts	CRT0101, CRT0110	FK_FormDefs	Foreign key required (FormDefs.OID)
0071	ItemGroupDefs	CRT0100, CRT0101	OID	Primary key
0072	ItemGroupDefs	CRT0111	OID	Must have corresponding value (ItemGroupDefItemRefs.ItemOID)
0073	ItemGroupDefs	CRT0101	Name	Required character value
0074	ItemGroupDefs	CRT0114	Repeating	Enumeration ("Yes", "No")
0075	ItemGroupDefs	CRT0101	Repeating	Required character value
0076	ItemGroupDefs	CRT0114	IsReferenceData	Enumeration ("Yes", "No")
0077	ItemGroupDefs	CRT0108	SASDatasetName	Character Format: sasName
0078	ItemGroupDefs	CRT0101	Label	Required character value
0079	ItemGroupDefs	CRT0101	ArchiveLocationID	Required character value
0080	ItemGroupDefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)

CRT-DDS Validation Number	Source Data Set	Check ID	Variable Being Checked	Check
0081	ItemGroupDefItemRefs	CRT0101, CRT0110	ItemOID	Foreign key required (ItemDefs.OID)
0082	ItemGroupDefItemRefs	CRT0114	Mandatory	Enumeration ("Yes", "No")
0083	ItemGroupDefItemRefs	CRT0101	Mandatory	Required character value
0084	ItemGroupDefItemRefs	CRT0110	ImputationMethodOID	Foreign key (ImputationMethods.OID)
0085	ItemGroupDefItemRefs	CRT0110	RoleCodeListOID	Foreign key (CodeLists.OID)
0086	ItemGroupDefItemRefs	CRT0101, CRT0110	FK_ItemGroupDefs	Foreign key required (ItemGroupDefs.OID)
0087	ItemGroupDefItemRefs	CRT0113	OrderNumber	No duplicates per unique value (ItemGroupDefItemRefs.FK_ ItemGroupDefs)
0088	ItemGroupDefItemRefs	CRT0113	ItemOID	No duplicates per unique value (ItemGroupDefItemRefs.FK_ ItemGroupDefs)
0154	ItemGroupDefItemRefs	CRT0101	Role	Required character value
0089	ItemGroupAliases	CRT0101	Context	Required character value
0090	ItemGroupAliases	CRT0101	Name	Required character value
0091	ItemGroupAliases	CRT0101, CRT0110	FK_ItemGroupDefs	Foreign key required (ItemGroupDefs.OID)
0092	ItemGroupLeaf	CRT0100, CRT0101	ID	Primary key
0093	ItemGroupLeaf	CRT0112	ID	Unique across data sets (DefineDocument.ID, MDVLeaf.ID)
0094	ItemGroupLeaf	CRT0101, CRT0110	FK_ItemGroupDefs	Foreign key required (ItemGroupDefs.OID)
0095	ItemGroupLeafTitles	CRT0101, CRT0110	FK_ItemGroupLeaf	Foreign key required (ItemGroupLeaf.ID)
0096	ItemDefs	CRT0100, CRT0101	OID	Primary key
0097	ItemDefs	CRT0101	Name	Required character value

CRT-DDS Validation Number	Source Data Set	Check ID	Variable Being Checked	Check
0098	ItemDefs	CRT0114	DataType	Enumeration ("integer", "float", "date", "datetime", "time", "text", "string")
0099	ItemDefs	CRT0101	DataType	Required character value
0100	ItemDefs	CRT0108	SASFieldName	Character format: sasName
0101	ItemDefs	CRT0108	SDSVarName	Character format: sasName
0102	ItemDefs	CRT0110	CodeListRef	Foreign key (CodeLists.OID)
0103	ItemDefs	CRT0110	ComputationMethodOID	Foreign key (ComputationMethods.OID)
0104	ItemDefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0105	ItemQuestionTranslatedText	CRT0106	lang	Character format: language
0106	ItemQuestionTranslatedText	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0107	ItemQuestionExternal	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0108	ItemMURefs	CRT0101, CRT0110	MeasurementUnitOID	Foreign key required (MeasurementUnits.OID)
0109	ItemMURefs	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0110	ItemRangeChecks	CRT0100, CRT0101	OID	Primary key
0111	ItemRangeChecks	CRT0111	OID	Must have corresponding value (ItemRangeCheckValues.OID)
0112	ItemRangeChecks	CRT0101	Comparator	Required character value
0113	ItemRangeChecks	CRT0114	Comparator	Enumeration ("LT", "LE", "GT", "GE", "EQ", "NE", "IN", "NOTIN")
0114	ItemRangeChecks	CRT0101	SoftHard	Required character value
0115	ItemRangeChecks	CRT0114	SoftHard	Enumeration ("Soft", "Hard")

CRT-DDS Validation Number	Source Data Set	Check ID	Variable Being Checked	Check
0116	ItemRangeChecks	CRT0101, CRT0110	MURefOID	Foreign key required (MeasurementUnits.OID)
0117	ItemRangeChecks	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0118	ItemRangeCheckValues	CRT0101, CRT0110	FK_ItemRangeChecks	Foreign key required (ItemRangeChecks.OID)
0119	RCErrorTranslatedText	CRT0106	lang	Character format: language
0120	RCErrorTranslatedText	CRT0101, CRT0110	FK_ItemRangeChecks	Foreign key required (ItemRangeChecks.OID)
0121	ItemRole	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0122	ItemAliases	CRT0101	Context	Required character value
0123	ItemAliases	CRT0101	Name	Required character value
0124	ItemAliases	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0125	ItemValueListRefs	CRT0101, CRT0110	ValueListOID	Foreign key required (ValueLists.OID)
0126	ItemValueListRefs	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0127	CodeLists	CRT0100, CRT0101	OID	Primary key
0128	CodeLists	CRT0101	Name	Required character value
0129	CodeLists	CRT0114	DataType	Enumeration ("integer", "float", "text")
0130	CodeLists	CRT0101	DataType	Required character value
0131	CodeLists	CRT0109	SASFormatName	Character format: sasFormat
0132	CodeLists	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0133	ExternalCodeLists	CRT0101, CRT0110	FK_CodeLists	Foreign key required (CodeLists.OID)

CRT-DDS Validation Number	Source Data Set	Check ID	Variable Being Checked	Check
0134	ExternalCodeLists	CRT0112	FK_CodeLists	Unique across data sets (CodeListItems.FK_CodeLists)
0135	CodeListItems	CRT0100, CRT0101	OID	Primary key
0137	CodeListItems	CRT0101, CRT0110	FK_CodeLists	Foreign key required (CodeLists.OID)
0138	CodeListItems	CRT0112	FK_CodeLists	Unique across data sets (ExternalCodeLists.FK_CodeLists)
0139	CodeListItems	CRT0113	CodedValue	No duplicates per unique value (FK_CodeLists)
0140	CLItemDecodeTranslatedText	CRT0106	lang	Character format: language
0141	CLItemDecodeTranslatedText	CRT0101, CRT0110	FK_CodeListItems	Foreign key required (CodeListItems.OID)
0142	ImputationMethods	CRT0100, CRT0101	OID	Primary key
0143	ImputationMethods	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0144	Presentation	CRT0100, CRT0101	OID	Primary key
0145	Presentation	CRT0106	lang	Character format: language
0146	Presentation	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)

The CDISC CRT-DDS validation checks that are listed in Table 6.17 on page 126 are performed by comparing the data against a set of expected values. The expected values have been stored in a format catalog (crtddsct.sas7bcat) and a data set (crtddsct.sas7bdat). They can be found in the `<`***`global standards library directory`***`>/standards/cdisc-crtdds-1.0-1.3/formats` folder. The following table lists the format names and values that are used during CRT-DDS validation. This methodology ensures case-sensitivity compliance required by the XML schema validation. For example, the ItemRangeChecks data set requires an enumeration edit for values such as LT and LE. If mixed case or lowercase values are detected, then the validation check reports an error. In this case, the validation check is CRT0114 , (see Table 6.17 on page 126) and it uses the Comp format to report this as an error.

*Table 6.18 Enumeration Validation Format Values**

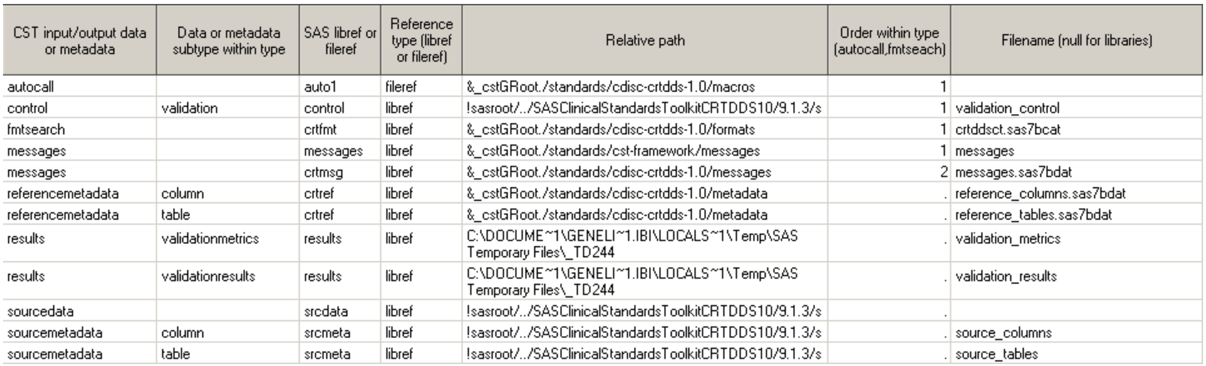
Format	Value 1	Value 2
Filetype	Snapshot Transactional	Snapshot Transactional
NY	Yes No	Yes No
Y	Yes	Yes
Gran	All Metadata AdminData ReferenceData AllClinicalData SingleSite SingleSubject	All Metadata AdminData ReferenceData AllClinicalData SingleSite SingleSubject
Type	Scheduled Unscheduled Common	Scheduled Unscheduled Common
IDType	integer float date datetime time text string	integer float date datetime time text string
Comp	LT LE GT GE EQ NE IN NOTIN	LT LE GT GE EQ NE IN NOTIN
Soft	Soft Hard	Soft Hard

Format	Value 1	Value 2
CLType	integer float text	integer float text

*Value 1 and Value 2 are case sensitive.

The SASReferences data set needs to contain a row for **fmtsearch**, with **SAS libref** set to **crtfmt** and the **Filename** should refer to **crtddsct.sas7bcat**.

Display 6.12 *Example SASReferences File*



CST input/output data or metadata	Data or metadata subtype within type	SAS libref or fileref	Reference type (libref or fileref)	Relative path	Order within type (autocall,fmtseach)	Filename (null for libraries)
autocall		auto1	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0/macros	1	
control	validation	control	libref	!sasroot/../SASClinicalStandardsToolkitCRTDDS10/9.1.3/s	1	validation_control
fmtsearch		crtfmt	libref	&_cstGRoot./standards/cdisc-crtdds-1.0/formats	1	crtddsct.sas7bcat
messages		messages	libref	&_cstGRoot./standards/cst-framework/messages	1	messages
messages		crtmsg	libref	&_cstGRoot./standards/cdisc-crtdds-1.0/messages	2	messages.sas7bdat
referencemetadata	column	crtref	libref	&_cstGRoot./standards/cdisc-crtdds-1.0/metadata	.	reference_columns.sas7bdat
referencemetadata	table	crtref	libref	&_cstGRoot./standards/cdisc-crtdds-1.0/metadata	.	reference_tables.sas7bdat
results	validationmetrics	results	libref	C:\DOCUME~1\GENELI~1.IBI\LOCALS~1\Temp\SAS Temporary Files_TD244	.	validation_metrics
results	validationresults	results	libref	C:\DOCUME~1\GENELI~1.IBI\LOCALS~1\Temp\SAS Temporary Files_TD244	.	validation_results
sourcedata		srcdata	libref	!sasroot/../SASClinicalStandardsToolkitCRTDDS10/9.1.3/s	.	
sourcemetadata	column	srcmeta	libref	!sasroot/../SASClinicalStandardsToolkitCRTDDS10/9.1.3/s	.	source_columns
sourcemetadata	table	srcmeta	libref	!sasroot/../SASClinicalStandardsToolkitCRTDDS10/9.1.3/s	.	source_tables

Special Topic: Validation Check Macros

The following SAS Clinical Standards Toolkit design requirements shaped the implementation of SAS Clinical Standards Toolkit validation code:

1. Code modules should be generic and reusable across standards. Fourteen check macros support CDISC SDTM 3.1.1 and 3.1.2 validation. CDISC CRT-DDS 1.0 uses six of these macros.
2. Code must run with SAS 9.1.3 (that is, no functionality new to SAS 9.2).
3. Code should be written as SAS macros.
4. SAS macros should have simple parameter signatures. All macros accept a single parameter, _cstControl, which is a single-observation data set that contains check-specific metadata.
5. SAS macros should be implemented as non-compiled open code.
6. SAS macros should be callable using the SAS autocall facility. The SAS Clinical Standards Toolkit framework supports a single sasmacros library. Each SAS Clinical Standards Toolkit standard supports an additional macros library, and the macro library is available using the SAS autocall path.
7. Code modules should be generic and reusable with multiple validation checks. The SAS Clinical Standards Toolkit check macros support 150 unique CDISC SDTM 3.1.1 validation checks, 243 unique CDISC SDTM 3.1.2 validation checks, and 11 unique CDISC CRT-DDS validation checks.

8. To support code generalization, use metadata-driven techniques to provide check-specific information to the check macros, even including which check macro to call.
9. Code should write processing results to a single validation Results data set. This Results data set should be available for post-process review and reporting.

These design requirements should be used when developing custom validation check macros. The following table identifies and describes the purpose of each of the 14 check macros provided with the SAS Clinical Standards Toolkit.

Table 6.19 *SAS Clinical Standards Toolkit Validation Check Macros*

Check Macro	Code Logic Style	Description of Purpose
cstcheck_notsorted	(not used)	Identifies any domain that is not sorted by the keys defined in the metadata.
cstcheck_zeroobs	(not used)	Identifies any data set with zero observations.
cstcheck_metamismatch	Step	Identifies inconsistencies between study and reference column metadata.
cstcheck_column	Statement	Identifies any invalid column values or attributes.
cstcheck_columncompare	Step	Supports comparison of column values.
cstcheck_dsmismatch	Step	Identifies any data set mismatches between study and template metadata and the source data library.
cstcheck_violatesstd	Statement	Identifies any invalid column values defined in a reference standard.
cstcheck_notunique	Not used for functions 1 through 3; DATA step for function 4	A multi-function macro that assesses the uniqueness of data sets, columns, or value-pairs from two columns. Function 1: Is data set unique by a set of columns? Function 2: For any subject, are column values unique? Function 3: Does a combination of two columns have unique values? Function 4: Are the values in one column (Column2) consistent in each value of another column (Column1)?

Check Macro	Code Logic Style	Description of Purpose
cstcheck_notconsistent	Step	Identifies any inconsistent column values across records.
cstcheck_recnotfound	Step	Compares the consistency of one or more columns across two tables or enables the comparison of the consistency of one <table>.<column> with another <table>.<column>.
cstcheck_comparedomains	Step	Compares values for one or more columns in one domain with values for those same columns in another domain.
cstcheck_recmismatch	Step	Identifies any record mismatches across domains (domain as referenced in another domain).
cstcheck_notincodelist	If lookuptype=DATASET, DATA step code logic required Else, DATA step code logic is optional	Identifies any column values inconsistent with controlled terminologies. Requires reference to the SAS format search path built based on type=FMTSEARCH records in the SASReferences control file. Example is a **STAT value is found other than 'NOT DONE.'
cstcheck_notimplemented	(not used)	Placeholder to report that a check is not yet implemented.

Each validation check macro follows a standard basic workflow. Several of the 14 validation check macros perform more complex operations and multiple functions. The basic workflow includes the following:

1. Call the utility macro %cstutil_readcontrol, which translates the validation check metadata passed as the input parameter into local macro variables for check macro processing.
2. Evaluate required check macro-specific metadata values.
3. Call the utility macro %cstutil_buildcollist (or, if processing only domains, %cstutil_builddomlist), which evaluates the requested scope of the specific validation check (that is, which tables and columns are to be included when running the check).
4. Loop through the target tables and columns identified in step 3.
5. Perform the logic required to properly assess the validation check. This might be the check macro code itself, or the code in the validation check metadata **codeLogic** field.
6. Write any informational or error messages to the Results data set. Metrics are written to the Metrics data set.

7. Clean up any Work files local to the check macro processing.

The following table provides the distribution of validation checks by check macro for both CDISC SDTM 3.1.1 and 3.1.2. For the distribution of validation checks by check macro for CDISC CRT-DDS 1.0, see Table 6.21 on page 140.

Table 6.20 *CDISC SDTM Validation Checks*

Check Macro (codesource)	Type of Check (checktype)	Unique Check Identifier (checkid) (add SDTM prefix)
cstcheck_notsorted	Multirecord	0601
cstcheck_zeroobs	Metadata	0001, 0002, 0003
cstcheck_metamismatch	Metadata	0011, 0012, 0013, 0014, 0015, 0019, 0020, 0022, 0023, 0030, 0031, 0032, 0033
cstcheck_column	Column	0271, 0493, 0494, 0860
	ColumnAttribute	0124, 0125, 0126, 0127, 0128, 0129, 0130, 0131
	ColumnValue	0204, 0205, 0206, 0207, 0217, 0220, 0222, 0251, 0352, 0354, 0355, 0452, 0490, 0506, 0521, 0562
	Date	0101, 0102
cstcheck_columncompare	Column	0208, 0212, 0213, 0214, 0215, 0216, 0219, 0223, 0225, 0226, 0231, 0232, 0233, 0351, 0353, 0405, 0406, 0408, 0409, 0410, 0411, 0412, 0413, 0414, 0415, 0416, 0417, 0418, 0419, 0422, 0423, 0462, 0463, 0500, 0501, 0502, 0503, 0507, 0511, 0534, 0541, 0551, 0561, 0843
	Date	0209, 0210, 0211, 0407
cstcheck_dsmismatch	Metadata	0004, 0005, 0006, 0017
cstcheck_violatesstd	Column	0201, 0202, 0203, 0606
cstcheck_notunique	Multirecord	0602, 0603, 0622, 0623, 0631, 0641, 0642, 0651, 0661, 0662, 0671, 0808, 0809
cstcheck_notconsistent	Multirecord	0604, 0605, 0607, 0621, 0643, 0644
	Multitable	0807
cstcheck_recnotfound	Multitable	0802, 0803, 0805, 0806, 0811, 0821, 0822, 0823, 0831, 0836, 0841

Check Macro (codesource)	Type of Check (checktype)	Unique Check Identifier (checkid) (add SDTM prefix)
cstcheck_comparedomains	Multitable	0645, 0801, 0804, 0812, 0842, 0844, 0845, 0846
cstcheck_recmismatch	Multitable	0851, 0861, 0862, 0863, 0864, 0865, 0866, 0871, 0872
cstcheck_notincodelist	Cntlterm	0218, 0221, 0302, 0401, 0402, 0403, 0450, 0451, 0453, 0454, 0455, 0456, 0457, 0458, 0459, 0460, 0461, 0464, 0465, 0466, 0467, 0470, 0471, 0472, 0473, 0475, 0476, 0477, 0478, 0479, 0480, 0481, 0482, 0483, 0484, 0485, 0486, 0487, 0488, 0489, 0491, 0492, 0495, 0496, 0497, 0498, 0499, 0504, 0505, 0508, 0509, 0510, 0512, 0513, 0514, 0515, 0516, 0517, 0518, 0522, 0523, 0531, 0532, 0533, 0570, 0571, 0572, 0573, 0574, 0575, 0576, 0580
	Column	0301, 0303
cstcheck_notimplemented	Cntlterm	0449, 0474
	Date	0190, 0191, 0192, 0193
	Derivation	0441, 0442, 0443
	Metadata	0016, 0034, 0035, 0036, 0037, 0038, 0039
	Multirecord	0672, 0673

Table 6.21 *CDISC CRT-DDS 1.0 Validation Checks*

Check Macro (codesource)	Check Type (checktype)	Unique Check Identifier (checkid))	Corresponding CRT-DDS Validation Number*
cstcheck_column	ColumnValue	CRT0106	0013, 0105, 0119, 0140, 0145
		CRT0107	0068
		CRT0108	0077, 0100, 0101
		CRT0109	0131

Check Macro (codesource)	Check Type (checktype)	Unique Check Identifier (checkid))	Corresponding CRT-DDS Validation Number*
cstcheck_violatesstd	Column	CRT0101	0000, 0001, 0008, 0009, 0010, 0011, 0012, 0014, 0015, 0016, 0017, 0018, 0019, 0020, 0021, 0022, 0025, 0026, 0027, 0028, 0029, 0030, 0031, 0035, 0036, 0037, 0038, 0040, 0043, 0044, 0046, 0048, 0049, 0051, 0052, 0055, 0056, 0058, 0059, 0060, 0062, 0063, 0066, 0067, 0070, 0071, 0073, 0075, 0078, 0079, 0080, 0081, 0083, 0086, 0089, 0090, 0091, 0092, 0094, 0095, 0096, 0097, 0099, 0104, 0106, 0107, 0108, 0109, 0110, 0112, 0114, 0116, 0117, 0118, 0120, 0121, 0122, 0123, 0124, 0125, 0126, 0127, 0128, 0130, 0132, 0133, 0135, 0137, 0141, 0142, 0143, 0144, 0146, 0147, 0148, 0149, 0150, 0151, 0152, 0153, 0154
cstcheck_notunique	Multirecord	CRT0100	0000, 0002, 0004, 0009, 0010, 0015, 0022, 0027, 0029, 0043, 0055, 0066, 0071, 0092, 0096, 0110, 0127, 0135, 0142, 0144
cstcheck_recnotfound	Multitable	CRT0110	0008, 0012, 0014, 0017, 0018, 0019, 0020, 0021, 0025, 0026, 0028, 0030, 0031, 0033, 0034, 0035, 0037, 0038, 0048, 0049, 0052, 0059, 0060, 0063, 0069, 0070, 0080, 0081, 0084, 0085, 0086, 0091, 0094, 0095, 0102, 0103, 0104, 0106, 0107, 0108, 0109, 0116, 0117, 0118, 0120, 0121, 0124, 0125, 0126, 0132, 0133, 0137, 0141, 0143, 0146
		CRT0111	0023, 0072, 0111
		CRT0112	0003, 0024, 0093, 0134, 0138
cstcheck_recmismatch	Multitable	CRT0113	0041, 0042, 0053, 0054, 0064, 0065, 0087, 0088, 0139

Check Macro (codesource)	Check Type (checktype)	Unique Check Identifier (checkid))	Corresponding CRT-DDS Validation Number*
cstcheck_notincodelist	Controlterm	CRT0114	0005, 0006, 0007, 0032, 0039, 0045, 0047, 0050, 0057, 0061, 0074, 0076, 0082, 0098, 0113, 0115, 0129

For a full listing of validation checks, see Appendix A5, "CDISC CRT-DDS 1.0 Validation Checks," on page 335.

More complete documentation is provided for each check macro in Appendix A3, "Macro Application Programming Interface," on page 225. This information is derived from the code header. For tips on building new check macros, see "Special Topic: Validation Customization" on page 158.

Special Topic: How SAS Clinical Standards Toolkit Interprets Validation Check Metadata

Overview

Four Validation Master metadata fields are key to how the SAS Clinical Standards Toolkit processes source data and source metadata: usesourcemetadata, tablescope, columnscope, and codelogic.

The SAS Clinical Standards Toolkit uses usesourcemetadata to point to the correct metadata. If usesourcemetadata is set to Y, then the SAS Clinical Standards Toolkit knows that the source metadata (source_tables and source_columns) is to be used to derive the domains and columns to be evaluated for compliance to the standard. If usesourcemetadata is set to N, reference metadata (reference_tables and reference_columns) is to be used.

The SAS Clinical Standards Toolkit uses the tablescope and columnscope values to build the work._csttablemetadata and work._cstcolumnmetadata data sets. Based on the values of these fields, the SAS Clinical Standards Toolkit creates a subset of source metadata or reference metadata that represents the union of tablescope and columnscope. The SAS Clinical Standards Toolkit builds columns specified in columnscope that also exist in the tables specified in tablescope.

For those checks that use codelogic, the SAS Clinical Standards Toolkit builds local macro variables to communicate tablescope and columnscope settings to the code. Simple examples are each domain is interpreted as &_cstDSName, and each column is interpreted as &_cstColumn.

Code logic is run. If the check code logic is a statement (codetype=1 or 3), then _cstError=1 is generally set. If the check code logic is a DATA step or PROC SQL code segment (codetype=2 or 4), then work.cstproblems is created.

Case Study 1: CDISC SDTM Check SDTM0604

In this case study, whether the sequence numbers (**SEQ) used in various domains are consecutively incremented beginning at 1 for each USUBJID is determined.

There are specific values to assign to usesourcemetadata, tablescope, and columnscope to set up a proper test of sequence numbers. First, you want to include the domains you actually have (that is, source data and metadata). So, set usesourcemetadata to Y. Next, you want to test all domains that contain sequence numbers. So, set tablescope to _ALL_. Because each domain uses a domain-specific name for sequence number, set columnscope to "**SEQ".

The following is the code logic for CDISC SDTM check SDTM0604:

```
%let _cstLastKey=%scan(%quote(&_cstSubjectKeys),-1,",");
data work._cstproblems (drop=count);
 set &_cstDSName (keep=&_cstDSKeys &_cstColumn);
 by &_cstDSKeys;
 if first.&_cstLastKey then count=1;
 else count+1;
 if &_cstcolumn ne count then output;
run;
```

The following five macro variables are used in this code. They are representative of variables set in many of the check macros before calling code logic. See each validation check macro for local macro variables available to code logic.

- _cstDSName is the name of the domain, as set in the calling code module.
- _cstSubjectKeys is the set of keys that define a subject. It is set once as a global macro variable in a properties file.
- _cstDSKeys contains the data set keys for _cstDSName. Keys are derived from the table metadata for that domain (source_tables.keys).
- _cstLastKey is the last subject key. In the CDISC SDTM case, the value is USUBJID.
- _cstColumn is the column of interest (sequence number). This variable is specific to the _cstDSName domain.

Processing based on Validation Master metadata fields results in records being added to work._cstproblems for any record that does not match the record counter within the subject.

However, there are two records in the Validation Master check data set for the CDISC SDTM check SDTM0604. The tablescope and columnscope settings for each record differ from the previous description. The CDISC SDTM TS (Trial Summary) domain does not contain the subject key USUBJID. The previous code logic does run against the TS domain without failing. (But, the SAS log indicates a problem: **`NOTE: Variable first.USUBJID is uninitialized.`**). A better solution is offered in the Validation Master check data set with the two records.

Table 6.22 Multiple Validation Check Invocations for a Specific CheckID

checkid	tablescope	columnscope	code logic

SDTM0604	_ALL_-TS	**SEQ	%let _cstLastKey=%scan(%quote(&_cstSubjectKeys),-1,",");
			data work._cstproblems (drop=count);
			set &_cstDSName (keep=&_cstDSKeys &_cstColumn);
			by &_cstDSKeys;
			if first.&_cstLastKey then count=1;
			else count+1;
			if &_cstcolumn ne count then output;
			run;

SDTM0604	TS	TSSEQ	data work._cstproblems;
			set &_cstDSName (keep=&_cstDSKeys &_cstColumn);
			if &_cstcolumn ne _n_ then output;
			run;

Case Study 2: CDISC SDTM Check SDTM0623

In this case study, whether the values for standard units (**STRESU) are consistent within each test code (**TESTCD) across all records in the CDISC SDTM findings domains is determined.

You want to include the domains you actually have (that is, source data and metadata). So, set usesourcemetadata to Y. Next, you want to test all findings domains, which typically contain these two domain columns (**STRESU and **TESTCD). So, you might want to set tablescope to CLASS:FINDINGS. Because you want to compare two columns in each domain, set columnscope to [**TESTCD][**STRESU]. For more information about tablescope and columnscope syntax, see Table 6.3 on page 90.)

The code logic for CDISC SDTM check SDTM0623 is listed:

```
data work._cstunique;
   set work._cstunique;
          by &_cstColumn1 &_cstColumn2;
   if first.&_cstColumn1=0 or last.&_cstColumn1=0 then _checkError=1;
run;
proc sort data=&_cstDSName out=&_cstclds;
    by &_cstColumn1 &_cstColumn2;
run;
data work._cstuniqueerrors;
    merge work._cstunique (where=(_checkerror=1) in=un)
                 &_cstclds (in=ds);
      by &_cstColumn1 &_cstColumn2;
    if un and ds and first.&_cstColumn2;
run;
```

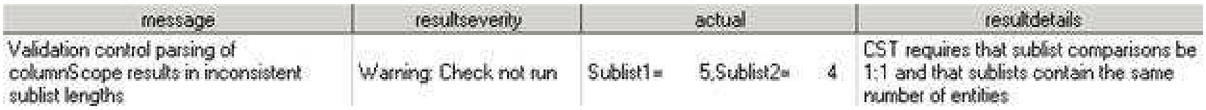
This case study shows how the SAS Clinical Standards Toolkit uses local macro variables for column comparisons. The columnscope syntax [**TESTCD][**STRESU] tells the

SAS Clinical Standards Toolkit to create two sublists. The first sublist is for all TESTCD columns, and the second is for all STRESU columns. These are referenced as &_cstColumn1 and &_cstColumn2 in code logic, respectively.

In this case, the validation check macro that calls and interprets code logic output (cstcheck_notunique) reports all work._cstuniqueerrors records as failing this instance of CDISC SDTM check SDTM0623.

It fails now because of the way it has been configured. The following sections shows how to solve the problem. The generated Results data set contains the following excerpt:

Display 6.13 *Results Data Set Excerpt for Check SDTM0623*



message	resultseverity	actual	resultdetails
Validation control parsing of columnScope results in inconsistent sublist lengths	Warning: Check not run	Sublist1= 5,Sublist2= 4	CST requires that sublist comparisons be 1:1 and that sublists contain the same number of entities

The **actual** and **resultdetails** values give clues about the problem. The SAS Clinical Standards Toolkit resolves the columnscope sublist [**TESTCD] to five columns. It resolves the sublist [**STRESU] to four columns. The SAS Clinical Standards Toolkit column comparisons require sublists of equal length so that valid comparisons can be made. There appears to be a findings domain that has TESTCD, but not STRESU. In this case, the domain IE does not have the column IESTRESU. Attempting to compare IETESTCD with LBSTRESU is not the intention.

Tablescope and columnscope syntax supports wildcarding and addition and subtraction operators. However, this flexible functionality is not required. You can submit explicit table and column references. CDISC SDTM check SDTM0623 could be defined in the Validation Master data set as the following:

tablescope	**columnscope**
EG	[EGTESTCD][EGSTRESU]
LB	[LBTESTCD][LBSTRESU]
SC	[SCTESTCD][SCSTRESU]
VS	[VSTESTCD][VSSTRESU]

Consider the following alternative definition for the check:

tablescope	**columnscope**
CLASS:FINDINGS-IE	[**TESTCD][**STRESU]

Both of the above definitions will run correctly, but do not yet match the record metadata for SDTM0623 in the SAS Validation Master data set:

tablescope	**columnscope**
CLASS:FINDINGS-LB-IE	[**TESTCD][**STRESU]

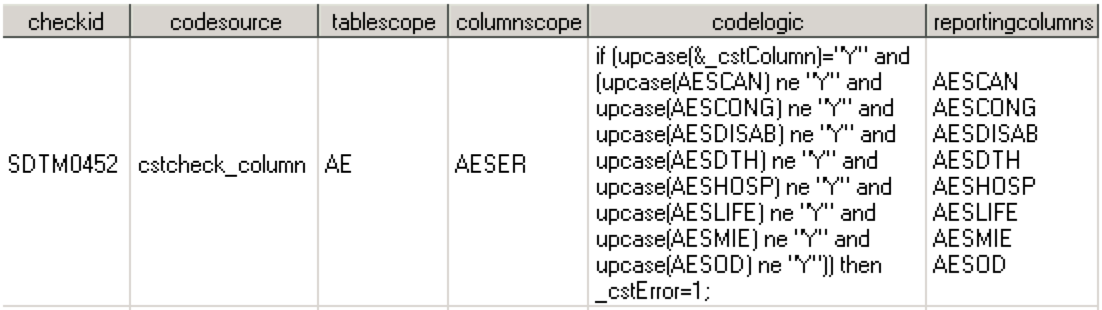
The reason `LB` is excluded from tablescope is because CDISC SDTM check SDTM0631 is a specific test of these LB domain columns (the Validation Master **checksource** and **sourceid** fields show SDTM0631 to be an implementation of the WebSDM check IR5006). SDTM0623 is simply a generalization of SDTM0631 to include all findings domains. There is no reason to redundantly test LB.

Case Study 3: CDISC SDTM Check SDTM0452

In the CDISC SDTM Adverse Events (AE) domain, AE is defined as serious (AESER="Y"), but none of the serious qualifier columns has been set to "Y". (For example, AE involves cancer (AESCAN).)

This case study is an example of a validation check with a specific implementation using hardcoded column references with no wildcarding.

Display 6.14 *Validation Check Metadata for Check SDTM0452*



checkid	codesource	tablescope	columnscope	codelogic	reportingcolumns
SDTM0452	cstcheck_column	AE	AESER	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;	AESCAN AESCONG AESDISAB AESDTH AESHOSP AESLIFE AESMIE AESOD

The **tablescope** and **columnscope** fields specify a single table and column. The **reportingcolumns** value is used by the cstcheck_column check macro to include these columns for check processing and to report the column values in the Results data set **actual** field.

But what happens if, in your source study, you did not collect all eight of the qualifier columns expected by the check metadata? By default, an error message such as the following is written to the SAS log:

```
ERROR: The variable AESCAN in the DROP, KEEP, or RENAME list has
never been referenced.
```

The following are the options for solving this problem:

1. Do not run the check at all. The check can be removed from the run-time Validation Control data set. Disabling a check can be done by removing it from the Validation Master data set or by setting the checkstatus flag to some value other than 1. (This assumes that the process that you use to extract checks into the run-time Validation Control data set references the **checkstatus** field).

2. Add missing (all null) columns to your AE domain, and include those columns in the source_columns metadata. (In this example, these columns are defined in the CDISC SDTM standard as permissible columns.) While this solves the immediate problem, it is not a recommended best practice, which is to exclude all-null columns defined as permissible. In fact, adding all-null columns triggers the reporting of another check (SDTM0605).

3. Can **columnscope** be modified to include the qualifier columns that you do have? The answer is no. The **columnscope** field defines the scope of the primary columns to be tested. The check macro code loops through the resolved set of columns from the

columnscope field, and it evaluates the validity of each column. In this case, you do not want to test each of the qualifier columns against each other.

4. Modify the current check record metadata with updated codelogic and **reportingcolumns** values. This seems simple enough. Remove the columns that are not collected in the source study. Your best practices control this action. For example:

   - Can existing checks be modified?
   - Will any change in the Validation Master super-set of checks require a new Validation Master data set?
   - Will any change require you to assign a new checkid (for example, CUST0452) and other new metadata (for example, checktype=CUSTOM)?
   - How should the **uniqueid** field be modified to reflect the new check?

   You should consider the implications of modifying existing checks with regard to future product updates and synchronization of changes. For more information, see "Special Topic: Validation Customization" on page 158.

5. Submit a request to SAS to modify the existing check to be more generic and flexible. For example, modify the code logic to check that each variable exists first before attempting to evaluate its value.

Special Topic: SAS Implementation of ISO 8601

ISO 8601 is a widely used data standard for dates, times, durations, and intervals. The values are stored as text strings. They are formatted in a way that ensures that all of the components are always unambiguous. ISO 8601 is both platform and software independent, which makes it suitable for data interchange.

Many data standards use a simplified subset of ISO 8601 for specifying their own dates, times, and durations. This is true of several CDISC standards, including SDTM.

A complete discussion of ISO 8601 and the CDISC subset of ISO 8601 is beyond the scope of this document. The following tables provide a general idea of what the text strings look like and how to interpret their values. Additional information can be found in the references.

The following list provides a summary of the SAS Clinical Standards Toolkit support of ISO 8601:

- Consistent with CDISC SDTM guidelines, the SAS Clinical Standards Toolkit does not support the ISO 8601 **basic** format. This means that the text strings must contain the hyphen delimiter for parts of the dates, and the colon delimiter for parts of the time.
- The SAS Clinical Standards Toolkit does not support some of the rarely used formats allowed by ISO 8601. The week (W) formats for dates, Julian dates, and extended dates (used to denote years greater than 9999) are not supported.
- The SAS Clinical Standards Toolkit requires a SAS hot fix for ISO informats.

  Several enhancements have been made to SAS informats $N8601B. and $N8601E. to enable them to provide even better support of the CDISC usage of ISO 8601. This includes backporting SAS informats for use with SAS 9.1.3. These enhancements are available as a free download as a SAS hot fix. (See **`http://ftp.sas.com/techsup/download/hotfix/hotfix.html`** and the SAS Clinical Standards Toolkit installation instructions for more information.)

This SAS hot fix is required to support ISO 8601-related SAS Clinical Standards Toolkit validation checks. If this hot fix is not installed, SAS 9.1.3 generates SAS errors, indicating that it cannot locate the SAS informats. In SAS 9.2, SAS errors are not generated, but some of the values might not be validated correctly.

SAS provides capabilities for processing ISO 8601 text strings that are far beyond those required by the SAS Clinical Standards Toolkit and CDISC standards.

- The SAS informats $N8601B. and $N8601E. convert an ISO 8601 text string to a special string called an ISO 8601 entity.

  The ISO 8601 entity is a complex binary value that is stored as a hexadecimal value in a SAS string variable.

  The ISO 8601 entity string is useful for reporting in the ISO 8601 format because it prevents the loss of valuable information from the input ISO 8601 text string.
- The ISO 8601 entity value should not be confused with the traditional numeric SAS date, time, or datetime value.
- The ISO 8601 entity should not be used in calculations or comparisons.
- The CALL IS8601_CONVERT routine can be used to generate traditional numeric SAS dates, times, and datetime values from an ISO 8601 string.
- For additional information, see the online SAS documentation.

The following table provides an overview of some commonly found values. It groups the comments based on the ISO 8601 string type.

Table 6.23 *Example ISO 8601 Values*

String	Interpretation	Comments
Dates and Times: Template		
YYYY-MM-DDTHH:MM:SS	A specific date and time	YYYY: Four-digit year. MM: # of month (01-12). DD: # of day of month (01-31). T: What follows is a time in a 24-hour clock. HH: Hours. MM: Minutes. SS: Seconds.
Dates and Times: Full Datetime Examples		
2009-03-25	March 25, 2009	Year must have four digits. Month, day, hour, minute, and second each must have two digits. Single-digit values must be preceded by a leading zero.

String	Interpretation	Comments
2009-03-25T22:29:30	March 25, 2009 10:29 and 30 seconds p.m.	T is always required before a time. Times must always be in military time (for example, 24-hour clock). Midnight must be written as 00:00. 24:00 is not valid. The individual parts of a date value must be separated by a hyphen (-). The individual parts of a time value must be separated by a colon (:).
2009-03-25T22:29:30.333+05:00	March 25, 2009 10:29 and 30.333 seconds p.m. in the time zone GMT + 5 hours	If provided, the time zone must be in HH:MM format. It cannot be truncated or a partial value. Some values in ISO 8601 formats can have decimal places. Most commonly, this is seen in seconds. The decimal place can be denoted as either a period (.) or a comma (,). When a time zone is provided, it must be accompanied by a complete date. The date cannot be truncated or a partial value. This is necessary because the 24 global time zones force the date to be considered as part of the time.
2009-03-25T22:29Z	March 25, 2009 10:29 p.m. Zulu time	Z can be used to substitute for times in GMT (or Zulu) time.
Dates and Times: Partial Datetime Examples (One or more components of the date or time are not known. Partial values are denoted by a single -, no matter how many digits are absent. Partial values can be expressed by truncating the missing parts.)		
-----T22:29	The time 10:29 p.m. No value for the date is provided.	A time value must always be prefixed by a date value. In this example, the date value is completely missing, which would be appropriate for time-only fields.
2009	Year 2009.	Trailing values can be truncated when the values are missing.

String	Interpretation	Comments
2009---25	The 25th day of an unknown month in the year 2009. The month is missing.	If a missing value is embedded in the string, then it must always be denoted by a hyphen (-).
--03-25	The 25th day of March in an unknown year.	Missing year.
--03--T-:15	The 15th minute of an unknown hour of an unknown day of the third month of an unknown year.	Missing year, day, and hour.
2009-03	Month of March 2009.	Trailing partial values can be omitted (truncated). If time is omitted, then T must also be omitted.
2009-03--T12	The 12th hour of an unknown day in March 2009.	Missing day of month.
Durations: Template		
PnYnMnDTnHnMnS	Duration	A span of time where n is the number of the unit that follows the unit. P: indicates that the value is a duration (period) nY: n elapsed years nM: n elapsed months nD: n elapsed days T: the elapsed time in hours, minutes, and seconds nH: n elapsed hours nM: n elapsed minutes nS: n elapsed seconds Typically, only the units with actual values are given. For example, P0Y1M would be P1M.
Durations: Examples		
P1D	The span of one day.	Durations always start with P for a period of time. Units of time that are not known are usually omitted. If time is omitted, then T must also be omitted.

String	Interpretation	Comments
P0000-00-01	The span of zero years + zero months + one day.	Durations can be expressed in an alternative format. When expressed, the length of time is stored in the same format as date and time, but preceded by a P. Instead of expressing a specific point in time, it expresses a period of time.
P1Y2M3DT4H5M6S	The span of 1 year, 2 months, 3 days, 4 hours, 5 minutes, and 6 seconds.	The units must be in the correct order. The T is required for all time values, but it should not be specified if no time value is given.
Intervals: Template		
PnYnMnDTnHnMnS/ YYYY-MM-DDTHH:MM:SS or YYYY-MM-DDTHH:MM:SS/ PnYnMnDTnHnMnS or YYYY-MM-DDTHH:MM:SS/ PnYnMnDTnHnMnS or YYYY-MM-DDTHH:MM:SS/YYYY-MM-DDTHH:MM:SS	Intervals	This is a duration that is anchored to a specific point in time.
Intervals: Examples		
2009-03-25T22:29/P1Y	The span of one year starting on March 25, 2009 at 10:29 p.m.	Intervals can express the period of time that starts at a given point in time. The end time is implied.
P0001-00-00/2009-03-25T22:29	The span of one year ending on March 25, 2009 at 10:29 p.m.	Intervals can express the period of time that ends at a given point in time. The start time is implied.

String	Interpretation	Comments
2008-03-25/2009-03-25	The span of time between March 25, 2008 and March 25, 2009, which happens to be one year.	Intervals can express the period of time that starts at a given point in time and ends at a given point in time. The duration value itself is implied.

Table 6.24 SAS ISO 8601 References

Topic	Link
SAS 9.2 Language Reference: Dictionary	`http://support.sas.com/documentation/cdl/en/lrdict/63026/HTML/default/viewer.htm#a002295669.htm`
Working with Dates and Times Using the ISO 8601 Basic and Extended Notations	`http://support.sas.com/documentation/cdl/en/lrdict/63026/HTML/default/viewer.htm#a003169814.htm`
CALL IS8601_CONVERT Routine	`http://support.sas.com/documentation/cdl/en/lrdict/63026/HTML/default/viewer.htm#a003156604.htm`
$N8601Bw.d Informat	`http://support.sas.com/documentation/cdl/en/lrdict/63026/HTML/default/viewer.htm#a003170563.htm`
$N8601Ew.d Informat	`http://support.sas.com/documentation/cdl/en/lrdict/63026/HTML/default/viewer.htm#a003170574.htm`
Reading Dates and Times Using the ISO 860 Basic and Extended Notations	`http://support.sas.com/documentation/cdl/en/lrdict/63026/HTML/default/viewer.htm#a003169817.htm`
SAS Hot Fixes	`http://ftp.sas.com/techsup/download/hotfix/hotfix.html`

Special Topic: Debugging a Validation Process

The SAS Clinical Standards Toolkit provides two properties or global macro variables for debugging problems occurring with all processes. These are _cstDebug and _cstDebugOptions.

The _cstDebug global macro variable toggles debugging options on and off. Many SAS Clinical Standards Toolkit code modules have conditional branching such as:

```
%if &_cstDebug %then
%do;
    /* perform some action */
end;
```

If debugging is toggled on (_cstDebug=1), several things can happen.

- If code is in place, like the following excerpt from the sample driver module (validate_data.sas) documented in "Running a Validation Process" on page 111, additional messaging to the SAS log can be enabled.

  ```
  data _null_;
    _cstDebug = input(symget('_cstDebug'),8.);
    if _cstDebug then call execute("options source source2
    &_cstDebugOptions;");
    else call execute("options source source2 nomlogic nomprint
          nosymbolgen;");
  run;
  ```

  By default, the &_cstDebugOptions global macro variable is set to:

  mprint mlogic symbolgen mautolocdisplay

  These SAS global macro variables generate a lot of information, and they quickly fill the SAS log when running interactively. You might consider running the process in batch or use PROC PRINTTO to redirect the SAS log to a file.

- Many Work files created during the process are not deleted. They remain available in the Work library to help with debugging.

Each SAS Clinical Standards Toolkit process consists of two primary tasks. The first task is to use set up routines to establish the SAS Clinical Standards Toolkit environment. The second task is to perform some primary SAS Clinical Standards Toolkit action. Debugging focus is different for these two tasks.

In SAS Clinical Standards Toolkit setup, errors most often occur because of problems with the SASReferences data set. The following table lists some common errors with possible causes: For recommendations on configuring the SASReferences data set appropriately, see "Building a SASReferences File" on page 69.

Table 6.25 Debugging Process Setup Errors

Error	Location Where Error Is Reported	Possible Cause and Corrective Action
Expected libraries are not allocated.	SAS Log, Libraries window, SAS DMS	(1) An invalid physical name for the libref has been used. Is the libref a valid SAS name? A SAS name can contain one to 32 characters. It must start with a letter or an underscore (_), not a number. Subsequent characters must be letters, numbers, or underscores. Blanks cannot appear in SAS names. Is the libref a reserved SAS libref name? You should not use Work, Sasuser, or Sashelp. (2) The path specified for the libref is invalid; it points to a nonexistent directory. Check the path in your SASReferences data set.
Error: SAS system library WORK cannot be reassigned.	SAS Log	Work is being used as a sasref value with or without a path being designated. A similar error occurs if Sasuser or Sashelp is used.
WARNING: One or more libraries specified in the concatenated library CSTTMP do not exist.	SAS Log	One of the paths specified for a libref is invalid; it points to a nonexistent directory.

Error	**Location Where Error Is Reported**	**Possible Cause and Corrective Action**
Warning: Process ending prematurely for CST0090-there were problems with the sasreferences data set.	SAS Log	There is a problem with the SASReferences data set being used. Check for these potential problems: The SASReferences data set does not exist. The SASReferences data set exists but it is empty. The structure of the SASReferences data set is incorrect. For example, it might have an extra column that is not required or an expected column that is missing. A column type might be incorrect. For example, the Order column might be character instead of numeric. An invalid TYPE or SUBTYPE or combination is used in the SASReferences data set. Valid TYPE and SUBTYPE values are provided in the Standardlookup data set found in ***`<global standards library directory>`*****`/standards/cst-framework-1.3/control`**. A TYPE value is missing. A SASREF value is missing or invalid. A REFTYPE value is missing or is not equal to libref or fileref (case insensitive).
Error: Physical file does not exist.	SAS Log	(1) The SASReferences data set references a file that does not exist. (2) The filename is not a valid SAS name.

Error	Location Where Error Is Reported	Possible Cause and Corrective Action
WARNING: Apparent invocation of macro SDTM_VALIDATE not resolved.	SAS Log	(1) The macro is misnamed or has not been added to the expected autocall library. Does the macros folder for this standard exist in the cstGlobalLibrary, in the !sasroot hierarchy, or in some correctly designated custom location? (2) The expected autocall path was not created correctly in the call to %cstutil_allocatesasreferences. Check that the SASReferences data set contains a type=autocall record, defined as a fileref, and points to the correct folder location. Check for an error occurring earlier in the SAS log suggesting that %cstutil_allocatesasreferences failed before setting the autocall path.

If the task to perform the primary SAS Clinical Standards Toolkit action begins (that is, the standard-specific validation macro, such as %sdtm_validate or %crtdds_validate, is found and begins processing), then setup has completed successfully, and remaining process failures are likely because of problems with the various validation components.

Most errors that halt a validation process are reported in the Results data set. As a general rule, the following Results data set fields signal process failures and provide information about the cause of the failure:

- the Process status field (_cst_rc), when the value is set to a nonzero value
- the Problem detected field (resultflag), when the value is set to -1
- the Source Data field (srcdata) identifies the macro reporting the problem
- the Resolved Message text field (message) provides a problem cause
- the Basis for Result field (resultdetails) can provide additional information pertinent to the problem

Depending on the severity of the problem and when it occurs, the Results data set might not be saved to the persisted location if that location was requested using a type=results record in the SASReferences data set. In this case, the Results data set defined with the &_cstResultsDS global macro variable might be referenced for the previous information. By default, &_cstResultsDS is set to work._cstresults.

Generally, the SAS Clinical Standards Toolkit does not halt the validation process when an error is detected in a specific check. The error is noted in the Results data set, the resultflag value for that check is set to -1, _cst_rc is set to 0, and processing continues with the next check. A validation process is most likely to be halted (by setting _cst_rc to 1) when there is a significant metadata error that suggests subsequent checks would likely fail to run.

The following table lists some common causes for premature process failure or the failure of specific checks to run:

Table 6.26 *Debugging Validation Process Errors*

Error	Resultid in Results Data Set	Possible Cause/Corrective Action
No tables evaluated-check validation control data set.	CST0002	No tables interpreted from the tablescope value could be found in the work._csttablemetadata data set.
	CST0003	This error usually indicates that a specific source column or data set could not be found. The code loops through a set of domains or columns built from the source metadata data sets. This error might result when the source metadata does not accurately reflect the source data.
No columns evaluated-check Validation Control specification.	CST0004	No columns interpreted from the columnscope value could be found in the work._cstcolumnmetadata data set. The SAS Clinical Standards Toolkit looks at the union of both tablescope and columnscope to build work._cstcolumnmetadata. The specified column might exist in a domain, but not in any column specified in a tablescope domain.
Lookup to SASreferences control data set failed.	CST0006	The SAS Clinical Standards Toolkit code has a call to the cstutil_getsasreference utility macro for a type or type and subtype combination that cannot be found in the SASReferences data set. This indicates that SASReferences has been incompletely defined for the SAS Clinical Standards Toolkit validation process.
Validation control parsing of tablescope/column results in inconsistent sublist lengths.	CST0023	This check involves a comparison of tables or columns, as indicated by multiple sets of brackets in tablescope or columnscope. Each set of brackets constitutes a sublist. However, the number of items in the specified sublist is inconsistent or unexpected by the check macro. Options typically include a more accurate specification of sublist items, either using explicit table or column names or more restrictive tablescope syntax (that is, removing the domain causing the inconsistency using minus sign (-) syntax, such as _ALL_-DM).

Error	Resultid in Results Data Set	Possible Cause/Corrective Action
One or more check metadata column values is invalid.	CST0026	A value in the Validation Control data set for the check being run is invalid in the context of the specific check macro. Examples include conditions that are required by the check macro but are not found, such as no code logic found, an unexpected usesourcemetadata value, or no lookuptype or lookupsource for valid value assessments.
Code failed due to SAS error-see log.	CST0050	A SAS DATA step or SAS procedure failed and the cause is reported in the SAS log. This most commonly occurs because of missing data sets, missing columns, incorrectly sorted data sets, and unexpected macro variable values.
<Message lookup failed to find matching record>	<varies>	The check macro code generates a resultid value that does not find a match in the Messages data set. Either the wrong resultid has been specified, or the standard-specific Messages data set has not been updated to include the resultid.

Other Debugging Tips

- Review available Work files for information about the errors (for example, _cstresults, _csttablemetadata, and _cstcolumnmetadata). These files might remain in the Work directory after a process by default. Toggling the _cstDebug global macro variable to 1 forces the Work files to remain after the process.
- When debugging, avoid setting the parameter flags in cstutil_cleanupcstsession to 1 (if that cleanup macro is called).

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

- Use work._cstcolumnmetadata and work._csttablemetadata to resolve missing domain and column issues. These data sets can also be used to resolve sublist length differences for checks using sublist syntax [] in tablescope and columnscope.
- Use the resultid code (for example, CST0003) in the Results data set to search the check macro code module used for a specific check for information about the error. The name of the macro code module is set in the Validation Control **codesource** field.

Special Topic: Validation Customization

Overview

One of the significant benefits of the SAS Clinical Standards Toolkit is that users can customize the solution to meet their needs. From a validation perspective, this includes:

- modifying an existing standard or defining a new reference standard
- using any set of source data and metadata
- modifying the SAS validation checks for supported standards
- adding new validation checks for supported standards
- modifying existing validation check macros or adding new macros
- modifying SAS Clinical Standards Toolkit messaging, including internationalization

Each of these customizations is described in the following case studies.

Case Study 1: Modifying an Existing Standard or Defining a New Reference Standard

Source data and metadata are validated in the SAS Clinical Standards Toolkit against a reference standard. For CDISC standards, the SAS Clinical Standards Toolkit provides a SAS interpretation of the supported CDISC standards. Because CDISC standards are guidelines, they are open to interpretation and customer-specific implementations. Not all clinical studies have all CDISC-defined standard domains, and most clinical studies have additional domains reflecting the focus of the clinical study. In addition, CDISC SDTM domain classes (findings, events, and interventions) enable the inclusion and exclusion of most columns, depending on the clinical data points collected in the study. CDISC guidelines generally do not specify column lengths.

Each of these factors suggests that the SAS Clinical Standards Toolkit CDISC reference standards will be modified or replaced with customer-derived standards. The SAS Clinical Standards Toolkit offers the option of building a reference standard to encompass domain and column customizations. Or, you can customize check macros and check logic to perform specific compliance assessments to a standard. For example, in CDISC SDTM, it is not uncommon to build multiple supplemental qualifier domains (for example, SUPPAE) associated with a core reference domain (for example, AE). It is at the customer's discretion whether the reference standard is modified to include each unique supplemental qualifier domain, or to use existing SAS Clinical Standards Toolkit validation check macros with unique code logic or custom check macros to validate the custom domains. These latter options are discussed in the following case studies.

It is likely that customers will derive multiple reference standards. From a SAS Clinical Standards Toolkit validation perspective, the only relevant reference standard is the one defined in the SASReferences data set (as type=referencemetadata).

For information about registering a new standard in the SAS Clinical Standards Toolkit, see “Registering a New Version of a Standard” on page 16.

Case Study 2: Using Any Set of Source Data and Metadata

From a SAS Clinical Standards Toolkit perspective, a source study is defined by the study domains, the study metadata represented in the source_tables and source_columns data sets, and anything that might be unique to a specific study, including controlled terminologies, properties, validation checks, and associated messages.

One key SAS Clinical Standards Toolkit requirement is that source study elements should be kept in synchronization. Another key requirement is that all relevant source study elements should be accurately represented in a SASReferences data set. The synchronization of study elements is a task that is often performed outside the SAS Clinical Standards Toolkit. The study data libraries must contain the domains of interest, the study metadata must provide the complete set of table-level and column-level metadata necessary to describe the source data, and any format catalogs and coding dictionaries supporting the study must be available.

Best Practice: If a standard folder hierarchy is adopted for source studies, such as in the SAS Clinical Standards Toolkit CDISC SDTM 3.1.2 sample study in SAS 9.2 (`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata`), using generic SASReferences files that use &studyRootPath in the path field might facilitate referencing new source studies.

Case Study 3: Modifying the SAS Validation Checks for Supported Standards

This case study addresses adding multiple instances of existing checks. The most common ways to modify SAS validation checks include:

- Altering the scope of the domains and columns to be validated. This change should not be required frequently. Many checks are defined to be run against specific domains or columns, against specific classes of domains (for example, CDISC SDTM findings, events, or interventions), or against all available domains or columns. Changes are likely to involve alterations to the Validation Control **tablescope** or **columnscope** fields.
- Changing the Validation Control **codelogic** field to alter the logic used to identify error conditions. This might be a necessary change if a check needs to be generalized to accommodate new domains or columns. Or, customer conventions might differ from those in the SAS Clinical Standards Toolkit checks.
- If customer code changes are sufficiently significant, then it might be better to create a new validation check macro. (See "Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros" on page 161.) If a new validation check macro is required, then the Validation Control **codesource** field needs to be modified to contain the name of the new check macro.
- The Validation Control **uniqueid** field provides a way to uniquely identify a specific validation check for reference. Any substantive change to any Validation Control data set check field normally leads to a new uniqueid. For information about the structure of uniqueid, see Table 6.3 on page 90.
- The Validation Control **checkstatus** field provides an easy way to identify selected checks with a user-defined status (for example, draft, deprecated, or not available for a given study). The SAS Clinical Standards Toolkit does not reference this field within any validation check macro.
- The Validation Control **lookupsource** field can be changed to reference a different SAS format or lookup data set (for example, a new version of MedDRA). In the latter case, a change to the **pathname**, **memname**, or both fields in the SASReferences data set might be a more appropriate action.

Case Study 4: Adding New Validation Checks for Supported Standards

To add a new validation check, consider the following checklist:

- Check metadata must conform to the Validation Master structure. (For more information, see Chapter 2, "Framework," on page 5.)
- Certain Validation Master fields accept any user-defined value (for example, checksource, sourceid, checktype, standardref, and checkstatus). These fields are not

referenced by the validation check macros. The remaining fields are used in the validation check macros, so users must abide by SAS Clinical Standards Toolkit conventions. These conventions are described in Chapter 2, "Framework," on page 5.

- A new check should be added to the (run-time) Validation Control data set for testing. After testing, it can be promoted to the Validation Master data set to be available to applications and processes. These requirements follow a typical development process.
- For each new validation check, a matching message is required. This is the message that you want written to the Results data set when an error condition is detected. For details, see "Messages" on page 101.

Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros

The SAS Clinical Standards Toolkit provides 14 validation check macros. These macros offer a variety of code samples that function in the general SAS Clinical Standards Toolkit framework. These 14 macros support CDISC SDTM validation; CDISC CRT-DDS 1.0 uses six of these macros. (For full descriptions of these macros, see "Special Topic: Validation Check Macros" on page 136.)

Some validation scenarios might require modifications to the SAS Clinical Standards Toolkit check macros or the derivations of new macros. If so, the following guidelines should be followed. These guidelines facilitate the use of these macros in the general SAS Clinical Standards Toolkit framework and in the specific SAS Clinical Standards Toolkit validation framework.

- Follow the current naming convention or adopt a consistent naming convention that conforms to SAS naming conventions.
- Use the current autocall library or use a customized autocall library that has been defined in the SASReferences data set (type=autocall).
- Conform to the basic check macro workflow. This workflow is described in "Special Topic: Validation Check Macros" on page 136.
- Ensure that the macro correctly accepts and interprets the metadata provided as input from the Validation Control data set. If the new macro fails to do so, then it can be hardcoded to provide any specific functionality that is desired.
- Ensure that the macro writes appropriate output to the Results and Metrics data sets.

Case Study 6: Modifying SAS Clinical Standards Toolkit Messaging, Including Internationalization

This case study considers the following three issues related to the support of SAS Clinical Standards Toolkit messaging:

1. Maintain the relationship between SAS Clinical Standards Toolkit standard-specific messages and standard-specific validation checks.
2. Maintain the relationship between messages and validation check macro code.

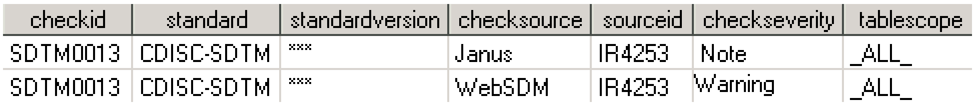
   (Deviations are acceptable to the extent that missing parameters have suitable defaults.)
3. Internationalize messages.

A SAS Clinical Standards Toolkit message is created for each distinct combination of the Validation Master **standard** and **checksource** fields. This allows the SAS Clinical Standards Toolkit to support checksource-specific messaging and severity. A unique SAS

Clinical Standards Toolkit message is required for each value of the Validation Master **standardversion** field if that value is not the wildcard ***.

Consider the following CDISC SDTM 3.1.1 Validation Master record excerpts:

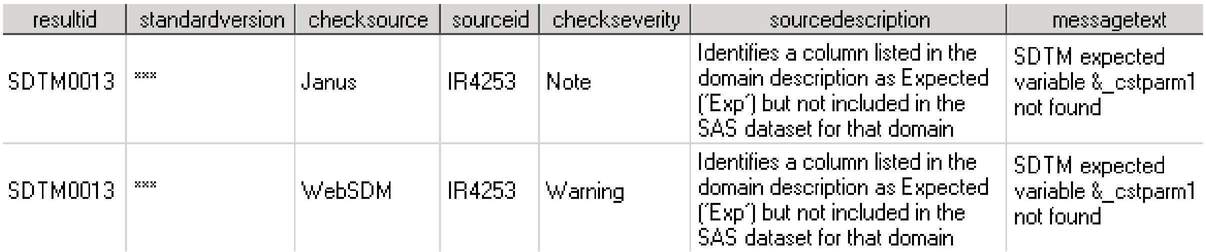
Display 6.15 Validation Master Data Set Excerpt for Check SDTM0013



checkid	standard	standardversion	checksource	sourceid	checkseverity	tablescope
SDTM0013	CDISC-SDTM	***	Janus	IR4253	Note	_ALL_
SDTM0013	CDISC-SDTM	***	WebSDM	IR4253	Warning	_ALL_

The SAS Clinical Standards Toolkit representation of the SDTM0013 check in the Messages data set is:

Display 6.16 Messages Data Set Excerpt for Check SDTM0013

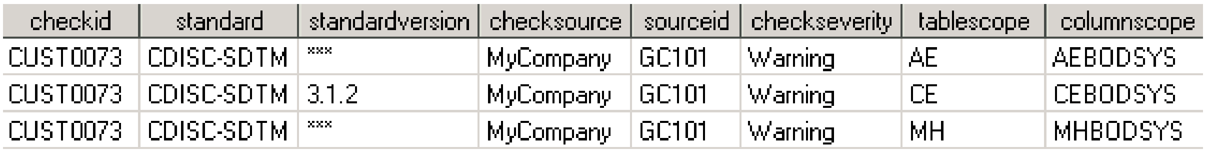


resultid	standardversion	checksource	sourceid	checkseverity	sourcedescription	messagetext
SDTM0013	***	Janus	IR4253	Note	Identifies a column listed in the domain description as Expected ('Exp') but not included in the SAS dataset for that domain	SDTM expected variable &_cstparm1 not found
SDTM0013	***	WebSDM	IR4253	Warning	Identifies a column listed in the domain description as Expected ('Exp') but not included in the SAS dataset for that domain	SDTM expected variable &_cstparm1 not found

The Messages data set contains two records because there are two distinct checksource values for Validation Master checkid SDTM0013.

Consider the following CDISC SDTM Validation Master record excerpts:

Display 6.17 Validation Master Data Set Excerpt for Check CUST0073

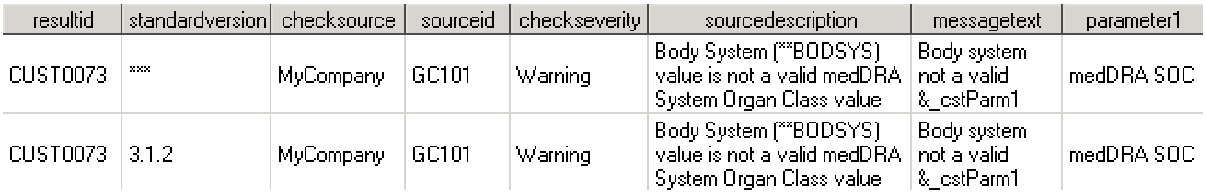


checkid	standard	standardversion	checksource	sourceid	checkseverity	tablescope	columnscope
CUST0073	CDISC-SDTM	***	MyCompany	GC101	Warning	AE	AEBODSYS
CUST0073	CDISC-SDTM	3.1.2	MyCompany	GC101	Warning	CE	CEBODSYS
CUST0073	CDISC-SDTM	***	MyCompany	GC101	Warning	MH	MHBODSYS

Three separate invocations of CUST0073 are represented. Each record points to a different domain (tablescope). This example assumes that the CDISC SDTM 3.1.2 standard has been registered. The first and third records (AE and MH domains) indicate that this specific implementation of the check is applicable to all versions of CDISC SDTM. However, the second record is applicable to only CDISC SDTM 3.1.2 (because CE is a new domain in SDTM 3.1.2).

Only two Messages data set records are required:

Display 6.18 Messages Data Set Excerpt for Check CUST0073



resultid	standardversion	checksource	sourceid	checkseverity	sourcedescription	messagetext	parameter1
CUST0073	***	MyCompany	GC101	Warning	Body System (**BODSYS) value is not a valid medDRA System Organ Class value	Body system not a valid &_cstParm1	medDRA SOC
CUST0073	3.1.2	MyCompany	GC101	Warning	Body System (**BODSYS) value is not a valid medDRA System Organ Class value	Body system not a valid &_cstParm1	medDRA SOC

It is the distinct combinations of the Validation Master **checkid**, **standardversion**, and **checksource** fields that control the associated Messages data set records.

It is important to maintain the relationship between messages and validation check macro code. If the validation check macro code references an unknown resultid, the text **`<Message lookup failed to find matching record>`** is written to the Results data set.

The CUST0073 check defines a substitution parameter (&_cstParm1). (The SAS Clinical Standards Toolkit code assumes that message substitution parameters begin with the string &_cst.) For the calling validation check macro to support parameters when writing output to the Results data set, the parameters that are passed should be syntactically consistent with the **messagetext** field in the Messages data set.

Building the message record to use a default value (as specified in the **parameter1** field) solves the problem when the calling macro fails to pass a substitution value. Using parameters is optional. Parameters might only be needed if the message is to be used in multiple contexts where substitutions of parameter values help interpret the message.

The SAS Clinical Standards Toolkit supports the internationalization of messages through specifying message file references in the SASReferences data set (type=messages). If referenced message files conform to the structure expected by the SAS Clinical Standards Toolkit, any text, including internationalized text, can be included.

Special Topic: Performance Considerations

Best Practice Recommendations:

- SAS Clinical Standards Toolkit validation should first be run on a subset of source data to identify general process problems, missing or inconsistent process control metadata, and common and perhaps correctable data errors.
- The SAS Clinical Standards Toolkit standard-specific Validation Master data set should be subsetted to remove duplicate checks. For example, CDISC SDTM 3.1.1 Janus checks are generally duplicates of WebSDM checks with occasionally different resultseverity values.
- The _cstDebug option should be toggled off except for when you want to debug specific program errors to avoid exceeding the SAS log-size limitations or to avoid generating large SAS log files.
- A SAS Clinical Standards Toolkit validation process that involves a large number of checks, should be run in batch or using PROC PRINTTO. This is also true for a SAS Clinical Standards Toolkit validation process that is run with the _cstDebug option toggled on. Doing so avoids exceeding the SAS log-size limitations.

Chapter 7
XML-Based Standards

SAS Support of XML-Based Standards

When processing XML-based standards (such as CDISC ODM and CDISC CRT-DDS), the SAS Clinical Standards Toolkit attempts to create a representation in SAS that is based on the standard. This typically includes a combination of metadata data sets, content data sets, and SAS format catalogs. Once the standard is represented in SAS, additional processing in SAS, such as model validation and reporting, is facilitated.

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. SAS Clinical Standards Toolkit 1.3 reads a CDISC ODM 1.3.0 or a CDISC CRT-DDS 1.0 XML file and converts the information into a SAS data set representation of each model. For CDISC CRT-DDS 1.0, this means that 39 data sets (such as ItemDefs) containing 176 columns are derived from the define.xml element and attribute structure. The SAS representation of each standard can be derived in part from other standards (such as CDISC-SDTM) and can include supporting metadata from other sources. SAS Clinical Standards Toolkit 1.3 can also create a CDISC CRT-DDS 1.0 XML file.

Reading XML Files

Overview of Basic Workflow

The following is the basic workflow for reading XML files:

1. Determine the existence of a valid XML file.
2. Use valid XSL style sheets for each target data set (such as ItemDefs.xsl).
3. Use the SAS DATA step component JavaObj to create a standardized intermediate cubeXML file using the XSL style sheets.
4. Read the standardized cubeXML file using the SAS XML LIBNAME engine and XMLMAP processing.

This basic workflow is used by all XML-based standards that are supported by the SAS Clinical Standards Toolkit.

Reading CDISC ODM XML Files: odm_read Macro

The current SAS Clinical Standards Toolkit release supports the reading of portions of an odm.xml file. It supports the translation of only the metadata (<Study>) and clinical data (<ClinicalData>) sections of the file into a SAS representation of the file content.

In order to read an odm.xml file, a specialized macro named odm_read is available in the ODM 1.3.0 standards macro folder. (For SAS 9.2, this folder is located at ***<global standards library directory>*** **/standards/cdisc-odm-1.3.0-1.3/macros**.) This macro is referenced from the create_sasodm_fromxml.sas driver program (described more fully below). There are no input parameters in the call to the odm_read macro. File references and other metadata that are required by the macro are set as global macro variable values. Currently, those global macro variable values are set through the framework initialization properties and the CDISC ODM 1.3.0 initialization properties. Throughout the processing of the odm_read macro, the Results data set contains all framework and ODM 1.3.0 specific messages generated during run time.

Based on file references from the SASReferences data set, odm_read accesses the odm.xml file.

The following is a partial listing of the sample odm.xml file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ODM
  xmlns="http://www.cdisc.org/ns/odm/v1.3"
  FileOID="Study1234"
  ODMVersion="1.3"
  FileType="Snapshot"
  CreationDateTime="2004-07-28T12:34:13-06:00"
  SourceSystem="ss00"
  AsOfDateTime="2004-07-29T12:34:13-06:00"
  Granularity="SingleSite"
  Description="Study to determine existence of ischemic stroke"
  Archival="Yes"
  PriorFileOID="Study-4321"
  Originator="SAS Institute"
  SourceSystemVersion="Version 0.0.0"
  Id="DSSignature123">
<Study OID="1234">
<GlobalVariables>
    <StudyName>1234</StudyName>
    <StudyDescription>1234 Data Definition</StudyDescription>
    <ProtocolName>1234</ProtocolName>
</GlobalVariables>
<BasicDefinitions>
    <MeasurementUnit Name="My Unit" OID="MU_0001">
       <Symbol>
         <TranslatedText xml:lang="enus">Hello there text</TranslatedText>
       </Symbol>
    </MeasurementUnit>
    <MeasurementUnit Name="My Other Unit" OID="MU_0002">
       <Symbol>
          <TranslatedText xml:lang="jpn">Bye there text</TranslatedText>
       </Symbol>
    </MeasurementUnit>
</BasicDefinitions>
<MetaDataVersion OID="CDISC.SDTM.3.1.0"
    Name="Study 1234, Data Definitions"
    Description="Study 1234, Data Definitions">
    <Include StudyOID="1234"
      MetaDataVersionOID="MDV000">
    </Include>
    <Protocol>
      <Description>
```

After the odm_read macro confirms that the odm.xml file exists, a call is made to the SAS DATA step component JavaObj. In SAS 9.1.3, you get a warning in the log that states that JavaObj is experimental. JavaObj processing converts the odm.xml file into the cubeXML file through transformations using XSL files and processes. The cubeXML file is created in the Work library. The name of the cubeXML file is **`_cubnnnn.xml`**, where ***nnnn*** is a randomly generated number. The cubeXML file is accessed using the SAS XML LIBNAME engine and XMLMAP processing. A default XMLMAP file is stored in the sample ODM 1.3.0 study folder hierarchy under **`/referencexml`** as odm.map. The odm.map file is required to process the cubeXML file. If it does not exist, then the odm_read macro attempts to create one using the ODM reference metadata.

The following is a partial listing of the odm.map file.

```
<?xml version="1.0" encoding="windows-1252"?>
<SXLEMAP version="1.2">

<TABLE name="Annotations">
   <TABLE-PATH syntax="XPath">/LIBRARY/Annotations</TABLE-PATH>
   <TABLE-DESCRIPTION>Annotations associated with data</TABLE-DESCRIPTION>

   <COLUMN name="ID">
     <PATH syntax="Xpath">/LIBRARY/Annotations/ID</PATH>
     <TYPE>character</TYPE>
     <DATATYPE>character</DATATYPE>
     <DESCRIPTION>Unique ID for a specific Annotation element</DESCRIPTION>
     <LENGTH>128</LENGTH>
   </COLUMN>
```

```
<COLUMN name="SeqNum">
  <PATH syntax="Xpath">/LIBRARY/Annotations/SeqNum</PATH>
  <TYPE>numeric</TYPE>
  <DATATYPE>numeric</DATATYPE>
  <DESCRIPTION>Uniquely identifies the annotation within its parent
          entity</DESCRIPTION>
  <LENGTH>8</LENGTH>
</COLUMN>
<COLUMN name="Comment">
  <PATH syntax="Xpath">/LIBRARY/Annotations/Comment</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>Free-text (uninterpreted) comment about clinical data</DESCRIPTION>
  <LENGTH>2000</LENGTH>
</COLUMN>
<COLUMN name="SponsorOrSite">
  <PATH syntax="Xpath">/LIBRARY/Annotations/SponsorOrSite</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>Comment source (Sponsor | Site)</DESCRIPTION>
  <LENGTH>2000</LENGTH>
</COLUMN>
<COLUMN name="FlagType">
  <PATH syntax="Xpath">/LIBRARY/Annotations/FlagType</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>Type of flag</DESCRIPTION>
  <LENGTH>2000</LENGTH>
</COLUMN>
<COLUMN name="FlagValue">
```

When the cubeXML is processed, the data sets (such as ItemDefs) that are included in the SAS representation of the CDISC ODM model are derived. The final step for the odm_read macro is the derivation of table and column metadata that describe the data sets in the SAS representation of the odm.xml file. At this point, the odm_read macro is ready to create the source_tables and source_columns data sets. The tables in the source_tables data sets are created and copied to the output library as defined in the SASReferences data set.

Sample Driver Program: create_sasodm_fromxml.sas

Overview

Each primary SAS Clinical Standards Toolkit task, such as reading CDISC ODM XML files, is guided by a sample driver module that is provided by SAS. For reading ODM XML files, this module is create_sasodm_fromxml.sas.

For SAS 9.1.3, this driver program is located at:

**!sasroot/../SASClinicalStandardsToolkitODM130/1.3/
sample/cdisc-odm-1.3.0/programs/create_sasodm_fromxml.sas**

For SAS 9.2, the driver program is located at:

**!sasroot/../../SASClinicalStandardsToolkitODM130/1.3/
sample/cdisc-odm-1.3.0/programs/create_sasodm_fromxml.sas**

The value for **!sasroot** is the location of your SAS installation directory.

The SASReferences Data Set

As a part of each SAS Clinical Standards Toolkit process setup, a valid SASReferences data set is required. It references the input files that are needed (such as the odm.xml file), the librefs and filenames to use, and the names and locations of data sets to be created by the process. It can be modified to point to study-specific files. For an explanation of the SASReferences data set, see Chapter 5, “SASReferences File,” on page 69.

In the SASReferences data set, there are two input file references and four output references that are key to the successful completion of the driver program. The following table lists these files and data sets, and they are discussed in separate sections. In the sample create_sasodm_fromxml.sas driver module, the following values are set for &studyRootPath and &studyOutputPath and are specific to a SAS release.

SAS 9.1.3

```
&studyRootPath=!sasroot/../
SASClinicalStandardsToolkitODM130/1.3/sample/cdisc-odm-1.3.0
```

```
&studyOutputPath=!sasroot/../
SASClinicalStandardsToolkitODM130/1.3/sample/cdisc-odm-1.3.0
```

SAS 9.2

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitODM130/1.3/sample/cdisc-odm-1.3.0
```

```
&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitODM130/1.3/sample/cdisc-odm-1.3.0
```

Table 7.1 Key Components of the SASReferences Data Set

Input or Output	Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	externalxml	odmxml	fileref	&studyRootPath/ sourcexml	odm.xml
Input	referencexml	odmmap	fileref	&studyRootPath/ referencexml	odm.map
Output	sourcedata	srcdata	LIBNAME	&studyOutputPath/data	*.*
Output	sourcemetadata	srcmeta	LIBNAME	&studyOutputPath/metadata	Source_ tables.sas7bdat
Output	sourcemetadata	srcmeta	LIBNAME	&studyOutputPath/metadata	Source_ columns.sas7bdat
Output	results	results	LIBNAME	&studyOutputPath/results	Read_ results.sas7bdat

Process Inputs

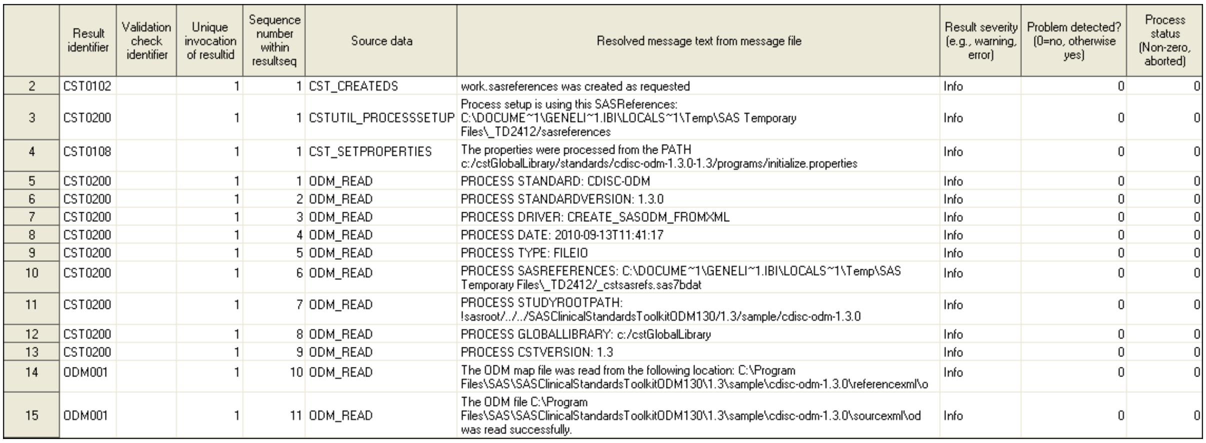
The metadata type externalxml refers to the odm.xml file that is being read. The filename odmxml is defined in the SASReferences data set. This filename is used in the submitted SAS code when referring to the ODM file.

The metadata type referencexml refers to the SAS map file that is used to generate the SAS data sets that represent the ODM file metadata and content. The filename odmmap is defined in the SASReferences data set. This filename is used in the submitted SAS code when referring to the SAS map file. If a path and filename for the map file is not specified, then a temporary map file is created as part of the odm_read processing.

Process Outputs

When the driver program finishes running, the read_results.sas7bdat data set is created in the Results library. This data set contains informational, warning, and any error messages that were generated by the submitted driver program. The following display shows an example of the contents of a Results data set that was built while reading the sample odm.xml file that was provided by SAS.

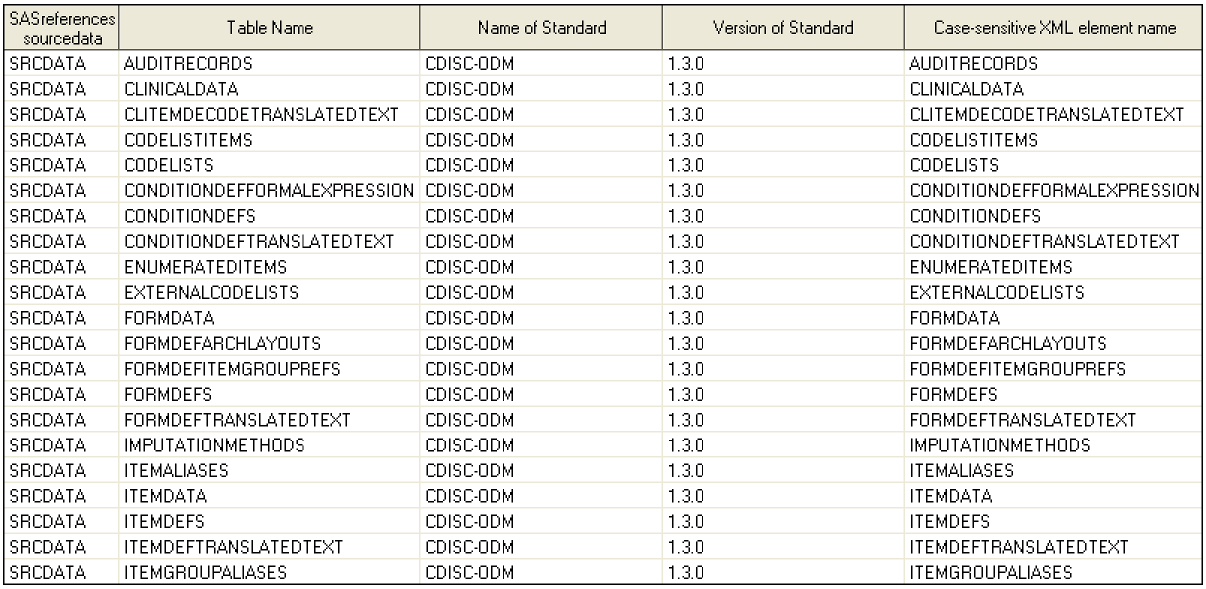
Display 7.1 *Example of a Partial Results Data Set Created by the create_sasodm_fromxml.sas Driver*



	Result identifier	Validation check identifier	Unique invocation of resultid	Sequence number within resultseq	Source data	Resolved message text from message file	Result severity (e.g., warning, error)	Problem detected? (0=no, otherwise yes)	Process status (Non-zero, aborted)
2	CST0102		1	1	CST_CREATEDS	work.sasreferences was created as requested	Info	0	0
3	CST0200		1	1	CSTUTIL_PROCESSSETUP	Process setup is using this SASReferences: C:\DOCUME~1\GENELI~1.IBI\LOCALS~1\Temp\SAS Temporary Files_TD2412/sasreferences	Info	0	0
4	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-odm-1.3.0-1.3/programs/initialize.properties	Info	0	0
5	CST0200		1	1	ODM_READ	PROCESS STANDARD: CDISC-ODM	Info	0	0
6	CST0200		1	2	ODM_READ	PROCESS STANDARDVERSION: 1.3.0	Info	0	0
7	CST0200		1	3	ODM_READ	PROCESS DRIVER: CREATE_SASODM_FROMXML	Info	0	0
8	CST0200		1	4	ODM_READ	PROCESS DATE: 2010-09-13T11:41:17	Info	0	0
9	CST0200		1	5	ODM_READ	PROCESS TYPE: FILEIO	Info	0	0
10	CST0200		1	6	ODM_READ	PROCESS SASREFERENCES: C:\DOCUME~1\GENELI~1.IBI\LOCALS~1\Temp\SAS Temporary Files_TD2412/_cstsasrefs.sas7bdat	Info	0	0
11	CST0200		1	7	ODM_READ	PROCESS STUDYROOTPATH: !sasroot/../../SASClinicalStandardsToolkitODM130/1.3/sample/cdisc-odm-1.3.0	Info	0	0
12	CST0200		1	8	ODM_READ	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info	0	0
13	CST0200		1	9	ODM_READ	PROCESS CSTVERSION: 1.3	Info	0	0
14	ODM001		1	10	ODM_READ	The ODM map file was read from the following location: C:\Program Files\SAS\SASClinicalStandardsToolkitODM130\1.3\sample\cdisc-odm-1.3.0\referencexml\o	Info	0	0
15	ODM001		1	11	ODM_READ	The ODM file C:\Program Files\SAS\SASClinicalStandardsToolkitODM130\1.3\sample\cdisc-odm-1.3.0\sourcexml\od was read successfully.	Info	0	0

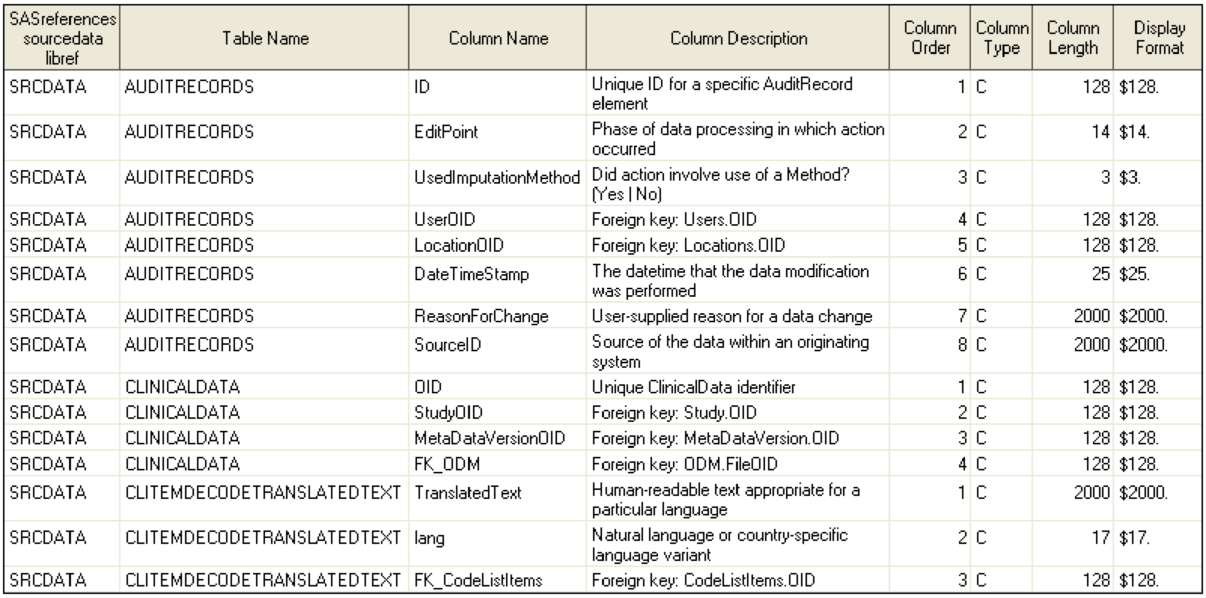
The odm_read macro creates the source_tables and source_columns data sets in the Srcmeta library. These data sets contain the table and column metadata for each of the SAS data sets that is derived from the odm.xml file.

Display 7.2 *Example of Partial Source_Tables Data Set Derived During odm_read*



SASreferences sourcedata	Table Name	Name of Standard	Version of Standard	Case-sensitive XML element name
SRCDATA	AUDITRECORDS	CDISC-ODM	1.3.0	AUDITRECORDS
SRCDATA	CLINICALDATA	CDISC-ODM	1.3.0	CLINICALDATA
SRCDATA	CLITEMDECODETRANSLATEDTEXT	CDISC-ODM	1.3.0	CLITEMDECODETRANSLATEDTEXT
SRCDATA	CODELISTITEMS	CDISC-ODM	1.3.0	CODELISTITEMS
SRCDATA	CODELISTS	CDISC-ODM	1.3.0	CODELISTS
SRCDATA	CONDITIONDEFFORMALEXPRESSION	CDISC-ODM	1.3.0	CONDITIONDEFFORMALEXPRESSION
SRCDATA	CONDITIONDEFS	CDISC-ODM	1.3.0	CONDITIONDEFS
SRCDATA	CONDITIONDEFTRANSLATEDTEXT	CDISC-ODM	1.3.0	CONDITIONDEFTRANSLATEDTEXT
SRCDATA	ENUMERATEDITEMS	CDISC-ODM	1.3.0	ENUMERATEDITEMS
SRCDATA	EXTERNALCODELISTS	CDISC-ODM	1.3.0	EXTERNALCODELISTS
SRCDATA	FORMDATA	CDISC-ODM	1.3.0	FORMDATA
SRCDATA	FORMDEFARCHLAYOUTS	CDISC-ODM	1.3.0	FORMDEFARCHLAYOUTS
SRCDATA	FORMDEFITEMGROUPREFS	CDISC-ODM	1.3.0	FORMDEFITEMGROUPREFS
SRCDATA	FORMDEFS	CDISC-ODM	1.3.0	FORMDEFS
SRCDATA	FORMDEFTRANSLATEDTEXT	CDISC-ODM	1.3.0	FORMDEFTRANSLATEDTEXT
SRCDATA	IMPUTATIONMETHODS	CDISC-ODM	1.3.0	IMPUTATIONMETHODS
SRCDATA	ITEMALIASES	CDISC-ODM	1.3.0	ITEMALIASES
SRCDATA	ITEMDATA	CDISC-ODM	1.3.0	ITEMDATA
SRCDATA	ITEMDEFS	CDISC-ODM	1.3.0	ITEMDEFS
SRCDATA	ITEMDEFTRANSLATEDTEXT	CDISC-ODM	1.3.0	ITEMDEFTRANSLATEDTEXT
SRCDATA	ITEMGROUPALIASES	CDISC-ODM	1.3.0	ITEMGROUPALIASES

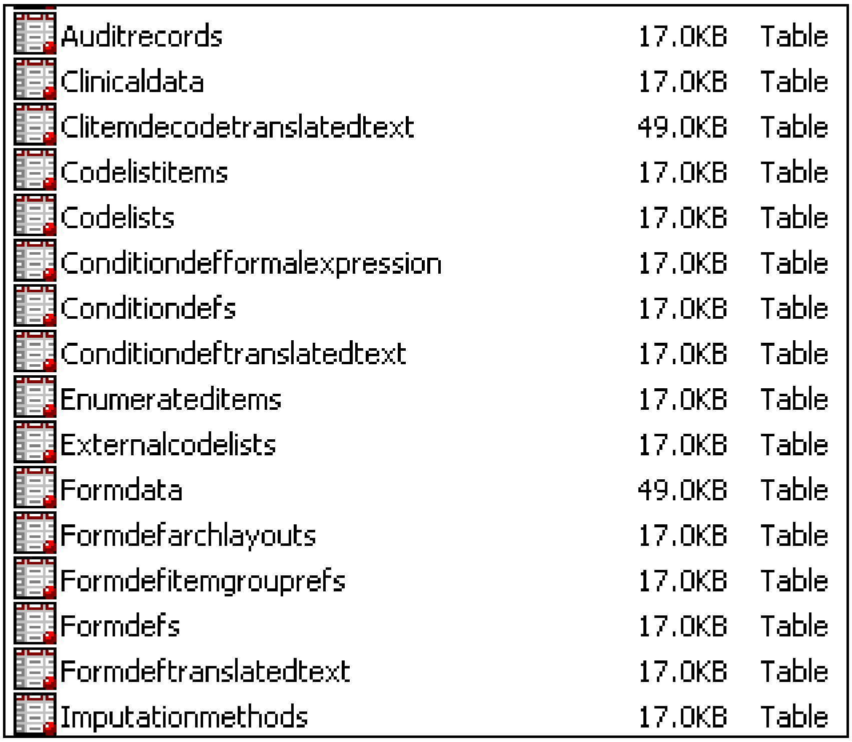
Display 7.3 Example of Partial Source_Columns Data Set Derived During odm_read



SASreferences sourcedata libref	Table Name	Column Name	Column Description	Column Order	Column Type	Column Length	Display Format
SRCDATA	AUDITRECORDS	ID	Unique ID for a specific AuditRecord element	1	C	128	$128.
SRCDATA	AUDITRECORDS	EditPoint	Phase of data processing in which action occurred	2	C	14	$14.
SRCDATA	AUDITRECORDS	UsedImputationMethod	Did action involve use of a Method? (Yes | No)	3	C	3	$3.
SRCDATA	AUDITRECORDS	UserOID	Foreign key: Users.OID	4	C	128	$128.
SRCDATA	AUDITRECORDS	LocationOID	Foreign key: Locations.OID	5	C	128	$128.
SRCDATA	AUDITRECORDS	DateTimeStamp	The datetime that the data modification was performed	6	C	25	$25.
SRCDATA	AUDITRECORDS	ReasonForChange	User-supplied reason for a data change	7	C	2000	$2000.
SRCDATA	AUDITRECORDS	SourceID	Source of the data within an originating system	8	C	2000	$2000.
SRCDATA	CLINICALDATA	OID	Unique ClinicalData identifier	1	C	128	$128.
SRCDATA	CLINICALDATA	StudyOID	Foreign key: Study.OID	2	C	128	$128.
SRCDATA	CLINICALDATA	MetaDataVersionOID	Foreign key: MetaDataVersion.OID	3	C	128	$128.
SRCDATA	CLINICALDATA	FK_ODM	Foreign key: ODM.FileOID	4	C	128	$128.
SRCDATA	CLITEMDECODETRANSLATEDTEXT	TranslatedText	Human-readable text appropriate for a particular language	1	C	2000	$2000.
SRCDATA	CLITEMDECODETRANSLATEDTEXT	lang	Natural language or country-specific language variant	2	C	17	$17.
SRCDATA	CLITEMDECODETRANSLATEDTEXT	FK_CodeListItems	Foreign key: CodeListItems.OID	3	C	128	$128.

The Srcdata library contains the SAS data sets that represent the ODM file metadata and content. By default, odm_read creates 52 unique data sets in SAS Clinical Standards Toolkit 1.3. Some of these data sets might be empty if no associated content was derived from the odm.xml file. There is a one-to-one correspondence between the tables listed in the Srcdata library and the tables contained in the source_tables metadata file in the Srcmeta library.

Display 7.4 Example of Partial Srcdata Library Derived During odm_read



Auditrecords	17.0KB	Table
Clinicaldata	17.0KB	Table
Clitemdecodetranslatedtext	49.0KB	Table
Codelistitems	17.0KB	Table
Codelists	17.0KB	Table
Conditiondefformalexpression	17.0KB	Table
Conditiondefs	17.0KB	Table
Conditiondeftranslatedtext	17.0KB	Table
Enumerateditems	17.0KB	Table
Externalcodelists	17.0KB	Table
Formdata	49.0KB	Table
Formdefarchlayouts	17.0KB	Table
Formdefitemgrouprefs	17.0KB	Table
Formdefs	17.0KB	Table
Formdeftranslatedtext	17.0KB	Table
Imputationmethods	17.0KB	Table

Reading CDISC CRT-DDS define.xml Files: crtdds_read Macro

The process for reading CDISC CRT-DDS define.xml files is similar to reading CDISC ODM XML files. SAS Clinical Standards Toolkit 1.3 supports reading a define.xml file and translating the file metadata into a SAS representation of the CDISC CRT-DDS model. To read the define.xml file, a specialized macro named crtdds_read.sas is available in the CRT-DDS 1.0 standards macro folder, located at **_<global standards library directory>_/standards/cdisc-crtdds-1.0-1.3/macros**. This macro is referenced from the create_sascrtdds_fromxml.sas driver program. There are no input parameters in the call to the crtdds_read macro. File references and other metadata that are required by the macro are set as global macro variables. Currently, their values are set through the framework initialization properties and the CDISC CRT-DDS 1.0 initialization properties processes. Throughout processing of the crtdds_read macro, the Results data set contains all framework and CRT-DDS 1.0 specific messages generated during run time.

Based on file references retrieved from the SASReferences data set, crtdds_read accesses the define.xml file.

The following is a partial listing of a define.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="define1-0-0.xsl"?>

<!--Produced from SAS data using the SAS Clinical Toolkit.-->
<ODM xmlns="http://www.cdisc.org/ns/odm/v1.2"
xmlns:def="http://www.cdisc.org/ns/def/v1.0"
xmlns:xlink="http://www.w3.org/1999/xlink" FileOID="1" CreationDateTime=
"2010-10-07T11:41:05-04:00" AsOfDateTime="2010-08-05T09:35:59"
Description="define1" FileType="Snapshot" Id="define1" ODMVersion="1.0"
Originator="SAS Institute">
   <Study OID="1">
        <GlobalVariables>
            <StudyName>study1</StudyName>
            <StudyDescription>first study</StudyDescription>
            <ProtocolName>Protocol abc</ProtocolName>
        </GlobalVariables>
        <MetaDataVersion OID="1" Name="CDISC-SDTM 3.1.2"
Description="CDISC-SDTM 3.1.2" def:DefineVersion="1.2"
def:StandardName="CDISC-SDTM" def:StandardVersion="3.1.2">
            <ItemGroupDef OID="AE1" Name="AE" Repeating="Yes" IsReferenceData="No"
SASDatasetName="AE" Domain="AE" Purpose="Tabulation" def:Label="Adverse Events"
def:Class="Events" def:Structure="One record per adverse event per subject"
def:DomainKeys="STUDYID USUBJID AEDECOD AESTDTC" def:ArchiveLocationID="AE1">
<ItemRef ItemOID="COL1" Mandatory="Yes" OrderNumber="1"
         KeySequence="1" Role="Identifier"/>
<ItemRef ItemOID="COL2" Mandatory="Yes" OrderNumber="2"
         Role="Identifier"/>
<ItemRef ItemOID="COL3" Mandatory="Yes" OrderNumber="3"
         KeySequence="2" Role="Identifier"/>
<ItemRef ItemOID="COL4" Mandatory="Yes" OrderNumber="4"
         Role="Identifier"/>
<ItemRef ItemOID="COL5" Mandatory="No" OrderNumber="5"
         Role="Identifier"/>
<ItemRef ItemOID="COL6" Mandatory="No" OrderNumber="6"
         Role="Identifier"/>
<ItemRef ItemOID="COL7" Mandatory="No" OrderNumber="7"
```

```
         Role="Identifier"/>
<ItemRef ItemOID="COL8" Mandatory="Yes" OrderNumber="8"
         Role="Topic"/>
<ItemRef ItemOID="COL9" Mandatory="No" OrderNumber="9"
         Role="SynonymQualifier"/>
<ItemRef ItemOID="COL10" Mandatory="Yes" OrderNumber="10"
         KeySequence="3" Role="SynonymQualifier"/>
<ItemRef ItemOID="COL11" Mandatory="No" OrderNumber="11"
         Role="GroupingQualifier"/>
<ItemRef ItemOID="COL12" Mandatory="No" OrderNumber="12"
         Role="GroupingQualifier"/>
<ItemRef ItemOID="COL13" Mandatory="No" OrderNumber="13"
         Role="RecordQualifier"/>
<ItemRef ItemOID="COL14" Mandatory="No" OrderNumber="14"
         Role="RecordQualifier"/>
<ItemRef ItemOID="COL15" Mandatory="No" OrderNumber="15"
         Role="RecordQualifier"/>
<ItemRef ItemOID="COL16" Mandatory="No" OrderNumber="16"
         Role="RecordQualifier"/>
<ItemRef ItemOID="COL17" Mandatory="No" OrderNumber="17"
         Role="RecordQualifier"/>
<ItemRef ItemOID="COL18" Mandatory="No" OrderNumber="18"
         Role="RecordQualifier"/>
<ItemRef ItemOID="COL19" Mandatory="No" OrderNumber="19"
         Role="RecordQualifier"/>
<ItemRef ItemOID="COL20" Mandatory="No" OrderNumber="20"
         Role="RecordQualifier"/>
```

After the crtdds_read macro confirms that the define.xml file exists, a call is made to the SAS data step component JavaObj. In SAS 9.1.3, you get a warning in the log that states that JavaObj is experimental. The JavaObj processing converts the define.xml file into the cubeXML file through transformations using XSL files and processes. The cubeXML file is created in the Work library. The name of the cubeXML file is **_cub*nnnn*.xml** , where ***nnnn*** is a randomly generated number. The cubeXML file is accessed using the SAS XML LIBNAME engine and XMLMAP processing. A default XMLMAP file is stored in the sample CRT-DDS 1.0 study folder hierarchy under **/referencexml** as define.map. The define.map file must exist to process the cubeXML file. If it does not exist, then the crtdds_read attempts to create one using the CRT-DDS reference metadata.

The following is a partial listing of the define.map file.

```
<?xml version="1.0" encoding="windows-1252"?>
<SXLEMAP version="1.2">

<TABLE name="AnnotatedCRFs">
   <TABLE-PATH syntax="XPath">/LIBRARY/AnnotatedCRFs</TABLE-PATH>
   <TABLE-DESCRIPTION></TABLE-DESCRIPTION>

   <COLUMN name="DocumentRef">
     <PATH syntax="Xpath">/LIBRARY/AnnotatedCRFs/DocumentRef</PATH>
     <TYPE>character</TYPE>
     <DATATYPE>character</DATATYPE>
     <DESCRIPTION></DESCRIPTION>
     <LENGTH>2000</LENGTH>
   </COLUMN>
   <COLUMN name="leafID">
     <PATH syntax="Xpath">/LIBRARY/AnnotatedCRFs/leafID</PATH>
```

```
      <TYPE>character</TYPE>
      <DATATYPE>character</DATATYPE>
      <DESCRIPTION></DESCRIPTION>
      <LENGTH>128</LENGTH>
    </COLUMN>
    <COLUMN name="FK_MetaDataVersion">
      <PATH syntax="Xpath">/LIBRARY/AnnotatedCRFs/FK_MetaDataVersion</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>character</DATATYPE>
      <DESCRIPTION></DESCRIPTION>
      <LENGTH>128</LENGTH>
    </COLUMN>

  </TABLE>
```

Processing of the cubeXML file results in the derivation of the data sets (such as ItemDefs) currently included in the SAS representation of the CDISC CRT-DDS model.

The final step in crtdds_read processing is the derivation of table and column metadata that describe the data sets in the SAS representation of the define.xml file. At this point, the crtdds_read macro is ready to create the source_tables and source_columns data sets. The tables in the source_tables data sets are created and copied to the output library as defined in the SASReferences data set.

Sample Driver Program: create_sascrtdds_fromxml.sas

Overview

Each primary SAS Clinical Standards Toolkit task, such as reading CDISC CRT-DDS XML files, is guided by a sample driver program that is provided by SAS. The create_sascrtdds_fromxml.sas driver program is used to read define.xml files.

For SAS 9.1.3, the driver program is located at:

```
!sasroot/../SASClinicalStandardsToolkitCRTDDS10/1.3/
sample/cdisc-crtdds-1.0/programs/create_sascrtdds_fromxml.sas
```

For SAS 9.2, the driver program is located at:

```
!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/
sample/cdisc-crtdds-1.0/programs/create_sascrtdds_fromxml.sas
```

The value for **!sasroot** is the location of your SAS installation directory.

The SASReferences Data Set

As a part of each SAS Clinical Standards Toolkit process setup, a valid SASReferences data set is required. It can be modified to point to study-specific files. For an explanation of the SASReferences data set, see Chapter 5, “SASReferences File,” on page 69.

In the SASReferences data set, there are two input file references and four output references that are key to successful completion of the driver program. The following table lists these files and data sets, and they are discussed in separate sections. In the sample create_sascrtdds_fromxml.sas driver program, the following values are set for &studyRootPath and &studyOutputPath and are specific to a SAS release.

SAS 9.1.3

```
&studyRootPath=!sasroot/../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

```
&studyOutputPath=!sasroot/../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

SAS 9.2

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

```
&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

Table 7.2 *Key Components of the SASReferences Data Set*

Input or Output	Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	externalxml	crtxml	fileref	&studyRootPath/ sourcexml	define.xml
Input	referencexml	crtmap	fileref	&studyRootPath/ referencexml	define.map
Output	sourcedata	srcdata	LIBNAME	&studyOutputPath/data	*.*
Output	sourcemetadata	srcmeta	LIBNAME	&studyOutputPath/metadata	Source_ tables.sas7bdat
Output	sourcemetadata	srcmeta	LIBNAME	&studyOutputPath/metadata	Source_ columns.sas7bdat
Output	results	results	LIBNAME	&studyOutputPath/results	Read_ results.sas7bdat

Process Inputs

The metadata type externalxml refers to the define.xml file that is being read. The filename crtxml is defined in the SASReferences data set. This filename is used in the submitted SAS code when referring to the define.xml file.

The metadata type referencexml refers to the SAS map file that is used to generate the SAS data sets that represent the define.xml file metadata and content. The filename crtmap is defined in the SASReferences data set that is used in the submitted SAS code when referring to the SAS map file. If a path and filename for the map file is not specified, then a temporary map file is created as part of the crtdds_read processing.

Process Outputs

The sourcedata type is the library where the metadata files are created. These metadata files are the data sets that comprise the CRT-DDS information. In the SAS Clinical Standards Toolkit sample study, these data sets are written to the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/deriveddata`** directory. This location is represented in the driver program by the Srcdata library name.

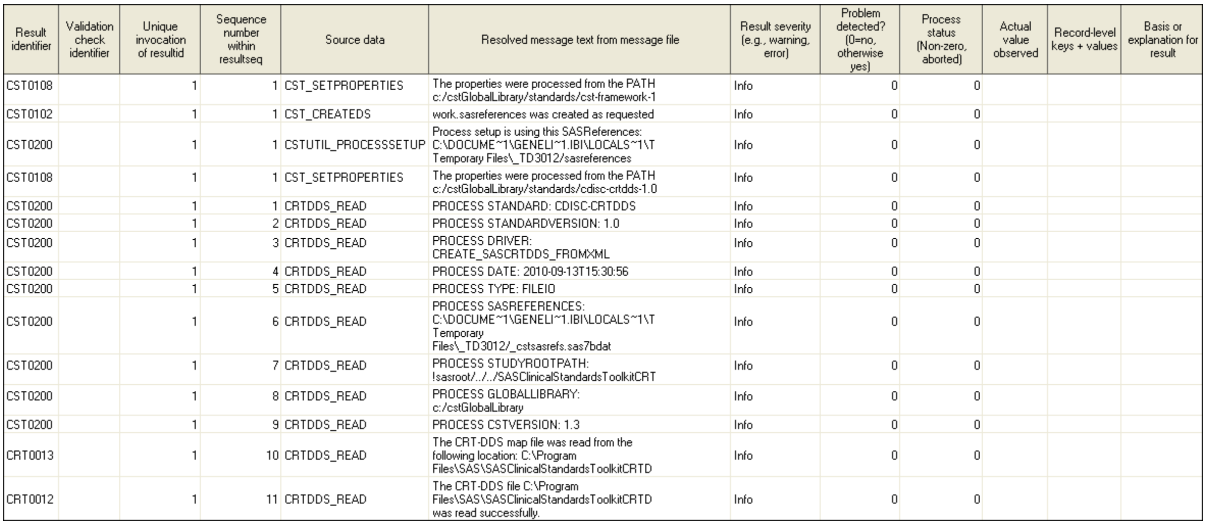
The sourcemetadata type refers to two data sets that are created from the cubeXML file, source_tables and source_columns. Both data sets are stored in the same library. The source_tables data set contains metadata about each table that is derived from the CRT-DDS process. The source_columns data set contains similar metadata, but it is at the column level. In the SAS Clinical Standards Toolkit sample study, this metadata is written to the **!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/derivedmetadata** directory. This location is represented in the driver program by the Srcmeta library name.

The results type refers to the Results data set that contains information from running the CRT-DDS process. In the SAS Clinical Standards Toolkit sample study, this information is written to the **!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/results** directory. This location is represented in the driver program by the Results library name.

Process Results

When the driver program finishes running, the read_results.sas7bdat data set is created in the Results library. This data set contains informational, warning, and any error messages that were generated by the submitted driver program. The following display shows an example of the contents of a Results data set in the CRT-DDS sample study.

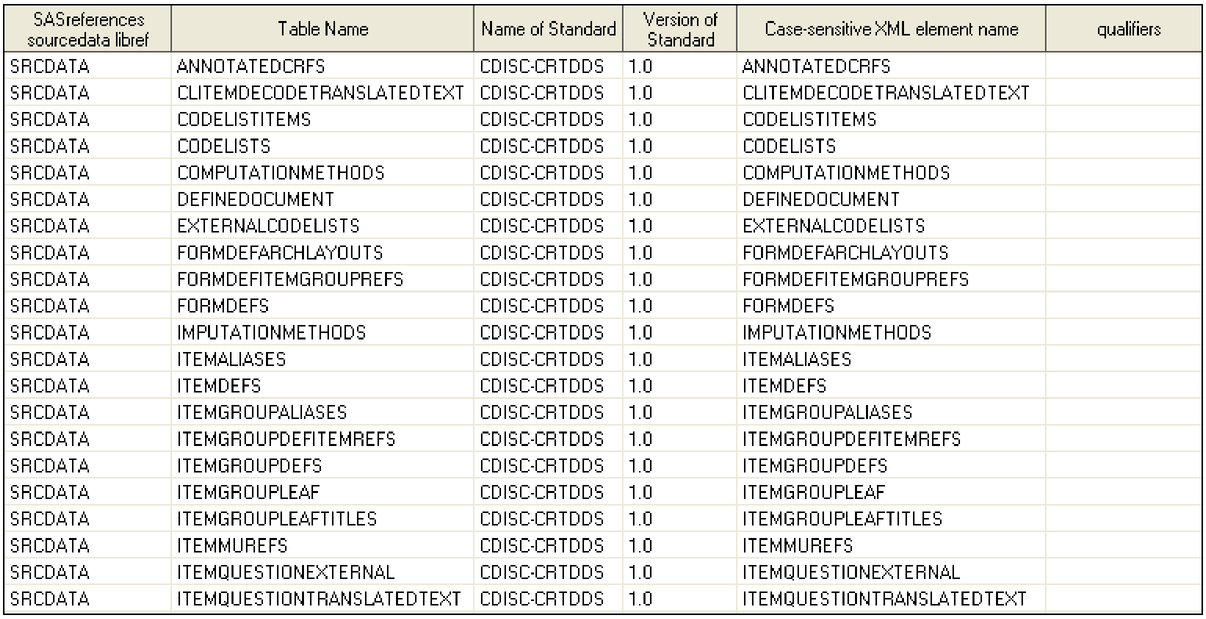
Display 7.5 *Example of a Partial Results Data Set Created by the create_sascrtdds_fromxml.sas Driver*



Result identifier	Validation check identifier	Unique invocation of resultid	Sequence number within resultseq	Source data	Resolved message text from message file	Result severity (e.g., warning, error)	Problem detected? (0=no, otherwise yes)	Process status (Non-zero, aborted)	Actual value observed	Record-level keys + values	Basis or explanation for result
CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cst-framework-1	Info	0	0			
CST0102		1	1	CST_CREATEDS	work.sasreferences was created as requested	Info	0	0			
CST0200		1	1	CSTUTIL_PROCESSSETUP	Process setup is using this SASReferences: C:\DOCUME~1\GENELI~1.IBI\LOCALS~1\T Temporary Files_TD3012/sasreferences	Info	0	0			
CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-crtdds-1.0	Info	0	0			
CST0200		1	1	CRTDDS_READ	PROCESS STANDARD: CDISC-CRTDDS	Info	0	0			
CST0200		1	2	CRTDDS_READ	PROCESS STANDARDVERSION: 1.0	Info	0	0			
CST0200		1	3	CRTDDS_READ	PROCESS DRIVER: CREATE_SASCRTDDS_FROMXML	Info	0	0			
CST0200		1	4	CRTDDS_READ	PROCESS DATE: 2010-09-13T15:30:56	Info	0	0			
CST0200		1	5	CRTDDS_READ	PROCESS TYPE: FILEIO	Info	0	0			
CST0200		1	6	CRTDDS_READ	PROCESS SASREFERENCES: C:\DOCUME~1\GENELI~1.IBI\LOCALS~1\T Temporary Files_TD3012/_cstsasrefs.sas7bdat	Info	0	0			
CST0200		1	7	CRTDDS_READ	PROCESS STUDYROOTPATH: !sasroot/../../SASClinicalStandardsToolkitCRT	Info	0	0			
CST0200		1	8	CRTDDS_READ	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info	0	0			
CST0200		1	9	CRTDDS_READ	PROCESS CSTVERSION: 1.3	Info	0	0			
CRT0013		1	10	CRTDDS_READ	The CRT-DDS map file was read from the following location: C:\Program Files\SAS\SASClinicalStandardsToolkitCRTD	Info	0	0			
CRT0012		1	11	CRTDDS_READ	The CRT-DDS file C:\Program Files\SAS\SASClinicalStandardsToolkitCRTD was read successfully.	Info	0	0			

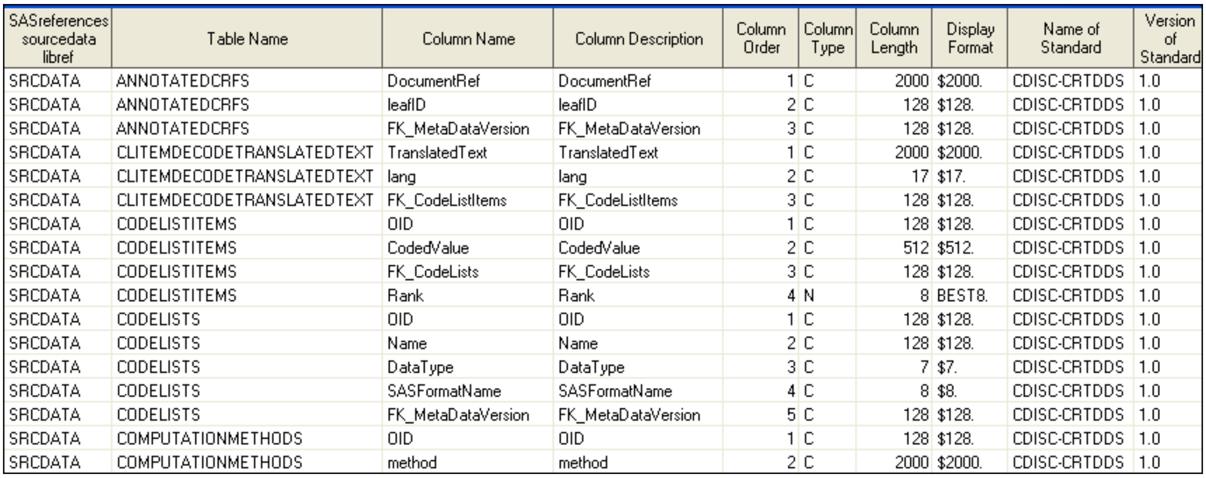
The crtdds_read macro creates the source_tables and source_columns data sets in the Srcmeta library. These data sets contain the table and column metadata for the SAS representation of CRT-DDS that is derived from the define.xml file. The Srcmeta library corresponds to the location specified in SASReferences (**&studyOutputPath/derivedmetadata**).

Display 7.6 *Example of Partial Source_Tables Data Set Derived During crtdds_read*



SASreferences sourcedata libref	Table Name	Name of Standard	Version of Standard	Case-sensitive XML element name	qualifiers
SRCDATA	ANNOTATEDCRFS	CDISC-CRTDDS	1.0	ANNOTATEDCRFS	
SRCDATA	CLITEMDECODETRANSLATEDTEXT	CDISC-CRTDDS	1.0	CLITEMDECODETRANSLATEDTEXT	
SRCDATA	CODELISTITEMS	CDISC-CRTDDS	1.0	CODELISTITEMS	
SRCDATA	CODELISTS	CDISC-CRTDDS	1.0	CODELISTS	
SRCDATA	COMPUTATIONMETHODS	CDISC-CRTDDS	1.0	COMPUTATIONMETHODS	
SRCDATA	DEFINEDOCUMENT	CDISC-CRTDDS	1.0	DEFINEDOCUMENT	
SRCDATA	EXTERNALCODELISTS	CDISC-CRTDDS	1.0	EXTERNALCODELISTS	
SRCDATA	FORMDEFARCHLAYOUTS	CDISC-CRTDDS	1.0	FORMDEFARCHLAYOUTS	
SRCDATA	FORMDEFITEMGROUPREFS	CDISC-CRTDDS	1.0	FORMDEFITEMGROUPREFS	
SRCDATA	FORMDEFS	CDISC-CRTDDS	1.0	FORMDEFS	
SRCDATA	IMPUTATIONMETHODS	CDISC-CRTDDS	1.0	IMPUTATIONMETHODS	
SRCDATA	ITEMALIASES	CDISC-CRTDDS	1.0	ITEMALIASES	
SRCDATA	ITEMDEFS	CDISC-CRTDDS	1.0	ITEMDEFS	
SRCDATA	ITEMGROUPALIASES	CDISC-CRTDDS	1.0	ITEMGROUPALIASES	
SRCDATA	ITEMGROUPDEFITEMREFS	CDISC-CRTDDS	1.0	ITEMGROUPDEFITEMREFS	
SRCDATA	ITEMGROUPDEFS	CDISC-CRTDDS	1.0	ITEMGROUPDEFS	
SRCDATA	ITEMGROUPLEAF	CDISC-CRTDDS	1.0	ITEMGROUPLEAF	
SRCDATA	ITEMGROUPLEAFTITLES	CDISC-CRTDDS	1.0	ITEMGROUPLEAFTITLES	
SRCDATA	ITEMMUREFS	CDISC-CRTDDS	1.0	ITEMMUREFS	
SRCDATA	ITEMQUESTIONEXTERNAL	CDISC-CRTDDS	1.0	ITEMQUESTIONEXTERNAL	
SRCDATA	ITEMQUESTIONTRANSLATEDTEXT	CDISC-CRTDDS	1.0	ITEMQUESTIONTRANSLATEDTEXT	

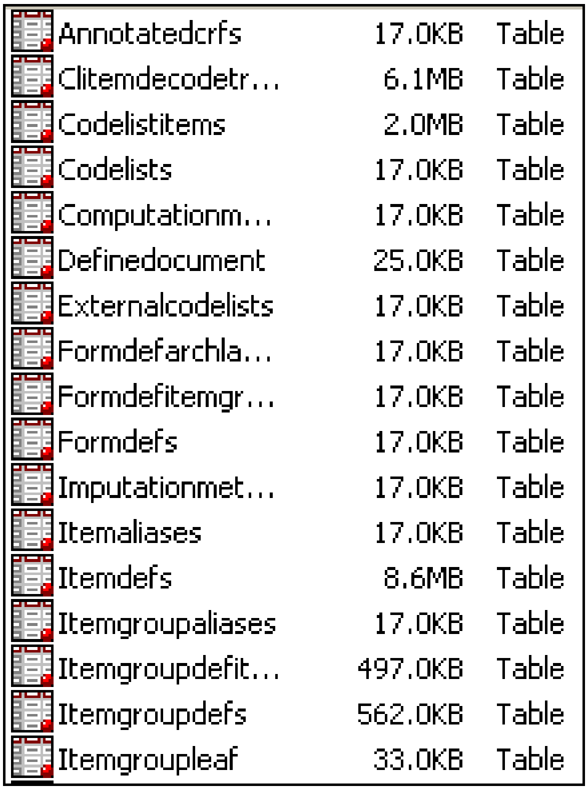
Display 7.7 *Example of Partial Source_Columns Data Set Derived During crtdds_read*



SASreferences sourcedata libref	Table Name	Column Name	Column Description	Column Order	Column Type	Column Length	Display Format	Name of Standard	Version of Standard
SRCDATA	ANNOTATEDCRFS	DocumentRef	DocumentRef	1	C	2000	$2000.	CDISC-CRTDDS	1.0
SRCDATA	ANNOTATEDCRFS	leafID	leafID	2	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	ANNOTATEDCRFS	FK_MetaDataVersion	FK_MetaDataVersion	3	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	CLITEMDECODETRANSLATEDTEXT	TranslatedText	TranslatedText	1	C	2000	$2000.	CDISC-CRTDDS	1.0
SRCDATA	CLITEMDECODETRANSLATEDTEXT	lang	lang	2	C	17	$17.	CDISC-CRTDDS	1.0
SRCDATA	CLITEMDECODETRANSLATEDTEXT	FK_CodeListItems	FK_CodeListItems	3	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTITEMS	OID	OID	1	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTITEMS	CodedValue	CodedValue	2	C	512	$512.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTITEMS	FK_CodeLists	FK_CodeLists	3	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTITEMS	Rank	Rank	4	N	8	BEST8.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTS	OID	OID	1	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTS	Name	Name	2	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTS	DataType	DataType	3	C	7	$7.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTS	SASFormatName	SASFormatName	4	C	8	$8.	CDISC-CRTDDS	1.0
SRCDATA	CODELISTS	FK_MetaDataVersion	FK_MetaDataVersion	5	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	COMPUTATIONMETHODS	OID	OID	1	C	128	$128.	CDISC-CRTDDS	1.0
SRCDATA	COMPUTATIONMETHODS	method	method	2	C	2000	$2000.	CDISC-CRTDDS	1.0

The Srcdata library contains the driver-generated tables that comprise the SAS representation of the CRT-DDS model. There is a one-to-one correspondence between the tables listed in the Srcdata library and the tables contained in the source_tables metadata file in the Srcmeta library. The Srcdata library corresponds to the location specified in SASReferences (`&studyOutputPath/deriveddata`).

Display 7.8 *Example of Partial Srcdata Library Derived During crtdds_read*



Annotatedcrfs	17.0KB	Table
Clitemdecodetr...	6.1MB	Table
Codelistitems	2.0MB	Table
Codelists	17.0KB	Table
Computationm...	17.0KB	Table
Definedocument	25.0KB	Table
Externalcodelists	17.0KB	Table
Formdefarchla...	17.0KB	Table
Formdefitemgr...	17.0KB	Table
Formdefs	17.0KB	Table
Imputationmet...	17.0KB	Table
Itemaliases	17.0KB	Table
Itemdefs	8.6MB	Table
Itemgroupaliases	17.0KB	Table
Itemgroupdefit...	497.0KB	Table
Itemgroupdefs	562.0KB	Table
Itemgroupleaf	33.0KB	Table

When running the driver programs against non-sample data, you need to populate the SASReferences data set in the driver program with the proper values. For an explanation of the SASReferences data set, see Chapter 5, “SASReferences File,” on page 69.

Writing XML Files

Overview

Support of CDISC XML-based standards such as CDISC CRT-DDS (define.xml) and CDISC ODM includes the ability to render these files in SAS data set format and the ability to create model-specific XML files from a SAS data set representation of those standards.

In SAS Clinical Standards Toolkit 1.3, you can create a CDISC CRT-DDS 1.0 define.xml file that references a CDISC SDTM 3.1.1 or 3.1.2 study. CDISC ODM write capabilities are under development. (For the latest updates, see the SAS Support Web site for SAS Clinical Standards Toolkit at **`http://support.sas.com/rnd/base/cdisc/cst/index.html`**).

The next section outlines the basic workflow for the creation of model-specific XML files.

Basic Workflow

The following is the basic workflow for writing XML files:

1. Build the SAS representation of a given XML-based standard by referencing an existing set of data and metadata about a clinical study, or by creating data and metadata about a new clinical study.
2. Validate the SAS representation of the XML-based standard (to include foreign key relationships, value conformance to a set of expected values, and so on). This step is optional.
3. Create a standardized intermediate cubeXML file using the data and metadata contained in the SAS representation of the standard.
4. (Build and) reference a set of valid XSL style sheets for each target data set (such as ItemDefs.xsl).
5. Use the SAS DATA step component JavaObj to read the cubeXML file using the XSL style sheets to create the target standard-specific XML file.
6. Validate the structure and syntax of the XML file that was created. This step is optional.

Creating the CDISC CRT-DDS 1.0 define.xml File

There are four key macros that are provided with the SAS Clinical Standards Toolkit that support creation of the define.xml file. The four macros are listed in the order in which they are executed:

- The crtdds_sdtm311todefine10.sas macro creates the 39 tables for the SAS representation of the CRT-DDS files from SDTM metadata. This macro, using SDTM table and column metadata as its source, populates a subset of 12 CRT-DDS data sets. Although the macro name implies that it is specific to SDTM 3.1.1, it operates on both CDISC SDTM 3.1.1 and 3.1.2 domains.
- The crtdds_validate.sas macro submits a set of validation checks based on what is defined in the Validation Control data set to validate the referenced SAS representation of the CRT-DDS files.
- The crtdds_write.sas macro creates the define.xml file from the SAS representation of the CRT-DDS files.
- The crtdds_xmlvalidate.sas macro validates that the XML file is syntactically correct. This macro is important if you customize the define.xml file outside of the workflow. For example, if you edit the define.xml file to add links for annotated CRF pages, this macro validates the syntax.

These macros are called by driver programs that are responsible for properly setting up each SAS Clinical Standards Toolkit process to perform a specific SAS Clinical Standards Toolkit task. Three sample driver modules are provided with the SAS Clinical Standards Toolkit CDISC CRT-DDS standard. The following lists the purpose of each of these drivers:

1. The create_crtdds10_from_sdtm311.sas driver program sets up the required metadata and SASReferences data set for the sample study. It runs the crtdds_sdtm311todefine10.sas macro. It creates the SAS representation of the CRT-DDS define data sets from the sample study SDTM data sets.
2. The validate_crtdds_data.sas driver program validates the SAS representation of the CRT-DDS define data sets based on the selected CRT-DDS validation checks. This driver program can be run multiple times until data validation has been reconciled.
3. The create_crtdds_define.sas driver program creates the define.xml file. It runs the crtdds_write and crtdds_xmlvalidate macros. This driver program creates and validates the XML syntax for the define.xml file.

These three driver programs are examples that are provided with the SAS Clinical Standards Toolkit. You can use these driver programs or create your own. The names of these driver programs are not important. However, the content is important and demonstrates how the various SAS Clinical Standards Toolkit framework macros are used to generate the required metadata files.

Sample Driver Program: create_crtdds10_from_sdtm311.sas

Overview

The create_crtdds10_from_sdtm311.sas driver program sets up the required environment variables and library references to initiate the crtdds_sdtm311todefine10.sas macro. This macro extracts data from the SDTM 3.1.1 or 3.1.2 metadata files. (For more information about the source_tables and source_columns data sets, see "Source Metadata" on page 90.) Depending on the available source information, the macro attempts to convert the information into the 39 tables that represent the SAS interpretation of the CDISC CRT-DDS 1.0 model. All 39 data sets are created, but only those data sets with the available data are populated. The other tables contain zero observations.

The following parameters must be set by the user before submitting the macro:

Table 7.3 *Parameters for the crtdds_sdtm311todefine10.sas Macro*

Parameter	Required	Description
_cstOutLib	Yes	Identifies the library reference (LIBNAME) where the tables are created.
_cstSourceTables	Yes	A data set that contains the SDTM metadata for the domains to be included in the CRT-DDS file.
_cstSourceColumns	Yes	A data set that contains the SDTM metadata for the domain columns to be included in the CRT-DDS file.
_cstSourceStudy	Yes	A data set that contains the SDTM metadata for the studies to be included in the CRT-DDS file.

The following is an example of a call to the crtdds_sdtm311todefine10.sas macro:

```
%crtdds_sdtm311todefine10(
_cstOutLib=srcdata,
_cstSourceTables=sampdata.source_tables,
_cstSourceColumns=sampdata.source_columns,
_cstSourceStudy=sampdata.source_study
);
```

In the example, the crtdds_sdtm311todefine10 macro sets **`_cstOutLib`** to **`srcdata`**. All of the CRT-DDS-defined tables are written to the SAS Srcdata library. The **`_cstSourceTables`** parameter accesses the source_tables data set that exists in the Sampdata library (**`sampdata.source_tables`**). The **`_cstSourceColumns`** parameter accesses the source_columns data set that exists in the Sampdata library (`sampdata.source_columns`). The `_cstSourceStudy` parameter accesses the source_study data set that exists in the sampdata library (`sampdata.source_study`).

The create_crtdds10_from_sdtm311.sas driver program is provided with SAS, and it is ready to run on any of the SDTM sample studies. Although the program name implies that it is specific to SDTM 3.1.1, it operates on both CDISC SDTM 3.1.1 and 3.1.2 domains. The driver program can be run interactively or in batch. To run the program interactively, start a SAS session, and load the driver program into the SAS editor.

For SAS 9.1.3, the driver program is located at:

```
!sasroot/../SASClinicalStandardsToolkitCRTDDS10/1.3/
sample/cdisc-crtdds-1.0/programs/create_crtdds10_from_
sdtm311.sas
```

For SAS 9.2, the driver program is located at:

```
!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/
sample/cdisc-crtdds-1.0/programs/create_crtdds10_from_
sdtm311.sas
```

The value for `!sasroot` is the location of your SAS installation directory.

The SASReferences Data Set

As a part of each SAS Clinical Standards Toolkit process setup, a valid SASReferences data set is required. It can be modified to point to study-specific files. For an explanation of the SASReferences data set, see Chapter 5, “SASReferences File,” on page 69.

In the SASReferences data set, there are two input file references and one output reference that are key to successful completion of the create_crtdds10_from_sdtm311.sas driver program. The following table lists these files and data sets, and they are discussed in separate sections. In the sample create_crtdds10_from_sdtm311.sas driver program, the following values are set for &studyRootPath and &studyOutputPath and are specific to a SAS release.

SAS 9.1.3

```
&studyRootPath=!sasroot/../
SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-
sdtm-3.1.2/sascstdemodata
```

```
&studyOutputPath=!sasroot/../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/
```

SAS 9.2

```
&studyRootPath=!sasroot/../../SASClinicalStandardsToolkit
SDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata
```

```
&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

Table 7.4 *Key Components of the SASReferences Data Set*

Input or Output	Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	sourcemetadata	sampdata	LIBNAME	&studyRootPath/ metadata	source_ tables.sas7bdat

Input or Output	Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	sourcemetadata	sampdata	LIBNAME	&studyRootPath/ metadata	source_ columns.sas7bdat
Output	sourcedata	srcdata	LIBNAME	&studyOutputPat h/data	

Process Inputs

The sourcemetadata type refers to two data sets that contain the SDTM domain metadata, source_tables and source_columns. Both data sets are stored in the same library. Because the sample create_crtdds10_from_sdtm311.sas driver program provided with the SAS Clinical Standards Toolkit references a source CDISC SDTM 3.1.2 study, the source_tables data set contains SDTM 3.1.2 metadata about each standard domain defined in the *CDISC-SDTM 3.1.2 Implementation Guide* and includes any customizations that you have added. The source_columns type contains similar metadata, but it is at the column level. This source metadata is read from the **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata/metadata`** directory. This location is represented in the driver program by the Srcmeta library name.

A source study data set (source_study.sas7bdat) is required by this macro. The following variables are required in this data set:

Table 7.5 *Variables Required in the Source Study Data Set (source_study.sas7bdat)*

Variable*	Required	Description
StudyName	Yes	Name of the study. This value is used to populate the srcdata.study.studyname column.
DefineDocumentName	Yes	Name of the define document being created. This value is used to populate the srcdata.definedocument.description and srcdata.definedocument.id columns.
SASref	Yes	Reference that ties the study name to the corresponding domains that are associated with this study in the source_tables and source_columns data sets.
ProtocolName	Yes	Name of the protocol for the study. This value is used to populate the srcdata.study.protocolname column.

Variable*	Required	Description
StudyDescription	Yes	Description of the study. This value is used to populate the srcdata.study.studydescription column. *Note:* You should not use commas, semicolons, or quotation marks in the description.

*All variables are required to be non-blank.

Multiple studies can be referenced in the source study data set, as well as source_columns and source_tables, by using different SASref values to link them across the tables.

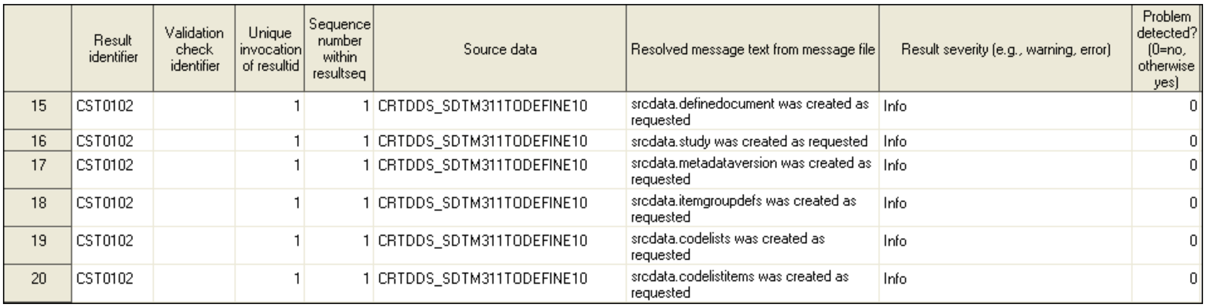
Process Outputs

The sourcedata type is the library where the metadata files are created. These metadata files are the data sets that constitute the SAS representation of the CDISC CRT-DDS 1.0 standard. The create_crtdds10_from_sdtm311.sas driver program creates 39 data sets. Most of these data sets have zero observations because there is no default SDTM metadata source. In the SAS Clinical Standards Toolkit sample study, these data sets are written to the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/data`** directory. This location is represented in the driver program by the srcdata library name.

Process Results

When the driver program finishes running, the work._cstresults.sas7bdat data set is created. This data set contains informational, warning, and any error messages that were generated by the submitted driver program. Because the create_crtdds10_from_sdtm311.sas sample SASreferences data set does not include a results record, this example does not save the process results data set after the SAS session ends.

Display 7.9 *Example of a Partial Results Data Set from CRT-DDS Sample Study*



	Result identifier	Validation check identifier	Unique invocation of resultid	Sequence number within resultseq	Source data	Resolved message text from message file	Result severity (e.g., warning, error)	Problem detected? (0=no, otherwise yes)
15	CST0102		1	1	CRTDDS_SDTM311TODEFINE10	srcdata.definedocument was created as requested	Info	0
16	CST0102		1	1	CRTDDS_SDTM311TODEFINE10	srcdata.study was created as requested	Info	0
17	CST0102		1	1	CRTDDS_SDTM311TODEFINE10	srcdata.metadataversion was created as requested	Info	0
18	CST0102		1	1	CRTDDS_SDTM311TODEFINE10	srcdata.itemgroupdefs was created as requested	Info	0
19	CST0102		1	1	CRTDDS_SDTM311TODEFINE10	srcdata.codelists was created as requested	Info	0
20	CST0102		1	1	CRTDDS_SDTM311TODEFINE10	srcdata.codelistitems was created as requested	Info	0

Sample Driver Program: create_crtdds_define.sas

Overview

The create_crtdds_define.sas driver program sets up the required environment variables and library references to initiate the crtdds_write.sas macro. This macro reads the 39 data sets that comprise the SAS representation of the CDISC CRT-DDS 1.0 model, and converts that information to the required define.xml structure. If source metadata or data are missing, then empty elements and attributes are not created in the define.xml file. The inputs and outputs are specified in the SASRferences data set. The following table lists the optional parameters that can be set by the user when submitting the macro:

Table 7.6 *Parameters for the crtdds_write.sas Macro*

Parameter	Required	Description
_cstCreateDisplayStyleSheet	Optional	Identifies whether the macro should create a style sheet in the same directory as the output XML file. If the value is 1, then the macro looks in the provided SASReferences file for a record with a type and subtype of referencexml and stylesheet and uses that file. If the value is 0, then the macro does not create the XSL, even if one is specified in the SASReferences file. The default setting is 1.
_cstOutputEncoding	Optional	XML encoding to use for the CRT-DDS file that is created. By default, UTF-8 is used.
_cstHeaderComment	Optional	A short comment is added at the top of the CRT-DDS file. If no comment is provided, then a default comment is used. The default comment notes that the file was produced by SAS Clinical Standards Toolkit.
_cstResultsOverrideDS	Optional	Provides the opportunity to designate [LIBNAME.]member as the name of the Results data set. If this parameter is omitted (default setting), then the Results data set specified by the &_cstResultsDS global macro variable is used.
_cstLogLevel	Optional	Identifies the level of error reporting. Valid values are Info, Warning, Error, and Fatal Error. The default setting is Info.

The following is an example of a call to the crtdds_write.sas macro:

```
%crtdds_write(_cstCreateDisplayStyleSheet=1, _cstOutputEncoding=UTF-16,
            _cstResultsOverrideDS=&_cstResultsDS);
```

In this example, a default style sheet is generated in the same directory as the XML output based on the information in the SASReferences data set. XML encoding is set to UTF-16, and process results are written to the default &_cstResultsDS data set.

The following is the call to the macro from the sample create_crtdds_define.sas driver program:

```
%crtdds_write(_cstCreateDisplayStyleSheet=1);
```

The call creates a display style sheet, and uses default values for the parameters.

The create_crtdds_define.sas driver program is ready to run on any of the CDISC SDTM sample studies. The driver program can be run interactively or in batch.

For SAS 9.1.3, the driver program is located at:

`!sasroot/../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/programs/create_crtdds_define.sas`

For SAS 9.2, the driver program is located at:

`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/programs/create_crtdds_define.sas`

The value for **`!sasroot`** is the location of your SAS installation directory.

Multiple tasks can be executed in any SAS Clinical Standards Toolkit driver program. The create_crtdds_define.sas driver program calls both the crtdds_write macro to create the define.xml file, and the crtdds_xmlvalidate macro to validate the syntax of the generated define.xml file. For more information about the crtdds_xmlvalidate macro, see “Validation of XML-Based Standards” on page 187.

The SASReferences Data Set

As a part of each SAS Clinical Standards Toolkit process setup, a valid SASReferences data set is required. It can be modified to point to study-specific files. For an explanation of the SASReferences data set, see Chapter 5, “SASReferences File,” on page 69.

In the SASReferences data set, there are two input file references and three output references that are key to successful completion of the create_crtdds_define.sas driver program. The following table lists these files and data sets, and they are discussed in separate sections. In the sample create_crtdds_define.sas driver program, the following values are set for &studyRootPath and &studyOutputPath and are specific to a SAS release.

SAS 9.1.3

`&studyRootPath=!sasroot/../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0`

`&studyOutputPath=!sasroot/../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0`

SAS 9.2

`&studyRootPath=!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0`

`&studyOutputPath=!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0`

Table 7.7 Key Components of the SASReferences Data Set

Input or Output	Metadata Type	LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	control	control	LIBNAME	&workpath	sasreferences.sas7bdat
Input	sourcedata	srcdata	LIBNAME	&studyRootPath/data	
Input or output	referencexml	xslt01	filename		
Output	results	results	LIBNAME	&studyOutputPath/results	write_results.sas7bdat
Output	externalxml	extxml	filename	&studyOutputPath/sourcexml	define.xml

Process Inputs

Use of the control library name that points to the path in the &workpath macro variable illustrates a technique of documenting the derivation of the SASReferences data set in the SAS Work library. The driver program initiates the macro variable &workpath with the following SAS code:

```
%let workPath=%sysfunc(pathname(work));
```

The sourcedata type is the library that contains the 39 data sets that might have been populated by the create_crtdds10_from_sdtm311.sas driver program. These metadata files are the data sets that constitute the SAS representation of the CDISC CRT-DDS 1.0 standard. In the SAS Clinical Standards Toolkit sample study, these data sets are read from the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/data`** directory. This location is represented in the driver program by the Srcdata library name.

Process Outputs

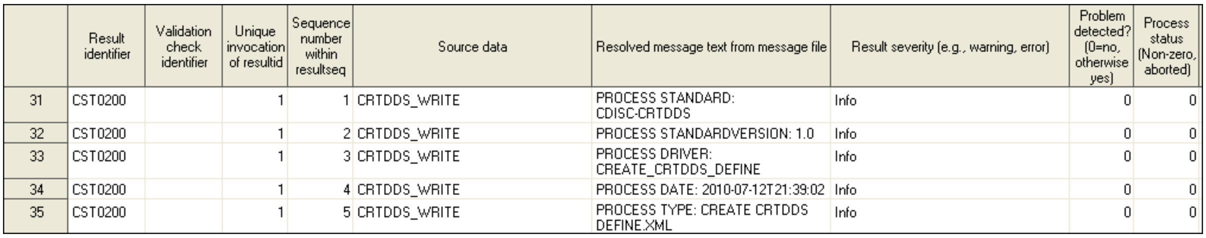
The externalxml type refers to the define.xml file. This file is accessed in the driver program from the extxml filename statement, and is written to the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/sourcexml`** directory.

The referencexml type can serve as either an input or output file reference. Because the path and filename are not provided, the crtdds_write macro interprets the _cstCreateDisplayStyleSheet=1 parameter to use the default style sheet that is provided by SAS Clinical Standards Toolkit in the Global Library. Had a path and filename been provided, the referencexml type would serve as an output file reference for the crtdds_write macro to copy the default style sheet from the Global Library to the path and filename that were specified. The results type refers to the write_results data set that documents the create define process results. In the SAS Clinical Standards Toolkit CDISC CRT-DDS folder hierarchy, this information is written to the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/results`** directory.

Process Results

Inclusion of the results record (row) in the SASReferences data set signals that the process results are to be copied to a write_results data set located in the specified SAS library.

Display 7.10 *Example of a Partial Results Data Set from the CRT-DDS Sample Study*



	Result identifier	Validation check identifier	Unique invocation of resultid	Sequence number within resultseq	Source data	Resolved message text from message file	Result severity (e.g., warning, error)	Problem detected? (0=no, otherwise yes)	Process status (Non-zero, aborted)
31	CST0200		1	1	CRTDDS_WRITE	PROCESS STANDARD: CDISC-CRTDDS	Info	0	0
32	CST0200		1	2	CRTDDS_WRITE	PROCESS STANDARDVERSION: 1.0	Info	0	0
33	CST0200		1	3	CRTDDS_WRITE	PROCESS DRIVER: CREATE_CRTDDS_DEFINE	Info	0	0
34	CST0200		1	4	CRTDDS_WRITE	PROCESS DATE: 2010-07-12T21:39:02	Info	0	0
35	CST0200		1	5	CRTDDS_WRITE	PROCESS TYPE: CREATE CRTDDS DEFINE.XML	Info	0	0

Validation of XML-Based Standards

XML Validation

When validating XML-based standards (such as CDISC ODM and CDISC CRT-DDS), SAS Clinical Standards Toolkit offers two complementary methodologies. The first methodology is described in Chapter 6, "Validation," on page 83. It relies on the definition of a master set of validation checks that are specific to the table and column metadata that define a set of data, and checks that are specific to the data itself. This method uses SAS files and SAS code to validate the SAS representation of the XML-based standard. Example checks include the assessment of foreign key relationships across data sets and value conformance to a set of expected values. The second methodology involves verification that an XML file is valid structurally and syntactically according to the XML schema for that standard.

SAS Clinical Standards Toolkit 1.3 provides both methodologies to support the validation of CDISC CRT-DDS 1.0 files. CDISC ODM validation capabilities are under development. (See the SAS Customer Support Web site for SAS Clinical Standards Toolkit at `http://support.sas.com/rnd/base/cdisc/cst/index.html` for the latest updates.)

Validating CDISC CRT-DDS 1.0 Files

The crtdds_xmlvalidate Macro

The crtdds_xmlvalidate.sas macro validates the structure and syntax of the define.xml file against the XML schema for the ODM standard. It can be run at any time. The SAS Clinical Standards Toolkit includes a call to the crtdds_xmlvalidate.sas macro immediately following the call to the crtdds_write.sas macro as the last step of the create_crtdds_define.sas sample driver program. If you customize the define.xml file after it is generated, then this macro can be used to validate the changes.

The following is an example of a call to the crtdds_xmlvalidate.sas macro:

```
%crtdds_xmlvalidate(_cstLogLevel=info,_cstResultsOverrideDS=work.xmlvalidate);
```

In this example, the %crtdds_xmlvalidate macro is being submitted with a log level of Info. The Results data set is named XMLVALIDATE and resides in the Work library.

Table 7.8 *Parameters for the crtdds_xmlvalidate.sas Macro*

Parameter	Required	Description
_cstLogLevel	Yes	Identifies the log level. Valid values are Info, Warning, Error, and Fatal Error. The default value is Info.
_cstResultsOverrideDS	Yes	Provides the opportunity to designate [LIBNAME.]member as the name of the Results data set. If this parameter is omitted (default setting), then the Results data set specified by the &_cstResultsDS global macro variable is used.

XML schema validation results are logged using four log level settings. These log levels refer to the XML-generated log, not the log that is generated by SAS.

Table 7.9 *Log Levels for the crtdds_xmlvalidate.sas Macro*

Log Level	Description
Info	Informational messages such as the system properties of the current Java environment, and progress messages. This is the default value.
Warning	Messages that indicate that there might be an issue with the CRT-DDS document or with the execution of the validation process.
Error	Messages that indicate that something in the define.xml document is invalid with respect to the normal XML schema for CRT-DDS. Or, a non-fatal error has occurred during processing.
Fatal Error	Messages that indicate that the XML document could not be processed at all. There are many causes, including, file system access errors, incorrect file paths, and malformed XML.

Each message that is generated during XML validation is associated with one of these levels. The level that you choose determines what other messages are generated. For example, if you choose **`warning`**, then all Warning messages and anything more severe, such as Error and Fatal Error messages, are generated. If you choose **`error`**, then only Error and Fatal Error messages are generated.

The following is an example of a call to the crtdds_xmlvalidate.sas macro:

```
%crtdds_xmlvalidate(_cstLogLevel=info,
                                    _cstResultsOverrideDS=work.xmlvalidate);
```

Validation of the SAS Representation: crtdds_validate Macro

The crtdds_validate.sas macro supports the first XML validation methodology outlined above. This method is based on SAS and validates the SAS representation of the XML-based standard.

In SAS Clinical Standards Toolkit, CDISC CRT-DDS validation uses the same types of metadata and the same workflow process that is common to validation of all data standards. SAS provides a set of validation checks for CDISC CRT-DDS that are designed to verify the metadata definitions and values of the 39 data sets that comprise the SAS representation of the CRT-DDS model. These checks were created by SAS. For more information about these checks, see Chapter 6, "Validation," on page 83 and Appendix A5, "CDISC CRT-DDS 1.0 Validation Checks," on page 335. Metadata about each check is provided in the Validation Master data set which can be found in `<global standards library directory>/standards/cdisc-crtdds-1.0-1.3/validation/control`.

The crtdds_validate.sas macro controls the validation workflow for CRT-DDS. As each check is processed from the run-time validation check data set, the check determines the source of the table and column metadata to use. The reference_tables and reference_columns data sets contain the metadata for the 39 data sets that comprise the SAS representation for CDISC CRT-DDS. Unless you make customizations or run-time modifications, the source metadata source_tables and source_columns data sets contain the same content as the reference metatadata reference_tables and reference_columns data sets.

If all 39 CRT-DDS tables contribute information to the define.xml file, then the validation process can run directly against the reference tables and columns data sets. In this case, the **Use source data** flag in the validation check data set needs to be set to N. However, most users will run validation against a subset of the 39 tables. In this case, a source_tables data set that contains the subset needs to be created from the reference_tables data set. And, a corresponding source_columns data set needs to be created from the reference_columns data set. The run-time validation check data set can contain all of the checks, and **Use source data** can be left set to Y, which is the default value.

There are no parameters for the crtdds_validate macro.

Sample Driver Program: validate_crtdds_data.sas

The validate_crtdds_data.sas driver program sets up the required environment variables and library references before a call is made to the crtdds_validate.sas macro.

For SAS 9.1.3, the driver program is located at:

```
!sasroot/../SASClinicalStandardsToolkitCRTDDS10/1.3/
sample/cdisc-crtdds-1.0/programs/validate_crtdds_data.sas
```

For SAS 9.2, the driver program is located at:

```
!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/
sample/cdisc-crtdds-1.0/programs/validate_crtdds_data.sas
```

The value for `!sasroot` is the location of your SAS installation directory.

The SASReferences Data Set

As a part of each SAS Clinical Standards Toolkit process setup, a valid SASReferences data set is required. It can be modified to point to study-specific files. For an explanation of the SASReferences data set, see Chapter 5, "SASReferences File," on page 69.

In the SASReferences data set, there are four input file references, one input library reference and, and one output file reference that are key to successful completion of the validation process. The following table lists these libraries and data sets, and they are discussed in separate sections. In the sample validate_crtdds_data.sas driver program, the following values are set for &studyRootPath and &studyOutputPath and are specific to a SAS release.

Note: The &studyRootPath and &studyOutputPath paths are the same for this driver. Two macro variables have been retained to maintain consistency across SAS Clinical Standards Toolkit driver programs.

SAS 9.1.3

```
&studyRootPath=!sasroot/../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

```
&studyOutputPath=!sasroot/../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

SAS 9.2

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

```
&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-
crtdds-1.0
```

Table 7.10 *Key Components of the SASReferences Data Set*

Input or Output	Metadata Type	LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	control	cntl_s	LIBNAME	&workpath	sasreferences.sas7bdat
Input	control	cntl_v	LIBNAME	&studyRootPath/control	validation_control.sas7bdat
Input	sourcemetadata	srcmeta	LIBNAME	&studyRootPath/metadata	source_tables.sas7bdat
Input	sourcemetadata	srcmeta	LIBNAME	&studyRootPath/metadata	source_columns.sas7bdat
Input	sourcedata	srcdata	LIBNAME	&studyRootPath/data	
Output	results	results	LIBNAME	&studyOutputPath/results	validation_results.sas7bdat

Process Inputs

The use of the cntl_s LIBNAME that points to the &workpath path illustrates a technique of documenting the derivation of the SASreferences data set in the SAS Work library. The driver program initiates the macro variable &workPath with the following statement:

```
%let workPath=%sysfunc(pathname(work));
```

In this case, the cntl_s LIBNAME points to the same directory as the Work LIBNAME. The second control record points to the validation_control.sas7bdat (run-time validation check) data set, and is accessed by the cntl_v LIBNAME statement. This LIBNAME is assigned to the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/control`** directory.

The sourcemetadata type references two metadata data sets that describe the table (source_tables) and column (source_columns) metadata for the 39 data sets that comprise the SAS representation of the CRT-DDS model. Both data sets are stored in the same library. In the SAS Clinical Standards Toolkit, this source metadata is read from the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/metadata`** directory. This location is represented in the driver program using the Srcmeta library name.

The sourcedata type is the library where the 39 data sets that comprise the SAS representation of the CRT-DDS model are stored. These are the data sets that are being validated. In the SAS Clinical Standards Toolkit, this library is read from the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/data`** directory. This location is represented in the driver program by the Srcdata library name.

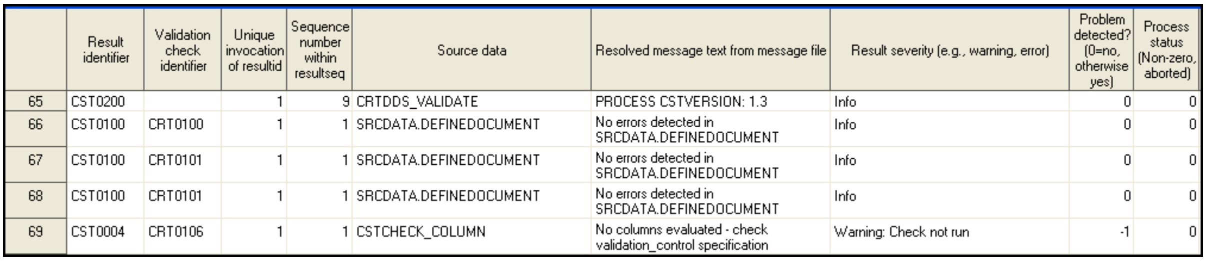
Process Outputs

For SAS Clinical Standards Toolkit validation processes, the only process outputs that are generated are the Validation Results and Validation Metrics data sets. These data sets are described in the following section.

Process Results

When the validate_crtdds_data.sas driver program finishes running, the validation_results.sas7bdat data set is created in the Results library. The Results data set contains informational, warning, and error messages that were generated by the validation program. Reporting of validation process metrics is supported, though it is not implemented for CDISC CRT-DDS validation.

Display 7.11 *Example of a CDISC CRT-DDS Results Data Set*



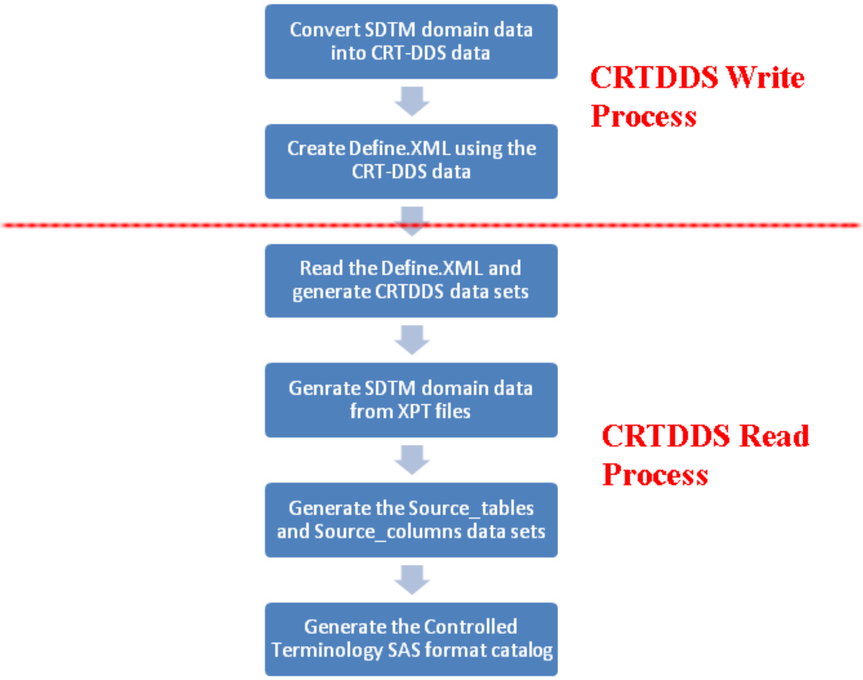
	Result identifier	Validation check identifier	Unique invocation of resultid	Sequence number within resultseq	Source data	Resolved message text from message file	Result severity (e.g., warning, error)	Problem detected? (0=no, otherwise yes)	Process status (Non-zero, aborted)
65	CST0200		1	9	CRTDDS_VALIDATE	PROCESS CSTVERSION: 1.3	Info	0	0
66	CST0100	CRT0100	1	1	SRCDATA.DEFINEDOCUMENT	No errors detected in SRCDATA.DEFINEDOCUMENT	Info	0	0
67	CST0100	CRT0101	1	1	SRCDATA.DEFINEDOCUMENT	No errors detected in SRCDATA.DEFINEDOCUMENT	Info	0	0
68	CST0100	CRT0101	1	1	SRCDATA.DEFINEDOCUMENT	No errors detected in SRCDATA.DEFINEDOCUMENT	Info	0	0
69	CST0004	CRT0106	1	1	CSTCHECK_COLUMN	No columns evaluated - check validation_control specification	Warning: Check not run	-1	0

Special Topic: A Round Trip Exercise Involving the CDISC SDTM and CDISC CRT-DDS Standards

The typical SAS Clinical Standards Toolkit workflow in support of the CDISC standards includes the definition and validation of SDTM submission data and the creation and validation of a define.xml file based on the SDTM domain data. This exercise illustrates how you can read a define.xml file to extract the data and metadata for the purposes of recreating the original source SDTM study. Recreating the original source study has value as a standalone exercise, either to extract a new SDTM study from a define.xml file or to create a new SDTM study using information in a define.xml file as a template.

As a round-trip exercise, this task validates the performance of the crtdds_write and crtdds_read SAS Clinical Standards Toolkit macros and allows a comparison of original and recreated SDTM metadata and data. The following display details the high-level workflow for this exercise.

Display 7.12 *Round Trip Process*


Convert SDTM domain data into CRT-DDS data
Create Define.XML using the CRT-DDS data
CRTDDS Write Process
Read the Define.XML and generate CRTDDS data sets
Genrate SDTM domain data from XPT files
Generate the Source_tables and Source_columns data sets
Generate the Controlled Terminology SAS format catalog
CRTDDS Read Process

The following steps describe the workflow in more detail. The first five steps describe the derivation of the CDISC CRT-DDS 1.0 define.xml file.

1. Access a study that contains valid CDISC SDTM data and metadata. This is a study that contains domain data (AE, DM, CO, and so on) and SAS Clinical Standards Toolkit metadata about that SDTM study, such as source_tables and source_columns. SAS Clinical Standards Toolkit also includes XSL style sheets, XML map files, and any metadata that is provided by SAS during the SAS Clinical Standards Toolkit installation.

2. Use the set of sample driver programs that are provided in the SAS Clinical Standards Toolkit to define the input and output files for each process task and to invoke the macros that support each standard-specific task. The driver programs are designed to run with the sample studies, but can be modified as needed. New custom drivers can also be created and used.

3. Submit the create_crtdds10_fromsdtm311.sas driver program to access the crtdds_sdtm311todefine10.sas macro, and create the 39 data sets that comprise the SAS representation of the CRT-DDS model. These 39 output data sets are written to the **!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/data** directory.

4. Validate the CRT-DDS data sets by submitting the validate_crtdds_data.sas driver program. This step is optional.

5. Create the define.xml file by submitting the create_crtdds_define.sas driver program. This driver program generates the define.xml file from the 39 CRT-DDS data sets that were created in step 3. It also calls the crtdds_xmlvalidate macro to validate the XML file structure. The define.xml file is written to the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/sourcexml`** directory.

   At this point, a valid define.xml file has been created from the SAS representation of the CRT-DDS model. In the next steps, the SDTM data and metadata are recreated using the XML read process.

6. Submit the create_sascrtdds_fromxml.sas driver program. This driver program reads the define.xml file created in step 5, and generates the SAS representation of the CRT-DDS model using the crtdds_read.sas macro. The data sets created in this step should match the data sets created in step 3. These data sets are written to the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/deriveddata`** directory. This driver program generates the source_tables and source_columns data sets in the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/derivedmetadata`** directory. By specifying new target folder locations (deriveddata and derivedmetadata), the data sets can be validated against the data sets that were created or referenced in step 3.

7. SDTM domain data sets are created based on a reachable set of SAS transport files that are specified in the define. xml file. Submit the create_sasdata_fromxpt.sas SDTM driver program. For SDTM 3.1.2, the program is in the **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata/programs`** directory. This driver program accesses the sdtmutil_createsasdatafromxpt.sas macro to generate the SDTM domain data sets from the SAS transport files. Creation of the SAS transport files is not performed by SAS Clinical Standards Toolkit. These files would have been produced as a prerequisite to the generation of the define.xml file as a part of the Electronic Common Document preparation process. The sdtmutil_createsasdatafromxpt.sas macro assumes that the SAS transport files are reachable from a folder relative to the location of the referenced define.xml file. In the create_sasdata_fromxpt.sas SDTM driver program, the XPT files are read from the **`!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.3/sample/cdisc-crtdds-1.0/transport`** directory. The generated data sets are written to the **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdiscsdtm-3.1.2/sascstdemodata/derived/data`** directory. At this point, the SDTM domain data sets should contain the same information as the original domain data sets that were accessed at the beginning of this process. By specifying a new target folder location, the SDTM data sets can be validated against those referenced in steps 1 and 3 above.

8. Source metadata that describes the SDTM domains and columns is derived using information contained in the CRT-DDS data sets derived in step 6. Submit the create_sourcemetadata.sas SDTM driver program. For SDTM 3.1.2, it is installed in the **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata/programs`** directory. In this exercise, this driver program calls the sdtmutil_createsrcmetafromcrtdds macro, which uses a library of SAS data sets that capture define.xml metadata (typically derived using the crtdds_read macro). The output of this step is a set of SDTM metadata in source_tables, source_columns, and source_study data sets. These data sets are written to the **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdiscsdtm-3.1.2/sascstdemodata/derived/metadata`** directory. At this point, the SDTM metadata should contain the same information as the original metadata that was accessed at the beginning of this process. By specifying

a new target folder location, the SDTM metadata data sets can be validated against those referenced in steps 1 and 3 above.

9. SAS formats that support SDTM controlled terminology are derived using information contained in the CRT-DDS data sets that were derived in step 6. Submit the create_formatsfromcrtdds.sas SDTM driver program. For SDTM 3.1.2, this program is installed in the **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdisc-sdtm-3.1.2/sascstdemodata/programs`** directory. The driver program accesses the sdtmutil_createformatsfromcrtdds.sas macro and generates the controlled terminology SAS formats catalog based on codelists specified in the define.xml file. The derived SAS format catalog is written to the **`!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.3/sample/cdiscsdtm-3.1.2/sascstdemodata/derived/formats`** directory. These formats should match those formats that were referenced by the SDTM columns at the beginning of this process. By specifying a new target folder location, the SAS format catalog can be validated against the catalog referenced in steps 1 and 3 above.

Note: **When running multiple driver programs:**

The SAS Clinical Standards Toolkit uses autocall macro libraries to contain and reference standard-specific code libraries. Once the autocall path is set, and one or more macros have been used in an autocall macro library, deallocation or reallocation of the autocall file reference cannot occur unless the autocall path is reset to exclude the specific file reference.

This becomes a problem with repeated calls to %cstutil_processsetup() or %cstutil_allocatesasreferences in the same SAS session. You might receive SAS errors, such as the following one, unless you submit some specific SAS code:

```
ERROR - At least one file associated with fileref SDTMAUTO is still in use.
ERROR - Error in the FILENAME statement.
```

If you call %cstutil_processsetup() or %cstutil_allocatesasreferences more than once in the same SAS session, which typically uses %let _cstReallocateSASRefs=1 to tell the SAS Clinical Standards Toolkit to attempt reallocation, use the following code between each code submission:

```
%let _cstReallocateSASRefs=1;
%include "&_cstGRoot/standards/cst-framework-1.3/programs/resetautocallpath.sas";
```

In the driver programs provided with the SAS Clinical Standards Toolkit, the previous code is commented so that it does not get submitted during run time.

Once the round trip exercise is complete, data derived from the process should match the original data. There might be some metadata collected that does not match exactly (particularly any date and time fields that collect real-time information). Differences can be detected by doing a PROC COMPARE with any of the derived data and metadata data sets against the original data and metadata data sets.

Chapter 8
Reporting

Sample Reports

Overview

To show how SAS Clinical Standards Toolkit metadata and results can be summarized in a report format, several sample reports are available with the SAS Clinical Standards Toolkit. These reports are offered as templates that can be modified to facilitate data review. The report templates are PROC REPORT implementations that use ODS to generate report output in a variety of formats supported by ODS. Three sample reports are provided:

- Report 1: This report is applicable to most SAS Clinical Standards Toolkit processes. It itemizes records that are written to the Results data by the process. In the case of validation processes, this report itemizes Results data set records by validation check.
- Report 2: This report is specific to SAS Clinical Standards Toolkit validation processes for standards that have the concept of source data domains (for example, CDISC SDTM and CDISC ADaM). Results are summarized by domain.
- Report 3: This report is specific to SAS Clinical Standards Toolkit validation functionality that summarizes all available metadata about validation checks for a supported standard. This report offers a multi-panel or one-page-per-check presentation format.

Process Results Reporting

Reports 1 and 2 have multiple sections or panels. Each section can be optionally generated. Several sections are common to each report, including a report summary, a listing of key process inputs and outputs as defined in the SASReferences data set, a summary of validation metrics, and a general process messaging panel.

A sample driver program is provided to define the SAS Clinical Standards Toolkit environment and to call the primary task framework macro (%cstutil_createreport). The following excerpt from the driver program header provides a brief overview:

```
cst_report.sas

Sample driver program to perform a primary Toolkit task, in this case,
reporting process results. This code performs any needed setup and data
management tasks, followed by one or more calls to the %cstutil_createreport()
macro to generate report output.

Two options for invoking this routine are addressed in these scenarios:
   (1) This code is run as a natural continuation of a CST process, in
       the same SAS session, with all required files available. The working
       assumption is that the SASReferences data set (referenced by the
       _cstSASRefs macro) exists and contains information on all input files
       required for reporting.
  (2) This code is run in another SAS session with no CST setup
       established, but the user has a CST Results data set and, therefore, can
       derive the location of the SASReferences data set that can
       provide the full CST setup needed to run the reports.

Assumptions:
  To generate all panels for both types of reports, the following metadata
  is expected:
        - the SASReferences data set must exist and be identified in the
            call to cstutil_processsetup if it is not work.sasreferences.
        - a Results data set.
        - a (validation-specific) Metrics data set.
        - the (validation-specific) run-time Control data set itemizing the
            validation checks requested.
        - access to the (validation-specific) check Messages data set.
```

The reporting as implemented in the SAS Clinical Standards Toolkit attempts to address the following two scenarios described in the driver module header above:

1. Some SAS Clinical Standards Toolkit task (such as validation against a reference standard) has been completed. The Results data set has been created. And, in the same SAS session (or batch job stream), you want to generate one or both reports. In this scenario, the reporting process uses the SASReferences data set defined by the global macro variable _cstSASRefs that was used by the previous process. The Results data set to be summarized in the report is the data set that was previously created and perhaps persisted to a location other than the SAS Work library. (Whether the data set was persisted was specified in the SASReferences data set.) Other files required by the report are identified in Table 8.1 on page 197.

   Best Practice Recommendation: The cleanup macro, %cstutil_cleanupcstsession, should not be called between primary tasks in a SAS Clinical Standards Toolkit SAS session (such as between validation and reporting). This keeps required files, macro variables, autocall paths, and so on, available for the reporting code.

2. The Results data set that was created in some prior SAS Clinical Standards Toolkit session is available. You want to generate one or both reports. The SAS Clinical Standards Toolkit processes add informational records to the Results data set, documenting the process itself. For example, a SAS Clinical Standards Toolkit CDISC SDTM validation process writes records to the Results data set that contains the following sample message text:

```
Message
PROCESS STANDARD: CDISC-SDTM
PROCESS STANDARDVERSION: 3.1.1
PROCESS DRIVER: SDTM_VALIDATE
PROCESS DATE: 2010-01-25T11:56:17
PROCESS TYPE: VALIDATION
PROCESS SASREFERENCES:
      !sasroot/../SASClinicalStandardsToolkitSDTM311/
      9.1.3/sample/cdisc-sdtm-
      3.1.1/SASDemo/control/sasreferences.sas7bdat
```

From this information, a reporting process can attempt to find and open the referenced SASReferences data set to derive information for some or all of the report sections.

Warning: There are obvious limits to how useful any SAS Clinical Standards Toolkit Results data set can be in rebuilding a session for reporting purposes. For example, if the SASReferences data set was built in the Work library in a previous session, then it will not be available and the report process fails. Similarly, if the SASReferences data set references library and file paths using a macro variable prefix (for example, &_cstGRoot or &studyRootPath), and those macro variables are not set or point to a different root path than the original process, then the report process might fail or yield unpredictable results. This scenario or technique is most appropriate for sites that adopt a consistent means of building and populating SASReferences data sets.

Table 8.1 *Metadata Sources for Reporting*

Data or Metadata Source	**Scenario 1: Continuation of an Active SAS Session**	**Scenario 2: Using a Results Data Set from a Previous SAS Session**
SASReferences	&_cstSASRefs used by the prior task that generated the Results data set.	The Results data set record containing the message PROCESS SASREFERENCES attempts to use the referenced file. &_cstSASRefs is set to this file.
Results	Precedence: 1. The data set referenced in &_cstSASRefs with type=results and subtype is either results or validationresults. 2. The data set referenced by &_cstResultsDS.	As provided in the cst_report.sas driver program _cstRptResultsDS macro variable.
Metrics	Precedence: 1. The data set referenced in &_cstSASRefs with type=results and subtype is either metrics or validationmetrics. 2. The data set referenced by &_cstMetricsDS.	The data set referenced in &_cstSASRefs with type=results and subtype is either metrics or validationmetrics.
Validation_Control	The data set referenced in &_cstSASRefs with type=control and subtype=validation.	The data set referenced in &_cstSASRefs with type=control and subtype=validation.

Data or Metadata Source	Scenario 1: Continuation of an Active SAS Session	Scenario 2: Using a Results Data Set from a Previous SAS Session
Messages	&_cstMessages used by the prior task.	&_cstMessages built by a call to %cstutil_allocatesasreferences.

Note: In the SAS Clinical Standards Toolkit 1.3, you are able to define report output locations in the SASReferences data set. These locations can be defined with type=report in SASReferences. They can be further specified in the framework Standardlookup data set. For more information, see Chapter 2, "Framework," on page 5.

The following code was excerpted from the cst_report.sas driver module and performs the setup tasks that are specific to reporting:

```
* Initialize macro variables used for this task *;
%let _cstRptControl=;
%let _cstRptLib=;
%let _cstRptMetricsDS=;
%let _cstRptOutputFile=&studyOutputPath/results/cstreport.pdf;
%let _cstRptResultsDS=;
%let _cstSetupSrc=SASREFERENCES;
%let _cstStandard=CDISC-SDTM;
%let _cstStandardVersion=3.1.2;

%cstutil_processsetup(_cstSASReferencesLocation=&studyrootpath/control);
%cstutil_reportsetup(_cstRptType=Results);
```

In this piece of code:

- The report output is specified in the _cstRptOutputFile variable and can be found in **&studyOutputPath/results/cstreport.pdf**. The studyOutputPath variable was previously defined to point to a folder with write permissions.
- The _cstSetupSrc=SASREFERENCES statement tells the process that a SASReferences data set is available and should be used to complete setup tasks.
- The call to the %cstutil_processsetup macro provides the location of the SASReferences data set using the previously defined &StudyRootPath variable.
- The call to the %cstutil_reportsetup macro completes the setup steps that are required to generate report 1, itemizing results data set records by validation check.

An alternative setup to support Scenario 2, as described on page 196, would include the following code excerpts:

```
%let _cstSetupSrc=RESULTS;
%cstutil_processsetup();
%let _cstRptResultsDS=work.validation_results;
%cstutil_reportsetup(_cstRptType=Results);
```

In this piece of code:

- The _cstSetupSrc=RESULTS statement tells the process that a SAS Clinical Standards Toolkit process results data set should be used as the initial metadata source to complete the setup tasks.

- The call to the %cstutil_processsetup macro without parameters, and with _cstSetupSrc=RESULTS, defers the remaining setup steps to the %cstutil_reportsetup macro.
- The call to the %cstutil_reportsetup macro completes the setup steps required to generate teport 1, itemizing work.validation_results records.

As the final step, the reporting driver program makes one or more calls to the utility reporting macro. At a minimum (using default parameter values), a simple macro call to create report 2 might include the following:

```
%cstutil_createreport(_cstsasreferencesdset=&_cstSASRefs,_cstreportbydomain=Y,
_cstreportoutput=&studyrootpath/results/cstchecktablereport.pdf);
```

The following table describes all supported parameters in the sample %cstutil_createreport macro:

Table 8.2 Supported Parameters for the %cstutil_createreport Macro

Parameter	Description
_cstsasreferencesdset	The libref.dataset of SASReferences data set used for a specific process. This parameter is optional. If it is specified, then _cstresultsdset and _cstmetricsdset parameters are ignored. Either _cstsasreferencesdset or _cstresultsdset must be provided.
_cstresultsdset	The libref.dataset of SAS Clinical Standards Toolkit process Results data set. This parameter is optional. Either _cstsasreferencesdset or _cstresultsdset must be provided. This parameter is ignored if _cstsasreferencesdset is specified.
_cstmetricsdset	The libref.dataset of SAS Clinical Standards Toolkit process Metrics data set. This parameter is optional. This parameter is ignored if _cstsasreferencesdset is specified.
_cstreporterrorsonly	If N (default), then this parameter reports all records in the Results data set, including information and non-error results. If Y, then this parameter reports only records in error (where the Results data set field results.resultflag=1).
_cstreportobs	If null (default), then this parameter reports all records in error (where results.resultflag=1) in the Results data set. Otherwise, set this parameter to any integer value > 0, signifying the number of records to print per checkid (where results.checkid is non-null). If _cstreportobs > 0 excludes any records, then a footnote is printed, noting that not all records were printed.

Parameter	Description
_cstreportbytable	If N (default), then this parameter does not report results by table (that is, run report 1). If Y, then this parameter reports results by table (that is, run report 2).
_csttablechecksdset	Report 2 parameter. A data set that provides a list of tables for each check. Using this parameter assumes that this data set has been built before running this report. For more information, see "Supplemental Validation Check Metadata: Domains by Check" on page 98. This parameter is optional. If this parameter is not used, then the data set is created.
_csttablecheckscode	Report 2 parameter. The code module (macro) to build _csttablechecksdset if it does not exist, or is not passed as a parameter. This parameter is required only if _cstreportbytable=Y and _csttablechecksdset is not provided.
_cstkeeptablechecklist	Report 2 parameter. The value is Y or N (default). If running report 2, then keep the derived list of tables (_csttablechecklist) to reuse in subsequent report requests. Building this file takes awhile.
_csttablesubset	Report 2 parameter. This parameter is optional. It produces a report based on a specific table, indicated by libref.data set. If the value is blank or the keyword _ALL_ is specified, then all tables are included in the report. This parameter is ignored if _cstreportbytable=N.
_cstreportoutput	The path and filename where report output is to be written. File types HTML, RTF, and PDF are supported. This parameter is required.
_cstsummaryReport	The value is Y (default) or N. If set to Y, then generate the report summary panel.
_cstioReport	The value is Y (default) or N. If set to Y, then generate the process inputs and outputs panel.
_cstmetricsReport	The value is Y (default) or N. If set to Y, then generate the process metrics panel. This parameter should be set to N for any non-validation reports and cases where metrics are not generated.
_cstgeneralResultsReport	The value is Y (default) or N. If set to Y, then generate the general process reporting panel.
_cstcheckIdResultsReport	The value is Y (default) or N. If set to Y, then generate the process results panel.

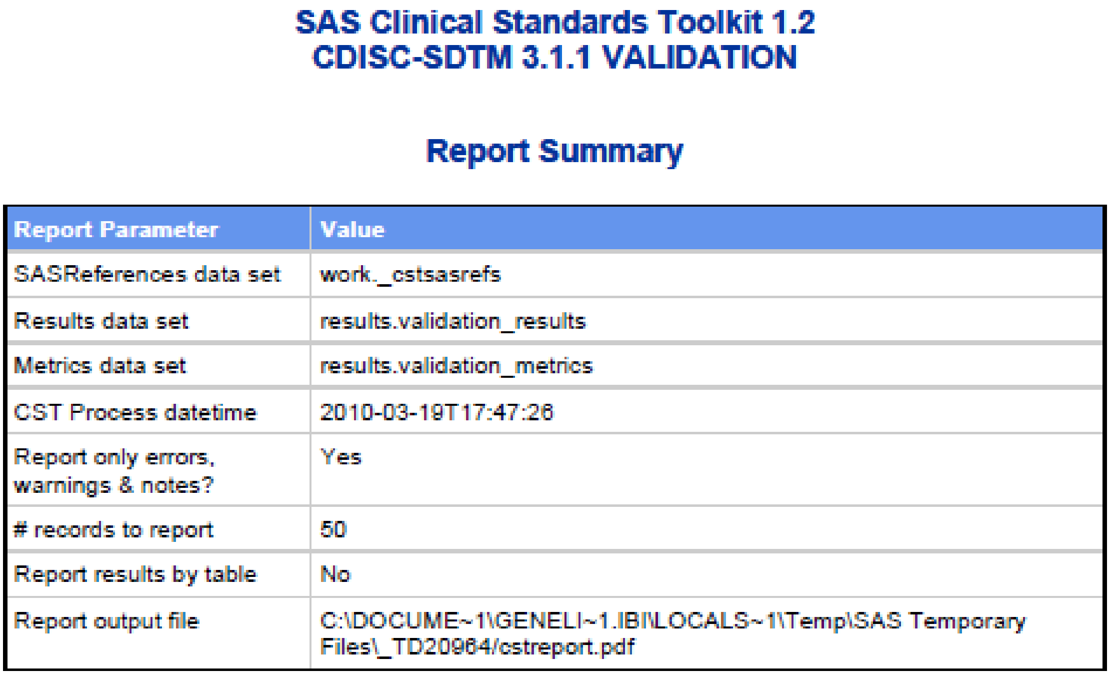
A more complete example of the %cstutil_createreport reporting macro includes the following macro call:

```
%cstutil_createreport(
   _cstsasreferencesdset=&_cstSASRefs,
   _cstresultsdset=&_cstRptResultsDS,
   _cstmetricsdset=&_cstRptMetricsDS,
   _cstreportbytable=N,
   _cstreporterrorsonly=Y,
   _cstreportobs=50,
   _cstreportoutput=%nrbquote(&_cstRptOutputFile),
   _cstsummaryReport=Y,
   _cstioReport=Y,
   _cstmetricsReport=Y,
   _cstgeneralResultsReport=Y,
   _cstcheckIdResultsReport=Y);
```

Interpretation of this request, based on the parameter descriptions in Table 9.2, produces a (validation) results listing that contains all five report panels and includes only the first 50 errors that are reported for each validation check.

The following displays show report content. The displays apply to report 1 (by checkid) unless otherwise indicated.

Display 8.1 *Report Summary*

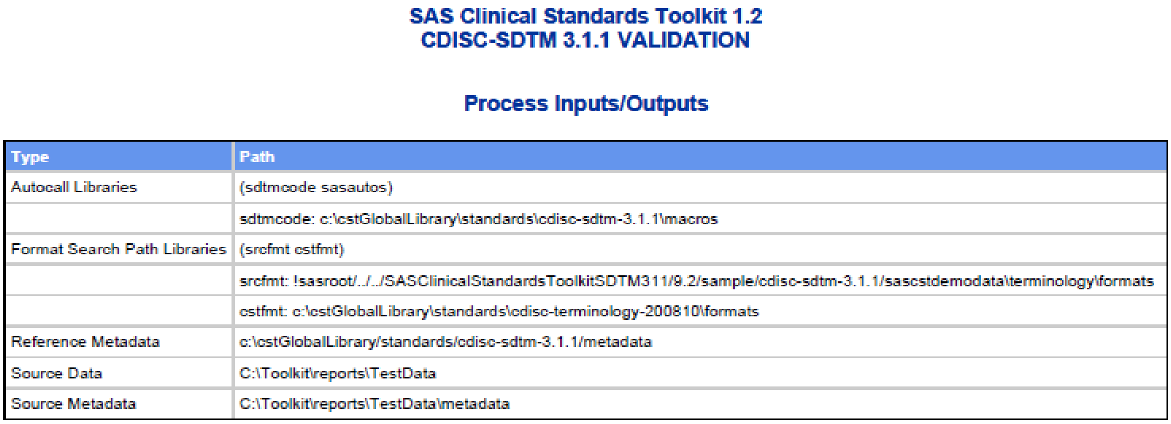


SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Report Summary

Report Parameter	Value
SASReferences data set	work._cstsasrefs
Results data set	results.validation_results
Metrics data set	results.validation_metrics
CST Process datetime	2010-03-19T17:47:26
Report only errors, warnings & notes?	Yes
# records to report	50
Report results by table	No
Report output file	C:\DOCUME~1\GENELI~1.IBI\LOCALS~1\Temp\SAS Temporary Files_TD20964/cstreport.pdf

Display 8.2 *Process Inputs and Outputs*

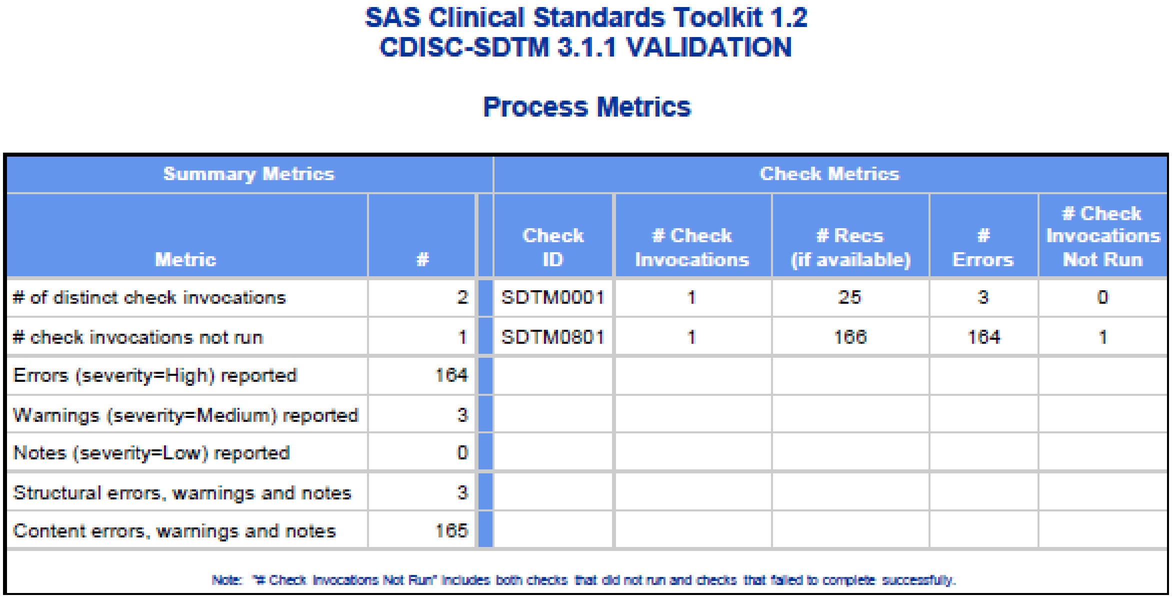


SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Inputs/Outputs

Type	Path
Autocall Libraries	(sdtmcode sasautos)
	sdtmcode: c:\cstGlobalLibrary\standards\cdisc-sdtm-3.1.1\macros
Format Search Path Libraries	(srcfmt cstfmt)
	srcfmt: !sasroot/../../SASClinicalStandardsToolkitSDTM311/9.2/sample/cdisc-sdtm-3.1.1/sascstdemodata\terminology\formats
	cstfmt: c:\cstGlobalLibrary\standards\cdisc-terminology-200810\formats
Reference Metadata	c:\cstGlobalLibrary/standards/cdisc-sdtm-3.1.1/metadata
Source Data	C:\Toolkit\reports\TestData
Source Metadata	C:\Toolkit\reports\TestData\metadata

Display 8.3 *Process Metrics (Report 1)*



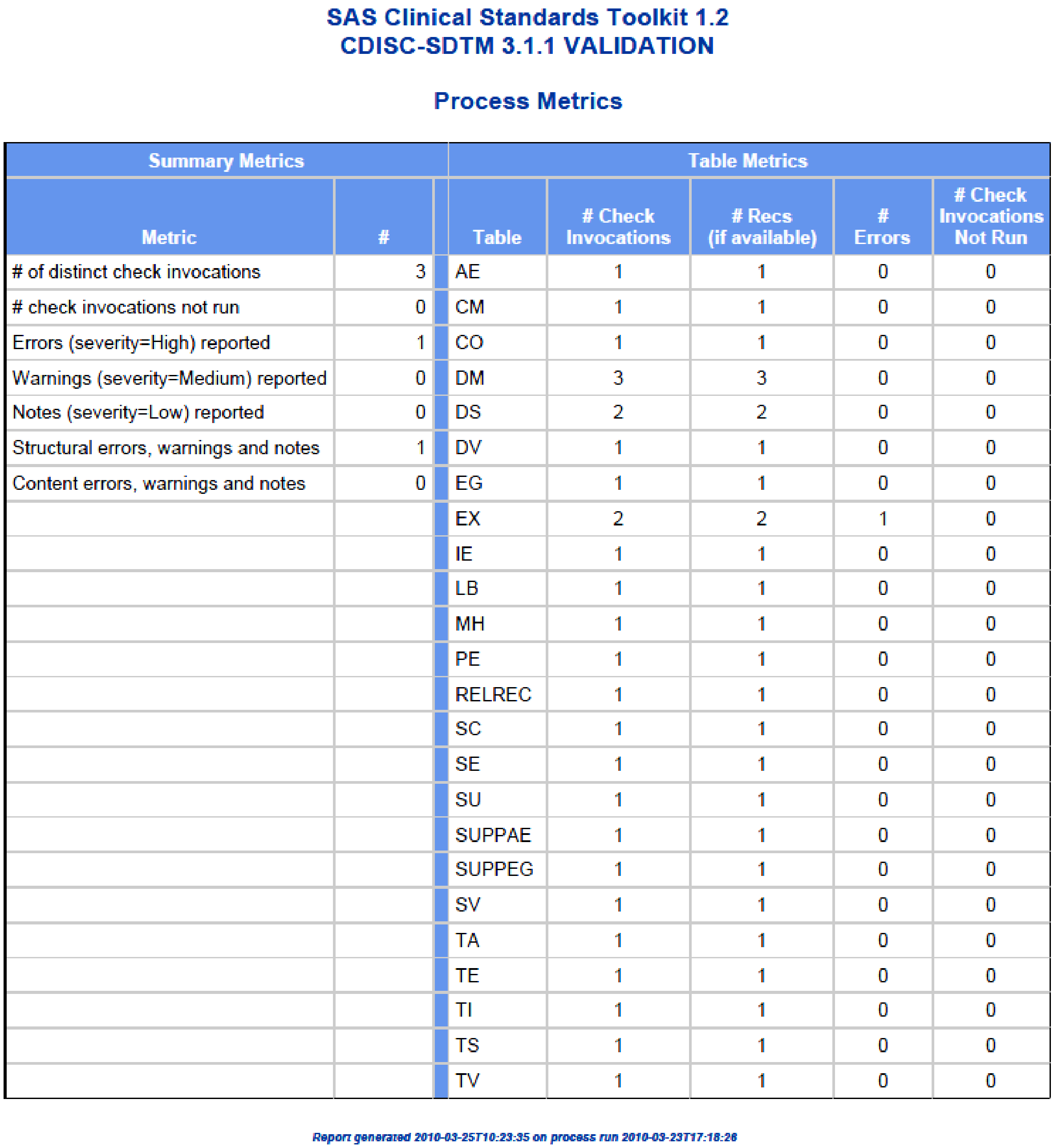
SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Metrics

Summary Metrics		Check Metrics				
Metric	#	Check ID	# Check Invocations	# Recs (if available)	# Errors	# Check Invocations Not Run
# of distinct check invocations	2	SDTM0001	1	25	3	0
# check invocations not run	1	SDTM0801	1	166	164	1
Errors (severity=High) reported	164					
Warnings (severity=Medium) reported	3					
Notes (severity=Low) reported	0					
Structural errors, warnings and notes	3					
Content errors, warnings and notes	165					

Note: "# Check Invocations Not Run" includes both checks that did not run and checks that failed to complete successfully.

Display 8.4 *Process Metrics by Domain (Report 2)*



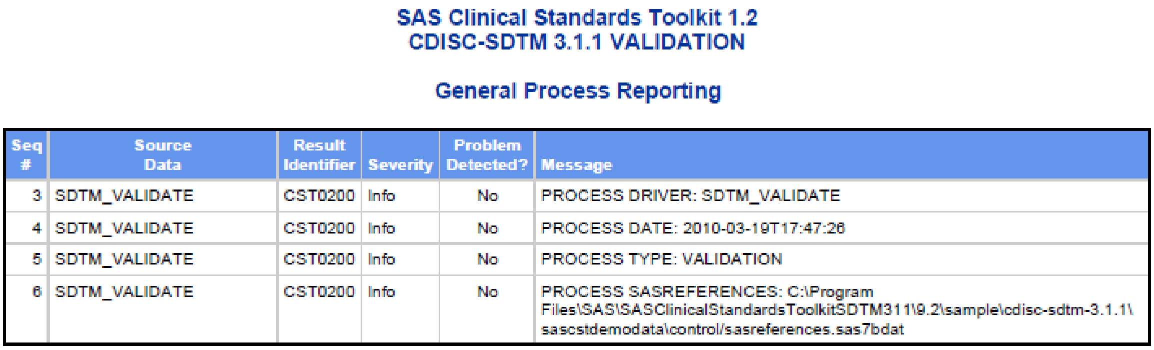
SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Metrics

Summary Metrics		Table Metrics				
Metric	#	Table	# Check Invocations	# Recs (if available)	# Errors	# Check Invocations Not Run
# of distinct check invocations	3	AE	1	1	0	0
# check invocations not run	0	CM	1	1	0	0
Errors (severity=High) reported	1	CO	1	1	0	0
Warnings (severity=Medium) reported	0	DM	3	3	0	0
Notes (severity=Low) reported	0	DS	2	2	0	0
Structural errors, warnings and notes	1	DV	1	1	0	0
Content errors, warnings and notes	0	EG	1	1	0	0
		EX	2	2	1	0
		IE	1	1	0	0
		LB	1	1	0	0
		MH	1	1	0	0
		PE	1	1	0	0
		RELREC	1	1	0	0
		SC	1	1	0	0
		SE	1	1	0	0
		SU	1	1	0	0
		SUPPAE	1	1	0	0
		SUPPEG	1	1	0	0
		SV	1	1	0	0
		TA	1	1	0	0
		TE	1	1	0	0
		TI	1	1	0	0
		TS	1	1	0	0
		TV	1	1	0	0

Report generated 2010-03-25T10:23:35 on process run 2010-03-23T17:18:26

Display 8.5 *General Process Reporting*

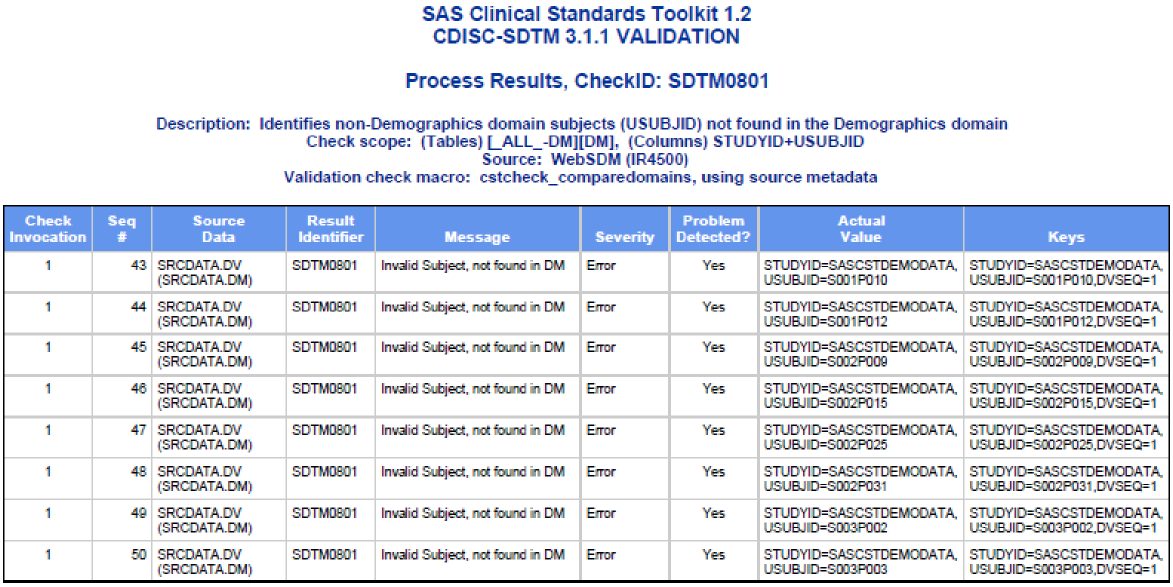


SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

General Process Reporting

Seq #	Source Data	Result Identifier	Severity	Problem Detected?	Message
3	SDTM_VALIDATE	CST0200	Info	No	PROCESS DRIVER: SDTM_VALIDATE
4	SDTM_VALIDATE	CST0200	Info	No	PROCESS DATE: 2010-03-19T17:47:26
5	SDTM_VALIDATE	CST0200	Info	No	PROCESS TYPE: VALIDATION
6	SDTM_VALIDATE	CST0200	Info	No	PROCESS SASREFERENCES: C:\Program Files\SAS\SASClinicalStandardsToolkitSDTM311\9.2\sample\cdisc-sdtm-3.1.1\sascstdemodata\control/sasreferences.sas7bdat

Display 8.6 *Validation Results by CheckID (Report 1)*



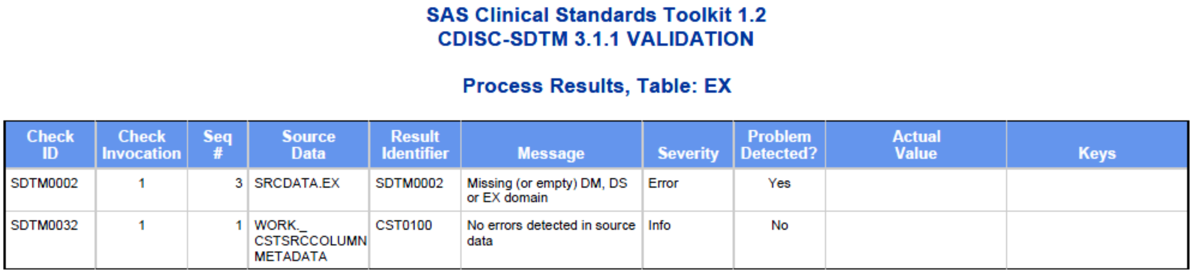
SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Results, CheckID: SDTM0801

Description: Identifies non-Demographics domain subjects (USUBJID) not found in the Demographics domain
Check scope: (Tables) [_ALL_-DM][DM], (Columns) STUDYID+USUBJID
Source: WebSDM (IR4500)
Validation check macro: cstcheck_comparedomains, using source metadata

Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
1	43	SRCDATA.DV (SRCDATA.DM)	SDTM0801	Invalid Subject, not found in DM	Error	Yes	STUDYID=SASCSTDEMODATA, USUBJID=S001P010	STUDYID=SASCSTDEMODATA, USUBJID=S001P010,DVSEQ=1
1	44	SRCDATA.DV (SRCDATA.DM)	SDTM0801	Invalid Subject, not found in DM	Error	Yes	STUDYID=SASCSTDEMODATA, USUBJID=S001P012	STUDYID=SASCSTDEMODATA, USUBJID=S001P012,DVSEQ=1
1	45	SRCDATA.DV (SRCDATA.DM)	SDTM0801	Invalid Subject, not found in DM	Error	Yes	STUDYID=SASCSTDEMODATA, USUBJID=S002P009	STUDYID=SASCSTDEMODATA, USUBJID=S002P009,DVSEQ=1
1	46	SRCDATA.DV (SRCDATA.DM)	SDTM0801	Invalid Subject, not found in DM	Error	Yes	STUDYID=SASCSTDEMODATA, USUBJID=S002P015	STUDYID=SASCSTDEMODATA, USUBJID=S002P015,DVSEQ=1
1	47	SRCDATA.DV (SRCDATA.DM)	SDTM0801	Invalid Subject, not found in DM	Error	Yes	STUDYID=SASCSTDEMODATA, USUBJID=S002P025	STUDYID=SASCSTDEMODATA, USUBJID=S002P025,DVSEQ=1
1	48	SRCDATA.DV (SRCDATA.DM)	SDTM0801	Invalid Subject, not found in DM	Error	Yes	STUDYID=SASCSTDEMODATA, USUBJID=S002P031	STUDYID=SASCSTDEMODATA, USUBJID=S002P031,DVSEQ=1
1	49	SRCDATA.DV (SRCDATA.DM)	SDTM0801	Invalid Subject, not found in DM	Error	Yes	STUDYID=SASCSTDEMODATA, USUBJID=S003P002	STUDYID=SASCSTDEMODATA, USUBJID=S003P002,DVSEQ=1
1	50	SRCDATA.DV (SRCDATA.DM)	SDTM0801	Invalid Subject, not found in DM	Error	Yes	STUDYID=SASCSTDEMODATA, USUBJID=S003P003	STUDYID=SASCSTDEMODATA, USUBJID=S003P003,DVSEQ=1

Display 8.7 *Validation Results by Domain (Report 2)*



SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Results, Table: EX

Check ID	Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
SDTM0002	1	3	SRCDATA.EX	SDTM0002	Missing (or empty) DM, DS or EX domain	Error	Yes		
SDTM0032	1	1	WORK._CSTSRCCOLUMNMETADATA	CST0100	No errors detected in source data	Info	No		

Validation Check Metadata Reporting

Report 3 offers the complete set of metadata about each validation check that is available in the SAS Clinical Standards Toolkit. The report can be printed in a multi-panel or one-page-per-check presentation format.

A sample driver program is provided to define the SAS Clinical Standards Toolkit environment and to call the primary task framework macro (%cstutil_createmetadatareport). The following excerpt from the driver program header provides a brief overview:

```
cst_metadatareport.sas

Sample driver program to perform the reporting of validation check metadata.
This code performs any needed setup and data management tasks, followed by
one or more calls to the %cstutil_createmetadatareport() macro to generate
report output.

Two options for invoking this routine are addressed in these scenarios:
  (1) This code is run as a natural continuation of a CST process, in
       the same SAS session, with all required files available. The working
       assumption is that the SASReferences data set (referenced by the
       _cstSASRefs macro) exists and
       contains information on all input files required for reporting.
  (2) This code is run in another SAS session with no CST setup
       established. In this case, the user assumes responsibility for
       defining all librefs and macro variables needed to run the reports,
       although defaults are set.

Assumptions:
(1) SASReferences is not required for this task. If it is found, it will be used.
     If it is not found, default libraries and macro variables are set and may be
     overridden by the user.
(2) The user of this code may override any cstutil_createmetadatareport
     parameter values.
(3) Only the cstutil_createmetadatareport &_cstRptControl and &_cstMessages
     parameters are required.
(4) If the _cststdrefds parameter is not set, the associated panel cannot be
     generated.
(5) By default, a PDF report format is assumed. This may be overridden.
(6) Report output is written to cstcheckmetadatareport.pdf in the SAS
     Work library unless another location is specified in SASReferences or
     in the setup code below.
(7) The report macro cstutil_createmetadatareport only produces panel 1
     (Check Overview) unless any of the last 3 parameters are set to Y.
```

Report setup is similar to reporting on process results. The only key difference is that the call to the %cstutil_reportsetup macro passes a different parameter value to request check metadata reporting:

```
%cstutil_reportsetup(_cstRptType=Metadata);
```

To generate the metadata report, the reporting driver program makes one or more calls to the utility reporting macro. At a minimum (using default parameter values), a simple macro call to create report 3 might include the following:

```
%cstutil_CreateMetadataReport(
                _cstValidationDS=&_cstRptControl
               ,_cstMessagesDS=&_cstMessages
               ,_cstReportOutput=%bquote(&_cstRptOutput)
               );
```

The following table describes all supported parameters in the sample %cstutil_createmetadatareport macro:

Table 8.3 Supported Parameters for the %cstutil_createmetadatareport Macro

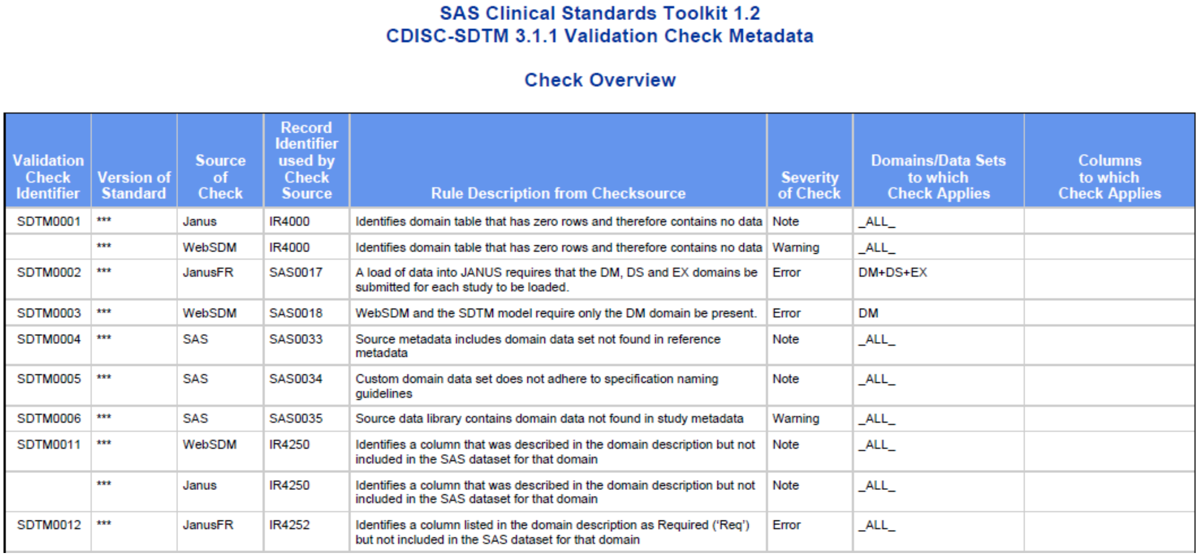
Parameter	Description
_cstStandardTitle	This parameter is optional. Title that defines the title2 statement.
_cstValidationDS	This parameter is required. The validation data set that is used by a SAS Clinical Standards Toolkit process. This is Validation Master, Validation Control, or a derivative as specified by the user.
_cstValidationDSWhClause	Optional WHERE clause applied to _cstValidationDS.
_cstMessagesDS	This parameter is required. The Messages data set used by a SAS Clinical Standards Toolkit process.
_cstStdRefDS	The Validation StdRef data set created for a SAS Clinical Standards Toolkit standard. This file is required if _cstStdRefReport=Y.
_cstReportOutput	This parameter is required. The path and filename where the report output is to be written. File types HTML, RTF, and PDF are supported.
_cstCheckMDReport	Specifies whether panel 2 additional check details is run. The default value is N.
_cstMessageReport	Specifies whether panel 3 message details is run. The default value is N.
_cstStdRefReport	Specifies whether panel 4 reference information is run. The default value is N.
_cstRecordView	If the value is Y, then all available check metadata is generated, by check, in a single listing. Either this listing, or the multi-panel report can be generated in a single invocation of this macro, but not both. The default value is N.

A more complete example of the %cstutil_createmetadatareport reporting macro includes the following macro call:

```
%cstutil_createmetadatareport(
   _cststandardtitle=%str(CDISC-SDTM 3.1.1 Validation Check Metadata),
   _cstvalidationds=refcntl.validation_master,
   _cstvalidationdswhclause=,
   _cstmessagesds=&_cstMessages,
   _cststdrefds=refcntl.validation_stdref,
   _cstreportoutput=%nrbquote(&studyOutputPath/results/cstcheckmetadatareport.pdf),
   _cstcheckmdreport=Y,
   _cstmessagereport=Y,
   _cststdrefreport=Y,
   _cstrecordview=N);
```

Interpretation of this request, based on the parameter descriptions in Table 9.3, produces a validation check metadata report (cstcheckmetadatareport.pdf) that contains all four report sections for the CDISC-SDTM 3.1.1 validation checks.

Display 8.8 *Check Overview*

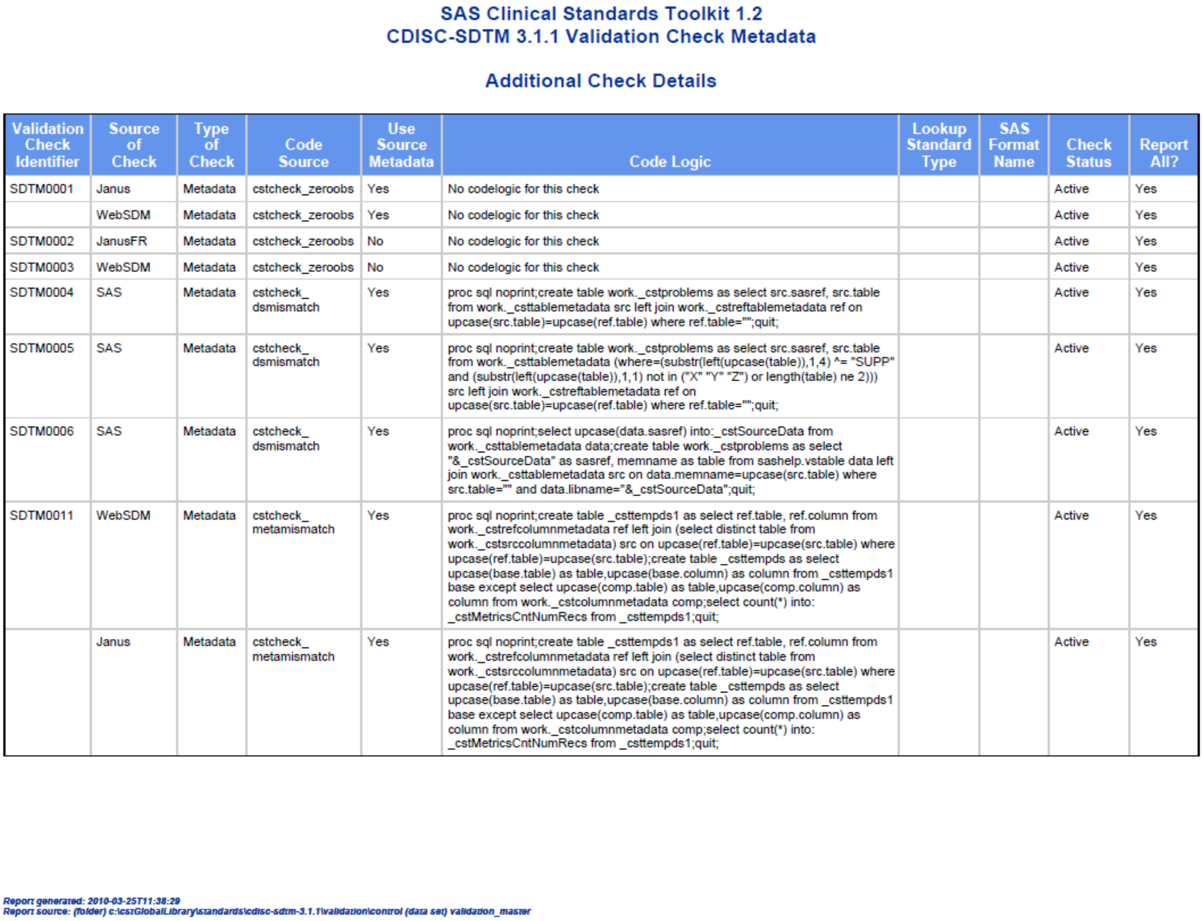


SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 Validation Check Metadata

Check Overview

Validation Check Identifier	Version of Standard	Source of Check	Record Identifier used by Check Source	Rule Description from Checksource	Severity of Check	Domains/Data Sets to which Check Applies	Columns to which Check Applies
SDTM0001	***	Janus	IR4000	Identifies domain table that has zero rows and therefore contains no data	Note	_ALL_	
	***	WebSDM	IR4000	Identifies domain table that has zero rows and therefore contains no data	Warning	_ALL_	
SDTM0002	***	JanusFR	SAS0017	A load of data into JANUS requires that the DM, DS and EX domains be submitted for each study to be loaded.	Error	DM+DS+EX	
SDTM0003	***	WebSDM	SAS0018	WebSDM and the SDTM model require only the DM domain be present.	Error	DM	
SDTM0004	***	SAS	SAS0033	Source metadata includes domain data set not found in reference metadata	Note	_ALL_	
SDTM0005	***	SAS	SAS0034	Custom domain data set does not adhere to specification naming guidelines	Note	_ALL_	
SDTM0006	***	SAS	SAS0035	Source data library contains domain data not found in study metadata	Warning	_ALL_	
SDTM0011	***	WebSDM	IR4250	Identifies a column that was described in the domain description but not included in the SAS dataset for that domain	Note	_ALL_	
	***	Janus	IR4250	Identifies a column that was described in the domain description but not included in the SAS dataset for that domain	Note	_ALL_	
SDTM0012	***	JanusFR	IR4252	Identifies a column listed in the domain description as Required ('Req') but not included in the SAS dataset for that domain	Error	_ALL_	

Display 8.9 *Additional Check Details (Panel 2) [_cstCheckMDReport=Y]*



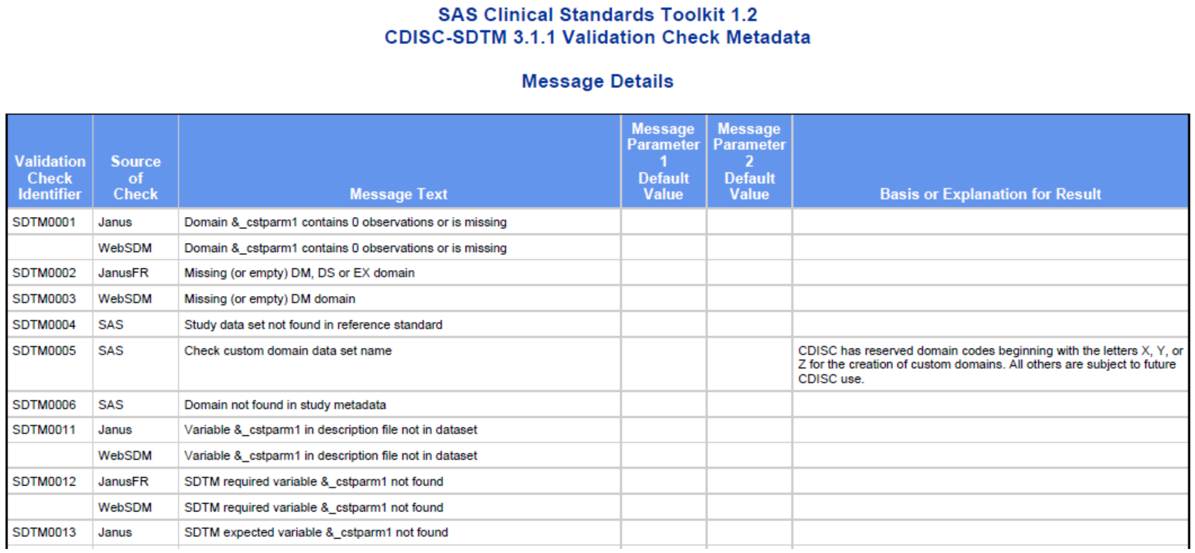
SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 Validation Check Metadata

Additional Check Details

Validation Check Identifier	Source of Check	Type of Check	Code Source	Use Source Metadata	Code Logic	Lookup Standard Type	SAS Format Name	Check Status	Report All?
SDTM0001	Janus	Metadata	cstcheck_zeroobs	Yes	No codelogic for this check			Active	Yes
	WebSDM	Metadata	cstcheck_zeroobs	Yes	No codelogic for this check			Active	Yes
SDTM0002	JanusFR	Metadata	cstcheck_zeroobs	No	No codelogic for this check			Active	Yes
SDTM0003	WebSDM	Metadata	cstcheck_zeroobs	No	No codelogic for this check			Active	Yes
SDTM0004	SAS	Metadata	cstcheck_ dsmismatch	Yes	proc sql noprint;create table work._cstproblems as select src.sasref, src.table from work._csttablemetadata src left join work._cstreftablemetadata ref on upcase(src.table)=upcase(ref.table) where ref.table="";quit;			Active	Yes
SDTM0005	SAS	Metadata	cstcheck_ dsmismatch	Yes	proc sql noprint;create table work._cstproblems as select src.sasref, src.table from work._csttablemetadata (where=(substr(left(upcase(table)),1,4) ^= "SUPP" and (substr(left(upcase(table)),1,1) not in ("X" "Y" "Z") or length(table) ne 2))) src left join work._cstreftablemetadata ref on upcase(src.table)=upcase(ref.table) where ref.table="";quit;			Active	Yes
SDTM0006	SAS	Metadata	cstcheck_ dsmismatch	Yes	proc sql noprint;select upcase(data.sasref) into:_cstSourceData from work._csttablemetadata data;create table work._cstproblems as select "&_cstSourceData" as sasref, memname as table from sashelp.vstable data left join work._csttablemetadata src on data.memname=upcase(src.table) where src.table="" and data.libname="&_cstSourceData";quit;			Active	Yes
SDTM0011	WebSDM	Metadata	cstcheck_ metamismatch	Yes	proc sql noprint;create table _csttempds1 as select ref.table, ref.column from work._cstrefcolumnmetadata ref left join (select distinct table from work._cstsrccolumnmetadata) src on upcase(ref.table)=upcase(src.table) where upcase(ref.table)=upcase(src.table);create table _csttempds as select upcase(base.table) as table,upcase(base.column) as column from _csttempds1 base except select upcase(comp.table) as table,upcase(comp.column) as column from work._cstcolumnmetadata comp;select count(*) into: _cstMetricsCntNumRecs from _csttempds1;quit;			Active	Yes
	Janus	Metadata	cstcheck_ metamismatch	Yes	proc sql noprint;create table _csttempds1 as select ref.table, ref.column from work._cstrefcolumnmetadata ref left join (select distinct table from work._cstsrccolumnmetadata) src on upcase(ref.table)=upcase(src.table) where upcase(ref.table)=upcase(src.table);create table _csttempds as select upcase(base.table) as table,upcase(base.column) as column from _csttempds1 base except select upcase(comp.table) as table,upcase(comp.column) as column from work._cstcolumnmetadata comp;select count(*) into: _cstMetricsCntNumRecs from _csttempds1;quit;			Active	Yes

Report generated: 2010-03-25T11:38:29
Report source: (folder) c:\cstGlobalLibrary\standards\cdisc-sdtm-3.1.1\validation\control (data set) validation_master

Display 8.10 *Message Details (Panel 3) [_cstMessageReport=Y]*

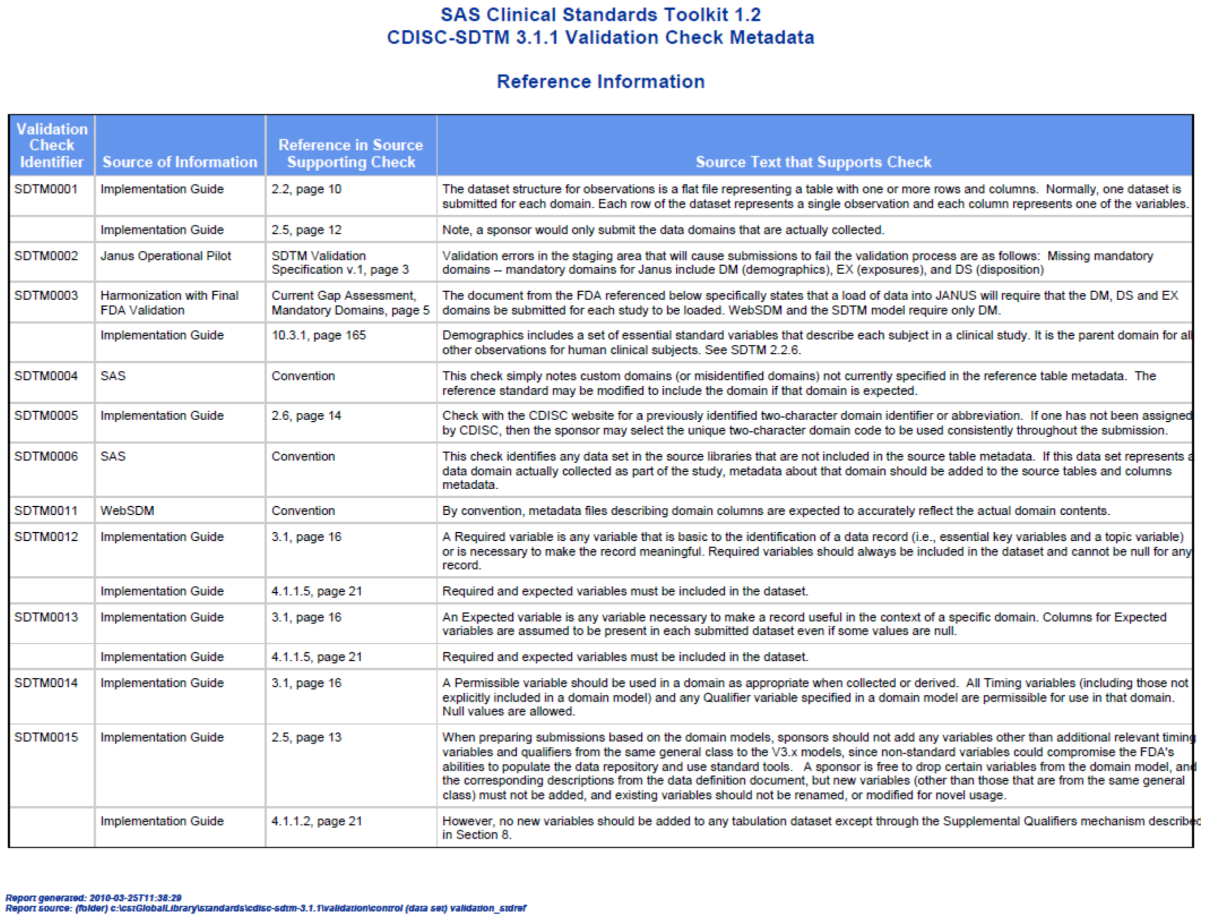


SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 Validation Check Metadata

Message Details

Validation Check Identifier	Source of Check	Message Text	Message Parameter 1 Default Value	Message Parameter 2 Default Value	Basis or Explanation for Result
SDTM0001	Janus	Domain &_cstparm1 contains 0 observations or is missing			
	WebSDM	Domain &_cstparm1 contains 0 observations or is missing			
SDTM0002	JanusFR	Missing (or empty) DM, DS or EX domain			
SDTM0003	WebSDM	Missing (or empty) DM domain			
SDTM0004	SAS	Study data set not found in reference standard			
SDTM0005	SAS	Check custom domain data set name			CDISC has reserved domain codes beginning with the letters X, Y, or Z for the creation of custom domains. All others are subject to future CDISC use.
SDTM0006	SAS	Domain not found in study metadata			
SDTM0011	Janus	Variable &_cstparm1 in description file not in dataset			
	WebSDM	Variable &_cstparm1 in description file not in dataset			
SDTM0012	JanusFR	SDTM required variable &_cstparm1 not found			
	WebSDM	SDTM required variable &_cstparm1 not found			
SDTM0013	Janus	SDTM expected variable &_cstparm1 not found			

Display 8.11 *Reference Information (Panel 4) [_cstSTDRefReport=Y]*



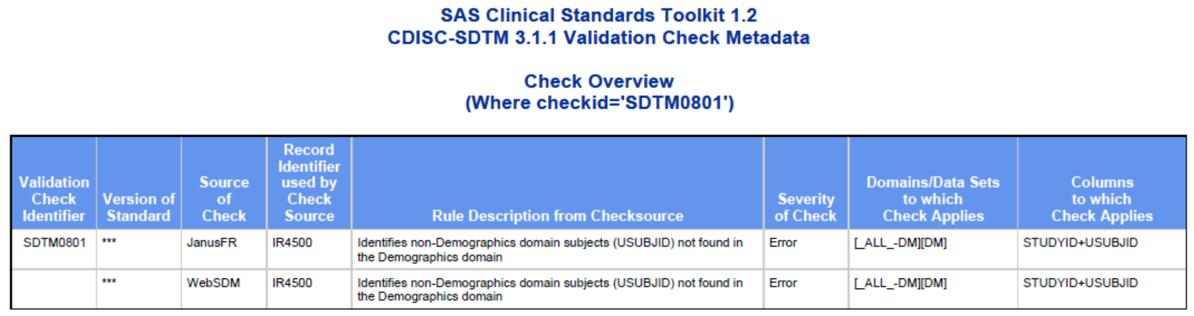
SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 Validation Check Metadata

Reference Information

Validation Check Identifier	Source of Information	Reference in Source Supporting Check	Source Text that Supports Check
SDTM0001	Implementation Guide	2.2, page 10	The dataset structure for observations is a flat file representing a table with one or more rows and columns. Normally, one dataset is submitted for each domain. Each row of the dataset represents a single observation and each column represents one of the variables.
	Implementation Guide	2.5, page 12	Note, a sponsor would only submit the data domains that are actually collected.
SDTM0002	Janus Operational Pilot	SDTM Validation Specification v.1, page 3	Validation errors in the staging area that will cause submissions to fail the validation process are as follows: Missing mandatory domains -- mandatory domains for Janus include DM (demographics), EX (exposures), and DS (disposition)
SDTM0003	Harmonization with Final FDA Validation	Current Gap Assessment, Mandatory Domains, page 5	The document from the FDA referenced below specifically states that a load of data into JANUS will require that the DM, DS and EX domains be submitted for each study to be loaded. WebSDM and the SDTM model require only DM.
	Implementation Guide	10.3.1, page 165	Demographics includes a set of essential standard variables that describe each subject in a clinical study. It is the parent domain for all other observations for human clinical subjects. See SDTM 2.2.6.
SDTM0004	SAS	Convention	This check simply notes custom domains (or misidentified domains) not currently specified in the reference table metadata. The reference standard may be modified to include the domain if that domain is expected.
SDTM0005	Implementation Guide	2.6, page 14	Check with the CDISC website for a previously identified two-character domain identifier or abbreviation. If one has not been assigned by CDISC, then the sponsor may select the unique two-character domain code to be used consistently throughout the submission.
SDTM0006	SAS	Convention	This check identifies any data set in the source libraries that are not included in the source table metadata. If this data set represents a data domain actually collected as part of the study, metadata about that domain should be added to the source tables and columns metadata.
SDTM0011	WebSDM	Convention	By convention, metadata files describing domain columns are expected to accurately reflect the actual domain contents.
SDTM0012	Implementation Guide	3.1, page 16	A Required variable is any variable that is basic to the identification of a data record (i.e., essential key variables and a topic variable) or is necessary to make the record meaningful. Required variables should always be included in the dataset and cannot be null for any record.
	Implementation Guide	4.1.1.5, page 21	Required and expected variables must be included in the dataset.
SDTM0013	Implementation Guide	3.1, page 16	An Expected variable is any variable necessary to make a record useful in the context of a specific domain. Columns for Expected variables are assumed to be present in each submitted dataset even if some values are null.
	Implementation Guide	4.1.1.5, page 21	Required and expected variables must be included in the dataset.
SDTM0014	Implementation Guide	3.1, page 16	A Permissible variable should be used in a domain as appropriate when collected or derived. All Timing variables (including those not explicitly included in a domain model) and any Qualifier variable specified in a domain model are permissible for use in that domain. Null values are allowed.
SDTM0015	Implementation Guide	2.5, page 13	When preparing submissions based on the domain models, sponsors should not add any variables other than additional relevant timing variables and qualifiers from the same general class to the V3.x models, since non-standard variables could compromise the FDA's abilities to populate the data repository and use standard tools. A sponsor is free to drop certain variables from the domain model, and the corresponding descriptions from the data definition document, but new variables (other than those that are from the same general class) must not be added, and existing variables should not be renamed, or modified for novel usage.
	Implementation Guide	4.1.1.2, page 21	However, no new variables should be added to any tabulation dataset except through the Supplemental Qualifiers mechanism described in Section 8.

Report generated: 2010-03-25T11:38:29
Report source: (folder) c:\cstGlobalLibrary\standards\cdisc-sdtm-3.1.1\validation\control (data set) validation_stdref

Display 8.12 *Sample Report Using WHERE Clause [_cstValidationDSWhClause=checkid='SDTM0801']*

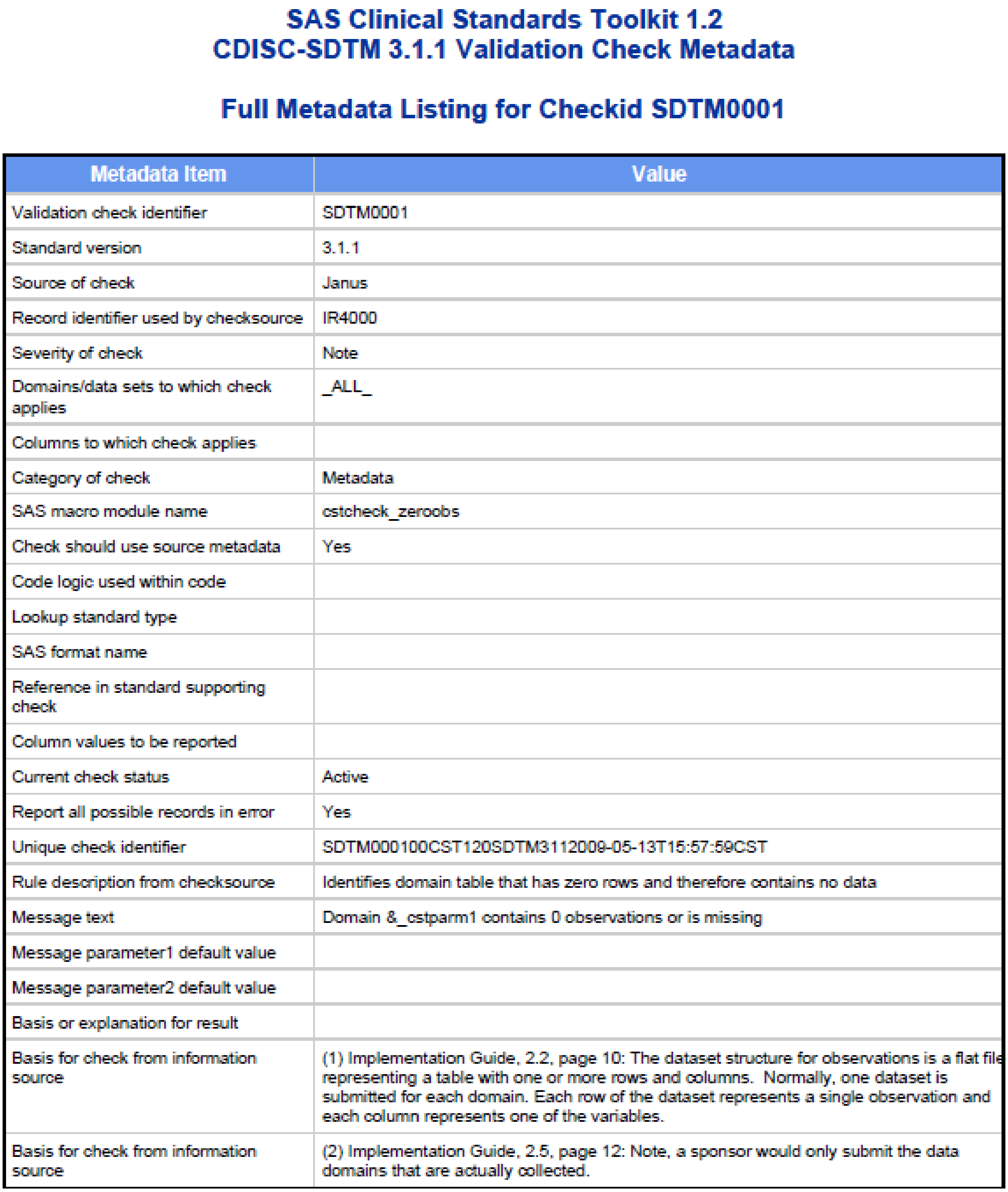


SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 Validation Check Metadata

Check Overview
(Where checkid='SDTM0801')

Validation Check Identifier	Version of Standard	Source of Check	Record Identifier used by Check Source	Rule Description from Checksource	Severity of Check	Domains/Data Sets to which Check Applies	Columns to which Check Applies
SDTM0801	***	JanusFR	IR4500	Identifies non-Demographics domain subjects (USUBJID) not found in the Demographics domain	Error	[_ALL_-DM][DM]	STUDYID+USUBJID
	***	WebSDM	IR4500	Identifies non-Demographics domain subjects (USUBJID) not found in the Demographics domain	Error	[_ALL_-DM][DM]	STUDYID+USUBJID

Display 8.13 *Sample Report By Record View [_cstRecordView=Y]*



SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 Validation Check Metadata

Full Metadata Listing for Checkid SDTM0001

Metadata Item	Value
Validation check identifier	SDTM0001
Standard version	3.1.1
Source of check	Janus
Record identifier used by checksource	IR4000
Severity of check	Note
Domains/data sets to which check applies	_ALL_
Columns to which check applies	
Category of check	Metadata
SAS macro module name	cstcheck_zeroobs
Check should use source metadata	Yes
Code logic used within code	
Lookup standard type	
SAS format name	
Reference in standard supporting check	
Column values to be reported	
Current check status	Active
Report all possible records in error	Yes
Unique check identifier	SDTM000100CST120SDTM3112009-05-13T15:57:59CST
Rule description from checksource	Identifies domain table that has zero rows and therefore contains no data
Message text	Domain &_cstparm1 contains 0 observations or is missing
Message parameter1 default value	
Message parameter2 default value	
Basis or explanation for result	
Basis for check from information source	(1) Implementation Guide, 2.2, page 10: The dataset structure for observations is a flat file representing a table with one or more rows and columns. Normally, one dataset is submitted for each domain. Each row of the dataset represents a single observation and each column represents one of the variables.
Basis for check from information source	(2) Implementation Guide, 2.5, page 12: Note, a sponsor would only submit the data domains that are actually collected.

Appendix 1
Global Macro Variables

The following global macro variables are used by the SAS Clinical Standards Toolkit. Most SAS Clinical Standards Toolkit global macro variables that are provided by SAS are defined in property files in the form of name and value pairs, such as:

_cstDebug=

Each registered standard, including CST-Framework, has an initialize.properties file. This file specifies global macro variables that are required by the standard and are available for use in any SAS Clinical Standards Toolkit processes that reference the standard. Each registered standard might have an action-related properties file that specifies global macro variables that are needed for processes performing the action. An example of this type of file is validation.properties.

A properties file is processed in one of two ways:

A direct call is made to the SAS Clinical Standards Toolkit utility macro %cstutil_setproperties in a code module, such as a driver program like validate_data.sas.

The file is included in the SASReferences data set (with type=properties), in which the %cstutil_allocatesasreferences macro calls %cstutil_setproperties.

Global macro variables can be deleted at the end of a process if the SAS Clinical Standards Toolkit utility macro %cstutil_cleanupcstsession is called with the _cstDeleteGlobalMacroVars parameter set to 1.

Two commonly used global macro variables are not defined in the properties files previously described. The _cstGRoot global macro variable defines the location of _cstGlobalLibrary and is set with the autocall macro %cstutil_setcstgroot. This macro is called in most framework macros. The &studyRootPath global macro variable defines the location of the study data and metadata. It is often set in user-defined driver programs (for example, validate_data.sas).

Table A1.1 *Global Macro Variables*

Global Macro Variable	Values * default value	Comments
	CST-Framework initialize.properties	
_cstDebug	0 (off)* 1 (on)	If on, then _cstDebugOptions are set. Many files remain in the Work library at process conclusion. *Note:* When _cstDebug=1, the size of the SAS log is significantly larger.

Global Macro Variable	Values * default value	Comments
_cstDebugOptions	mprint mlogic symbolgen mautolocdisplay*	SAS system options set when _cstDebug=1.
_cst_rc	0 (no error)* 1 (error)	Set to 1 during processing if an error is encountered that should halt the process.
_cst_MsgID	<blank>*	The result or validation check ID that is used for reporting process results. A value is set in each code module.
_cst_MsgParm1	<blank>*	Any result message parameter (1) that is used for reporting process results. A value is set in each code module.
_cst_MsgParm2	<blank>*	Any result message parameter (2) that is used for reporting process results. A value is set in each code module.
_cstResultSeq	0*	Sequence indicator that is used to signal multiple instances of the same event (such as running the same validation check multiple times). This variable should be initialized to 0. This variable is used for reporting process results. Values are incremented in each code module. This variable is used to join the Results and Metrics data sets.
_cstSeqCnt	0*	Sequence indicator that is used to count the number of records that were output to the Results data set in _cstResultSeq. This variable should be initialized to 0. This variable is used for reporting process results. Values are incremented in each code module
_cstResultsDS	work._cstresults *	The default data set name that is used to accumulate results during a process. This variable might be persisted at the end of the process based on the SASReferences (type=results) entry.
_cstSrcData	<blank>*	This variable is used for reporting process results. A value is set in each code module.

Global Macro Variable	**Values** *** default value**	**Comments**
_cstResultFlag	0* -1 1	This variable reports the status of any result. A value of 0 indicates an informational or non-error status. A positive integer indicates an error status. A negative integer indicates that the assessment could not be completed, often because of metadata problems or SAS errors.
_cstReallocateSASRefs	0* (no) 1 (yes)	This variable specifies whether the SAS Clinical Standards Toolkit should attempt to reallocate any SAS librefs and filerefs if they are already allocated. If the value is yes, then allocation is based on SASReferences content.
_cstFMTLibraries	<blank>* ** work.fmt (example)	This variable enables users to change the format search path built from SASReferences (type=fmtsearch) entries with <libref> or <libref.catalog> references. If only <libref> is provided, then SAS assumes a catalog name of FORMATS. If the value begins with ** (such as ** WORK), then the SAS Clinical Standards Toolkit moves WORK.FORMATS to the end of the format search path.
_cstMessageOrder	APPEND* MERGE	This variable is used in the derivation of _cstMessages. The value APPEND appends message files based on the order of SASReferences (type=messages) entries. The value MERGE allows references to multiple standard-specific message files (including internationalized messages), retaining a single message per message ID, standardversion, and checksource.
_cstMessages	work._cstmessages*	The default data set name that is used to aggregate all standard messages based on SASReferences (type=messages) entries. This file is used during processing to fully resolve the results.message field.

Global Macro Variable	Values * default value	Comments
_cstSASRefsLoc	&workpath*	The path to a directory that contains the SASReferences data that is specified in _cstSASRefsName. By default, the SAS Clinical Standards Toolkit assumes that the SASRferences data set is located in the SAS Work library (signified by &workpath). Use of &workpath is not required.
_cstSASRefsName	sasreferences*	The name of the SASReferences data set (in _cstSASRefsLoc) to be used as the initial source of information about all inputs and outputs defined for a SAS Clinical Standards Toolkit process. The name of the data set that is a SASReferences data set. This allows more than one SASReferences data set to be stored in a directory.
_cstSASRefs	work._cstsasrefs*	The SASReferences data set that is used during processing that contains fully resolved records (for example, paths) based on using standard-level SASReferences data sets for default values.
CDISC SDTM (3.1.1 and 3.1.2) initialize.properties		
_cstSubjectColumns	studyid usubjid*	The standard-specific set of columns that identify a subject. Columns are used by standard-specific macros and for metrics calculations. Columns do not need to be in all source tables (for example, non-patient-level domains like CDISC trial design domains).
_cstTableMetadata	work._csttablemetadata*	Data set that is used during processing that contains table-level metadata (derived from either the reference or study table metadata) that is used by the process.
_cstColumnMetadata	work._cstcolumnmetadata*	Data set that is used during processing that contains column-level metadata (derived from either the reference or study column metadata) that is used by the process.
CDISC SDTM (3.1.1 and 3.1.2) validation.properties		

Global Macro Variable	Values * default value	Comments
_cstCheckSortOrder	_DATA_* <keys>	This variable enables specification of the order in which the checks are to be run. The _DATA_ value indicates that checks are to be processed in the order defined in the Validation Control data set. Users can specify a set of space-delimited keys from Validation Control columns (for example, checksource checkid).
_cstMetrics	0 (off)* 1 (on)	Toggle this variable to enable or disable metrics reporting. This variable attempts to provide a denominator for the errors that are detected. Increased processing time can result.
_cstMetricsDS	work._cstmetrics*	The default data set name that is used to accumulate results during a process. This variable is typically stored at the end of the process based on the SASReferences (type=results) entry.
_cstMetricsTimer	0 (off) 1 (on)*	This variable estimates the elapsed time to perform an action. Results are added to _cstMetricsDS. The value is ignored if _cstMetrics=0.
_cstMetricsNumSubj	0 (off) 1 (on)*	This variable enables counts on a subject level. The value is ignored if _cstMetrics=0.
_cstMetricsNumRecs	0 (off) 1 (on)*	This variable enables counts on the number of records tested. The value is ignored if _cstMetrics=0.
_cstMetricsNumChecks	0 (off) 1 (on)*	This variable specifies whether to report the number of distinct validation check invocations. The value is ignored if _cstMetrics=0.
_cstMetricsNumBadChecks	0 (off) 1 (on)*	This variable specifies whether to report the number of check invocations that were not run. The value is ignored if _cstMetrics=0.
_cstMetricsNumErrors	0 (off) 1 (on)*	This variable specifies whether to report the number of resultseverity="Error" records in the Results data set. This value is ignored if _cstMetrics=0.

Global Macro Variable	Values * default value	Comments
_cstMetricsNumWarnings	0 (off) 1 (on)*	This variable specifies whether to report the number of resultseverity="Warning" records in the Results data set. This value is ignored if _cstMetrics=0.
_cstMetricsNumNotes	0 (off) 1 (on)*	This variable specifies whether to report the number of resultseverity="Note" records in the Results data set. The value is ignored if _cstMetrics=0.
_cstMetricsNumStructural	0 (off) 1 (on)*	This variable specifies whether to report the number of structural errors that were detected. This variable is based on the errors reported for checks where checktype= "Metadata". This excludes informational records in the Results data set. The value is ignored if _cstMetrics=0.
_cstMetricsNumContent	0 (off) 1 (on)*	This variable specifies whether to report the number of content errors that were detected. This variable is based on the errors reported for checks where checktype ^= "Metadata". This excludes informational records in the Results data set. The value is ignored if _cstMetrics=0.
_cstMetricsCntNumSubj	0*	Actual count of the number of subjects that were tested. The value is not calculated if _cstMetrics=0.
_cstMetricsCntNumRecs	0*	Actual count of the number of records that were tested. The value is not calculated if _cstMetrics=0.
_cstMetricsCntNumChecks	0*	Actual count of the number of validation checks that were run. The value is not calculated if _cstMetrics=0.
_cstMetricsCntNumBadChecks	0*	Actual count of the number of check invocations that were not run. The value is not calculated if _cstMetrics=0.
_cstMetricsCntNumErrors	0*	Actual count of the number of errors that were reported. The value is not calculated if _cstMetrics=0.

Global Macro Variable	Values * default value	Comments
_cstMetricsCntNumWarnings	0*	Actual count of the number of warnings that were reported. The value is not calculated if _cstMetrics=0.
_cstMetricsCntNumNotes	0*	Actual count of the number of notes that were reported. The value is not calculated if _cstMetrics=0.
_cstMetricsCntNumStructural	0*	Actual count of the number of structural errors that were reported. The value is not calculated if _cstMetrics=0.
_cstMetricsCntNumContent	0*	Actual count of the number of content errors that were reported. The value is not calculated if _cstMetrics=0.
_cstCRTVersion	1.0*	Current CDISC CRT-DDS version. *Note:* This variable might be deprecated in a future release.
General Purpose (not set in properties files)		
_cstGRoot	Example: C:\cstGlobalLibrary	This variable is required. It defines the location of _cstGlobalLibrary. It is set with the autocall macro %cstutil_setcstgroot, which is called in most framework macros. It is used most often in SASReferences paths to enable relative path mobility.
studyRootPath	Example: C:\Study1	This variable is optional. It defines the location of study data and metadata. It is often set in user-defined driver programs (for example, validate_data.sas). It is used in SASReferences paths to limit the changes that are required when changing input data sources, which facilitates portability.

* default value

Appendix 2
Framework Messages

Table A2.1 *Result IDs and Associated Message Text*

Result ID	Check Severity	Message Text
CST0001	Error	Fatal error encountered, process cannot continue
CST0002	Warning: Check not run	No tables evaluated-check validation control data set
CST0003	Warning: Check not run	&_cstparm1 could not be found
CST0004	Warning: Check not run	No columns evaluated - check validation_control specification
CST0005	Error	Input parameters to macro insufficient for &_cstparm1 macro to run
CST0006	Warning: Check not run	Lookup to SASReferences control data set failed
CST0007	Error	SASReferences lookup returned no records
CST0008	Error	&_cstparm1 could not be found
CST0009	Error	&_cstparm1 macro variable not defined
CST0010	Error	SASReferences lookup returned multiple records
CST0012	Error	SASReferences lookup returned inconsistent SASref and memname values
CST0014	Warning: Check not run	Global macro variable &_cstparm1 cannot be null
CST0015	Warning: Check not run	Invalid &_cstparm1 input parameter, &_cstparm2 macro cannot run
CST0016	Warning: Check not run	&_cstparm1 could not be found

Result ID	Check Severity	Message Text
CST0020	Info	Check run but nonmissing codeLogic is not used
CST0021	Warning: Check not run	Table &_cstparm1 does not contain &_cstparm2 column
CST0022	Warning: Check not run	&_cstparm1 keys could not be found
CST0023	Warning: Check not run	Validation control parsing of &_cstparm1 results in inconsistent sublist lengths
CST0024	Warning: Check incomplete	The column &_cstparm1 was not found in &_cstparm2 - compliance not assessed
CST0025	Warning: Check incomplete	Data set not found in reference standard - compliance not assessed
CST0026	Warning: Check not run	One or more check metadata column values is invalid - &_cstparm1
CST0027	Warning: Check not run	Global macro variable &_cstparm1 could not be found or contains an invalid value
CST0028	Warning: Check not run	Format search path has not been set
CST0029	Info	Format catalog &_cstparm1 in fmtsearch could not be found
CST0030	Warning: Check not run	No catalogs in fmtsearch could be found
CST0031	Warning: Check not run	Reference terminology &_cstparm1 not found
CST0032	Info	Reference terminology data set &_cstparm1 was set to &_cstparm2
CST0033	Info	Format search path has been set to &_cstparm1
CST0034	Warning: Check not run	&_cstParm1 has no observations for &_cstParm2
CST0050	Warning: Check not run	Code failed due to SAS error - &_cstparm1
CST0051	Error	Code failed due to SAS error - &_cstparm1

Result ID	Check Severity	Message Text
CST0070	Error	Selected target directory, &_cstparm1, does not exist. Please create the Target Directory.
CST0071	Error	The Standards directory, &_cstparm1, is not available.
CST0072	Error	Unable to create directory, &_cstparm1.
CST0073	Info	Study directories created in &_cstparm1.
CST0074	Info	Study reference data created in &_cstparm1.
CST0075	Error	Unable to allocate &_cstparm1 for &_cstparm2.
CST0076	Info	SAS &_cstparm1 from SASref=&_cstparm2 SASReferences record not allocated
CST0080	Info	SASReferences for &_cstParm1 were copied to &_cstParm2
CST0081	Error	A required parameter was not supplied &_cstParm1
CST0082	Error	The standard &_cstParm1 is not registered
CST0083	Error	The version &_cstParm1 does not exist for &_cstParm2
CST0084	Error	The SASReferences type &_cstParm1 is not defined for &_cstParm2
CST0085	Error	No version was supplied and there is no default for the &_cstParm1 standard
CST0086	Error	The SASReferences type &_cstParm1 has more than one subtype and none was specified
CST0087	Error	The type/subtype &_cstParm1 is not defined for &_cstParm2
CST0088	Error	The following columns of &_cstParm1 cannot be empty: &_cstParm2
CST0089	Error	Only libraries are supported for this operation

Result ID	Check Severity	Message Text
CST0090	Error	There were problems with the sasreferences data set
CST0099	Warning: Check not run	&_cstparm1 is not supported in the current release of CST
CST0100	Info	No errors detected in &_cstparm1
CST0101	Error	The libref &_cstparm1 must be assigned prior to calling the macro
CST0102	Info	&_cstparm1 was created as requested
CST0103	Error	A SASReferences file must be passed as a parameter or specified using the CST global environment variable
CST0104	Error	Unable to acquire exclusive locks on the global metadata data sets
CST0106	Error	The standard &_cstParm1 does not have a properties file registered for &_cstParm2
CST0107	Error	Invalid location type &_cstParm1
CST0108	Info	The properties were processed from the &_cstParm1 &_cstParm2
CST0109	Info	The default version for &_cstParm1 has been set to &_cstParm2
CST0110	Info	&_cstParm1 is no longer registered as a standard
CST0111	Error	Unable to open data set &_cstParm1
CST0112	Error	Data set &_cstParm1 has no observations
CST0114	Error	No lookup table found in registered standards data set where standard=&_cstParm1 and version=&_cstParm2
CST0115	Error	Null values are not permitted for column &_cstParm2 in data set &_cstParm1
CST0116	Error	Invalid value for column &_cstParm1 in data set &_cstParm2

Result ID	Check Severity	Message Text
CST0117	Error	No template data set found for type=&_cstParm1, subtype=&_cstParm2 in the registered data standards
CST0118	Error	The standard &_cstParm1 is not a data standard
CST0119	Error	The standard &_cstParm1 is missing referencemetadata for &_cstParm2
CST0120	Error	Could not continue due to errors encountered in assigning libraries
CST0121	Error	Errors were encountered creating the &_cstParm1 tables - check the log
CST0122	Info	The tables were created for &_cstParm1 in library &_cstParm2
CST0123	Warning	The lookup table has no entries for standard=&_cstParm1 and version=&_cstParm2
CST0124	Error	The default version &_cstParm1 for &_cstParm2 cannot be unregistered while other versions exist
CST0125	Error	Differences found between data set &_cstParm1 and the template data set &_cstParm2
CST0200	Info	&_cstParm1

Note: Not all message data set fields are displayed.

Appendix 3
Macro Application Programming Interface

Module CRT-DDS V1.0 (Run Time)

Overview

This is the CDISC CRT-DDS 1.0 run-time macro library.

Table A3.1 *Module CRT-DDS V1.0 (Run Time) Macro Summary*

Exposure	Macro
	%crtdds_clitemdecodetrans(_cstsourcestudy=, _cstsourcecolumns=, _cstcodelistitemsds=, _cstmdvDS=, _cststudyds=, _cstcodelistsds=, _cstCLlang=, _cstoutclitemdecodetransds=);
	%crtdds_codelistitems(_cstsourcecolumns=, _cstcodelistsds=, _cstoutcodelistitemsds=);
	%crtdds_codelists(_cstsourcecolumns=, _cstmdvds=, _cstmdvname=, _cstoutcodelistsds=);

Exposure	Macro
	%crtdds_computationmethods(_cstsourcecolumns=, _cstsourcestudy=, _cstmdvds=, _cstitemdefsds=, _cststudyds=, _cstoutcomputationmethodsds=);
	%crtdds_definedocument(_cstname=, _cstdescr=, _cstoutdefinedocds=);
	%crtdds_getStatic(_cstName=, _cstVar=);
	%crtdds_itemdefs(_cstsourcecolumns=, _cstsourcestudy=, _cststudyds=, _cstmdvds=, _cstcodelistsDS=, _cstoutitemdefsds=, _cstoutitemdefsds2=);
	%crtdds_itemgroupdefitemrefs(_cstsourcecolumns=, _cstsourcetables=, _cstsourcestudy=, _cstitemdefsds2=, _cstmdvds=, _cstitemgroupdefsds=, _cststudyds=, _cstoutitemgroupdefitemrefsds=);
	%crtdds_itemgroupdefs(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupdefsds=);
	%crtdds_itemgroupleaf(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupleafds=);
	%crtdds_itemgroupleaftitles(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupleaftitlesds=);
	%crtdds_metadataversion(_cstname=, _cstdescr=, _cststandard=, _cstversion=, _cstdefineversion=, _cststudyds=, _cststudyname=, _cstoutmdvds=);
`External` `CRTDDS`	%crtdds_read;
`External` `CRTDDS`	%crtdds_sdtm311todefine10(_cstOutLib=, _cstSourceTables=, _cstSourceColumns=, _cstSourceStudy=);
	%crtdds_study(_cstname=, _cstdescr=, _cstprotocol=, _cstdefineds=, _cstdefinename=, _cstoutstudyds=);

`External`
`CRTDDS`
`Validation`
`Process`
	%crtdds_validate /des='CST: Validate CDISC CRTDDS model files';

`External` `CRT-DDS`	%crtdds_write(_cstCreateDisplayStyleSheet=1, _cstOutputEncoding=, _cstHeaderComment=, _cstResultsOverrideDS=, _cstLogLevel=info);
`External` `CRTDDS`	%crtdds_xmlvalidate(_cstLogLevel=info, _cstResultsOverrideDS=);

`Internal`
`Framework Utility`
	%crtddsutil_buildchecktablelist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=);

Macro Detail

%crtdds_clitemdecodetrans

%crtdds_clitemdecodetrans(_cstsourcestudy=, _cstsourcecolumns=, _cstcodelistitemsds=, _cstmdvDS=, _cststudyds=, _cstcodelistsds=, _cstCLlang=, _cstoutclitemdecodetransds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcestudy
- _cstsourcecolumns
- _cstcodelistitemsds
- _cstmdvDS
- _cststudyds
- _cstcodelistsds
- _cstCLlang
- _cstoutclitemdecodetransds

File: crtdds_clitemdecodetrans.sas

%crtdds_codelistitems

%crtdds_codelistitems(_cstsourcecolumns=, _cstcodelistsds=, _cstoutcodelistitemsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns
- _cstcodelistsds
- _cstoutcodelistitemsds

File: crtdds_codelistitems.sas

%crtdds_codelists

%crtdds_codelists(_cstsourcecolumns=, _cstmdvds=, _cstmdvname=, _cstoutcodelistsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns
- _cstmdvds
- _cstmdvname
- _cstoutcodelistsds

File: crtdds_codelists.sas

%crtdds_computationmethods

%crtdds_computationmethods(_cstsourcecolumns=, _cstsourcestudy=, _cstmdvds=, _cstitemdefsds=, _cststudyds=, _cstoutcomputationmethodsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns
- _cstsourcestudy
- _cstmdvds
- _cstitemdefsds
- _cststudyds
- _cstoutcomputationmethodsds

File: crtdds_computationmethods.sas

%crtdds_definedocument

%crtdds_definedocument(_cstname=, _cstdescr=, _cstoutdefinedocds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstname
- _cstdescr
- _cstoutdefinedocds

File: crtdds_definedocument.sas

%crtdds_getStatic

%crtdds_getStatic(_cstName=, _cstVar=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstName
- _cstVar

File: crtdds_getstatic.sas

%crtdds_itemdefs

%crtdds_itemdefs(_cstsourcecolumns=, _cstsourcestudy=, _cststudyds=, _cstmdvds=, _cstcodelistsDS=, _cstoutitemdefsds=, _cstoutitemdefsds2=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns
- _cstsourcestudy
- _cststudyds
- _cstmdvds
- _cstcodelistsDS

- _cstoutitemdefsds
- _cstoutitemdefsds2

File: crtdds_itemdefs.sas

%crtdds_itemgroupdefitemrefs

%crtdds_itemgroupdefitemrefs(_cstsourcecolumns=, _cstsourcetables=, _cstsourcestudy=, _cstitemdefsds2=, _cstmdvds=, _cstitemgroupdefsds=, _cststudyds=, _cstoutitemgroupdefitemrefsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns
- _cstsourcetables
- _cstsourcestudy
- _cstitemdefsds2
- _cstmdvds
- _cstitemgroupdefsds
- _cststudyds
- _cstoutitemgroupdefitemrefsds

File: crtdds_itemgroupdefitemrefs.sas

%crtdds_itemgroupdefs

%crtdds_itemgroupdefs(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupdefsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcetables
- _cstsourcestudy
- _cststudyds
- _cstmdvDS
- _cstoutitemgroupdefsds

File: crtdds_itemgroupdefs.sas

%crtdds_itemgroupleaf

%crtdds_itemgroupleaf(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupleafds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcetables
- _cstsourcestudy
- _cststudyds
- _cstmdvDS

- _cstoutitemgroupdefsds

File: crtdds_itemgroupleaf.sas

%crtdds_itemgroupleaftitles

%crtdds_itemgroupleaftitles(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupleaftitlesds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcetables
- _cstsourcestudy
- _cststudyds
- _cstmdvDS
- _cstoutitemgroupdefsds

File: crtdds_itemgroupleaftitles.sas

%crtdds_metadataversion

%crtdds_metadataversion(_cstname=, _cstdescr=, _cststandard=, _cstversion=, _cstdefineversion=, _cststudyds=, _cststudyname=, _cstoutmdvds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstname
- _cstdescr
- _cststandard
- _cstversion
- _cstdefineversion
- _cststudyds
- _cststudyname
- _cstoutmdvds

File: crtdds_metadataversion.sas

%crtdds_read

%crtdds_read;

[Exposure: external] [Macro Type: CRTDDS]

Reads a CDISC CRT-DDS 1.0 (define.xml) XML file into the SAS representation of CRT-DDS 1.0.

This macro uses the SAS representation of a CDISC CRT-DDS XML file as source, and converts it into SAS data sets. The inputs and outputs are specified in a SASReferences file.

Required global macro variables:

- Framework initialization properties.
- CDISC CRT-DDS 1.0 initialization properties.

- _cstResultsDS should point to an existing Results data set. Otherwise, work._cstResults is used.

File: crtdds_metadataversion.sas

%crtdds_sdtm311todefine10

%crtdds_sdtm311todefine10(_cstOutLib=, _cstSourceTables=, _cstSourceColumns=, _cstSourceStudy=);

[Exposure: external] [Macro Type: CRT-DDS]

Populates 12 of the 39 tables in the SAS representation of the CRT-DDS standard.

This macro extracts data from the SDTM metadata files, and converts the metadata into a subset (12) of the tables in the SAS representation of the CRT-DDS model. The following CRT-DDS tables are created:

- clitemdecodetranslatedtext
- codelistitems
- codelists
- computationmethods
- definedocument
- itemdefs
- itemgroupdefitemrefs
- itemgroupdefs
- itemgroupleaf
- itemgroupleaftitles
- metadataversion
- study

The metadata source is specified in a SASReferences file.

Required global macro variables:

- framework initialization properties
- CRT-DDS 1.0 initialization properties
- _cstresultsds should point to an existing Results data set

Parameters:

- _cstOutLib—Required. Identifies library reference where the resulting tables should be written to.
- _cstSourceTables—Required. A data set that contains the SDTM metadata for the domains to be included in the CRT-DDS file.
- _cstSourceColumns—Required. A data set that contains the SDTM metadata for the domain columns to be included in the CRT-DDS file.
- _cstSourceStudy—Required. A data set that contains the metadata for the studies to be included in the CRT-DDS file.

File: crtdds_sdtm311todefine10.sas

%crtdds_study

%crtdds_study(_cstname=, _cstdescr=, _cstprotocol=, _cstdefineds=, _cstdefinename=, _cstoutstudyds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstname
- _cstdescr
- _cstprotocol
- _cstdefineds
- _cstdefinename
- _cstoutstudyds

File: crtdds_study.sas

%crtdds_validate

%crtdds_validate /des='CST: Validate CDISC CRTDDS model files';

[Exposure: external] [Macro Type: CRTDDS Validation Process]

crtdds_validate

Validate CDISC CRT-DDS model files.

The basic function of this code module is to cycle through the validation checks to be run, writing validation results to the process Results and Metrics data sets. These data sets are persisted to any permanent location based on type=results records in a SASReferences file. Process cleanup is based on the _cstDebug global macro variable.

Required global macro variables (beyond reporting and debugging variables):

(none)

Required File Inputs:

run-time (type=control,subtype=validation in a SASReferences file) check data set

File: crtdds_validate.sas

%crtdds_write

%crtdds_write(_cstCreateDisplayStyleSheet=1, _cstOutputEncoding=, _cstHeaderComment=, _cstResultsOverrideDS=, _cstLogLevel=info);

[Exposure: external] [Macro Type: CRT-DDS]

Writes a CDISC CRT-DDS V1.0 XML file.

This macro uses the SAS representation of a CRT-DDS file as source data, and converts it to the required XML structure. The inputs and outputs are specified in a SASReferences file.

Required global macro variables:

- framework initialization properties
- CRT-DDS 1.0 initialization properties
- _cstresultsds should point to an existing Results data set, or it should override this value using the _cstResultsOverrideDS parameter to this macro

Parameters:

- _cstCreateDisplayStyleSheet—Optional. Identifies whether the macro should create a style sheet in the same directory as the output XML file. If this is set to 1, then the macro looks in the provided SASReferences file for a record with a type and subtype of referencexml and stylesheet, and uses that file. If this is set to 0, then the macro does not create the XSL, even if one is specified in the SASReferences file.
- _cstOutputEncoding—Optional. The XML encoding to use for the CRT-DDS file that is created.
- _cstHeaderComment—Optional. A short comment is added to the top of the CRT-DDS file that is produced. If none is provided, then a default is used.
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.
- _cstLogLevel—Optional. Identifies the level of error reporting. Valid values are Info, Warning, Error, and Fatal Error.

File: crtdds_write.sas

%crtdds_xmlvalidate

%crtdds_xmlvalidate(_cstLogLevel=info, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: CRT-DDS]

Performs XML-level (not SAS) validation on a CRT-DDS V1.0 XML file.

General use of this macro is in combination with another macro (such as crtdds_write or crtdds_read). Conditional code is included that writes metadata to the Results data set, and checks the validity of the SASReferences data set if this macro is run independently.

Parameters:

- _cstLogLevel—Optional. Identifies the level of error reporting. Valid values are Info, Warning, Error, and Fatal Error.
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: crtdds_xmlvalidate.sas

%crtddsutil_buildchecktablelist

%crtddsutil_buildchecktablelist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=);

[Exposure: internal] [Macro Type: Framework utility]

Builds a data set that identifies the domains to be validated by each check. This is based on the contents of the validation check data set columns tablescope and columnscope.

Required global macro variables:

(none)

Required File Inputs:

only as specified in the parameters

Parameters:

- _cstCheckDS—The validation check data set that contains the checks for a standard and standardversion. Typically, this is the Validation Master data set.

- _cstWhereClause—Optional. A WHERE clause to subset _cstCheckDS. The syntax should comply with a SAS statement argument, such as any of the following: `VAR1=1` or `upcase(var2)="Y" or checkstatus>0`.
- _cstOutputDS—The output data set that is returned to the calling program. This data set contains a record for each domain that is referenced by a checkid, standardversion, and checksource.

File: crtddsutil_buildchecktablelist.sas

Module Framework

Overview

This is the framework description. It describes what the framework does, and how it fits together.

Since: 1.2

See Appendix A1, “Global Macro Variables,” on page 211

Table A3.2 Module Framework Macro Summary

Exposure	Macro
External Framework	%cst_createDS(_cstStandard=, _cstStandardVersion=, _cstType=, _cstSubType=, _cstOutputDS=, _cstResultsOverrideDS=);
External standard_name	%cst_createEmptyTables; Deprecated
External Study Creation	%cst_createStudyFromStandard(_cstModel=, _cstVersion=, _cstStudyRootPath=);
External Framework	%cst_createTablesForDataStandard(_cstStandard=, _cstStandardVersion=, _cstOutputLibrary=, _cstResultsOverrideDS=);
External Framework	%cst_deleteProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);
External Framework	%cst_getRegisteredStandards(_cstOutputDS=, _cstResultsDS=);
External standard_name	%cst_getStandardMetadata(_cstSASReferences=, _cstResultsOverrideDS=); Deprecated
External Framework	%cst_getStandardSASReferences(_cstStandard=, _cstStandardVersion=, _cstOutputDS=, _cstResultsOverrideDS=);

Exposure	Macro
External Framework	%cst_getStatic(_cstName=, _cstVar=);
External	%cst_insertStandardSASRefs(_cstSASReferences=, _cstOutputDS=, _cstAddRequiredCSTRefs=0, _cstResultsOverrideDS=);
External Framework	%cst_registerStandard(_cstRootPath=, _cstControlSubPath=, _cstStdDSName=, _cstStdSASRefsDSName=, _cstOutputDS=);
External Framework	%cst_setProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);
External Framework	%cst_setStandardProperties(_cstStandard=, _cstStandardVersion=, _cstSubType=, _cstResultsOverrideDS=);
External Framework	%cst_setStandardVersionDefault(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);
External Framework	%cst_unregisterStandard(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);
External Framework	%cst_unsetProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);
External Validation Check	%cstcheck_column(_cstControl=);
External Validation Check	%cstcheck_columncompare(_cstControl=);
External Validation Check	%cstcheck_comparedomains(_cstControl=);
External Validation Check	%cstcheck_dsmismatch(_cstControl=);
External Validation Check	%cstcheck_metamismatch(_cstControl=);
External Validation Check	%cstcheck_notconsistent(_cstControl=);
External Validation Check	%cstcheck_notimplemented(_cstControl=);
External Validation Check	%cstcheck_notincodelist(_cstControl=);

Exposure	Macro
External Validation Check	%cstcheck_notsorted(_cstControl=);
External Validation Check	%cstcheck_notunique(_cstControl=);
External Validation Check	%cstcheck_recmismatch(_cstControl=);
External Validation Check	%cstcheck_recnotfound(_cstControl=);
External Validation Check	%cstcheck_violatesstd(_cstControl=);
Internal Framework Utility	%cstcheck_zeroobs(_cstControl=);
Internal SAS Clinical Standards Toolkit Validation Check Utility	%cstcheckutil_formatlookup(_cstCol2=, _cstCol2Value=, _cstCol1=&_cstColumn, _cstDomOnly=, _cstDSN=&_cstDSName, _cstRowCt=&_cstDSRowCount, _cstC2Val=&_cstColumn2Value);
Internal Framework Utility	%cstutil_allocatesasreferences / des='CST: Allocate sasreferences';
External Framework	%cstutil_allocGlobalMetadataLib(_cstLibname=);
Internal Framework Utility	%cstutil_appendresultds(_cstErrorDS=, _cstVersion=&_cstStandardVersion, _cstSource=&_cstCheckSource, _cstStdRef=, _cstOrderBy=);
Internal Framework Utility	%cstutil_buildcollist(_cstFormatType=DATASET, _cstColWhere=, _cstDomWhere=, _cstColDSName=&_cstColumnMetadata, _cstDomDSName=&_cstTableMetadata, _cstColSubOverride=N, _cstDomSubOverride=N);
Internal Framework Utility	%cstutil_builddomlist(_cstFormatType=DATASET, _cstDomWhere=, _cstDomDSName=&_cstTableMetadata, _cstSubOverride=N);
Internal Framework Check	%cstutil_checkds(_cstdsname=, _csttype=, _cstsubtype=, _cststandard=*, _cststandardversion=*);
Internal Framework Check Internal macro for the cstutil_checkds macro	%chkvals;

Exposure	**Macro**
Internal Framework Utility	%cstutil_cleanupcstsession(_cstClearCompiledMacros=0, _cstClearLibRefs=0, _cstResetSASAutos=0, _cstResetFmtSearch=0, _cstResetSASOptions=1, _cstDeleteFiles=1, _cstDeleteGlobalMacroVars=0);
External Framework Utility	%cstutil_CreateMetadataReport(_cstStandardTitle=, _cstValidationDS=, _cstValidationDSWhClause=, _cstMessagesDS=, _cstStdRefDS=, _cstReportOutput=, _cstCheckMDReport=N, _cstMessageReport=N, _cstStdRefReport=N, _cstRecordView=N);
External Framework Utility	%cstutil_createreport(_cstsasreferencesdset=, _cstresultsdset=&_cstRptResultsDS, _cstmetricsdset=&_cstRptMetricsDS, _cstreporterrorsonly=N, _cstreportobs=, _cstreportbytable=N, _csttablechecksdset=, _csttablecheckscode=, _cstkeeptablechecklist=N, _csttablesubset=, _cstreportoutput=, _cstsummaryReport=Y, _cstioReport=Y, _cstmetricsReport=Y, _cstgeneralResultsReport=Y, _cstcheckIdResultsReport=Y);
Internal Framework	%cstutil_createTempMessages(_cstCreationFlag=);
Internal standard_name	%cstutil_deleteDataSet(_cstDataSetName=);
Internal Framework	%cstutil_getRandomNumber(_cstVarname=);
Internal Framework Utility	%cstutil_getsasreference(_cstStandard=, _cstStandardVersion=, _cstSASRefType=, _cstSASRefSubtype=, _cstSASRefsasref=, _cstSASRefmember=, _cstConcatenate=0, _cstFullname=0, _cstAllowZeroObs=0);
Internal Framework Utility	%cstutil_getsubjectcount(_cstDS=, _cstsubid=&_cstSubjectColumns);
External Framework	%cstutil_internalmanageresults(_cstAction=);
Internal Framework Utility	%cstutil_messagesdsattr /des='CST: Messages data set column attributes';
Internal Framework Utility	%cstutil_metricsdsattr /des='CST: Metrics data set column attributes';
Internal Framework Utility	%cstutil_parsecolumnscope(_cstscopestr=, _cstopsource=, _cstsublistnum=);
Internal Framework Utility	%cstutil_parsescopesegment(_cstPart=, _cstVarName=, _cstMessageID=CST0004);
Internal Framework Utility	%cstutil_parsetablescope(_cstscopestr=, _cstopsource=, _cstsublistnum=);

Exposure	Macro
Internal SAS Clinical Standards Toolkit Framework	%cstutil_processsetup(_cstSASReferencesSource=SASREFERENCES, _cstSASReferencesName=sasreferences, _cstSASReferencesLocation=);
Internal Framework Utility	%cstutil_readcontrol /des="CST: Create control file macro variables";
External Framework Utility	%cstutil_reportgeneralprocess;
External Framework Utility	%cstutil_reportinputsoutputs;
External Framework Utility	%cstutil_reportprocessmetrics;
External Framework Utility	%cstutil_reportprocessresults;
External Framework Utility	%cstutil_reportprocesssummary;
External Framework Utility	%cstutil_reportsetup(_cstRptType=Metadata);
External Framework Utility	%cstutil_reporttabledata;
Internal Framework Utility	%cstutil_resultsdsattr /des='CST: Results data set column attributes';
Internal Framework Utility	%cstutil_resultsdskeep /des='CST: Results data set columns';
Internal Framework Utility	%cstutil_saveresults(_cstIncludeValidationMetrics=0);
Automatically generated by the CST-Framework post-installation configuration component	%cstutil_setcstgroot;
Internal Framework Utility	%cstutil_setmodel /des="Set Which Model Definition to Use";
Internal CDISC CRT-DDS	%cstutil_writecubexml(_cstXMLOut=, _cstMDPFile=, _cstDebug=);

Exposure	**Macro**
Internal Framework Utility	%cstutil_writemetric(_cstMetricParameter=, _cstResultID=, _cstResultSeqParm=, _cstMetricCnt=, _cstSrcDataParm=, _cstMetricsDSParm=&_cstMetricsDS);
Internal Framework Utility	%cstutil_writeresult(_cstResultID=, _cstValCheckID=, _cstResultParm1=, _cstResultParm2=, _cstResultSeqParm=1, _cstSeqNoParm=1, _cstSrcDataParm=, _cstResultFlagParm=0, _cstRCParm=0, _cstActualParm=, _cstKeyValuesParm=, _cstResultDetails=, _cstResultsDSParm=&_cstResultsDS);

Macro Detail

%cst_createDS

%cst_createDS(_cstStandard=, _cstStandardVersion=, _cstType=, _cstSubType=, _cstOutputDS=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Creates a zero observation data set based on those provided by a registered standard.

Parameters:

- _cstStandard—Required. The name of a registered standard.
- _cstStandardVersion—Optional. The version of the standard that the data set should be created from. If this is omitted, then the default version for the standard is used. If a default version is not defined, then an error is generated.
- _cstType—Required. The type of data set to be created. This value comes from the TYPE column in the SASReferences file for the standard-version combination.
- _cstSubType—Optional. Specifies the subtype for the type. This value comes from the SUBTYPE column in the SASReferences file for the standard-version combination. If the type has no subtypes, then the value can be omitted. Otherwise, it must be provided.
- _cstOutputDS—Required. The name of the data set to be created.
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_createds.sas

%cst_createEmptyTables

%cst_createEmptyTables;

[Exposure: external] [Macro Type: standard_name]

Create empty table shells using reference metadata.

Full, multi-line explanation

Required Global Macro Variables:

- _cstVar1
- _cstVar2

Deprecated. Explanation

File: cst_createemptytables.sas

%cst_createStudyFromStandard

%cst_createStudyFromStandard(_cstModel=, _cstVersion=, _cstStudyRootPath=);

[Exposure: external] [Macro Type: study creation]

cst_createStudyFromStandard

Creates a study from selected model and version.

Required Global Macro Variables: (none)

Required File Inputs: (none)

Parameters:

- _cstModel—The name of the data model to use for this study.
- _cstVersion—The version of the data model to use for this study.
- _cstStudyRootPath

File: cst_createStudyFromStandard.sas

%cst_createTablesForDataStandard

%cst_createTablesForDataStandard(_cstStandard=, _cstStandardVersion=, _cstOutputLibrary=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Creates tables from registered reference metadata. This macro generates all of the table shells that are defined for the standard in a library specified by the caller where a standard is registered .

Required Global Macro Variables: CST-Framework standard variables

Parameters:

- _cstStandard—Required. The name of a registered standard.
- _cstStandardVersion—Optional. The version of the standard from which the data set should be created. If this is omitted, then the default version for the standard is used. If a default version is not defined, then an error is generated.
- _cstOutputLibrary—Required. Specifies the LIBNAME in which the table shells should be created.
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_createtablesfordatastandard.sas

%cst_deleteProperties

%cst_deleteProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Reads a properties file or data set and unsets global macros, accordingly. Property files should have the format name=value. Property data sets should have a character field for name and value. They might have a comment field, but this field is ignored.

Parameters:

- _cstPropertiesLocation—Required. The location of the property file. The format depends on the value of _cstLocationType.

- _cstLocationType—Required. Identifies the format for the value of _cstPropertiesLocation. Valid values are: PATH (the path to a properties file), FILENAME (a valid, assigned SAS filename to the properties file), and DATA (a (LIBNAME.)membername of a SAS data set that contains the properties).
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_deleteproperties.sas

%cst_getRegisteredStandards

%cst_getRegisteredStandards(_cstOutputDS=, _cstResultsDS=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstOutputDS
- _cstResultsDS

File: cst_getregisteredstandards.sas

%cst_getStandardMetadata

%cst_getStandardMetadata(_cstSASReferences=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: standard_name]

Retrieves the standard metadata for standards.

A valid SASReferences data set is passed into the macro. It should contain records that point to the metadata for the data standard. A row should exist for each metadata table that is to be returned. The row should identify the standard, standardversion, type, and subtype that can be mapped to the standard's registered information. In addition, the SASRef and memName columns should identify where the new data set is to be created. The RefType must be set to libref.

For example, to retrieve SDTM 3.1.1 reference metadata about tables, the data set should have the columns standard=CDISC-SDTM and standardVersion=3.1.1. Type should be set to “referencemetadata” and subtype to “table.” SASRef could be set to “Work” and memname to “refTableMD.”

Deprecated. explanation

Parameters:

- _cstSASReferences—Required. The (LIBNAME.)member that refers to a valid SASReferences file.
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_getstandardmetadata.sas

%cst_getStandardSASReferences

%cst_getStandardSASReferences(_cstStandard=, _cstStandardVersion=, _cstOutputDS=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: Framework]

Retrieves the global SASReference records for the standard.

If the macro succeeds, then the global variable _cst_rc is set to 0. If it fails, then _cst_rc is set to 1. The Results data set contains more information as to why it failed.

Parameters:

- _cstStandard—Required. The name of a registered standard.
- _cstStandardVersion—Optional. The version of the standard for which the caller wants to retrieve the global SASReferences. This might be omitted if the caller is requesting the default version for the standard.
- _cstOutputDS—Required. The (LIBNAME.)member name of the output data set to be created.
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_getstandardsasreferences.sas

%cst_getStatic

%cst_getStatic(_cstName=, _cstVar=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstName
- _cstVar

File: cst_getstatic.sas

%cst_insertStandardSASRefs

%cst_insertStandardSASRefs(_cstSASReferences=, _cstOutputDS=, _cstAddRequiredCSTRefs=0, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: Not specified]

Inserts missing standards information into a SASReferences file.

It is possible to specify only the standard, standardversion, type, and subtype for information that has been registered by the standard where a SASReferences uses a standard. Calling this macro fills in the missing information. If a standardversion is not specified, then the information for the default version of that standard is used.

Parameters:

- _cstSASReferences—Optional. The(LIBNAME.)member that points to a SASReferences file to be completed. If this is not specified, then the global macro variables _cstSASRefsLoc and _cstSASRefsName might be used to specify the SASReferences file information. The _cstSASRefs macro variable is used if none of the other mechanisms are provided or available.
- _cstOutputDS—Required. The output data set to create that contains the completed information.
- _cstAddRequiredCSTRefs
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_insertstandardsasrefs.sas

See Appendix A1, “Global Macro Variables,” on page 211.

%cst_registerStandard

%cst_registerStandard(_cstRootPath=, _cstControlSubPath=, _cstStdDSName=, _cstStdSASRefsDSName=, _cstOutputDS=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstRootPath
- _cstControlSubPath
- _cstStdDSName
- _cstStdSASRefsDSName
- _cstOutputDS

File: cst_registerstandard.sas

%cst_setProperties

%cst_setProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Reads a properties file or data set and sets global macros, accordingly. Property files should have the format name=value. Property data sets should have a character field for name and value. They might have a comment field, but this field is ignored.

Parameters:

- _cstPropertiesLocation—Required. The location of the property file. The format depends on the value of _cstLocationType.
- _cstLocationType—Required. Identifies the format for the value of _cstPropertiesLocation. Valid values are PATH (the path to a properties file), FILENAME (a valid, assigned SAS filename to the properties file), and DATA (a (LIBNAME.)membername of a SAS data set that contains the properties).
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_setproperties.sas

%cst_setStandardProperties

%cst_setStandardProperties(_cstStandard=, _cstStandardVersion=, _cstSubType=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

When a standard is registered, it most likely also registers values in a SASReferences file. A number of these values might be for properties files that are used by the standard, or provided by the standard to help users. For example, CST_FRAMEWORK provides a property subType of 'required' that points to a property file that has default settings for required properties. A user can call this method using the following code to set these properties:

```
%cst_setStandardProperties(
_cstStandard=CST_FRAMEWORK,
_cstStandardVersion=1.2,
_cstSubType=required);
```

Parameters:

- _cstStandard—Required. The name of a registered standard.
- _cstStandardVersion—Optional if the standard has a default set. Otherwise, it is mandatory. This specifies the version of the standard.
- _cstSubType—Required. The name of the properties subtype that is to be read and from where properties are set.
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_setstandardproperties.sas

%cst_setStandardVersionDefault

%cst_setStandardVersionDefault(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstStandard
- _cstStandardVersion
- _cstResultsOverrideDS

File: cst_setstandardversiondefault.sas

%cst_unregisterStandard

%cst_unregisterStandard(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstStandard
- _cstStandardVersion
- _cstResultsOverrideDS

File: cst_unregisterstandard.sas

%cst_unsetProperties

%cst_unsetProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Reads a properties file or data set and unsets global macros, accordingly. Property files should have the format name=value. Property data sets should have a character field for name and value. They might have a comment field, but this field is ignored.

Parameters:

- _cstPropertiesLocation—Required. The location of the property file. The format depends on the value of _cstLocationType.

- _cstLocationType—Required. Identifies the format for the value of _cstPropertiesLocation. Valid values are: PATH (the path to a properties file), FILENAME (a valid, assigned SAS filename to the properties file), and DATA (a (LIBNAME.)membername of a SAS data set that contains the properties).
- _cstResultsOverrideDS—Optional. The (LIBNAME.)member that refers to a Results data set to be created. If omitted, then the Results data set specified by the &_cstResultsDS is used.

File: cst_unsetproperties.sas

%cstcheck_column

%cstcheck_column(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_column

Identifies any invalid column value or attribute.

Note: Macro requires use of _cstCodeLogic at a statement level in a SAS DATA step context. _cstCodeLogic identifies records in errors by setting _cstError=1.

Example validation checks that use this macro include:

- Value of Visit Number is formatted to > 3 decimal places
- A column character value is not left-justified
- Study day of Visit/Collection/Exam (**DY) equals 0
- Length of **TEST > 40

Required Global Macro Variables (beyond reporting and debugging variables): _cstSubjectColumns

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_column.sas

%cstcheck_columncompare

%cstcheck_columncompare(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_columncompare

Supports comparison of column values (much like cstcheck_multicolumn), providing additional functionality in the form of step-level code (for example, optional reference to column metadata).

Note: Macro requires use of _cstCodeLogic at a SAS DATA step level (that is, a full DATA step or PROC SQL invocation). _cstCodeLogic creates a Work file (_cstproblems) that contains records in error.

Example validation checks that use this macro: **DOSE and **DOSU inconsistencies for expected columns

Required Global Macro Variables:

- _cstSubjectColumns
- _cstMetrics*
- <messaging, error>

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_columncompare.sas

%cstcheck_comparedomains

%cstcheck_comparedomains(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_comparedomains

Generally compares values for 1+ columns in one domain with values for those same columns in another domain. For example, USUBJID value in any domain does not have a matching USUBJID value in the DM domain.

Note: Macro requires use of _cstCodeLogic at a statement level in a SAS DATA step context. _cstCodeLogic identifies records in error by setting _cstError=1.

Example validation checks that use this macro: Unique USUBJID+VISIT+VISITNUM combinations in each domain not found in SV.

Required Global Macro Variables: (none)

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_comparedomains.sas

%cstcheck_dsmismatch

%cstcheck_dsmismatch(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_dsmismatch

Identifies any data set mismatches between study and template metadata and the source data library.

Note: This macro module currently ignores tablescope and columnscope in the _cstControl input data set.

Required Global Macro Variables: (none)

Required File Inputs: Single-record control data set identified by the control input parameter.

Parameters:

- _cstControl—The single observation data set containing check-specific metadata.

File: cstcheck_dsmismatch.sas

%cstcheck_metamismatch

%cstcheck_metamismatch(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_metamismatch

Identifies inconsistencies between study and reference column metadata.

Note: Macro requires use of _cstCodeLogic as a full SAS DATA step or PROC SQL invocation. This DATA step or PROC SQL step assumes as input a Work copy of the column metadata data set returned by the cstutil_buildcollist macro. Any resulting records in the derived data set represent errors to be reported.

ASSUMPTIONS:

- No data content is accessed for this check.
- Both study and reference metadata must be present to assess compliance.
- Current coding approach assumes no reporting on non-errors.

Example validation checks that use this macro include:

- Required column not found (Error).
- Expected column not found (Warning).
- Permissible column not found (Note).
- Column found in data set but not in specification.
- Supplemental qualifier data set without USUBJID column.
- Column metadata attribute differences (for example, type, length, label, order, CT, and so on).

Required Global Macro Variables: (none)

Required File Inputs: Single-record control data set identified by a control input parameter.

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_metamismatch.sas

%cstcheck_notconsistent

%cstcheck_notconsistent(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_notconsistent

Identifies any inconsistent column values across records.

Note: This macro requires use of _cstCodeLogic at a SAS DATA step level (that is, a full DATA step or PROC SQL invocation). _cstCodeLogic creates a Work file (_cstproblems) that contains records in error.

Example validation checks that use this macro include:

- **SEQ not consecutively incremented beginning at 1.
- Standard units inconsistent within **TESTCD across records.

Required Global Macro Variables:

- _cstSubjectColumns
- _cstMetrics*
- <messaging, error>

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_notconsistent.sas

%cstcheck_notimplemented

%cstcheck_notimplemented(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

Placeholder to report that a check has not yet been implemented.

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_notimplemented.sas

%cstcheck_notincodelist

%cstcheck_notincodelist(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_notincodelist

Identifies any column values inconsistent with controlled terminologies. For example, a **STAT value is found other than 'NOT DONE'.

Note: This macro requires reference to the SAS format search path built based on type=FMTSEARCH records in the SASReferences control file.

Processing is based on the value of the check metadata LOOKUPTYPE field. When LOOKUPTYPE=FORMAT, the code compares column values against a SAS format in the format search path. Code logic is optional (that is, if the user does not specify any code logic, then cstcheck_notincodelist uses default logic, which is PROC SQL code that creates work._cstproblems if one or more errors are detected). The SAS format is specified in the check metadata LOOKUPSOURCE field.

When LOOKUPTYPE=DATASET, the code requires the use of code logic to create the data set work._cstproblems. LOOKUPSOURCE must contain the reference data set (for example, MedDRA for AE preferred term lookups) used in code logic. Given that any reference dictionary with any structure might be used, it is the responsibility of the user to code correct joins and lookup logic in code logic.

When LOOKUPTYPE=CODELIST, functionality is deferred for SAS Clinical Standards Toolkit 1.3.

When LOOKUPTYPE=METADATA, the code compares column values against a SAS format in the format search path. Code logic is optional (that is, if the user does not specify any code logic, then cstcheck_notincodelist uses default logic, which is PROC SQL code that creates work._cstproblems if one or more errors are detected). The SAS format is specified in the source column metadata XMLCODELIST field.

Required Global Macro Variables: (none)

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_notincodelist.sas

%cstcheck_notsorted

%cstcheck_notsorted(_cstControl=);

[Exposure: external] [Macro Type: Validation check]

cstcheck_notsorted

Identifies any domain that is not sorted by the keys defined in the metadata.

Example validation check that uses this macro: Identifies domain table that is not correctly sorted.

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_notsorted.sas

%cstcheck_notunique

%cstcheck_notunique(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_notunique

This is a multi-function macro that assesses the uniqueness of data sets, columns, or value-pairs from two columns. Each of these three functions accesses different code sections within the macro.

Function 1: Is data set unique by a set of columns?

Data sets—It is assumed that if control column columnscope is blank, then code cycles through domains that are specified in control column tablescope. Code identifies any records that are not unique by the domain keys defined in the table-level metadata.

Multiple columns—This option allows the specification of a single set of columns (in the form var1+var2+...varn). Code identifies any records that are not unique by the specified set of columns within each domain specified in tablescope. For the purposes of reporting, the specified columns are treated as the domain keys. No code logic is used or currently checked.

Function 2: For any subject, are column values unique?

Single columns—For single columns (for example, **SEQ), code checks for uniqueness in USUBJID (except TSSEQ, in TSPARMCD). No code logic is used or currently checked.

Function 3: Does a combination of two columns have unique values?

Column pairs—For multiple columns (for example, **TEST and **TESTCD), code checks that there are a unique set of values for the pair of columns. These must be specified in the form of matching columnscope sublists. Exactly and only two sublists can be specified. No code logic is used or currently checked.

Function 4: Are the values in one column (Column2) consistent with the values in another column (Column1)?

Column pairs—For multiple columns (for example, **TESTCD and **STRESU), code checks that there is a unique value in Column2 for each value of Column1. These must be specified in the form of matching columnscope sublists. Exactly and only two sublists can be specified, with the first sublist containing Column1 (for example, VSTESTCD), and the second sublist containing Column2 (for example, VSSTRESU). Code logic is required. It is the presence of code logic that distinguishes Function 3 and Function 4 processing.

The columnscope sublists should be bounded by brackets in the following style:

[LBTEST+VSTEST][LBTESTCD+VSTESTCD]

The following limitations apply:

- The two lists must resolve to the same number of columns.
- The columns to be compared must be in the same data set.
- The first item in sublist 1 is paired with the first item in sublist 2, and so on.

The following are the example combinations of tablescope and columnscope:

```
tableScope columnScope        How code interprets *;
---------- -----------        -------------------------------------------
ALL                           For all domains, is each unique by its keys?
FINDINGS   [**TEST][**TESTCD] For all FINDINGS domains, **TEST and **TESTCD
                              must map 1:1
ALL        **SEQ              For all domains, check **SEQ for uniqueness
                              within USUBJID
DM                            Is DM unique by its keys (STUDYID+USUBJID)?
DV         [DVTERM][DVDECOD]  For DV, DVTERM and DVDECOD must map 1:1
SUPP**                        For all SUPP** domains, are records unique by
                              their keys?
DV         USUBJID+DVTERM     For DV, are records unique by USUBJID and
                              DVTERM?
```

Required Global Macro Variables:

- _cstSubjectColumns
- _cstMetrics*
- <messaging, error>

Required File Inputs: Single-record control data set identified by _cstControl input parameter.

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_notunique.sas

%cstcheck_recmismatch

%cstcheck_recmismatch(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_recmismatch

Identifies any record mismatches across domains.

Note: Macro requires use of _cstCodeLogic at a SAS DATA step level (that is, a full DATA step or PROC SQL invocation). _cstCodeLogic creates a Work file (_cstproblems) containing records in error.

Example CDISC SDTM validation checks that use this macro: Comments, Relrec, or Supplemental Qualifier RDOMAIN references to other domains or domain records that do not exist.

Required Global Macro Variables:

- _cstMetrics*
- <messaging, error>

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_recmismatch.sas

%cstcheck_recnotfound

%cstcheck_recnotfound(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_recnotfound

Generally compares the consistency of one or more columns across two tables. Or, it allows the comparison of the consistency of one <table>.<column> with another <table>.<column>. (For example, in CDISC SDTM, STUDYID in the TA domain does not match STUDYID in the DM domain).

Note: This macro requires the use of _cstCodeLogic at a statement level in a SAS DATA step context. _cstCodeLogic identifies records in error by setting _cstError=1.

Note: This macro requires that tablescope syntax specifies two sublists in the form [DM] [TA], comparing one or more columnscope fields across the tables in these sublists.

CDISC SDTM example validation check that uses this macro: DM subjects where no record for the subject is found in the DS table.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_recnotfound.sas

%cstcheck_violatesstd

%cstcheck_violatesstd(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_violatesstd

Identifies any invalid column value or values that are defined in a reference standard.

Note: This macro requires use of _cstCodeLogic at a statement level in a SAS DATA step context. _cstCodeLogic identifies records in errors by setting _cstError=1.

Example validation checks that use this macro include:

- Identifies a null value found in a column where core attribute is REQ.
- Identifies a null value found in a column where core attribute is EXP.
- A column character value is not correctly in uppercase.
- A numeric column that contains nonnumeric entries.

Required Global Macro Variables:

- _cstSubjectColumns—Currently used only with the SDTM model. CRT-DDS does not require this global macro. CRT-DDS does not use _cstMetricsNumSubj when running metrics (not subject based).

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_violatesstd.sas

%cstcheck_zeroobs

%cstcheck_zeroobs(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_zeroobs

Identifies any data set with zero observations.

Required Global Macro Variables: (none)

Required File Inputs: Single-record control data set identified by control input parameter.

Parameters:

- _cstControl—The single observation data set that contains check-specific metadata.

File: cstcheck_zeroobs.sas

%cstcheckutil_formatlookup

%cstcheckutil_formatlookup(_cstCol2=, _cstCol2Value=, _cstCol1=&_cstColumn, _cstDomOnly=, _cstDSN=&_cstDSName, _cstRowCt=&_cstDSRowCount, _cstC2Val=&_cstColumn2Value);

[Exposure: external] [Macro Type: SAS Clinical Standards Toolkit Validation Check Utility]

cstcheckutil_formatlookup

Creates work._cstproblems that contains any records that are included in the _cstSourceDS data set where the value of a column is not found in the format value column. For example, in the TS domain, TSPARMCD has a value of SEX. The $SEXPOP format is associated with this variable and has the following values: BOTH, F, and M. TSVAL has to contain one of these values to be correct. An error condition exists otherwise.

Note: This macro is called within _cstCodeLogic at a SAS DATA step level (that is, a full DATA step or PROC SQL invocation).

Required Global Macro Variables: (none)

Required File Inputs: Single-record control data set identified by control input parameter.

Parameters:

- _cstCol2—The variable that contains the value to check (TSPARMCD).
- _cstCol2Value—The actual value to check from _cstCol2.
- _cstCol1
- _cstDomOnly—The domain or table that contains _cstCol2.
- _cstDSN
- _cstRowCt
- _cstC2Val

File: cstcheckutil_formatlookup.sas

%cstutil_allocatesasreferences

%cstutil_allocatesasreferences / des='CST: Allocate sasreferences';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_allocatesasreferences

Method to allocate any librefs and filerefs in the SASReferences data set, and set the autocall and format search paths based on the SASReferences settings.

Must be called outside the context of a DATA step, typically as an initial step in any SAS Clinical Standards Toolkit driver program (for example, cst_validate).

Note: A call to a framework macro to validate the structure and content of the SASReferences data set is a required initial step.

Required Global Macro Variables:

- _cstResultsDS
- _cstSASRefsLoc (provides location of the SASReferences input file)
- _cstSASRefsName (provides name of the SASReferences input file)
- _cstSASRefs (Work library version of SASReferences)
- _cstFMTLibraries (include Work and Library in fmtsearch?)
- _cstMessageOrder (append or merge, where merge honors order precedence)

Required File Inputs: sasreferences.sas7bdat

File: cstutil_allocatesasreferences.sas

%cstutil_allocGlobalMetadataLib

%cstutil_allocGlobalMetadataLib(_cstLibname=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstLibname

File: cstutil_allocglobalmetadatalib.sas

%cstutil_appendresultds

%cstutil_appendresultds(_cstErrorDS=, _cstVersion=&_cstStandardVersion, _cstSource=&_cstCheckSource, _cstStdRef=, _cstOrderBy=);

[Exposure: internal] [Macro Type: Framework utility]

Appends a check-level work Results data set to the process work Results data set. Parameters passed are check-level, not record-level values.

Must be called outside the context of a DATA step.

Required File Inputs: (none)

Required Global Macro Variables: (none)

Parameters:

- _cstErrorDS—A SAS Work data set that contains one or more observations documenting the results of check-level validation processing on a source data set record level.
- _cstVersion—The specific version of the model. This defaults to the global _cstStandardVersion macro variable value. Used to look up an associated message from the Messages data set.
- _cstSource—The source of the check, allowing source-specific messaging. Used to look up an associated message from the Messages data set.
- _cstStdRef—Optional. Reference in standard supporting checks.
- _cstOrderBy—Optional. The order of the records is important, so specify the column order (SQL form, comma-separated columns) that the _cstErrorDS should have when exiting this macro.

File: cstutil_appendresultds.sas

%cstutil_buildcollist

%cstutil_buildcollist(_cstFormatType=DATASET, _cstColWhere=, _cstDomWhere=, _cstColDSName=&_cstColumnMetadata, _cstDomDSName=&_cstTableMetadata, _cstColSubOverride=N, _cstDomSubOverride=N);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_buildcollist

Builds a set of columns (in either list or data set format) based on the value from the validation check control file Validation_Control.columnscope.

The expected result is that the work._csttablemetadata and work._cstcolumnmetadata data sets are created and are in synchronization. This means that they are consistent with regard to the tables based on resolving the tablescope and columnscope check macro fields.

Rules used to interpret columnscope values (using mostly CDISC SDTM examples):

- Validation_Control.columnscope might be null.
- Blanks are translated to + (for example, LBDTC LBENDTC becomes LBDTC +LBENDTC).
- Value should not begin with a + or -.
- If the blank translation results in multiple + characters, then all but one of these characters are removed (for example, AE1 +DM1 becomes AE1++DM1, which becomes AE1+DM1).
- No attempt is made to assess the validity of the columnscope value (for example, **TEST-AE1 is allowed, although no change to the resolved set of **TEST columns occurs).
- The derived set of columns is built by parsing columnscope from left to columns).
- If <libref> is included, then it must be listed in the SASReferences.SASRef column.

Wildcard Conventions:

- must use the string **
- might appear as a suffix (for example, SUPP** for all columns that start with SUPP)
- might appear as a prefix (for example, **DTC for all columns that end with DTC)
- might appear alone (for example, **), equivalent to _ALL_
- <table>.** for all columns in the specified data set
- **.USUBJID for all USUBJID columns across referenced data sets
- sublists are delimited by brackets, and resolved lengths (that is, # columns) must be the same unless _cst*SubOverride is set to Y, and they must conform to non-sublist rules stated above
- A special naming convention of <column>:<value>, such as QUALIFIERS:DATETIME allows specification of a _cstColumnMetadata column and column value to subset columns. In this example, all _cstColumnMetadata.QUALIFIERS= 'DATETIME' columns are returned.

Sample columnscope values:

- _ALL_ (all columns)
- AESEQ (a single column)
- LBDTC+LBENDTC (multiple columns)
- QUALIFIERS:DATETIME (_cstColumnMetadata.QUALIFIERS='DATETIME')

- **TEST (all columns ending in TEST)
- DM** (all columns beginning with DM)
- **TEST+**TESTCD (all columns ending in TEST or TESTCD)
- [AESTDY+CMSTDY+EXSTDY][AEENDY+CMENDY+EXENDY] (two paired sublists)
- SRCDATA1.AE.AESTDY+SRCDATA2.AE.AESTDY (AESTDY column from AE data sets in two different libraries)
- AE.** (all columns in the AE table)
- **.USUBJID (all USUBJID columns from all tables)

Required Global Macro Variables (beyond reporting and debugging variables):

- _cstTableMetadata
- _cstColumnMetadata

Required File Inputs: work._cstcolumnmetadata

Parameters:

- _cstFormatType—If the value is LIST, it sets macro variables of # tables and space-delimited list of tables. The value DATASET is the default. Returns a data set of tables matching tablescope specification.
- _cstColWhere—WHERE clause to subset returned set of columns. Any WHERE clause is applied as the last step.
- _cstDomWhere—WHERE clause to subset returned set of tables. Any WHERE clause is applied as the last step.
- _cstColDSName—Name of data set with column metadata returned when _cstFormatType=DATASET.
- _cstDomDSName—Name of data set with table metadata returned when _cstFormatType=DATASET.
- _cstColSubOverride—Y or N (default). If Y, then overrides sublist processing to allow sublists of different lengths (such as columnScope=[**DTC][RFSTDTC]).
- _cstDomSubOverride—Y or N (default). If Y, then overrides sublist processing to allow sublists of different lengths (such as tableScope=[_ALL_-DM][DM]).

File: cstutil_buildcollist.sas

%cstutil_builddomlist

%cstutil_builddomlist(_cstFormatType=DATASET, _cstDomWhere=, _cstDomDSName=&_cstTableMetadata, _cstSubOverride=N);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_builddomlist

Builds set of tables (in either list or data set format) based on the value from the validation check control file Validation_Control.tablescope.

Rules used to interpret tablescope values (using mostly CDISC SDTM examples) include:

- Validation_Control.tablescope might not be null.
- Blanks are translated to + (for example, AE DM becomes AE+DM).
- Value should not begin with a + or -.

- If the blank translation results in multiple + characters, then all but one of the + characters are removed (for example, AE +DM becomes AE++DM, which becomes AE+DM).
- No attempt is made to assess the validity of the tablescope value (for example, CLASS:FINDINGS-AE is allowed, although no change to the resolved set of CLASS:FINDINGS tables occurs).
- The derived set of tables is built by parsing tablescope from left to right (for example, _ALL_-CLASS:RELATES builds a set of all tables removing RELREC and SUPP**).
- If <libref> is included, then it must be listed in the SASReferences.SASRef column.

Wildcard Conventions:

- must use the string **
- might appear as a suffix (for example, SUPP** for all tables that start with SUPP)
- might appear as a prefix (for example, **DM for all tables that end with DM)
- might appear alone (for example, **), equivalent to _ALL_
- <libref>.** for all tables in the specified library
- **.AE for all AE tables across referenced libraries
- sublists are delimited by brackets, and resolved lengths (that is, # columns) must be the same unless _cst*SubOverride is set to Y, and they must conform to non-sublist rules stated above
- A special naming convention of <column>:<value>, such as: CLASS:EVENTS allows specification of a _cstTableMetadata column and column value to subset tables. In this example, all CLASS='EVENTS' tables are returned.

Sample tablescope values:

- _ALL_ (all tables)
- AE (a single table)
- DM+DS (multiple tables)
- CLASS:EVENTS (_cstTableMetadata.CLASS='EVENTS')
- SUPP** (all Supplemental Qualifier tables)
- _ALL_-SUPP** (all tables except Supplemental Qualifier tables)
- [DM][EX] (two sublists comparing DM with EX)
- SRCDATA1.AE+SRCDATA2.AE (AE table from two different libraries)
- SRCDATA.** (all tables from the SRCDATA library)
- **.AE (all AE tables from all sourcedata libraries)

Required Global Macro Variables (beyond reporting and debugging variables):

_cstTableMetadata

Required File Inputs: none

Parameters:

- _cstFormatType—If the value is LIST, it sets macro variables of # tables and space-delimited list of tables. The value DATASET is the default. Returns a data set of tables matching tablescope specification.

- _cstDomWhere—WHERE clause to subset returned set of tables. Any WHERE clause is applied as the last step.
- _cstDomDSName—Name of data set returned when _cstFormatType=DATASET.
- _cstSubOverride—Y or N (default). If Y, then overrides sublist processing to allow sublists of different lengths (such as tableScope=[_ALL_-DM][DM]).

File: cstutil_builddomlist.sas

%cstutil_checkds

%cstutil_checkds(_cstdsname=, _csttype=, _cstsubtype=, _cststandard=*, _cststandardversion=*);

[Exposure: internal] [Macro Type: framework check]

cstutil_checkDS

Validates the structure of the data set against the template data set structure that is provided with the standard.

Required Global Macro Variables: assumes &_cstResultsDS macro is set to a valid two-level name.

Required File Inputs:

Parameters:

- _cstdsname—Required. The two-level name of the data set to validate.
- _csttype—Required. The type of data set to be created. This value comes from the TYPE column in the SASReferences file for the standard-version combination.
- _cstsubtype—Optional. Specifies the subtype for the type. This value comes from the SUBTYPE column in the SASReferences file for the standard-version combination. If the type has no subtypes, then this value can be omitted. Oherwise it must be provided.
- _cststandard—Optional. The name of the data standard to validate against. By default, all standards are included.
- _cststandardversion—Optional. The version of the data standard to validate against. By default, all values of standardversion are included.

File: cstutil_checkds.sas

%chkvals

%chkvals;

[Exposure: Not specified] [Macro Type: Not specified]

***********macro parameters not defined ************

Data set exists and has some records.

Dump contents of template table, compare it to the data set that is passed in.

Only keep those columns that have lookup values provided with the standard.

Load the list of columns to check into macro variable column.

Load the number of columns to check into macro variable numcol.

Separate which lookup columns have a dependency on the refcolumn in the lookup table.

Load the list of columns that have a dependency on a reference column into macro variable refcol.

Load the number of columns that depend on a refcolumn into macro variable numrefcol.

Sort the lookup table.

File: cstutil_checkds.sas

%cstutil_cleanupcstsession

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

[Exposure: internal] [Macro Type: Framework utility]

cstutil_cleanupcstsession

Cleans up after a SAS Clinical Standards Toolkit session, including removing any process-level SAS files and clearing the work.sasmacr catalog.

Most often used at the end of a SAS Clinical Standards Toolkit driver program, such as validate_data. Should be called where a DATA step or PROC is allowed.

Required Global Macro Variables:

- _cstDeBug
- _cstsasrefs
- _cstmessages

Parameters:

- _cstClearCompiledMacros—Remove all compiled macros from the work.sasmacr catalog. Values: 0 (No, default), 1 (Yes).
- _cstClearLibRefs—Deallocate all librefs and filerefs set based on the SASReferences content. Values: 0 (No, default), 1 (Yes).
- _cstResetSASAutos—Reset the autocall search path to its initial state. Values: 0 (No, default), 1 (Yes).
- _cstResetFmtSearch—Reset the format search path to its initial state. Values: 0 (No, default), 1 (Yes).
- _cstResetSASOptions—Reset SAS options to their initial state. Values: 0 (No), 1 (Yes, default).
- _cstDeleteFiles—Delete all SAS Clinical Standards Toolkit Work files and catalogs. Values: 0 (No), 1 (Yes, default). If _cstDebug=1, then files are not deleted even if _cstDeleteFiles=1.
- _cstDeleteGlobalMacroVars—Delete all SAS Clinical Standards Toolkit global macro variables set based on property filename or value pairs. Values: 0 (No, default), 1 (Yes).

File: cstutil_cleanupcstsession.sas

%cstutil_CreateMetadataReport

%cstutil_CreateMetadataReport(_cstStandardTitle=, _cstValidationDS=, _cstValidationDSWhClause=, _cstMessagesDS=, _cstStdRefDS=, _cstReportOutput=, _cstCheckMDReport=N, _cstMessageReport=N, _cstStdRefReport=N, _cstRecordView=N);

[Exposure: external] [Macro Type: Framework utility]

cstutil_createmetadatareport

Create a report documenting a SAS Clinical Standards Toolkit process, based on the Validation Master or Validation Control, Messages, and the Validation StdRef data sets.

Parameters:

- _cstStandardTitle—Optional. Title that defines the title2 statement for all reports.
- _cstValidationDS—Required. The validation data set that is used by a SAS Clinical Standards Toolkit process. This would be Validation Master or Validation Control, or a derivative provided by the user.
- _cstValidationDSWhClause—Optional. WHERE clause applied to _cstValidationDS.
- _cstMessagesDS—Required. The Messages data set used by a SAS Clinical Standards Toolkit process.
- _cstStdRefDS—The Validation StdRef data set created for a SAS Clinical Standards Toolkit standard. This file is required if _cstStdRefReport=Y.
- _cstReportOutput—The file that contains the report. Acceptable files are PDF, RTF, and HTML. The extension is used to determine ODS output.
- _cstCheckMDReport—Specifies whether panel 2 Check Details is run. Default is N.
- _cstMessageReport—Specifies whether panel 3 Message Details is run. Default is N.
- _cstStdRefReport—Specifies whether panel 4 Reference Information is run. Default is N.
- _cstRecordView—If Y, then a full listing of all available check metadata is generated, by check, in a single listing. Either this listing or the multi-panel report can be generated in a single invocation of this macro, but not both. Default is N.

File: cstutil_createmetadatareport.sas

%cstutil_createreport

%cstutil_createreport(_cstsasreferencesdset=, _cstresultsdset=&_cstRptResultsDS, _cstmetricsdset=&_cstRptMetricsDS, _cstreporterrorsonly=N, _cstreportobs=, _cstreportbytable=N, _csttablechecksdset=, _csttablecheckscode=, _cstkeeptablechecklist=N, _csttablesubset=, _cstreportoutput=, _cstsummaryReport=Y, _cstioReport=Y, _cstmetricsReport=Y, _cstgeneralResultsReport=Y, _cstcheckIdResultsReport=Y);

[Exposure: external] [Macro Type: Framework utility]

Creates a report documenting a SAS Clinical Standards Toolkit process, based on the Results and Metrics data sets generated by that process.

Parameters:

- _cstsasreferencesdset—The SASReferences data set used by a SAS Clinical Standards Toolkit process. Either this data set or the _cstresultsdset must exist.
- _cstresultsdset—The Results data set created by a SAS Clinical Standards Toolkit process. Either this data set or the _cstsasreferencesdset must exist.
- _cstmetricsdset—Optional. The Metrics data set created by a SAS Clinical Standards Toolkit process.
- _cstreporterrorsonly—(Y/N), If Y (default), then print only non-informational Results data set records.
- _cstreportobs—The number of Results data set records (per checkid) to be printed. If blank, then all records are printed.

- _cstreportbytable—Y/N. If N (default), then generate Report1 (by checkid) results. If Y, then generate Report2 (by table) results. Any value that is not equal to Y is assumed to be N.
- _csttablechecksdset—A data set providing a list of tables for each check. Use of this parameter assumes that this data set has been built before running this report.
- _csttablecheckscode—The code module (macro) to build _csttablechecksdset if it does not exist or is not passed as a parameter. Required only if _cstreportbytable=Y and _csttablechecksdset is not provided.
- _cstkeeptablechecklist—Y or N (default). If running Report2, then keep the derived list of tables (_csttablechecklist) to reuse in subsequent report requests. Building this file might take awhile. Any value that is not equal to Y is assumed to be N.
- _csttablesubset—Report 2 parameter, subset Results data set to specified source data set. If blank or _ALL_, then all records are printed. Example: DM.
- _cstreportoutput—Required. The path and filename where the report output is to be written.
- _cstsummaryReport—Specifies whether to generate Report Summary panel. Valid values are Y (default) and N. Any value that is not equal to N is assumed to be Y.
- _cstioReport—Specifies whether to generate Process Inputs/Outputs panel. Valid values are Y (default) and N. Any value that is not equal to N is assumed to be Y.
- _cstmetricsReport—Specifies whether to generate Process Metrics panel. Valid values are Y (default) and N. Any value that is not equal to N is assumed to be Y.
- _cstgeneralResultsReport—Specifies whether to generate General Process Reporting panel. Valid values are Y (default) and N. Any value that is not equal to N is assumed to be Y.
- _cstgeneralResultsReport—Specifies whether to generate General Process Reporting panel. Valid values are Y (default) and N. Any value that is not equal to N is assumed to be Y.
- _cstcheckIdResultsReport—Specifies whether to generate Process Results panel. Valid values are Y (default) and N. Any value that is not equal to N is assumed to be Y.

File: cstutil_createreport.sas

%cstutil_createTempMessages

%cstutil_createTempMessages(_cstCreationFlag=);

[Exposure: internal] [Macro Type: Framework]

Creates a temporary Messages data set using the CST-FRAMEWORK messages. If the Messages data set specified by the macro variable &_cstMessages does not exist, then this macro creates a temporary version. It looks for the default version of the SAS Clinical Standards Toolkit framework. It copies the Messages data set specified in the default SASReferences file to the name specified in the &_cstMessages macro variable. If the caller supplies the name of a macro variable in _cstCreationFlag, then this is set if the data set was created in this macro.

Parameters:

- _cstCreationFlag—Optional. The name of a macro variable that is set in the macro. It is set to 0 if the macro did not create the Messages data set (because it existed). It is set to 1 if this macro created the data set. It is strongly suggested that the caller use this variable to ensure that the temporary data set is cleaned up afterward.

File: cstutil_createtempmessages.sas

%cstutil_deleteDataSet

%cstutil_deleteDataSet(_cstDataSetName=);

[Exposure: internal] [Macro Type: standard_name]

Deletes a data set if it exists. _cst_rc is set to 0 if it is successful, and 1 otherwise. If the library is not assigned, or the data set does not exist, then this still returns 0.

Parameters:

- _cstDataSetName—Required. The (LIBNAME.)memname of the data set to be deleted.

File: cstutil_deletedataset.sas

%cstutil_getRandomNumber

%cstutil_getRandomNumber(_cstVarname=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstVarname

File: cstutil_getrandomnumber.sas

%cstutil_getsasreference

%cstutil_getsasreference(_cstStandard=, _cstStandardVersion=, _cstSASRefType=, _cstSASRefSubtype=, _cstSASRefsasref=, _cstSASRefmember=, _cstConcatenate=0, _cstFullname=0, _cstAllowZeroObs=0);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_getsasreference

Gets the row-level metadata from the SASReferences data set given the type and subtype.

Assumptions: SASReferences exists and has interpretable content.

Required Global Macro Variables (beyond reporting and debugging variables):

- _cstTableMetadata
- _cstColumnMetadata
- _cstSASRefs

Required File Inputs: SASReferences data set (as defined by &_cstSASRefs)

Parameters:

- _cstStandard—Identifies the name of a registered standard. If blank, then no subsetting by standard is attempted.
- _cstStandardVersion—Identifies the version of a registered standard. If blank, then no subsetting by version is attempted.
- _cstSASRefType—Required. File or data type from sasreferences.type. Representative values: autocall, control, fmtsearch, messages, properties, referencecontrol, referencemetadata, results, sourcedata, and sourcemetadata.
- _cstSASRefSubtype—Optional. File or data subtype from sasreferences.subtype. Values are specific to type. Some types do not have subtypes. Representative values: column, data, log, lookup, metrics, package, reference, results, table, and validation.
- _cstSASRefsasref—Identifies the calling macro variable name to populate with the value of sasreferences.sasref.

- _cstSASRefmember—Identifies the calling macro variable name to populate with the value of sasreferences.memname, based on the value of the _cstFullname parameter.
- _cstConcatenate—If 1, then return multiple row values, space delimited, for each macro variable requested (sasref, member).
- _cstFullname—If 1, then return full name from sasreferences.memname.
- _cstAllowZeroObs—If 1, then allow SASReferences to operate without warnings when a row that is requested is not found and returns zero observations. Default=0. Create warning when zero observations are encountered.

File: cstutil_getsasreference.sas

%cstutil_getsubjectcount

%cstutil_getsubjectcount(_cstDS=, _cstsubid=&_cstSubjectColumns);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_getsubjectcount

Part of metrics processing. Populates the Metrics global macro variable _cstMetricsCntNumSubj with the count of the number of subjects.

Called by any macro code module for which a count of the number of subjects is wanted.

Required Global Macro Variables (beyond reporting and debugging variables): _cstSubjectColumns (used by default for a null _cstsubid input parameter)

Required File Inputs: source data set to be processed (as parameter _cstDS)

Parameters:

- _cstDS—The source data set that contains subject data of interest.
- _cstsubid—The set of subject identifiers appropriate for the _cstDS.

File: cstutil_getsubjectcount.sas

%cstutil_internalmanageresults

%cstutil_internalmanageresults(_cstAction=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstAction

File: cstutil_internalmanageresults.sas

%cstutil_messagesdsattr

%cstutil_messagesdsattr /des='CST: Messages data set column attributes';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_messagesdsattr

Defines Messages data set column attributes.

Use: Statement level in a SAS DATA step, where a SAS ATTRIB statement might be used.

Required Global Macro Variables: (none)

Required File Inputs: (none)

File: cstutil_messagesdsattr.sas

%cstutil_metricsdsattr

%cstutil_metricsdsattr /des='CST: Metrics data set column attributes';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_metricsdsattr

Defines Metrics data set column attributes.

Use: Statement level in a SAS DATA step, where a SAS ATTRIB statement might be used.

Required Global Macro Variables: (none)

Required File Inputs: (none)

File: cstutil_metricsdsattr.sas

%cstutil_parsecolumnscope

%cstutil_parsecolumnscope(_cstscopestr=, _cstopsource=, _cstsublistnum=);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_parsecolumnscope

Parses input parameter strings to add or remove columns from the Work data set _cstColumnMetadata.

Called only by cstutil_buildcollist.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs:

- work._csttempcolumnmetadata
- work._cstcolumnmetadata

Parameters:

- _cstscopestr—The string value being parsed. Generally, this is the entire columnscope value if there are no sublists, or a specific sublist.
- _cstopsource—A modified string value used to populate the _cstRefValue macro value.
- _cstsublistnum—The sublist number in columnscope. If there is no sublist, then this is set to 1.

File: cstutil_parsecolumnscope.sas

%cstutil_parsescopesegment

%cstutil_parsescopesegment(_cstPart=, _cstVarName=, _cstMessageID=CST0004);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_parsescopesegment

Parses validation check metadata columns tablescope and columnscope to handle extended values such as <libref>.<table>.<column> and wildcarding to build a logical SAS code string to subset _cstTableMetadata and _cstColumnMetadata.

Called only by cstutil_parsecolumnscope and cstutil_parsetablescope.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs: (none)

Parameters:

- _cstPart—Which part of the tablescope or columnscope string is to be interpreted. Expected values are _cstLibPart, _cstTabPart, or _cstColPart.
- _cstVarName—The column name in either _csttablemetadata or _cstcolumnmetadata. Typical values are sasref, table, or column.
- _cstMessageID

File: cstutil_parsescopesegment.sas

%cstutil_parsetablescope

%cstutil_parsetablescope(_cstscopestr=, _cstopsource=, _cstsublistnum=);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_parsetablescope

Parses input parameter strings to add or remove tables from the Work data set _cstTableMetadata.

Called only by cstutil_builddomlist.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs:

- work._csttablemetadata
- work._csttemptablemetadata

Parameters:

- _cstscopestr—The string value being parsed. Generally, this is the entire tablescope value if there are no sublists, or a specific sublist.
- _cstopsource—A modified string value used to populate the _cstRefValue macro value.
- _cstsublistnum—The sublist number within tablescope. If there is no sublist, then this is set to 1.

File: cstutil_parsetablescope.sas

%cstutil_processsetup

%cstutil_processsetup(_cstSASReferencesSource=SASREFERENCES, _cstSASReferencesName=sasreferences, _cstSASReferencesLocation=);

[Exposure: external] [Macro Type: SAS Clinical Standards Toolkit Framework]

cstutil_processsetup

Set up model-specific study metadata.

The basic function of this code module is to set up study metadata when using the various SAS driver programs (for example, validate_data, cst_reports, and so on).

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs: (none)

Parameters:

- _cstSASReferencesSource—Setup should be based on what initial source? Valid values are SASREFERENCES (default) or RESULTS data set. If RESULTS, then no other parameters are required, and set up responsibility is passed to the cstutil_reportsetup macro.

- _cstSASReferencesName—The name of the SASReferences data set (default is SASREFERENCES).
- _cstSASReferencesLocation—The path (folder location) of the SASReferences data set (default is the path to the Work library).

File: cstutil_processsetup.sas

%cstutil_readcontrol

%cstutil_readcontrol /des="CST: Create control file macro variables";

[Exposure: internal] [Macro Type: Framework utility]

cstutil_readcontrol

To read a single Validation Control record, as passed in through the data set referenced by the _cstThisCheckDS global macro variable, and to create local macro variables for each column in the control file. These macro variables are available in the context of each specific check macro.

Called by each check macro.

Required Global Macro Variables: _cstThisCheckDS

Required File Inputs: Control file as stored in _cstThisCheckDS

File: cstutil_readcontrol.sas

%cstutil_reportgeneralprocess

%cstutil_reportgeneralprocess;

[Exposure: external] [Macro Type: Framework utility]

cstutil_reportinputsoutputs

Creates the General Process Reporting panel.

Parameters: (none)

File: cstutil_reportgeneralprocess.sas

%cstutil_reportinputsoutputs

%cstutil_reportinputsoutputs;

[Exposure: external] [Macro Type: Framework utility]

cstutil_reportinputsoutputs

Creates the Process Inputs/Outputs panel.

Parameters: (none)

File: cstutil_reportinputsoutputs.sas

%cstutil_reportprocessmetrics

%cstutil_reportprocessmetrics

[Exposure: external] [Macro Type: Framework utility]

cstutil_reportprocessmetrics

Creates the Process Metrics panel.

Parameters: (none)

File: cstutil_reportprocessmetrics.sas

%cstutil_reportprocessresults

%cstutil_reportprocessresults;

[Exposure: external] [Macro Type: Framework utility]

cstutil_reportprocessresults

Creates the Process Results panel.

Parameters: (none)

File: cstutil_reportprocessresults.sas

%cstutil_reportprocesssummary

%cstutil_reportprocesssummary;

[Exposure: external] [Macro Type: Framework utility]

cstutil_reportprocesssummary

Creates the Process Summary panel.

Parameters: (none)

File: cstutil_reportprocesssummary.sas

%cstutil_reportsetup

%cstutil_reportsetup(_cstRptType=Metadata);

[Exposure: external] [Macro Type: Framework utility]

cstutil_reportsetup

Performs a setup function for SAS Clinical Standards Toolkit reporting. If _cstSetupSrc=RESULTS, then the code interprets information from a Results data set referenced by the _cstRptResultsDS macro variable. Otherwise, the code interprets information from the SASReferences data set referenced by the _cstSASRefs global macro variable.

Parameters:

- _cstRptType—Identifies the type of report to be generated. Valid values include metadata (report on SAS Clinical Standards Toolkit validation check metadata) and results (report on SAS Clinical Standards Toolkit process results and metrics).

Assumptions:

_cstSASRefs global macro variable exists and specifies a valid SASReferences data set. (Either SASREFERENCES (default) or RESULTS).

File: cstutil_reportsetup.sas

%cstutil_reporttabledata

%cstutil_reporttabledata;

[Exposure: external] [Macro Type: Framework utility]

cstutil_reporttabledata

Creates work._cstrptresultsdom, which represents work._cstrptresults expanded to include records for each table applicable to the originally reported results.

Assumptions:

- This module is applicable only to Report2 and CDISC standards reporting table-level results (that is, CDISC SDTM and CDISC ADaM).

- This module includes a call to a CDSIC SDTM specific macro that only is known or found in a CDISC SDTM autocall path.

Parameters: (none)

File: cstutil_reporttabledata.sas

%cstutil_resultsdsattr

%cstutil_resultsdsattr /des='CST: Results data set column attributes';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_resultsdsattr

Defines Results data set column attributes.

Use: Statement level in a SAS DATA step, where a SAS ATTRIB statement might be used.

Required Global Macro Variables: (none)

Required File Inputs: (none)

File: cstutil_resultsdsattr.sas

%cstutil_resultsdskeep

%cstutil_resultsdskeep /des='CST: Results data set columns';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_resultsdskeep

Specifies Results data set columns to keep in a DATA step.

Use: Statement level in a SAS DATA step, where a SAS KEEP statement might be used.

Required Global Macro Variables: (none)

Required File Inputs: (none)

File: cstutil_resultsdskeep.sas

%cstutil_saveresults

%cstutil_saveresults(_cstIncludeValidationMetrics=0);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_saveresults

Saves process results to a file or files that are specified in SASReferences with type= RESULTS values.

Required Global Macro Variables:

- _cstMetricsDS
- _cstResultsDS

Parameters:

- _cstIncludeValidationMetrics—Specifies whether process results includes validation metrics. Valid values are 0 (No, default) and 1 (Yes).

File: cstutil_saveresults.sas

%cstutil_setcstgroot

%cstutil_setcstgroot;

[Exposure: Not specified] [Macro Type: Not specified]

File: cstutil_setcstgroot.sas

%cstutil_setmodel

%cstutil_setmodel /des="Set Which Model Definition to Use";

[Exposure: internal] [Macro Type: Framework utility]

cstutil_setmodel

To establish the comparison reference metadata for a check. This is based on the Validation_Control.usesourcemetadata flag. If this flag is Y, then sourcemetadata.* serves as the comparison metadata. Otherwise, reference metadata.* does.

Called for each check, but only by builddomlist and buildcollist macros.

Required Global Macro Variables (beyond reporting and debugging variables):

- _cstTableMetadata
- _cstColumnMetadata

Required File Inputs:

- Source tables and column metadata (derived from SASReferences)
- Reference tables and column metadata (derived from SASReferences)

Assumptions:

- While there should generally be only a single source of referencemetadata.* from the SASReferences control data set, the code allows multiple sources. These are concatenated in the derivation of the work._cstTableMetadata and work._cstColumnMetadata data sets.
- There might be multiple sources of sourcemetadata.* from the SASReferences control data set. These are concatenated in the derivation of the work._cstTableMetadata and work._cstColumnMetadata data sets.

File: cstutil_setmodel.sas

%cstutil_writecubexml

%cstutil_writecubexml(_cstXMLOut=, _cstMDPFile=, _cstDebug=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstXMLOut
- _cstMDPFile
- _cstDebug

File: cstutil_writecubexml.sas

%cstutil_writemetric

%cstutil_writemetric(_cstMetricParameter=, _cstResultID=, _cstResultSeqParm=, _cstMetricCnt=, _cstSrcDataParm=, _cstMetricsDSParm=&_cstMetricsDS);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_writemetric

Adds a single record to the Metrics data set based solely on parameter values.

Must be called outside the context of a DATA step.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs: &_cstMetricsDS (as parameter _cstMetricsDSParm)

Parameters:

- _cstMetricParameter—Metric parameter. Extensible set of metrics. Examples include: # of subjects, # of records tested, # of distinct check invocations, Errors (severity=High) reported, Warnings (severity=Medium) reported, Notes (severity=Low) reported, # of structural errors, and # of content errors. METRICS value.
- _cstResultID
- _cstResultSeqParm—Generally 1, unless duplicate values of resultid need to be distinguished, such as multiple invocations of the same validation checkid.
- _cstMetricCnt—Record counter for _cstMetricParameter.
- _cstSrcDataParm—Information to link metric back to source (for example, SDTM domain name or calling validation code module).
- _cstMetricsDSParm—The base (cross-check) Metrics data set to which this record is to be appended. By default, this is the data set referenced by the _cstMetricsDS global macro variable.

File: cstutil_writemetric.sas

%cstutil_writeresult

%cstutil_writeresult(_cstResultID=, _cstValCheckID=, _cstResultParm1=, _cstResultParm2=, _cstResultSeqParm=1, _cstSeqNoParm=1, _cstSrcDataParm=, _cstResultFlagParm=0, _cstRCParm=0, _cstActualParm=, _cstKeyValuesParm=, _cstResultDetails=, _cstResultsDSParm=&_cstResultsDS);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_writeresult

Adds a single record to the Results data set based solely on parameter values.

Must be called outside the context of a DATA step.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs:

- &_cstMessages (created by cstutil_allocatesasreferences)
- &_cstResultsDS (as parameter _cstResultsDSParm)

Parameters:

- _cstResultID—Set to validation process ID (for example, CST0017). Should have matching entry in Messages data set.
- _cstValCheckID—Validation check identifier from Validation Control data set.
- _cstResultParm1—An optional parameter to appear in first substitution field of the associated message with the same resultid.
- _cstResultParm2—An optional parameter to appear in second substitution field of the associated message with the same resultid.
- _cstResultSeqParm—Generally 1, unless duplicate values of resultid need to be distinguished, such as multiple invocations of the same validation checkid.

- _cstSeqNoParm—Sequence number within _cstResultSeqParm, beginning with 1 and incremented by 1 for each observation that is written to a data set.
- _cstSrcDataParm—Information to link result back to source (for example, SDTM domain name or calling validation code module).
- _cstResultFlagParm—Problem detected? Set to 0 if this is an informational rather than error record. A positive value indicates that an error was detected. A negative value indicates that the check failed to run for some reason.
- _cstRCParm—Value of _cst_rc at the point the result is written to data set.
- _cstActualParm—Source data value or values that are causing result to be written to data set.
- _cstKeyValuesParm—Information to link result back to a specific source data record (for example, data set key or XML row and column values).
- _cstResultDetails—Provides the ability to specify run-time details about the result. These take precedence over metadata result details.
- _cstResultsDSParm—The base (cross-check) result data set to which this record is to be appended. By default, this is the data set referenced by the _cstResultsDS global macro variable.

File: cstutil_writeresult.sas

Module SDTM V3.1.1 (Run Time)

Overview

This is the SDTM 3.1.1 run-time macro library.

Since: V1.2

Macro Summary

Table A3.3 Module SDTM V3.1.1 (Run Time) Macro Summary

Exposure	Macro
External SDTM Validation Process	%sdtm_validate /des='CST: Validate CDISC SDTM model files';
Internal SDTM Validation Check Utility	%sdtmcheckutil_recordlookup(_cstSourceDS=&_cstDSName, _cstSourceLib=&_cstRefOnly);
Internal Framework Utility	%sdtmutil_buildcheckdomainlist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=);
External Framework	%sdtmutil_buildsasreferences;

Exposure	Macro
External SDTM Tool	%sdtmutil_createsrcmetafromsaslib /des='CST: Create SDTM metadata from SAS library';
	%sdtmutil_getchecks(_cstControl=, _cstMeta=, _cstMsg=, _cstDomain="*", _cstOutDS=, _cstIncludeDraft=true);
Internal SDTM Utility	%sdtmutil_iso8601(_cstString=);
	%sdtmutil_listsettings(group=_ALL_);

Macro Detail

%sdtm_validate

%sdtm_validate /des='CST: Validate CDISC SDTM model files';

[Exposure: external] [Macro Type: SDTM Validation Process]

sdtm_validate

Validate CDISC SDTM model files.

The basic function of this code module is to cycle through the validation checks to be run, writing validation results to the process Results and Metrics data sets. These are persisted to any permanent location based on type=results records in the SASReferences file. Process cleanup is based on the _cstDebug global macro variable.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs: run-time (type=control,subtype=validation in SASReferences) check data set

File: sdtm_validate.sas

%sdtmcheckutil_recordlookup

%sdtmcheckutil_recordlookup(_cstSourceDS=&_cstDSName, _cstSourceLib=&_cstRefOnly);

[Exposure: internal] [Macro Type: SDTM Validation Check Utility]

sdtmcheckutil_recordlookup

Creates work._cstproblems that contains any records that are included in the _cstSourceDS data set that cannot be found in the referenced lookup data set. For example, SUPPAE includes a record that points to a record in the AE domain that does not exist with the key values specified.

Note: This macro is called in _cstCodeLogic at a SAS DATA step level (that is, a full DATA step or PROC SQL invocation).

Required Global Macro Variables: (none)

Parameters:

- _cstSourceDS—The source data set that is evaluated by the validation check.

- _cstSourceLib—The source libref for the lookup domain.

File: sdtmcheckutil_recordlookup.sas

%sdtmutil_buildcheckdomainlist

%sdtmutil_buildcheckdomainlist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=);

[Exposure: internal] [Macro Type: Framework utility]

sdtmutil_buildcheckdomainlist

Builds a data set that identifies the domains to be validated by each check based on the contents of the validation check data set columns tablescope and columnscope.

Required Global Macro Variables: (none)

Required File Inputs: Only as specified in the parameters

Parameters:

- _cstCheckDS—The validation check data set that contains the checks for a standard and standardversion. Typically, this is the Validation Master data set.
- _cstWhereClause—An optional WHERE clause to subset _cstCheckDS. The syntax should comply with a SAS statement argument such as any of the following: VAR1=1, upcase(var2)="Y", or checkstatus>0.
- _cstOutputDS—The output data set that is returned to the calling program. This data set contains a record for each domain that is referenced by a checkid, standardversion, and checksource.

File: sdtmutil_buildcheckdomainlist.sas

%sdtmutil_buildsasreferences

%sdtmutil_buildsasreferences;

[Exposure: Not specified] [Macro Type: Not specified]

File: sdtmutil_buildsasreferences.sas

%sdtmutil_createsrcmetafromsaslib

%sdtmutil_createsrcmetafromsaslib /des='CST: Create SDTM metadata from SAS library';

[Exposure: external] [Macro Type: SDTM Tool]

sdtmutil_createsrcmetafromsaslib

SAS library for a CDISC SDTM study.

The following source metadata files are used by the SAS Clinical Standards Toolkit to support CDISC SDTM validation and derivation of CDISC CRT-DDS (define.xml) files:

- source_tables
- source_columns
- source_study

The following is the general strategy that is used:

1. Use PROC CONTENTS output as the primary source of information.
2. Use reference_tables and reference_columns for matching columns.
3. Use class_columns as a generic source for metadata.

Note: This is only an attempted approximation of source metadata. No assumptions should be made that the results accurately represent the study data.

Assumptions:

- Source data is read from a single SAS library. The code can be modified to reference multiple libraries using library concatenation.
- Data set keys are estimated by the sort order of the source data (if set) and, if not, assumed based on the presence of columns SAS uses to define keys in the reference standard.
- For any unknown domain, the domain class (events, interventions, or findings) is estimated based on the presence of the class-specific topic variable (that is, _TERM (events), _TRT (interventions), and _TESTCD (findings)).
- Most column values in source_study are hardcoded because there is no metadata source. These values are used only to build the define.xml file. These are marked as <--- HARDCODE.

Limitations:

The following are two scenarios that have not yet been addressed:

- Split domains, such as QS**
- SDTM 3.1.2 FA multiple domains (for example, FACM)

File: sdtmutil_createsrcmetafromsaslib.sas

%sdtmutil_getchecks

%sdtmutil_getchecks(_cstControl=, _cstMeta=, _cstMsg=, _cstDomain=*, _cstOutDS=, _cstIncludeDraft=true);

[Exposure: Not specified] [Macro Type: Not specified]

- _cstControl
- _cstMeta
- _cstMsg
- _cstDomain
- _cstOutDS
- _cstIncludeDraft

File: sdtmutil_getchecks.sas

%sdtmutil_iso8601

%sdtmutil_iso8601(_cstString=);

[Exposure: internal] [Macro Type: SDTM utility]

sdtmutil_iso8601

Verifies whether a string is in a valid ISO 8601 format. The verification includes tests that are specific to SDTM and clinical trials data in general. Must be called from within a DATA step. It might be called more than once within a single DATA step.

Required Global Macro Variables: (none)

Required File Inputs: (none)

Parameters:

- _cstString—String that designates the name of the SAS data set variable that is being checked. Returns SAS data set variables. The programmer is expected to copy any values he or she wants to keep. All variables are automatically dropped at the end of the current data set.

  _cstISOisValid—Numeric. Binary flag that denotes whether the ISO string is a valid ISO 8601 string. Valid values are 0 (string is invalid) and 1 (string is valid).

  _cstISOrc—Numeric return code. A value of 0 indicates that no problems were found. Any other value is a coding error number.

  _cstISOmsg—String. A message that describes the validity of the input string.

  _cstISOinfo—String. An informational message that provides additional details about the string.

  _cstISOtype—String.

File: sdtmutil_iso8601.sas

%sdtmutil_listsettings

%sdtmutil_listsettings(group=_ALL_);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- group

File: sdtmutil_listsettings.sas

Module SDTM V3.1.2 (Run Time)

Overview

This is the SDTM 3.1.2 run-time macro library.

Since: V1.3

Macro Summary

Table A3.4 *Module SDTM V3.1.2 (Run Time) Macro Summary*

Exposure	Macro
External SDTM Validation Process	%sdtm_validate /des='CST: Validate CDISC SDTM model files';
Internal SDTM Validation Check Utility	%sdtmcheckutil_recordlookup(_cstSourceDS=&_cstDSName, _cstSourceLib=&_cstRefOnly);

Exposure	Macro
Internal Framework Utility	%sdtmutil_buildcheckdomainlist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=);
External SDTM Tool	%sdtmutil_createformatsfromcrtdds;
External SDTM Tool	%sdtmutil_createsasdatafromxpt /des='CST: Create SAS Data Sets from XPT';
External SDTM Tool	%sdtmutil_createsrcmetafromcrtdds /des='CST: Create SDTM metadata from CRTDDS Data';
External SDTM Tool	%sdtmutil_createsrcmetafromsaslib /des='CST: Create SDTM metadata from SAS library';
Internal SDTM Utility	%sdtmutil_iso8601(_cstString=);

Macro Detail

%sdtm_validate

%sdtm_validate /des='CST: Validate CDISC SDTM model files';

[Exposure: external] [Macro Type: SDTM Validation Process]

sdtm_validate

Validate CDISC SDTM model files.

The basic function of this code module is to cycle through the validation checks to be run, writing validation results to the process Results and Metrics data sets. These are persisted to any permanent location based on type=results records in the SASReferences file. Process cleanup is based on the _cstDebug global macro variable.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs: run-time (type=control,subtype=validation in SASsreferences) check data set

File: sdtm_validate.sas

%sdtmcheckutil_recordlookup

%sdtmcheckutil_recordlookup(_cstSourceDS=&_cstDSName, _cstSourceLib=&_cstRefOnly);

[Exposure: internal] [Macro Type: SDTM Validation Check Utility]

sdtmcheckutil_recordlookup

Creates work._cstproblems that contains any records that are included in the _cstSourceDS data set that cannot be found in the referenced lookup data set. For example, SUPPAE includes a record that points to a record in the AE domain that does not exist with the key values specified.

Note: This macro is called in _cstCodeLogic at a SAS DATA step level (that is, a full DATA step or PROC SQL invocation).

Required Global Macro Variables: (none)

Parameters:

- _cstSourceDS—The source data set that is evaluated by the validation check.
- _cstSourceLib—The source libref for the lookup domain.

File: sdtmcheckutil_recordlookup.sas

%sdtmutil_buildcheckdomainlist

%sdtmutil_buildcheckdomainlist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=);

[Exposure: internal] [Macro Type: Framework utility]

sdtmutil_buildcheckdomainlist

Builds a data set that identifies the domains to be validated by each check based on the contents of the validation check data set columns tablescope and columnscope.

Required Global Macro Variables: (none)

Required File Inputs: Only as specified in the parameters

Parameters:

- _cstCheckDS—The validation check data set that contains the checks for a standard and standardversion. Typically, this is the Validation Master data set.
- _cstWhereClause—An optional WHERE clause to subset _cstCheckDS. The syntax should comply with a SAS statement argument such as any of the following: VAR1=1, upcase(var2)="Y", or checkstatus>0.
- _cstOutputDS—The output data set that is returned to the calling program. This data set contains a record for each domain that is referenced by a checkid, standardversion, and checksource.

File: sdtmutil_buildcheckdomainlist.sas

%sdtmutil_createformatsfromcrtdds

%sdtmutil_createformatsfromcrtdds;

[Exposure: external] [Macro Type: SDTM Tool]

sdtmutil_createformatsfromcrtdds

This sample utility macro attempts to derive code lists from a CRT-DDS data library derived from define.xml file for a CDISC SDTM study.

The following source metadata files are used by the SAS Clinical Standards Toolkit to create code lists as provided in CDISC CRT-DDS (define.xml) files:

- codelists
- codelistitems
- clitemdecodetranslatedtext

The following is the general strategy that is used:

1. Combine CRT-DDS data to create the cntlin data set.

2. Read the cntlin data set using PROC FORMAT to create a format catalog.

Assumptions:

- Source data is read from a single SAS library. The code can be modified to reference multiple libraries using library concatenation.
- Only one study reference can be specified at this time. Multiple study references require modification of the code.

File: sdtmutil_createformatsfromcrtdds.sas

%sdtmutil_createsasdatafromxpt

%sdtmutil_createsasdatafromxpt /des='CST: Create SAS Data Sets from XPT';

[Exposure: external] [Macro Type: SDTM Tool]

sdtmutil_createsasdatafromxpt

This sample utility macro attempts to derive source metadata files from a CRT-DDS data library derived from define.xml file for a CDISC SDTM study.

The itemgroupleaf data set is used by this macro to generate a list of XPT files.

The following is the general strategy that is used:

1. Read itemgroupleaf to create a list of XPT files and generate SAS code to create SAS data sets using the XPORT LIBNAME option.
2. Submit generated code to create SAS data sets.

Assumptions:

- CRT-DDS data is read from a single SAS library.
- Currently, you must supply the libref for the location of the XPT files.

File: sdtmutil_createsasdatafromxpt.sas

%sdtmutil_createsrcmetafromcrtdds

%sdtmutil_createsrcmetafromcrtdds /des='CST: Create SDTM metadata from CRTDDS Data';

[Exposure: external] [Macro Type: SDTM Tool]

sdtmutil_createsrcmetafromcrtdds

This sample utility macro attempts to derive source metadata files from a CRT-DDS data library derived from define.xml file for a CDISC SDTM study.

The following source metadata files are used by the SAS Clinical Standards Toolkit to support CDISC SDTM validation and derivation of CDISC CRT-DDS (define.xml) files:

- source_tables
- source_columns
- source_study

The following is the general strategy that is used:

1. Use PROC CONTENTS output as the primary source of information.
2. Use reference_tables and reference_columns for matching columns.

Assumptions:

- Source data is read from a single SAS library. The code can be modified to reference multiple libraries using library concatenation.
- Only one study reference can be specified at this time. Multiple study references require modification of the code.

File: sdtmutil_createsrcmetafromcrtdds.sas

%sdtmutil_createsrcmetafromsaslib

%sdtmutil_createsrcmetafromsaslib /des='CST: Create SDTM metadata from SAS library';

[Exposure: external] [Macro Type: SDTM Tool]

sdtmutil_createsrcmetafromsaslib

SAS library for a CDISC SDTM study.

The following source metadata files are used by the SAS Clinical Standards Toolkit to support CDISC SDTM validation and derivation of CDISC CRT-DDS (define.xml) files:

- source_tables
- source_columns
- source_study

The following is the general strategy that is used:

1. Use PROC CONTENTS output as the primary source of information.
2. Use reference_tables and reference_columns for matching columns.
3. Use class_columns as a generic source for metadata.

Note: This is only an attempted approximation of source metadata. No assumptions should be made that the results accurately represent the study data.

Assumptions:

- Source data is read from a single SAS library. The code can be modified to reference multiple libraries using library concatenation.
- Data set keys are estimated by the sort order of the source data (if set) and, if not, assumed based on the presence of columns SAS uses to define keys in the reference standard.
- For any unknown domain, the domain class (events, interventions, or findings) is estimated based on the presence of the class-specific topic variable (that is, _TERM (events), _TRT (interventions), and _TESTCD (findings)).
- Most column values in source_study are hardcoded because there is no metadata source. These values are used only to build the define.xml file. These are marked as <--- HARDCODE.

Limitations:

The following are two scenarios that have not yet been addressed:

- Split domains, such as QS**
- SDTM 3.1.2 FA multiple domains (for example, FACM)

File: sdtmutil_createsrcmetafromsaslib.sas

%sdtmutil_iso8601

%sdtmutil_iso8601(_cstString=);

[Exposure: internal] [Macro Type: SDTM utility]

sdtmutil_iso8601

Verifies whether a string is in a valid ISO 8601 format. The verification includes tests that are specific to SDTM and clinical trials data in general. Must be called from within a DATA step. It might be called more than once within a single DATA step.

Required Global Macro Variables: (none)

Required File Inputs: (none)

Parameters:

- _cstString—String that designates the name of the SAS data set variable that is being checked. Returns SAS data set variables. The programmer is expected to copy any values he or she wants to keep. All variables are automatically dropped at the end of the current data set.

  _cstISOisValid—Numeric. Binary flag that denotes whether the ISO string is a valid ISO 8601 string. Valid values are 0 (string is invalid) and 1 (string is valid).

  _cstISOrc—Numeric return code. A value of 0 indicates that no problems were found. Any other value is a coding error number.

  _cstISOmsg—String. A message that describes the validity of the input string.

  _cstISOinfo—String. An informational message that provides additional details about the string.

  _cstISOtype—String.

File: sdtmutil_iso8601.sas

Module ODM V1.3.0 (Run Time)

Overview

This is the ODM V1.3.0 run-time macro library.

Since: V1.3

Macro Summary

Table A3.5 *Module ODM V1.3.0 (Run Time) Macro Summary*

Exposure	Macro
External Framework	%odm_getStatic(_cstName=, _cstVar=);
External CDISC ODM	%odm_read;

Macro Detail

%odm_getStatic

%odm_getStatic(_cstName=, _cstVar=);

[Exposure: Not Specified] [Macro Type: Not Specified]

Parameters:

- _cstName
- _cstVar

File: odm_getstatic.sas

%odm_read

%odm_read;

[Exposure: external] [Macro Type: CDISC-ODM]

Reads a CDISC ODM V1.3.0 XML file into the SAS representation of ODM 1.3.0.

This macro uses the SAS representation of a CDISC ODM XML file as source, and converts it into SAS data sets. The inputs and outputs are specified in a SASReferences file.

Required Global Macro Variables:

File: sdtmcheckutil_recordlookup.sas

- framework initialization properties
- CDISC ODM 1.3.0 initialization properties
- _cstResultsDS should point to an existing Results data set. Otherwise, work._cstResults is used.

File: odm_read.sas

Appendix 4
CDISC SDTM Validation Checks

The following table provides a complete list of CDISC SDTM validation checks. WebSDM V3.0 changed the IR numbers for 3.1.2, and all begin with 5000. These numbers are listed in parentheses.

Table A4.1 *Validation Checks*

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0001	Janus	IR4000 (IR5000)	Identifies domain table that has zero rows and, therefore, contains no data.	_ALL_		X	
SDTM0001	WebSDM	IR4000 (IR5000)	Identifies domain table that has zero rows and, therefore, contains no data.	_ALL_		X	X
SDTM0002	JanusFR	SAS0017	A load of data into Janus requires that the DM, DS, and EX domains be submitted for each study to be loaded.	DM+DS +EX		X	
SDTM0002	SAS	SAS0017	A load of data into JANUS requires that the DM, DS, and EX domains be submitted for each study to be loaded.	DM+DS +EX			X
SDTM0003	WebSDM	SAS0018	WebSDM and the SDTM model require only the DM domain be present.	DM		X	X
SDTM0004	SAS	SAS0033	Source metadata includes domain data set not found in reference metadata.	_ALL_		X	X
SDTM0005	SAS	SAS0034	Custom domain data set does not conform to specification naming guidelines.	_ALL_		X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0006	SAS	SAS0035	Source data library contains domain data not found in study metadata.	_ALL_		X	X
SDTM0011	Janus	IR4250	Identifies a column that was described in the domain description, but not included in the SAS data set for that domain.	_ALL_		X	
SDTM0011	WebSDM	IR4250 (IR5250)	Identifies a column that was described in the domain description, but not included in the SAS data set for that domain.	_ALL_		X	X
SDTM0012	JanusFR	IR4252	Identifies a column listed in the domain description as required (Req), but not included in the SAS data set for that domain.	_ALL_			X
SDTM0012	WebSDM	IR4252 (IR5252)	Identifies a column listed in the domain description as required (Req), but not included in the SAS data set for that domain.	_ALL_		X	X
SDTM0013	Janus	IR4253	Identifies a column listed in the domain description as expected (Exp), but not included in the SAS data set for that domain.	_ALL_		X	
SDTM0013	WebSDM	IR4253 (IR5253)	Identifies a column listed in the domain description as expected (Exp), but not included in the SAS data set for that domain.	_ALL_		X	X
SDTM0014	SAS	SAS0008	Identifies a column listed in the domain description as permissible (Perm), but not included in the SAS data set for that domain.	_ALL_		X	X
SDTM0015	Janus	IR4254	Identifies a column that appears in the SAS data set, but is not listed in the domain description.	_ALL_		X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0015	WebSDM	IR4254 (IR5254)	Identifies a column that appears in the SAS data set, but is not listed in the domain description.	_ALL_		X	X
SDTM0016	WebSDM	IR5260	Identifies a variable that is present in the SAS data set, but not present in the (study-specific) description file.	_ALL_			X
SDTM0017	Janus	IR4258	Identifies a domain that appears to contain supplemental qualifier data, but does not contain the unique subject identifier (USUBJID).	SUPP**		X	
SDTM0017	WebSDM	IR4258 (IR5258)	Identifies a domain that appears to contain supplemental qualifier data, but does not contain the unique subject identifier (USUBJID).	SUPP**		X	X
SDTM0019	JanusFR	IR4259	Identifies a variable where data type in (study-specific) description is not consistent with data type implicit in SAS data set.	_ALL_		X	
SDTM0019	WebSDM	IR4259 (IR5259)	Identifies a variable where data type in (study-specific) description is not consistent with data type implicit in SAS data set.	_ALL_		X	X
SDTM0020	SAS	SAS0006	Column order does not match standard.	_ALL_		X	X
SDTM0022	SAS	SAS0001	Column length < length defined in standard.	_ALL_		X	X
SDTM0023	SAS	SAS0002	Column length > length defined in standard.	_ALL_		X	X
SDTM0030	WebSDM	IR4264 (IR5264)	Column label inconsistent with label defined in standard.	_ALL_		X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0031	SAS	SAS0004	Column format found, but column not subject to controlled terminology.	_ALL_		X	X
SDTM0032	SAS	SAS0005	Column format found, but format name mismatch with standard controlled terminology name.	_ALL_		X	X
SDTM0033	WebSDM	IR4266 (IR5266)	Identifies a variable that has been deprecated according to the CDISC SDTM standard.	_ALL_		X	X
SDTM034	WebSDM	IR5251	Identifies a column where the expected data type that is inferred from the description file does not match the data type in the data set.	_ALL_			X
SDTM035	WebSDM	IR5261	Identifies a domain that is referenced in a description file, but for which there is no source data.	_ALL_			X
SDTM036	WebSDM	IR5262	Identifies a domain whose source data failed to load.	_ALL_			X
SDTM037	WebSDM	IR5263	Identifies a variable that uses an unsupported event dictionary.	_ALL_			X
SDTM038	WebSDM	IR5265	Identifies a variable whose referenced codelist is not properly defined in the associated define.xml file.	_ALL_			X
SDTM039	WebSDM	IR5267	Identifies a domain for which metadata has not been provided.	_ALL_			X
SDTM0101	JanusFR	IR4002	Identifies values that do not conform to the ISO 8601 standard for datetimes.	_ALL_	**DTC +**STDTC +**ENDTC +BRTHDTC +RFSTDTC +RFENDTC	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0101	WebSDM	IR4002 (IR5002)	Identifies values that do not conform to the ISO 8601 standard for datetimes.	_ALL_	**DTC +**STDTC +**ENDTC +BRTHDTC +RFSTDTC +RFENDTC	X	X
SDTM0102	JanusFR	IR4002	Identifies values that do not conform to the ISO 8601 standard for durations.	_ALL_	**DUR	X	
SDTM0102	WebSDM	IR4002 (IR5002)	Identifies values that do not conform to the ISO 8601 standard for durations.	_ALL_	**DUR	X	X
SDTM0124	Janus	IR4113	Identifies records that violate the condition [LENGTH(Name of Measurement, Test, or Examination (**TEST)) less than or equal to 40 characters].	CLASS:FINDINGS	**TEST	X	
SDTM0124	WebSDM	IR4113 (IR5113)	Identifies records that violate the condition [LENGTH(Name of Measurement, Test, or Examination (**TEST)) less than or equal to 40 characters].	CLASS:FINDINGS	**TEST	X	X
SDTM0125	Janus	IR4114	Identifies records that violate the condition [LENGTH(Sort Name of Measurement, Test, or Examination (**TESTCD)) less than or equal to 8 characters, cannot start with a number, or contain special characters].	CLASS:FINDINGS	**TESTCD	X	
SDTM0125	WebSDM	IR4114 (IR5114)	Identifies records that violate the condition [LENGTH(Sort Name of Measurement, Test, or Examination (**TESTCD)) less than or equal to 8 characters, cannot start with a number, or contain special characters].	CLASS:FINDINGS	**TESTCD	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0126	SAS	SAS0017	Qualifier variable label (QLABEL) length > 40.	SUPP**	QLABEL	X	X
SDTM0127	SAS	SAS0018	Qualifier variable name (QNAM) length > 8, starts with a number, or contains special characters.	SUPP**	QNAM	X	X
SDTM0128	Janus	IR4115	Identifies records that violate the condition [LENGTH(Trial Summary Parameter (**PARM)) less than or equal to 40 characters].	TS	TSPARM	X	
SDTM0128	WebSDM	IR4115 (IR5115)	Identifies records that violate the condition [LENGTH(Trial Summary Parameter (**PARM)) less than or equal to 40 characters].	TS	TSPARM	X	X
SDTM0129	Janus	IR4116	Identifies records that violate the condition [LENGTH(Trial Summary Parameter Sort Name (**PARMCD)) less than or equal to 8 characters, cannot start with a number, or contain special characters].	TS	TSPARMC D	X	
SDTM0129	WebSDM	IR4116 (IR5116)	Identifies records that violate the condition [LENGTH(Trial Summary Parameter Sort Name (**PARMCD)) less than or equal to 8 characters, cannot start with a number, or contain special characters].	TS	TSPARMC D	X	X
SDTM0130	OpenCDISC	SD1004	The value of planned arm code (ARMCD) must not be more than 20 characters in length.				X
SDTM0131	OpenCDISC	SD1009	The value of the element code (ETCD) must not be no more than 8 characters in length.				X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0190	WebSDM	IR5517	Identifies subjects with records that violate the condition [(Start date/time of adverse event less than or equal to start date/time of collection less than or equal to start date/time of disposition event) or (latest exposure date less than or equal to start date/time of disposition event)], limited to records where the cited variables are not null.	DS+AE +EG+LB +VS+EX			X
SDTM0191	OpenCDISC	SD0080	Start date/time of adverse event (AESTDTC) must be less than or equal to the start date/time of the latest disposition event (DSSTDTC).	AE	[AESTDTC] [DSSTDTC]		X
SDTM0192	OpenCDISC	SD0081	Date/time of collection (DTC) must be less than or equal to the start date/time of the latest disposition event (DSSTDTC).	EG+LB +VS	[**DTC] [DSSTDTC]		X
SDTM0193	OpenCDISC	SD0082	End of date/time of treatment (EXENDTC) must be less than or equal to the start date/time of the latest disposition event (DSSTDTC).	EX	[EXENDTC] [DSSTDTC]		X
SDTM0201	Janus	IR4001	Identifies a null (empty) value found in a column where (Standard) Core attribute is Req.	_ALL_		X	
SDTM0201	WebSDM	IR4001 (IR5001)	Identifies a null (empty) value found in a column where (Standard) Core attribute is Req.	_ALL_		X	X
SDTM0202	SAS	SAS0015	Identifies a null (empty) value found in a column where (Standard) Core attribute is Exp.	_ALL_		X	X
SDTM0203	SAS	SAS0010	Character column value is not correctly uppercased per specification.	_ALL_		X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0204	SAS	SAS0011	Character column value contains the numeric missing '.' or any special missing value like '.N'.	_ALL_		X	X
SDTM0205	SAS	SAS0012	Column value is not left-justified.	_ALL_		X	X
SDTM0206	Janus	IR4003	Identifies records where the value in the Domain Abbreviation column (DOMAIN) does not match the name of Domain.	_ALL_ -SUPP**-RELREC	DOMAIN	X	
SDTM0206	WebSDM	IR4003 (IR5003)	Identifies records where the value in the Domain Abbreviation column (DOMAIN) does not match the name of Domain.	_ALL_ -SUPP**-RELREC	DOMAIN	X	X
SDTM0207	Janus	IR4010	Identifies records where the value for Visit Number (VISITNUM) is formatted to more than three decimal places.	_ALL_	VISITNUM	X	
SDTM0207	WebSDM	IR4010 (IR5010)	Identifies records where the value for Visit Number (VISITNUM) is formatted to more than 3 decimal places.	_ALL_	VISITNUM	X	X
SDTM0208	Janus	IR4009	Identifies records where Result or Finding in Original Units (**ORRES) and Status (**STAT) both have a value, or where both are null and Derived Flag (**DRVFL) is not equal to 'Y'.	CLASS:FINDINGS-IE	[**ORRES] [**STAT]	X	
SDTM0208	WebSDM	IR4009	Identifies records where Result or Finding in Original Units (**ORRES) and Status (**STAT) both have a value, or where both are null and Derived Flag (**DRVFL) is not equal to 'Y'.	CLASS:FINDINGS-IE	[**ORRES] [**STAT]	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0209	JanusFR	IR4100	Identifies records that violate the condition [Study Day of Start of Observation (**STDY) less than or equal to Study Day of End of Observation (**ENDY)], limited to records where **STDY is not null and **ENDY is not null.	_ALL_-DS	[**STDY] [**ENDY]	X	
SDTM0209	WebSDM	IR4100 (IR5100)	Identifies records that violate the condition [Study Day of Start of Observation (**STDY) less than or equal to Study Day of End of Observation (**ENDY)], limited to records where **STDY is not null and **ENDY is not null.	_ALL_-DS	[**STDY] [**ENDY]	X	X
SDTM0210	JanusFR	IR4101	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **STDTC is not null and **ENDTC is not null.	_ALL_ -DS-LB	[**STDTC] [**ENDTC]	X	
SDTM0210	WebSDM	IR5101	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **STDTC is not null and **ENDTC is not null.	_ALL_ -DS-LB-PC-SV	[**STDTC] [**ENDTC]		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0210	WebSDM	IR4101	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **STDTC is not null and **ENDTC is not null.	_ALL_ -DS-LB	[**STDTC] [**ENDTC]	X	
SDTM0211	Janus	IR4130	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) or Start Relative to Reference Period (**STRF) is not null], limited to records where [End Date/Time of Observation (**ENDTC) or End Relative to Reference Period (**ENRF) is not null].	CM+SU	[**STRF] [**ENRF]	X	
SDTM0211	Janus	IR4130	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) or Start Relative to Reference Period (**STRF) is not null], limited to records where [End Date/Time of Observation (**ENDTC) or End Relative to Reference Period (**ENRF) is not null].	_ALL_ -DS-LB	[**STDTC] [**ENDTC]	X	
SDTM0211	WebSDM	IR4130	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) or Start Relative to Reference Period (**STRF) is not null], limited to records where [End Date/Time of Observation (**ENDTC) or End Relative to Reference Period (**ENRF) is not null].	CM+SU	[**STRF] [**ENRF]	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0211	WebSDM	IR5130	Identifies records that violate the condition [start date/time of observation (**STDTC) or start relative to reference period (**STRF) is not null], limited to records where [end date/time of observation (**ENDTC) or end relative to reference period (**ENDRF) is not null].	CE+CM +SU	[**STRTF] [**ENRF]		X
SDTM0211	WebSDM	IR4130	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) or Start Relative to Reference Period (**STRF) is not null], limited to records where [End Date/Time of Observation (**ENDTC) or End Relative to Reference Period (**ENRF) is not null].	_ALL_ -DS-LB	[**STDTC] [**ENDTC]	X	
SDTM0211	WebSDM	IR5130	Identifies records that violate the condition [start date/time of observation (**STDTC) or start relative to reference period (**STRF) is not null], limited to records where [end date/time of observation (**ENDTC) or end relative to reference period (**ENDRF) is not null].	_ALL_ -DS-LB-PC-SV	[**STDTC] [**ENDTC]		X
SDTM0212	Janus	IR4131	Identifies records that violate the condition [Planned Time Point Name (**TPT) is not null], limited to records where [Planned Time Point Number (**TPTNUM) is not null].	_ALL_	[**TPT] [**TPTNUM]	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0212	WebSDM	IR4131 (IR5131)	Identifies records that violate the condition [Planned Time Point Name (**TPT) is not null], limited to records where [Planned Time Point Number (**TPTNUM) is not null].	_ALL_	[**TPT] [**TPTNUM]	X	X
SDTM0213	Janus	IR4132	Identifies records that violate the condition [Planned Time Point Number (**TPTNUM) is not null], limited to records where [Planned Time Point Name (**TPT) is not null].	_ALL_	[**TPT] [**TPTNUM]	X	
SDTM0213	WebSDM	IR4132 (IR5132)	Identifies records that violate the condition [Planned Time Point Number (**TPTNUM) is not null], limited to records where [Planned Time Point Name (**TPT) is not null].	_ALL_	[**TPT] [**TPTNUM]	X	X
SDTM0214	Janus	IR4133	Identifies records that violate the condition [Time Point Reference (**TPTREF) is not null], limited to records where [Elapsed Time from Reference Point (**ELTM) is not null].	_ALL_-PP	[**TPTREF] [**ELTM]	X	
SDTM0214	WebSDM	IR4133 (IR5133)	Identifies records that violate the condition [Time Point Reference (**TPTREF) is not null], limited to records where [Elapsed Time from Reference Point (**ELTM) is not null].	_ALL_-PP	[**TPTREF] [**ELTM]	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0215	WebSDM	IR4117	Identifies records that violate the condition [End Relative to Reference Period (**ENRF) is not null], limited to records where [End Date/Time of Observation (**ENDTC) is null] and [Occurrence (**OCCUR) does not equal 'N'].	AE+CM +MH+SU	[**ENRF] [**ENDTC]	X	
SDTM0215	OpenCDISC	SD0021	Identifies records that violate the condition [End relative to reference period (**ENRF) is not null], limited to records where [end date/time of observation (**ENDTC) is null] and [occurrence (**OCCUR) does not equal 'N'].	AE+CE +CM+MH +SU	[**ENRF] [**ENDTC]		X
SDTM0216	WebSDM	IR4118	Identifies records that violate the condition [Start Relative to Reference Period (**STRF) is not null], limited to records where [Start Date/Time of Observation (**STDTC) is null] and [Occurrence (**OCCUR) does not equal 'N'].	CM+SU	[**STRF] [**STDTC]	X	
SDTM0216	OpenCDISC	SD022	Identifies records that violate the condition [start relative to reference period (**STRF) is not null], limited to records where [start date/time of observation (**STDTC) is null] and [occurrence (**OCCUR) does not equal 'N'].	CE+CM +SU	[**STRF] [**STDTC]		X
SDTM0217	JanusFR	IR4120	Identifies records that violate the condition [Evaluation Interval (**EVLINT) greater than or equal to 0], limited to records where **EVLINT is not null.	_ALL_	**EVLINT	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0217	WebSDM	IR4120	Identifies records that violate the condition [Evaluation Interval (**EVLINT) greater than or equal to 0], limited to records where **EVLINT is not null.	_ALL_	**EVLINT	X	
SDTM0218	Janus	IR4107	Identifies records that violate the condition [Status (**STAT) equals NOT DONE], limited to records where **STAT is not null.	_ALL_	**STAT	X	
SDTM0218	WebSDM	IR4107 (IR5107)	Identifies records that violate the condition [Status (**STAT) equals NOT DONE], limited to records where **STAT is not null.	_ALL_	**STAT	X	X
SDTM0219	Janus	IR4122	Identifies records that violate the condition [Reason Not Done (**REASND) is null], limited to records where [Status (**STAT) is null].	CM+EG +LB+MH +PE+QS +SC+SU +VS	[**REASN D][**STAT]	X	
SDTM0219	WebSDM	IR4122 (IR5122)	Identifies records that violate the condition [Reason Not Done (**REASND) is null], limited to records where [Status (**STAT) is null].	CM+EG +LB+MH +PE+QS +SC+SU +VS	[**REASN D][**STAT]	X	
SDTM0219	WebSDM	IR5122	Identifies records that violate the condition [reason not done (**REASND) is null], limited to records where [status (**STST) is null].	_ALL_	[**REASN D] [**STAT]		X
SDTM0220	Janus	IR4110	Identifies records that violate the condition [Duration (**DUR) greater than or equal to 0], limited to records where **DUR is not null.	_ALL_	**DUR	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0220	WebSDM	IR4110 (IR5110)	Identifies records that violate the condition [Duration (**DUR) greater than or equal to 0], limited to records where **DUR is not null.	_ALL_	**DUR	X	X
SDTM0221	Janus	IR4136	Identifies records where values are not found in the study-specific codelist attached to a variable.	_ALL_		X	
SDTM0221	WebSDM	IR4136 (IR5136)	Identifies records where values are not found in the study-specific codelist attached to a variable.	_ALL_		X	X
SDTM0222	Janus	IR4137	Identifies records that violate the condition [Study Day of Visit/ Collection/Exam (**DY) does not equal 0].	_ALL_	**DY +**STDY +**ENDY +VISITDY	X	
SDTM0222	WebSDM	IR4137 (IR5137)	Identifies records that violate the condition [Study Day of Visit/ Collection/Exam (**DY) does not equal 0].	_ALL_	**DY +**STDY +**ENDY +VISITDY	X	X
SDTM0223	SAS	SAS0030	Identifies records with the condition [Subcategory (**SCAT) is not null when category of related records (**CAT) is null].	AE+CM +DS+EG +EX+IE +LB+MH +QS+SC +SU+VS	[**SCAT] [**CAT]	X	
SDTM0223	SAS	SAS0030	Identifies records with the condition [subcategory (**SCAT) is not null when category of related records (**CAT) is null].	_ALL_-TI	[**SCAT] [**CAT]		X
SDTM0223	SAS	SAS0030	Identifies records with the condition [subcategory (**SCAT) is not null when category of related records (**CAT) is null].	TI	[IESCAT] [IECAT]		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0225	WebSDM	IR5162	Identifies records that violate the condition [result or finding in original units cannot be null unless status='NOT DONE'], limited to records where [derived flag does not equal 'Y'].	CLASS-FINDINGS-IE	[**ORRES] [**STAT]	X	X
SDTM0226	WebSDM	IR5163	Identifies records that violate the condition [if non-null result or finding in original units is provided, then status must be null].	CLASS-FINDINGS-IE	[**ORRES] [**STAT]	X	X
SDTM0231	OpenCDISC	SD1003	When age units (AGEU) are not null, then age (AGE) should be provided.	DM	[AGE] [AGEU]		X
SDTM0232	OpenCDISC	SD1010	When subjects experience for a particular period of time is represented as an unplanned element, where element code (ETCD) is equal to 'UNPLAN”, then the description element (ELEMENT) should be null.	SE	[ETCD] [ELEMENT]		X
SDTM0233	OpenCDISC	SD1019	For unplanned visits where the description of the unplanned visit (SVUPDES) is populated, the planned study day of visit (VISITDY) should be null.	SV	[SVUPDES] [VISITDY]		X
SDTM0251	Janus	IR4121	Identifies records that violate the condition [Toxicity Grade (**TOXGR) is a valid number], limited to records where **TOXGR is not null.	CLASS:EVENTS	**TOXGR	X	
SDTM0251	SAS	IR4121	Identifies records that violate the condition [toxicity grade (**TOXGR) is a valid number], limited to records where **TOXGR is not null.	_ALL_	**TOXGR	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0251	WebSDM	IR5121	Identifies records that violate the condition [Toxicity Grade (**TOXGR) is a valid number], limited to records where **TOXGR is not null.	_ALL_	**TOXGR		X
SDTM0251	WebSDM	IR4121	Identifies records that violate the condition [Toxicity Grade (**TOXGR) is a valid number], limited to records where **TOXGR is not null.	CLASS:EVENTS	**TOXGR	X	
SDTM0271	SAS	SAS0036	Value for column defined as a data set key is null.	_ALL_		X	X
SDTM0301	JanusFR	IR4104	Identifies records that violate the condition [End Relative to Reference Period (**ENRF) in (BEFORE , DURING , AFTER , DURING/AFTER , U)], limited to records where **ENRF is not null.	CLASS:EVENTS +CLASS:INTERVENTIONS	**ENRF	X	
SDTM0301	WebSDM	IR4104 (IR5104)	Identifies records that violate the condition [End Relative to Reference Period (**ENRF) in (BEFORE , DURING , AFTER , DURING/AFTER , U)], limited to records where **ENRF is not null.	CLASS:EVENTS +CLASS:INTERVENTIONS	**ENRF	X	X
SDTM0302	JanusFR	IR4106	Identifies records that violate the condition [Occurrence (**OCCUR) in (Y , N)], limited to records where **OCCUR is not null.	CLASS:EVENTS +CLASS:INTERVENTIONS	**OCCUR	X	
SDTM0302	WebSDM	IR4106 (IR5106)	Identifies records that violate the condition [Occurrence (**OCCUR) in (Y , N)], limited to records where **OCCUR is not null.	CLASS:EVENTS +CLASS:INTERVENTIONS	**OCCUR	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0303	JanusFR	IR4108	Identifies records that violate the condition [Start Relative to Reference Period (**STRF) in (BEFORE , DURING , AFTER ,'U')], limited to records where **STRF is not null.	CLASS:EVENTS +CLASS:INTERVENTIONS	**STRF	X	
SDTM0303	WebSDM	IR4108 (IR5108)	Identifies records that violate the condition [Start Relative to Reference Period (**STRF) in (BEFORE , DURING , AFTER ,'U')], limited to records where **STRF is not null.	CLASS:EVENTS +CLASS:INTERVENTIONS	**STRF	X	X
SDTM0351	JanusFR	IR4134	Identifies records that violate the condition [Dose units (**DOSU) is not null], limited to records where [Dose (**DOSE) is not null] and (Standard) Core attribute is 'Perm'.	CLASS:INTERVENTIONS	[**DOSE] [**DOSU]	X	
SDTM0351	WebSDM	IR4134 (IR5134)	Identifies records that violate the condition [Dose units (**DOSU) is not null], limited to records where [Dose (**DOSE) is not null] and (Standard) Core attribute is 'Perm'.	CLASS:INTERVENTIONS	[**DOSE] [**DOSU]	X	X
SDTM0352	JanusFR	IR4109	Identifies records that violate the condition [Dose (**DOSE) greater than or equal to 0], limited to records where **DOSE is not null.	CLASS:INTERVENTIONS	**DOSE	X	
SDTM0352	WebSDM	IR4109 (IR5109)	Identifies records that violate the condition [Dose (**DOSE) greater than or equal to 0], limited to records where **DOSE is not null.	CLASS:INTERVENTIONS	**DOSE	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0353	JanusFR	IR4138	Identifies records that violate the condition [Dose units (**DOSU) is not null], limited to records where [Dose (**DOSE) is not null] and (Standard) Core attribute is 'Exp'.	CLASS:INTERVENTIONS	[**DOSE] [**DOSU]	X	
SDTM0353	WebSDM	IR4138 (IR5138)	Identifies records that violate the condition [Dose units (**DOSU) is not null], limited to records where [Dose (**DOSE) is not null] and (Standard) Core attribute is 'Exp'.	CLASS:INTERVENTIONS	[**DOSE] [**DOSU]	X	X
SDTM0354	WebSDM	IR4139 (IR5139)	Identifies records that violate the condition [Related Domain (RDOMAIN) is not null].	SUPP** +RELREC	RDOMAIN	X	X
SDTM0355	SAS	SAS0040	Value for Related Domain (RDOMAIN) is inconsistent with data set name.	SUPP**-SUPPQUAL	RDOMAIN	X	X
SDTM0401	Janus	IR4102	Identifies records that violate the condition [Baseline Flag (**BLFL) either 'Y' or null].	CLASS:FINDINGS	**BLFL	X	
SDTM0401	WebSDM	IR4102 (IR5102)	Identifies records that violate the condition [Baseline Flag (**BLFL) either 'Y' or null].	CLASS:FINDINGS	**BLFL	X	X
SDTM0402	JanusFR	IR4103	Identifies records that violate the condition [Derived Flag (**DRVFL) either 'Y' or null].	CLASS:FINDINGS	**DRVFL	X	
SDTM0402	WebSDM	IR4103 (IR5103)	Identifies records that violate the condition [Derived Flag (**DRVFL) either 'Y' or null].	CLASS:FINDINGS	**DRVFL	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0403	JanusFR	IR4105	Identifies records that violate the condition [Fasting Status (**FAST) in (Y , N , U)], limited to records where **FAST is not null.	CLASS:FINDINGS	**FAST	X	
SDTM0403	WebSDM	IR4105 (IR5105)	Identifies records that violate the condition [Fasting Status (**FAST) in (Y , N , U)], limited to records where **FAST is not null.	CLASS:FINDINGS	**FAST	X	X
SDTM0405	Janus	IR4112	Identifies records that violate the condition [Result or Finding in Standard Format (**STRESC) is not null], limited to records where [Derived Flag (**DRVFL) equals 'Y'].	CLASS:FINDINGS-DA-IE-PE-PP-SC	[**STRESC][**DRVFL]	X	
SDTM0405	WebSDM	IR4112 (IR5112)	Identifies records that violate the condition [Result or Finding in Standard Format (**STRESC) is not null], limited to records where [Derived Flag (**DRVFL) equals 'Y'].	CLASS:FINDINGS-DA-IE-PE-PP-SC	[**STRESC][**DRVFL]	X	X
SDTM0406	Janus	IR4123	Identifies records that violate the condition [Date/Time of Collection (**DTC) is not null], limited to records where [End Date/Time of Observation (**ENDTC) is not null].	LB	[LBDTC] [LBENDTC]	X	
SDTM0406	WebSDM	IR4123	Identifies records that violate the condition [Date/Time of Collection (**DTC) is not null], limited to records where [End Date/Time of Observation (**ENDTC) is not null].	LB	[LBDTC] [LBENDTC]	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0406	WebSDM	IR5123	Identifies records that violate the condition [date/time of collection (**DTC) is not null], limited to records where [end date/time of observation (**ENDTC) is not null].	LB+MH +PC	[**DTC] [**ENDTC]		X
SDTM0407	JanusFR	IR4124	Identifies records that violate the condition [Date/Time of Collection (**DTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **DTC is not null and **ENDTC exists.	LB	[LBDTC] [LBENDTC]	X	
SDTM0407	WebSDM	IR4124	Identifies records that violate the condition [Date/Time of Collection (**DTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **DTC is not null and **ENDTC exists.	LB	[LBDTC] [LBENDTC]	X	
SDTM0407	WebSDM	IR5124	Identifies records that violate the condition [date/time of collection (**DTC) less than or equal to end date/time of observation (**ENDTC)], limited to records where **DTC is not null and **ENDTC exists.	LB+MH +PC	[**DTC] [**ENDTC]		X
SDTM0408	Janus	IR4125	Identifies records that violate the condition [Original units (**ORRESU) is not null], limited to records where [Result or Finding in Original Units (**ORRES) is not null].	CLASS:FINDINGS-IE	[**ORRES] [**ORRESU]	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0408	WebSDM	IR4125 (IR5125)	Identifies records that violate the condition [Original units (**ORRESU) is not null], limited to records where [Result or Finding in Original Units (**ORRES) is not null].	CLASS:FINDINGS-IE	[**ORRES] [**ORRESU]	X	X
SDTM0409	Janus	IR4126	Identifies records that violate the condition [Original units (**ORRESU) is null], limited to records where [Result or Finding in Original Units (**ORRES) is null].	CLASS:FINDINGS-IE	[**ORRES] [**ORRESU]	X	
SDTM0409	WebSDM	IR4126	Identifies records that violate the condition [Original units (**ORRESU) is null], limited to records where [Result or Finding in Original Units (**ORRES) is null].	CLASS:FINDINGS-IE	[**ORRES] [**ORRESU]	X	X
SDTM0410	JanusFR	IR4127	Identifies records that violate the condition [Normal Range Upper Limit-Standard Units (**STNRHI) greater than or equal to Normal Range Lower Limit-Standard Units (**STNRLO)], limited to records where **STNRHI is not null and **STNRLO is not null.	CLASS:FINDINGS	[**STNRHI] [**STNRLO]	X	
SDTM0410	WebSDM	IR4127	Identifies records that violate the condition [Normal Range Upper Limit-Standard Units (**STNRHI) greater than or equal to Normal Range Lower Limit-Standard Units (**STNRLO)], limited to records where **STNRHI is not null and **STNRLO is not null.	CLASS:FINDINGS	[**STNRHI] [**STNRLO]	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0410	WebSDM	IR5127	Identifies records that violate the condition [normal range upper limit-standard units (**STNRLO)], limited to records where **STNRHI is not null and **STNRLO is not null.	LB	[LBSTNRHI] [LBSTNRL O]		X
SDTM0411	SAS	SAS0029	Identifies records that violate the condition [Normal Range Upper Limit-Standard Units (**STNRHI) is null and Normal Range Lower Limit-Standard Units (**STNRLO) is null], or the condition [**STNRHI is not null and **STNRLO is not null].	CLASS:FI NDINGS	[**STNRHI] [**STNRLO]	X	X
SDTM0412	Janus	IR4128	Identifies records that violate the condition [Standard Units (**STRESU) is not null], limited to records where [Result or Finding in Standard Format (**STRESC) is not null].	CLASS:FI NDINGS-IE	[**STRESC] [**STRESU]	X	
SDTM0412	WebSDM	IR4128	Identifies records that violate the condition [Standard Units (**STRESU) is not null], limited to records where [Result or Finding in Standard Format (**STRESC) is not null].	CLASS:FI NDINGS-IE	[**STRESC] [**STRESU]	X	
SDTM0412	WebSDM	IR5128	Identifies records that violate the condition [standard units (**STRESU) are not null], limited to records where [result or finding in standard format (**STRESC) is not null].	CLASS:FI NDINGS-IE-PE	[**STRESC] [**STRESU]		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0413	Janus	IR4129	Identifies records that violate the condition [Standard Units (**STRESU) is null], limited to records where [Result or Finding in Standard Format (**STRESC) is null].	CLASS:FINDINGS-IE	[**STRESC] [**STRESU]	X	
SDTM0413	WebSDM	IR4129	Identifies records that violate the condition [Standard Units (**STRESU) is null], limited to records where [Result or Finding in Standard Format (**STRESC) is null].	CLASS:FINDINGS-IE	[**STRESC] [**STRESU]	X	
SDTM0413	WebSDM	IR5129	Identifies records that violate the condition [standard units (**STRESU) is null], limited to records where [result or finding in standard format (**STRESC) is null].	CLASS:FINDINGS-IE-PE	[**STRESC] [**STRESU]		X
SDTM0414	JanusFR	IR4135	Identifies records that violate the condition [Result or Finding in Standard Format (**STRESC) is not null], limited to records where [Result or Finding in Original Units (**ORRES) is not null].	CLASS:FINDINGS	[**ORRES] [**STRESC]	X	
SDTM0414	WebSDM	IR4135 (IR5135)	Identifies records that violate the condition [Result or Finding in Standard Format (**STRESC) is not null], limited to records where [Result or Finding in Original Units (**ORRES) is not null].	CLASS:FINDINGS	[**ORRES] [**STRESC]	X	X
SDTM0415	WebSDM	IR5143	Identifies records that violate the condition [if non-null occurrence (**OCCUR) is provided, then pre-specified (**PRESP) must equal 'Y'].	CE+CM+SU+MH	[**OCCUR] [**PRESP]		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0416	WebSDM	IR5144	Identifies records that violate the condition [if non-null occurrence (**OCCUR) is provided and pre-specified (**PRESP) equal 'Y', then completion status (**STAT) must equal 'NOT DONE'].	CLASS:EVENTS +CLASS:INTERVENTIONS	[**OCCUR] [**STAT]		X
SDTM0417	WebSDM	IR5145	Identifies records that violate the condition [treatment vehicle (**TRTV) is not null], limited to records where [treatment vehicle amount (**VAMT) is not null].	EX	[**TRTV] [**VAMT]		X
SDTM0418	WebSDM	IR5146	Identifies records that violate the condition [treatment vehicle amount units (**VAMTU) is not null], limited to records where [treatment vehicle amount (**VAMT) is not null].	EX	[**VAMTU] [**VAMT]		X
SDTM0419	WebSDM	IR5147	Identifies records that violate the condition [result or finding in standard format (**STRESC) is not null], limited to records where [result category (**RESCAT) is not null].	MB+MS	[**STRESC] [**RESCAT]		X
SDTM0422	WebSDM	IR5168	Identifies records that violate the condition [if non-null start relative to reference time point (**STRTPT) is provided, then start reference time point (**STTPT) must be non-null].	_ALL_	[**STRTPT] [**STTPT]		X
SDTM0423	WebSDM	IR5169	Identifies records that violate the condition [if non-null end relative to reference time point (**ENRTPT) is provided, then the end reference time point (**ENTPT) must be non-null].	_ALL_	[**ENRTPT] [**ENTPT]		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0441	WebSDM	IR5255	Identifies a sponsor-derived flag variable for 'treatment emergent AE' where the derivation cannot be executed.	SUPPAE	AETERM		X
SDTM0442	WebSDM	IR5256	Identifies a sponsor-derived flag variable for 'clinically significant lab' where the derivation cannot be executed.	SUPPLB	LBCLSIG		X
SDTM0443	WebSDM	IR5257	Identifies a sponsor-derived flag variable for 'clinically significant vital sign' where the derivation cannot be executed.	SUPPVS	VSCLSIG		X
SDTM0449	WebSDM	IR5141	Identifies user-defined codelist values that are not found in the corresponding SDTM codelist.	_ALL_			X
SDTM0450	SAS	SAS0037	Identifies records where the lookup value for a coded field (such as **DECOD, **BODSYS or **LOINC) could not be found in the associated dictionary.	_ALL_	**DECOD	X	X
SDTM0451	JanusFR	IR4007	Identifies records where the value for the preferred term could not be found in the MedDRA dictionary.	AE	AEDECOD	X	
SDTM0451	WebSDM	IR4007 (IR5007)	Identifies records where the value for the preferred term could not be found in the MedDRA dictionary.	AE	AEDECOD	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0452	Janus	IR4008	Identifies records where Serious Event (AESER)='Y' but none of Involves Cancer (AESCAN), Congenital Anomaly or Birth Defect (AESCONG), Persist or Signif Disability/ Incapacity (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'.	AE	AESER	X	
SDTM0452	WebSDM	IR4008 (IR5008)	Identifies records where Serious Event (AESER)='Y' but none of Involves Cancer (AESCAN), Congenital Anomaly or Birth Defect (AESCONG), Persist or Signif Disability/ Incapacity (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'.	AE	AESER	X	X
SDTM0453	JanusFR	R4019	Identifies records where value for [Serious Event (AESER)] is not found in codelist [YESNO].	AE	AESER	X	
SDTM0453	WebSDM	R4019 (IR5019)	Identifies records where value for [Serious Event (AESER)] is not found in Codelist [YESNO].	AE	AESER	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0454	JanusFR	R4023	Identifies records where value for [Congenital Anomaly or Birth Defect (AESCONG)] is not found in Codelist [YESNO], limited to records where AESCONG is not null.	AE	AESCONG	X	
SDTM0454	WebSDM	R4023 (IR5023)	Identifies records where value for [Congenital Anomaly or Birth Defect (AESCONG)] is not found in Codelist [YESNO], limited to records where AESCONG is not null	AE	AESCONG	X	X
SDTM0455	JanusFR	R4024	Identifies records where value for [Persist or Signif Disability/Incapacity (AESDISAB)] is not found in Codelist [YESNO], limited to records where AESDISAB is not null.	AE	AESDISAB	X	
SDTM0455	WebSDM	R4024 (IR5024)	Identifies records where value for [Persist or Signif Disability/Incapacity (AESDISAB)] is not found in Codelist [YESNO], limited to records where AESDISAB is not null.	AE	AESDISAB	X	X
SDTM0456	JanusFR	R4025	Identifies records where value for [Results in Death (AESDTH)] is not found in Codelist [YESNO], limited to records where AESDTH is not null.	AE	AESDTH	X	
SDTM0456	WebSDM	R4025 (R5025)	Identifies records where value for [Results in Death (AESDTH)] is not found in Codelist [YESNO], limited to records where AESDTH is not null.	AE	AESDTH	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0457	JanusFR	R4026	Identifies records where value for [Requires or Prolongs Hospitalization (AESHOSP)] is not found in Codelist [YESNO], limited to records where AESHOSP is not null.	AE	AESHOSP	X	
SDTM0457	WebSDM	R4026 (R5026)	Identifies records where value for [Requires or Prolongs Hospitalization (AESHOSP)] is not found in Codelist [YESNO], limited to records where AESHOSP is not null.	AE	AESHOSP	X	X
SDTM0458	JanusFR	R4027	Identifies records where value for [Is Life Threatening (AESLIFE)] is not found in Codelist [YESNO], limited to records where AESLIFE is not null.	AE	AESLIFE	X	
SDTM0458	WebSDM	R4027 (R5027)	Identifies records where value for [Is Life Threatening (AESLIFE)] is not found in Codelist [YESNO], limited to records where AESLIFE is not null.	AE	AESLIFE	X	X
SDTM0459	JanusFR	R4045	Identifies records where value for [Involves Cancer (AESCAN)] is not found in Codelist [YESNO], limited to records where AESCAN is not null.	AE	AESCAN	X	
SDTM0459	WebSDM	R4045 (R5045)	Identifies records where value for [Involves Cancer (AESCAN)] is not found in Codelist [YESNO], limited to records where AESCAN is not null.	AE	AESCAN	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0460	JanusFR	R4046	Identifies records where value for [Other Medically Important Serious Event (AESMIE)] is not found in Codelist [YESNO], limited to records where AESMIE is not null.	AE	AESMIE	X	
SDTM0460	WebSDM	R4046 (R5046)	Identifies records where value for [Other Medically Important Serious Event (AESMIE)] is not found in Codelist [YESNO], limited to records where AESMIE is not null.	AE	AESMIE	X	X
SDTM0461	JanusFR	R4047	Identifies records where value for [Occurred with Overdose (AESOD)] is not found in Codelist [YESNO], limited to records where AESOD is not null.	AE	AESOD	X	
SDTM0461	WebSDM	R4047 (R5047)	Identifies records where value for [Occurred with Overdose (AESOD)] is not found in Codelist [YESNO], limited to records where AESOD is not null.	AE	AESOD	X	X
SDTM0462	Janus	R4102	Identifies records that violate the condition [Results in Death (AESDTH)= Y], limited to records where [Outcome of Adverse Event (AEOUT)='FATAL'].	AE	[AESDTH] [AEOUT]	X	
SDTM0462	WebSDM	R4102 (R5102)	Identifies records that violate the condition [Results in Death (AESDTH)= Y], limited to records where [Outcome of Adverse Event (AEOUT)='FATAL'].	AE	[AESDTH] [AEOUT]	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0463	Janus	R4103	Identifies records that violate the condition [Outcome of Adverse Event (AEOUT)='FATAL'], limited to records where [Results in Death (AESDTH)='Y'].	AE	[AESDTH] [AEOUT]	X	
SDTM0463	WebSDM	R4103 (R5103)	Identifies records that violate the condition [Outcome of Adverse Event (AEOUT)='FATAL'], limited to records where [Results in Death (AESDTH)='Y'].	AE	[AESDTH] [AEOUT]	X	X
SDTM0464	JanusFR	R4043	Identifies records where value for [Concomitant or Additional Trtmnt Given (AECONTRT)] is not found in Codelist [YESNO], limited to records where AECONTRT is not null.	AE	AECONTR T	X	
SDTM0464	WebSDM	R4043 (R5043)	Identifies records where value for [Concomitant or Additional Trtmnt Given (AECONTRT)] is not found in Codelist [YESNO], limited to records where AECONTRT is not null.	AE	AECONTR T	X	X
SDTM0465	WebSDM	R5108	Identifies records where value for [action taken with study treatment (AEACN)] is not found in codelist [ACN], limited to records where AEACN is not null.	AE	AEACN		X
SDTM0466	WebSDM	R5109	Identifies records where value for [outcome of adverse event (AEOUT)] is not found in codelist [OUT], limited to records where AEOUT is not null.	AE	AEOUT		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0467	WebSDM	R5110	Identifies records where value for [severity or intensity (AESEV)] is not found in codelist [AESEV], limited to records where AESEV is not null.	AE	AESEV		X
SDTM0470	OpenCDISC	CT0003	Variable values should be populated with terms found in AGESPAN (C66780) CDISC controlled terminology codelist.	TS	TSVAL		X
SDTM0471	OpenCDISC	CT0005	Variable values should be populated with terms found in AGEU (C66781) CDISC terminology codelist.	TS	TSVAL		X
SDTM0472	OpenCDISC	CT0007	Variable values should be populated with terms found in drug accountability test name (C66731) CDISC controlled terminology codelist.	DA	DATEST		X
SDTM0473	OpenCDISC	CT0008	Variable values should be populated with terms found in drug accountability test code (C66732) CDISC controlled terminology codelist.	DA	DATESTCD		X
SDTM0474	OpenCDISC	CT0009	Variable values should be populated with terms found in 'Domain Abbreviation' (C66734) CDISC controlled terminology codelist.	_ALL_	DOMAIN		X
SDTM0475	OpenCDISC	CT0016	Variable values should be populated with terms found in evaluator (C78735) CDISC controlled terminology codelist.	CLASS:FINDINGS	**EVAL		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0476	OpenCDISC	CT0017	Variable values should be populated with terms found in evaluator (C78735) CDISC controlled terminology codelist.	SUPP**	QEVAL		X
SDTM0477	OpenCDISC	CT0024	Variable values should be populated with terms found in MARISTAT (C76348) CDISC controlled terminology codelist.	SC	SCSTRESC		X
SDTM0478	OpenCDISC	CT0026	Variable values should be populated with terms found in 'Reference Range Indicator' (C78736) CDISC controlled terminology codelist.	CLASS:FINDINGS	**NRIND		X
SDTM0479	OpenCDISC	CT0032	Variable values should be populated with terms found in ROUTE (C66729) CDISC controlled terminology codelist.	TS	TSVAL		X
SDTM0480	OpenCDISC	CT0035	Variable values should be populated with terms found in SEXPOP (C66732) CDISC controlled terminology codelist.	TS	TSVAL		X
SDTM0481	OpenCDISC	CT0037	Variable values should be populated with terms found in AGESPAN (C66780) CDISC controlled terminology codelist.	CLASS:FINDINGS +CLASS:EVENTS	**BODSYS		X
SDTM0482	OpenCDISC	CT0040	Variable values should be populated with terms found in TBLIND (C66735) CDISC controlled terminology codelist.	TS	TSVAL		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0483	OpenCDISC	CT0041	Variable values should be populated with terms found in TCNTRL (C66785) CDISC controlled terminology codelist.	TS	TSVAL		X
SDTM0484	OpenCDISC	CT0042	Variable values should be populated with terms found in TDIGRP (C66787) CDISC controlled terminology codelist.	TS	TSVAL		X
SDTM0485	OpenCDISC	CT0043	Variable values should be populated with terms found in TINDTP (C66736) CDISC controlled terminology codelist.	TS	TSVAL		X
SDTM0486	OpenCDISC	CT0045	Variable values should be populated with terms found in TPHASE (C66737) CDISC controlled terminology codelist.	TS	TSVAL		X
SDTM0487	OpenCDISC	CT0046	Variable values should be populated with terms found in Trial Summary Parameter Test Name (C67152) CDISC controlled terminology codelist.	TS	TSPARM		X
SDTM0488	OpenCDISC	CT0047	Variable values should be populated with terms found in Trial Summary Parameter Test Code (C66738) CDISC controlled terminology codelist.	TS	TSPARMCD		X
SDTM0489	OpenCDISC	CT0035	Variable values should be populated with terms found in TTYPE (C66739) CDISC controlled terminology codelist.	TS	TSVAL		X
SDTM0490	WebSDM	IR5150	Identifies records that violate the condition [Pre-specified (**PRESP) is either 'Y' or null].	_ALL_	**PRESP		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0491	WebSDM	IR5159	Identifies records that violate the condition [route of administration (**ROUTE) is in codelist ROUTE or is null].	CLASS:INTERVENTIONS	**ROUTE		X
SDTM0492	WebSDM	IR5164	Identifies records that violate the condition [position of subject during observation (**POS) is in codelist POSITION or is null].	CLASS:FINDINGS	**POS		X
SDTM0493	WebSDM	IR5165	Identifies records that violate the condition [start relative to reference time point (**STRTPT) is in codelist STRTPT or is null].	_ALL_	**STRTPT		X
SDTM0494	WebSDM	IR5166	Identifies records that violate the condition [end relative to reference time point (**ENRTPT) is in codelist ENRTPT or is null].	_ALL_	**ENRTPT		X
SDTM0495	WebSDM	IR5173	Identifies records that violate the condition [dose units (**DOSU) is in codelist UNIT or is null].	_ALL_	**DOSU		X
SDTM0496	WebSDM	IR5174	Identifies records that violate the condition [Original Units (**ORRESU) is in codelist UNIT or is null].	_ALL_-VS	**ORRESU		X
SDTM0497	WebSDM	IR5175	Identifies records that violate the condition [Standard Units (**STRESU) is in codelist UNIT or is null].	_ALL_-VS	**STRESU		X
SDTM0498	WebSDM	IR5176	Identifies records that violate the condition [location used for the measurement (**LOC) is in codelist LOC or is null].	_ALL_	**LOC		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0499	WebSDM	IR5177	Identifies records that violate the condition [dosing frequency per interval (**DOSFRQ) is in codelist FREQ or is null].	CLASS:INTERVENTIONS	**DOSFRQ		X
SDTM0500	WebSDM	IR4172 (R5172)	Identifies records that violate the condition [if arm code (ARMCD)='NOTASSGN' then description of arm (ARM) must equal 'Not Assigned', and vice versa].	DM+TA	[ARM] [ARMCD]	X	X
SDTM0501	Janus	IR4011	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFAIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa].	DM	[ARM] [ARMCD]	X	
SDTM0501	WebSDM	IR4011 (ir5011)	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFAIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa].	DM+TA	[ARM] [ARMCD]	X	X
SDTM0502	JanusFR	R4096	Identifies records that violate the condition [Subject Reference Start Date and Time (RFSTDTC) is not null], limited to records where upper(Arm Code (ARMCD)) does not equal 'SCRNFAIL'.	DM	[RFSTDTC] [ARMCD]	X	
SDTM0502	WebSDM	R4096 (R5096)	Identifies records that violate the condition [Subject Reference Start Date and /Time (RFSTDTC) is not null], limited to records where upper(Arm Code (ARMCD)) does not equal 'SCRNFAIL'.	DM	[RFSTDTC] [ARMCD]	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0503	JanusFR	R4097	Identifies records that violate the condition [Subject Reference End Date and Time (RFENDTC) is not null], limited to records where upper(Arm Code (ARMCD)) does not equal 'SCRNFAIL'.	DM	[RFENDTC] [ARMCD]	X	
SDTM0503	WebSDM	R4097 (R5097)	Identifies records that violate the condition [Subject Reference End Date and Time (RFENDTC) is not null], limited to records where upper(Arm Code (ARMCD)) does not equal 'SCRNFAIL'.	DM	[RFENDTC] [ARMCD]	X	X
SDTM0504	JanusFR	R4007	Identifies records where value for [SEX] is not found in codelist [SEX].	DM	SEX	X	
SDTM0504	WebSDM	R4007 (R5007)	Identifies records where value for [SEX] is not found in codelist [SEX].	DM	SEX	X	X
SDTM0505	Janus	R4008	Identifies records where value for [COUNTRY] is not found in codelist [COUNTRY].	DM	COUNTRY	X	
SDTM0505	WebSDM	R4008 (R5008)	Identifies records where value for [COUNTRY] is not found in codelist [COUNTRY].	DM	COUNTRY	X	X
SDTM0506	JanusFR	R4006	Identifies records that violate the condition [age (AGE) greater than or equal to 0], limited to records where AGE is not null.	DM	AGE	X	
SDTM0506	WebSDM	R4006 (R5006)	Identifies records that violate the condition [age (AGE) greater than or equal to 0], limited to records where AGE is not null.	DM	AGE	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0507	Janus	R4106	Identifies records that violate the condition [age units (AGEU) is not null], limited to records where AGE is not null.	DM	[AGE] [AGEU]	X	
SDTM0507	WebSDM	R4106 (R5106)	Identifies records that violate the condition [age units (AGEU) is not null], limited to records where AGE is not null.	DM	[AGE] [AGEU]	X	X
SDTM0508	JanusFR	R4062	Identifies records where value for [age units (AGEU)] is not found in codelist [AGEUNITS2], limited to records where AGEU is not null.	DM	AGEU	X	
SDTM0508	WebSDM	R4062 (R5062)	Identifies records where value for [age units (AGEU)] is not found in codelist [AGEUNITS2], limited to records where AGEU is not null.	DM	AGEU	X	X
SDTM0509	WebSDM	R5113	Identifies records where value for [ethnicity (ETHNIC)] is not found in codelist [ETHNIC], limited to records where ETHNIC is not null.	DM	ETHNIC		X
SDTM0510	WebSDM	R5130	Identifies records where value for [race] is not found in codelist [RACE], limited to records where [race is not null].	DM	RACE		X
SDTM0511	WebSDM	R5121	Identifies records that violate the condition [category for disposition event (DSCAT) = 'DISPOSITION EVENT'], limited to records where [epoch (EPOCH) is not null].	DS	[DSCAT] [EPOCH]		X
SDTM0512	WebSDM	R5122	Identifies records where value for [category for disposition event (DSCAT)] is not found in codelist [DSCAT].	DS	DSCAT		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0513	WebSDM	R5131	Identifies records where value for [subject characteristic short name (SCTESTCD)] is not found in codelist [SCCD].	SC	SCTESTCD		X
SDTM0514	WebSDM	R5126	Identifies records where value for [dose form (CMDOSFRM)] is not found in codelist [FRM], limited to records where [CMDOSFRM is not null].	CM	CMDOSFRM		X
SDTM0515	WebSDM	R5123	Identifies records where value for [ECG Test or Examination Name (EGTEST)] is not found in Codelist [EGTEST].	EG	EGTEST		X
SDTM0516	WebSDM	R5124	Identifies records where value for [ECG test or examination short name (EGTESTCD)] is not found in codelist [EGTESTCD].	EG	EGTESTCD		X
SDTM0517	WebSDM	R5125	Identifies records where value for [method of ECG rest (EGMETHOD)] is not found in codelist [EGMETHOD], limited to records where [EGMETHOD is not null].	EG	EGMETHOD		X
SDTM0518	WebSDM	R5129	Identifies records where value for [character result or finding in standard format (EGSTRESC)] is not found in codelist [EGSTRESC], limited to records where [EGSTRESC is not null].	EG	EGSTRESC		X
SDTM0521	Janus	IR4119	Identifies records that violate the condition [Planned Elapsed Time from Reference Point (**ELTM) greater than or equal to 0], limited to records where **ELTM is not null.	EX	EXELTM	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0521	WebSDM	IR4119 (IR5119)	Identifies records that violate the condition [Planned Elapsed Time from Reference Point (**ELTM) greater than or equal to 0], limited to records where **ELTM is not null.	EX	EXELTM	X	
SDTM0522	WebSDM	R5127	Identifies records where value for [dose form (EXDOSFRM)] is not found in codelist [FRM], limited to records where [EXDOSFRM is not null].	EX	EXDOSFRM		X
SDTM0523	WebSDM	R5128	Identifies records where value for [treatment vehicle amount units (EXVAMTU)] is not found in codelist [UNIT], limited to records where [EXVAMTU is not null].	EX	EXVAMTU		X
SDTM0531	JanusFR	R4031	Identifies records where value for [Inclusion or Exclusion Category (IECAT)] is not found in codelist [INCEX], limited to records where IECAT is not null.	IE	IECAT	X	
SDTM0531	WebSDM	R4031 (R5031)	Identifies records where value for [Inclusion or Exclusion Category (IECAT)] is not found in codelist [INCEX], limited to records where IECAT is not null.	IE	IECAT	X	X
SDTM0532	JanusFR	R4071	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES)] is not found in codelist[YESNO], limited to records where IEORRES is not null.	IE	IEORRES	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0532	WebSDM	R4071 (R5071)	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES)] is not found in codelist[YESNO], limited to records where IEORRES is not null.	IE	IEORRES	X	X
SDTM0533	JanusFR	R4072	Identifies records that violate the condition [I/E Criterion Original Result in Standard Format (IESTRESC)] is not found in codelist[YESNO], limited to records where IESTRESC is not null.	IE	IESTRESC	X	
SDTM0533	WebSDM	R4072 (R5072)	Identifies records that violate the condition [I/E Criterion Original Result in Standard Format (IESTRESC)] is not found in codelist[YESNO], limited to records where IESTRESC is not null.	IE	IESTRESC	X	X
SDTM0534	Janus	R4073	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES) = I/E Criterion Original Result in Std Format (IESTRESC)].	IE	[IEORRES] [IESTRESC]	X	
SDTM0534	WebSDM	R4073 (R5073)	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES) = I/E Criterion Original Result in Std Format (IESTRESC)].	IE	[IEORRES] [IESTRESC]	X	X
SDTM0541	Janus	R4105	Identifies records that violate the condition [Description of Unplanned Element (SEUPDES) is not null], limited to records where Subject Element Code (ETCD) ='UNPLAN'.	SE	[SEUPDES] [ETCD]	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0541	WebSDM	R4105 (R5105)	Identifies records that violate the condition [Description of Unplanned Element (SEUPDES) is not null], limited to records where Subject Element Code (ETCD) ='UNPLAN'.	SE	[SEUPDES] [ETCD]	X	X
SDTM0551	JanusFR	IR4012	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFAIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa].	TA	[ARM] [ARMCD]	X	
SDTM0551	WebSDM	IR4012 (IR5012)	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFAIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa].	TA	[ARM] [ARMCD]	X	
SDTM0561	Janus	R4101	Identifies records that violate the condition [Rule for End of Element (TEENRL) is not null or Planned Duration of Element (TEDUR) is not null].	TE	[TEENRL] [TEDUR]	X	
SDTM0561	WebSDM	R4101 (R5101)	Identifies records that violate the condition [Rule for End of Element (TEENRL) is not null or Planned Duration of Element (TEDUR) is not null].	TE	[TEENRL] [TEDUR]	X	X
SDTM0562	OpenCDISC	SD1008	When comments are related to a specific parent record or group of parent records in a domain, then the value of Date and Time of Comment (CODTC) should be null because the timing of the parent record or records is inherited by the comment record.	CO	CODTC		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0570	WebSDM	R5114	Identifies records where value for [lab test or examination name (LBTEST)] is not found in codelist [LBTEST].	LB	LBTEST		X
SDTM0571	WebSDM	R5115	Identifies records where value for [lab test or examination code (LBTESTCD)] is not found in codelist [LBTESTCD].	LB	LBTESTCD		X
SDTM0572	WebSDM	R5116	Identifies records where value for [original units (VSORRESU)] is not found in codelist [VSRESU], limited to records where [VSORRESU is not null].	VS	VSORRESU		X
SDTM0573	WebSDM	R5117	Identifies records where value for [character result or finding in std format (VSSTRESC)] is not found in codelist [SIZE], limited to records where [vital signs test short name (VSTESTCD) = 'FRMSIZE'].	VS	VSSTRESC		X
SDTM0574	WebSDM	R5118	Identifies records where value for [standard units (VSSTRESU)] is not found in codelist [VSRESU], limited to records where [VSSTRESU is not null].	VS	VSSTRESU		X
SDTM0575	WebSDM	R5119	Identifies records where value for [vital signs test name (VSTEST)] is not found in codelist [VSTEST].	VS	VSTEST		X
SDTM0576	WebSDM	R5120	Identifies records where value for [vital signs test short name (VSTESTCD)] is not found in codelist [VSTESTCD].	VS	VSTESTCD		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0580	WebSDM	R5112	Variable values should be populated with terms found in completion/ reason for non-completion (C66727) CDISC controlled terminology codelist.	DS	DSDECOD		X
SDTM0601	SAS	SAS0013	Domain not sorted by keys as defined in standard.	_ALL_		X	X
SDTM0602	SAS	SAS0007	Records are not unique by the expected keys	_ALL_		X	X
SDTM0603	JanusFR	IR4004	Identifies records where non-unique values for Sequence Number variable (**SEQ) exist within a subject.	_ALL_-TS	**SEQ	X	
SDTM0603	WebSDM	IR4004 (IR5004)	Identifies records where non-unique values for Sequence Number variable (**SEQ) exist within a subject.	_ALL_-TS	**SEQ	X	X
SDTM0604	SAS	SAS0009	Sequence Number (**SEQ) values are not consecutively incremented beginning at 1 for each USUBJID.	TS	TSSEQ	X	X
SDTM0604	SAS	SAS0009	Sequence Number (**SEQ) values are not consecutively incremented beginning at 1 for each USUBJID.	_ALL_-TS	**SEQ	X	X
SDTM0605	SAS	SAS0014	Report any variable for the domain that contains all missing or null values.	_ALL_	_ALL_	X	X
SDTM0606	JanusFR	SAS0022	Identify any column defined as numeric in the standard that contains non-numeric values.	_ALL_		X	
SDTM606	SAS	SAS0022	Identify any columns defined as numeric in the standard that contains non-numeric values.	_ALL_			X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0607	JanusFR	SAS0038	Site Study Identifier (SITEID) is null for all records.	DM	SITEID	X	
SDTM0607	SAS	SAS0038	Site study identifier (SITEID) is null for all records.	DM	SITEID		X
SDTM0621	WebSDM	IR4005 (IR5005)	Identifies subjects where there are no records with a value of 'Y' in the baseline flag variable (**BLFL), excluding Arm Code (ARMCD)='SCRNFAIL'.	EG+LB +QS+VS	**BLFL	X	X
SDTM0622	WebSDM	IR4142 (IR5142)	Inconsistency between test (**TEST) and test code (**TESTCD).	CLASS:FINDINGS	[**TEST] [**TESTCD]	X	X
SDTM0623	SAS	SAS0027	Identifies Test Code (**TESTCD) values where Standard Units (**STRESU) value is not consistent across all records.	CLASS:FINDINGS-IE-PE	[**TESTCD] [**STRESU]	X	X
SDTM0631	JanusFR	IR4006	Identifies Short Name of Measurement, Test, or Examination (**TESTCD) values where Standard Units (**STRESU) value is not consistent across all records.	EG+LB +QS+VS	[**TESTCD] [**STRESU]	X	
SDTM0631	WebSDM	IR4006 (IR5006)	Identifies Short Name of Measurement, Test, or Examination (**TESTCD) values where Standard Units (**STRESU) value is not consistent across all records.	EG+LB +QS+VS	[**TESTCD] [**STRESU]	X	X
SDTM0641	JanusFR	R4005	Identifies records where values for Unique Subject ID (USUBJID) are not unique, limited to records where USUBJID is not null.	DM		X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0641	WebSDM	R4005 (R5005)	Identifies records where values for Unique Subject ID (USUBJID) are not unique, limited to records where USUBJID is not null.	DM		X	X
SDTM0642	SAS	SAS0028	Inconsistency between Description of Arm (ARM) and Arm Code (ARMCD) values across all records.	DM	[ARM] [ARMCD]	X	X
SDTM0643	SAS	SAS0016	AGE precision inconsistent across records.	DM	AGE	X	X
SDTM0644	JanusFR	SAS0019	The current version of JANUS requires that the STUDYID column have the same value for all records within a study.	_ALL_ -DM	STUDYID	X	
SDTM0644	JanusFR	SAS0019	The current version of JANUS requires that the STUDYID column have the same value for all records within a study.	DM	STUDYID	X	
SDTM0644	SAS	SAS0019	STUDYID should have the same value for all records within a study.	DM	STUDYID		X
SDTM0645	OpenCDISC	SD1005	Study identifier (STUDYID) values must match the STUDYID in demographics (DM) domain.	[_ALL_ -DM][DM]	STUDYID		X
SDTM0661	JanusFR	IR4083	Identifies records where values for [Study Identifier (STUDYID), Unique Subject Identifier (USUBJID), Identifying Variable (IDVAR), Identifying Variable Value (IDVARVAL), and Qualifier Variable Name (QNAM)] variable or variables are not unique.	SUPP**		X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0661	WebSDM	IR4083 (IR5083)	Identifies records where values for [Study Identifier (STUDYID), Unique Subject Identifier (USUBJID), Identifying Variable (IDVAR), Identifying Variable Value (IDVARVAL), and Qualifier Variable Name (QNAM)] variable or variables are not unique.	SUPP**		X	X
SDTM0662	WebSDM	IR4161 (IR5161)	Identifies qualifier variable name (QNAM) where variable label value (Qualifier Variable Label QLABEL) is not consistent across all records.	SUPP**	[QNAM] [QLABEL]	X	X
SDTM0671	SAS	SAS0032	Inconsistency between Trial Summary Parameter (TSPARM) and Trial Summary Parameter Short Name (TSPARMCD).	TS	[TSPARM] [TSPARMCD]	X	X
SDTM0672	OpenCDISC	SD0083	The value of unique subject identifier (USUBJID) variable must be unique for each subject across all trials in the submission.	_ALL_	USUBJID		X
SDTM0673	OpenCDISC	SD1001	The value of subject identifier for the study (SUBJID) variable must be unique for each subject with the study.	DM	SUBJID		X
SDTM0801	JanusFR	IR4500	Identifies non-demographics domain subjects (USUBJID) not found in the demographics domain.	[_ALL_ -DM][DM]	STUDYID +USUBJID	X	
SDTM0801	WebSDM	IR4500 (IR5500)	Identifies non-demographics domain subjects (USUBJID) not found in the demographics domain.	[_ALL_ -DM][DM]	STUDYID +USUBJID	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0802	Janus	IR4505	Identifies demographics subjects where no record for the subject is found in the disposition domain.	[DM][DS]	STUDYID +USUBJID	X	
SDTM0802	WebSDM	IR4505 (IR5505)	Identifies demographics subjects where no record for the subject is found in the disposition domain.	[DM][DS]	STUDYID +USUBJID	X	X
SDTM0803	Janus	IR4506	Identifies demographics subjects where no record for the subject is found in the exposure domain.	[DM][EX]	STUDYID +USUBJID	X	
SDTM0803	WebSDM	IR4506 (IR5506)	Identifies demographics subjects where no record for the subject is found in the exposure domain.	[DM][EX]	STUDYID +USUBJID	X	X
SDTM0804	Janus	IR4501	Identifies Unique Subject Identifier (USUBJID) + Visit Name (VISIT) + Visit Number (VISITNUM) combinations not found in the SV domain.	[_ALL_ -SV][SV]	USUBJID +VISITNUM+VISIT	X	
SDTM0804	WebSDM	IR4501 (IR5501)	Identifies Unique Subject Identifier (USUBJID) + Visit Name (VISIT) + Visit Number (VISITNUM) combinations not found in the SV domain.	[_ALL_ -SV][SV]	USUBJID +VISITNUM+VISIT	X	X
SDTM0805	Janus	IR4502	Identifies records where the value for ARM code (ARMCD) is not found in the TA domain, excluding 'SCRNFAIL'.	[DM][TA]	ARMCD	X	
SDTM0805	WebSDM	IR4502 (IR5502)	Identifies records where the value for ARM code (ARMCD) is not found in the TA domain, excluding 'SCRNFAIL'.	[DM][TA]	ARMCD	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0806	JanusFR	IR4507	Identifies demographics treatment arms (Description of Arm (ARM) + Arm Code (ARMCD) combination) not found in the TA domain, excluding 'Screen Failure', 'SCRNFAIL'.	[DM][TA]	ARM +ARMCD	X	
SDTM0806	WebSDM	IR4507 (IR5507)	Identifies demographics treatment arms (Description of Arm (ARM) + Arm Code (ARMCD) combination) not found in the TA domain, excluding 'Screen Failure', 'SCRNFAIL'.	[DM][TA]	ARM +ARMCD	X	X
SDTM0807	JanusFR	SAS0039	TA domain is not provided and Planned Arm Code (ARMCD) is null for all rows in the demographics domain.	DM	ARMCD	X	
SDTM0807	SAS	SAS0039	TA domain is not provided and Planned Arm Code (ARMCD) is null for all rows in the demographics domain.	DM	ARMCD		X
SDTM0808	WebSDM	IR4170 (IR5170)	Identifies records that violate the condition [Visit Name (VISIT) must be the same for a given value of Visit Number (VISITNUM)].	SV	[VISIT] [VISITNUM]	X	X
SDTM0809	WebSDM	IR4171 (IR5171)	Identifies records that violate the condition [Visit Number (VISITNUM) must be the same for a given value of Visit Name (VISIT)].	SV	[VISITNUM] [VISIT]	X	X
SDTM0811	Janus	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[TA][TE]	ETCD	X	

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0811	Janus	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[SE][TE]	ETCD	X	
SDTM0811	WebSDM	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[TA][TE]	ETCD	X	X
SDTM0811	WebSDM	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[SE][TE]	ETCD	X	X
SDTM0812	WebSDM	IR5516	Identifies records in exposure that should not be present since the subject has Arm Code (ARMCD)='NOTASSGN'.	[EX][DM]	USUBJID		X
SDTM0821	JanusFR	IR4504	Identifies records where the value for Inclusion/ Exclusion Criterion Short Name (IETESTCD) is not found in the TI domain.	[IE][TI]	IETESTCD	X	
SDTM0821	WebSDM	IR4504	Identifies records where the value for Inclusion/ Exclusion Criterion Short Name (IETESTCD) is not found in the TI domain.	[IE][TI]	IETESTCD	X	X
SDTM0822	Janus	SAS0023	Identifies records where the value for Inclusion/ Exclusion Category (IECAT) in the IE domain does not exist in the TI domain if the TI domain was supplied.	[IE][TI]	IECAT	X	
SDTM0822	SAS	SAS0023	Identifies records where the value for Inclusion/ Exclusion Category (IECAT) in the IE domain does not exist in the TI domain if the TI domain was supplied.	[IE][TI]	IECAT		

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0823	OpenCDISC	SD1016	The combination of Inclusion/Exclusion Criterion Short Name (IETESTCD), Criterion (IETEST), and Category (IECAT) values must match entries in the Trial Inclusion/Exclusion Criteria (TI) data set.	[IE][TI]	IETESTCD+ IETEST+ IECAT		X
SDTM0831	JanusFR	SAS0020	The Study Identifier (STUDYID) in the TA domain does not match STUDYID in the DM domain.	[TA][DM]	STUDYID	X	
SDTM0831	SAS	SAS0020	The study identifier (STUDYID) in the TA domain does not match STUDYID in the DM domain.	[TA][DM]	STUDYID		
SDTM0836	JanusFR	SAS0021	The study identifier (STUDYID) in the TV domain does not match STUDYID in the DM domain.	[TV][DM]	STUDYID	X	
SDTM0836	SAS	SAS0021	The study identifier (STUDYID) in the TV domain does not match STUDYID in the DM domain.	[TV][DM]	STUDYID		
SDTM0841	Janus	SAS0026	Identifies records where a value for VISITNUM in the SV domain is not found in the TV domain, limited to records where both the SV and TV domains exist and the Description of Unplanned Visit (SVUPDES) is null.	[SV][TV]	VISITNUM	X	
SDTM0841	OpenCDISC	SD1017	Identifies records where a value for VISITNUM in the SV domain is not found in the TV domain, limited to records where both the SV and TV domains exist and the Description of Unplanned Visit (SVUPDES) is null.	[SV][TV]	VISITNUM		X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0842	OpenCDISC	SD1012	The combination of Element Code (ETCD) and Description of Element (ELEMENT) values must match entries in the Trial Elements (TE) data set, except for unplanned Element (ETCD = 'UNPLAN').	[SE+TA] [TE]	ETCD+ ELEMENT		X
SDTM0843	OpenCDISC	SD1013	When subjects experience for a particular period of time is represented as an unplanned element, where Element Code (ETCD) is equal to 'UNPLAN', then Planned Order of Elements within Arm (TAETORD) should be null.	SE	[ETCD] [TAETORD]		X
SDTM0844	OpenCDISC	SD1014	Order of Element within Arm (TAETORD) values must match the entries in the Trial Arms (TA) data set.	[_ALL_ -TA][TA]	TAETORD		X
SDTM0845	OpenCDISC	SD1015	Epoch (EPOCH) values must match the entries in the Trial Arms (TA) data set.	[_ALL_ -TA][TA]	EPOCH		X
SDTM0846	OpenCDISC	SD1018	For planned visits, where Description of Unplanned Visit (SVUPDES) is null, the combination of Visit Number (VISITNUM), Visit Name (VISIT), and Planned Study Day of Visit (VISITDY) values must match the entries in the Trial Visits (TV) data set.	[SV][TV]	VISITNUM + VISIT+ VISITDY		X
SDTM0851	JanusFR	IR4508	Identifies comments (CO) domain reference to an unknown related domain.	CO	RDOMAIN	X	
SDTM0851	WebSDM	IR4508 (IR5508)	Identifies comments (CO) domain reference to an unknown related domain.	CO	RDOMAIN	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0860	WebSDM	R5132	Identifies records where value for [Relationship Type (RELTYPE)] is not found in Codelist [CARDINALITY], limited to records where [RELTYPE is not null].	RELREC	RELTYPE		X
SDTM0861	Janus	IR4509	Identifies Related Records (RELREC) domain reference to an unknown related domain.	RELREC	RDOMAIN	X	
SDTM0861	WebSDM	IR4509 (IR5509)	Identifies Related Records (RELREC) domain reference to an unknown related domain.	RELREC	RDOMAIN	X	X
SDTM0862	JanusFR	IR4510	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to an unknown related domain.	SUPP**	RDOMAIN	X	
SDTM0862	WebSDM	IR4510 (IR5510)	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to an unknown related domain.	SUPP**	RDOMAIN	X	X
SDTM0863	Janus	IR4511	Identifies Related Records (RELREC) domain reference to a key variable that is not defined in the target domain.	RELREC	IDVAR	X	
SDTM0863	WebSDM	IR4511 (IR5511)	Identifies Related Records (RELREC) domain reference to a key variable that is not defined in the target domain.	RELREC	IDVAR	X	X
SDTM0864	JanusFR	IR4512	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a key variable that is not defined in the target domain.	SUPP**	IDVAR	X	
SDTM0864	WebSDM	IR4512 (IR5512)	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a key variable that is not defined in the target domain.	SUPP**	IDVAR	X	X

checkid	check source	sourceid*	source description	table scope	column scope	3.1.1	3.1.2
SDTM0865	Janus	IR4513	Identifies Related Records (RELREC) domain reference to a record that does not exist in the target domain.	RELREC	IDVAR	X	
SDTM0865	WebSDM	IR4513 (IR5513)	Identifies Related Records (RELREC) domain reference to a record that does not exist in the target domain.	RELREC	IDVAR	X	X
SDTM0866	JanusFR	IR4514	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a record that does not exist in the target domain.	SUPP**	IDVAR	X	
SDTM0866	WebSDM	IR4514 (IR5514)	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a record that does not exist in the target domain.	SUPP**	IDVAR	X	X
SDTM0871	Janus	SAS0024	Identifies comments (CO) domain reference to a record that does not exist in the target domain.	CO	IDVAR	X	
SDTM0871	OpenCDISC	SD1007	Identifies comments (CO) domain reference to a record that does not exist in the target domain.	CO	IDVAR		X
SDTM0872	OpenCDISC	SD1006	When comments are related to a specific parent record or group of parent records in a domain, then the value of Identifying Variable (IDVAR) must reference a key variable name in the parent domain.	CO	IDVAR		X

Appendix 5

CDISC CRT-DDS 1.0 Validation Checks

The following table provides a complete list of all CDISC CRT-DDS 1.0 validation checks.

Table A5.1 *Validation Checks*

checkid	checktype	tablescope	columnscope	messagetext
CRT0100	Structural	CodeListItems		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	CodeLists		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	ComputationMethods		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	DefineDocument		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	FormDefArchLayouts		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	FormDefs		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	ImputationMethods		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	ItemDefs		No two values for the source column can be equivalent within the same source data set.

checkid	checktype	tablescope	columnscope	messagetext
CRT0100	Structural	ItemGroupDefs		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	ItemGroupLeaf		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	ItemRangeChecks		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	MDVLeaf		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	MeasurementUnits		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	MetadataVersion		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	Presentation		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	Study		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	StudyEventDefs		No two values for the source column can be equivalent within the same source data set.
CRT0100	Structural	ValueLists		No two values for the source column can be equivalent within the same source data set.
CRT0101	Content	AnnotatedCRFs		Data is required for this field.
CRT0101	Content	CLItemDecodeTranslatedText		Data is required for this field.

checkid	checktype	tablescope	columnscope	messagetext
CRT0101	Content	CodeListItems		Data is required for this field.
CRT0101	Content	CodeLists		Data is required for this field.
CRT0101	Content	ComputationMethods		Data is required for this field.
CRT0101	Content	DefineDocument		Data is required for this field.
CRT0101	Content	ExternalCodeLists		Data is required for this field.
CRT0101	Content	FormDefArchLayouts		Data is required for this field.
CRT0101	Content	FormDefItemGroupRefs		Data is required for this field.
CRT0101	Content	FormDefs		Data is required for this field.
CRT0101	Content	ImputationMethods		Data is required for this field.
CRT0101	Content	ItemAliases		Data is required for this field.
CRT0101	Content	ItemDefs		Data is required for this field.
CRT0101	Content	ItemGroupAliases		Data is required for this field.
CRT0101	Content	ItemGroupDefItemRefs		Data is required for this field.
CRT0101	Content	ItemGroupDefs		Data is required for this field.
CRT0101	Content	ItemGroupLeaf		Data is required for this field.
CRT0101	Content	ItemGroupLeafTitles		Data is required for this field.
CRT0101	Content	ItemMURefs		Data is required for this field.
CRT0101	Content	ItemQuestionExternal		Data is required for this field.

checkid	checktype	tablescope	columnscope	messagetext
CRT0101	Content	ItemQuestionTranslatedText		Data is required for this field.
CRT0101	Content	ItemRangeCheckValues		Data is required for this field.
CRT0101	Content	ItemRangeChecks		Data is required for this field.
CRT0101	Content	ItemRole		Data is required for this field.
CRT0101	Content	ItemValueListRefs		Data is required for this field.
CRT0101	Content	MDVLeaf		Data is required for this field.
CRT0101	Content	MDVLeafTitles		Data is required for this field.
CRT0101	Content	MUTranslatedText		Data is required for this field.
CRT0101	Content	MeasurementUnits		Data is required for this field.
CRT0101	Content	MetaDataVersion		Data is required for this field.
CRT0101	Content	Presentation		Data is required for this field.
CRT0101	Content	ProtocolEventRefs		Data is required for this field.
CRT0101	Content	RCErrorTranslatedText		Data is required for this field.
CRT0101	Content	Study		Data is required for this field.
CRT0101	Content	StudyEventDefs		Data is required for this field.
CRT0101	Content	StudyEventFormRefs		Data is required for this field.
CRT0101	Content	SupplementalDocs		Data is required for this field.
CRT0101	Content	ValueListItemRefs		Data is required for this field.

checkid	checktype	tablescope	columnscope	messagetext
CRT0101	Content	ValueLists		Data is required for this field.
CRT0106	Content	CLItemDecodeTranslatedText	lang	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9].
CRT0106	Content	ItemQuestionTranslatedText	lang	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9].
CRT0106	Content	MUTtranslatedText	lang	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9].
CRT0106	Content	Presentation	lang	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9].
CRT0106	Content	RCErrorTranslatedText	lang	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9].
CRT0107	Content	FormDefArchLayouts	PdfFileName	The data in the &_cstparm1 field is an improperly constructed filename. Must be in the form [A-Za-z0-9_.].
CRT0108	Content	ItemDefs	SASFieldName SDSVarName	The data in the &_cstparm1 field is an improperly constructed SAS name. Must be in the form [A-Za-z0-9_].
CRT0108	Content	ItemGroupDefs	SASDatasetName	The data in the &_cstparm1 field is an improperly constructed SAS name. Must be in the form [A-Za-z0-9_].

checkid	checktype	tablescope	columnscope	messagetext
CRT0109	Content	CodeLists	SASFormatName	The data in the &_cstparm1 field is an improperly constructed SAS format name. Must be in the form [($)A-Za-z0-9_].
CRT0110	Content	[AnnotatedCRFs] [MDVLeaf]	[AnnotatedCRFs.leafID] [MDVLeaf.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[AnnotatedCRFs] [MetaDataVersion]	[AnnotatedCRFs.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[CLItemDecodeTranslatedText] [CodeListItems]	[CLItemDecode TranslatedText.FK_ CodeListItems] [CodeListItems.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[CodeListItems] [CodeLists]	[CodeListItems.FK_ CodeLists] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[CodeLists] [MetaDataVersion]	[CodeLists.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ComputationMethods] [MetaDataVersion]	[ComputationMethods. FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ExternalCodeLists] [CodeLists]	[ExternalCodeLists.FK_ CodeLists] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[FormDefArchLayouts] [FormDefs]	[FormDefArchLayouts. FK_FormDefs] [FormDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[FormDefArchLayouts] [Presentation]	[FormDefArchLayouts. PresentationOID] [Presentation.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[FormDefItemGroupRefs] [FormDefs]	[FormDefItemGroup Refs.FK_FormDefs] [FormDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[FormDefItemGroupRefs] [ItemGroupDefs]	[FormDefItemGroupRef s.ItemGroupOID] [ItemGroupDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[FormDefs] [MetaDataVersion]	[FormDefs.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1; does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ImputationMethods] [MetaDataVersion]	[ImputationMethods.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemAliases] [ItemDefs]	[ItemAliases.FK_ ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemDefs] [CodeLists]	[ItemDefs.CodeListRef] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemDefs] [ComputationMethods]	[ItemDefs.Computation MethodOID] [ComputationMethods. OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemDefs] [MetaDataVersion]	[ItemDefs.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[ItemGroupAliases] [ItemGroupDefs]	[ItemGroupAliases.FK_ItemGroupDefs] [ItemGroupDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemGroupDefItemRefs] [CodeLists]	[ItemGroupDefItemRefs.RoleCodeListOID] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemGroupDefItemRefs] [ImputationMethods]	[ItemGroupDefItemRefs.ImputationMethodOID] [ImputationMethods.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemGroupDefItemRefs] [ItemDefs]	[ItemGroupDefItemRefs.ItemOID] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemGroupDefItemRefs] [ItemGroupDefs]	[ItemGroupDefItemRefs.FK_ItemGroupDefs] [ItemGroupDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemGroupDefs] [MetaDataVersion]	[ItemGroupDefs.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemGroupLeafTitles] [ItemGroupLeaf]	[ItemGroupLeafTitles.FK_ItemGroupLeaf] [ItemGroupLeaf.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemGroupLeaf] [ItemGroupDefs]	[ItemGroupLeaf.FK_ItemGroupDefs] [ItemGroupDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemMURefs] [ItemDefs]	[ItemMURefs.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[ItemMURefs] [MeasurementUnits]	[ItemMURefs.MeasurementUnitOID] [MeasurementUnits.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemQuestionExternal] [ItemDefs]	[ItemQuestionExternal.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemQuestionTranslatedText] [ItemDefs]	[ItemQuestionTranslatedText.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemRangeCheckValues] [ItemRangeChecks]	[ItemRangeCheckValues.FK_ItemRangeChecks] [ItemRangeChecks.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemRangeChecks] [ItemDefs]	[ItemRangeChecks.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemRangeChecks] [MeasurementUnits]	[ItemRangeChecks.MURefOID] [MeasurementUnits.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemRole] [ItemDefs]	[ItemRole.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemValueListRefs] [ItemDefs]	[ItemValueListRefs.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemValueListRefs] [ValueLists]	[ItemValueListRefs.ValueListOID] [ValueLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[MDVLeafTitles] [MDVLeaf]	[MDVLeafTitles.FK_ MDVLeaf] [MDVLeaf.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[MDVLeaf] [MetaDataVersion]	[MDVLeaf.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[MUTranslatedText] [MeasurementUnits]	[MUTranslatedText.FK_ MeasurementUnits] [MeasurementUnits.OI D]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[MeasurementUnits] [Study]	[MeasurementUnits.FK_ Study] [Study.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[MetaDataVersion] [Study]	[MetaDataVersion.FK_ Study] [Study.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[Presentation] [MetaDataVersion]	[Presentation.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ProtocolEventRefs] [MetaDataVersion]	[ProtocolEventRefs.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ProtocolEventRefs] [StudyEventDefs]	[ProtocolEventRefs.Stu dyEventOID] [StudyEventDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[RCErrorTranslatedText] [ItemRangeChecks]	[RCErrorTranslatedText .FK_ItemRangeChecks] [ItemRangeChecks.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[StudyEventDefs] [MetaDataVersion]	[StudyEventDefs.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[StudyEventFormRefs] [FormDefs]	[StudyEventFormRefs.FormOID] [FormDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[StudyEventFormRefs] [StudyEventDefs]	[StudyEventFormRefs. FK_StudyEventDefs] [StudyEventDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[Study] [DefineDocument]	[Study.FK_ DefineDocument] [DefineDocument.FileOID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[SupplementalDocs] [MDVLeaf]	[SupplementalDocs.leafID] [MDVLeaf.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[SupplementalDocs] [MetaDataVersion]	[SupplementalDocs.FK_ MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ValueListItemRefs] [CodeLists]	[ValueListItemRefs.RoleCodeListOID] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ValueListItemRefs] [ImputationMethods]	[ValueListItemRefs.ImputationMethodOID] [ImputationMethods.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ValueListItemRefs] [ItemDefs]	[ValueListItemRefs.ItemOID] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[ValueListItemRefs] [ValueLists]	[ValueListItemRefs.FK_ ValueLists] [ValueLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ValueLists] [ValueLists]	[ValueLists.FK_ MetaDataVersion] [ValueLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0111	Content	[ItemGroupDefs] [ItemGroupDefItemRefs]	[ItemGroupDefs.OID] [ItemGroupDefItemRefs.FK_ItemGroupDefs]	Each distinct value of &_cstparm1 must have a corresponding value in the target data set &_cstparm2.
CRT0111	Content	[ItemRangeChecks] [ItemRangeCheckValues]	[ItemRangeChecks.OID] [ItemRangeCheckValues.FK_ ItemRangeChecks]	Each distinct value of &_cstparm1 must have a corresponding value in the target data set &_cstparm2.
CRT0111	Content	[MDVLeaf] [MDVLeafTitles]	[MDVLeaf.ID] [MDVLeafTitles.FK_ MDVLeaf]	Each distinct value of &_cstparm1 must have a corresponding value in the target data set &_cstparm2.
CRT0112	Content	[CodeListItems] [ExternalCodeLists]	[CodeListItems.FK_ CodeLists] [ExternalCodeLists.FK_ CodeLists]	No value in &_cstparm1 can be equal to any value in &_cstparm2.
CRT0112	Content	[DefineDocument] [ItemGroupLeaf]	[DefineDocument.ID] [ItemGroupLeaf.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2.
CRT0112	Content	[DefineDocument] [MDVLeaf]	[DefineDocument.ID] [MDVLeaf.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2.
CRT0112	Content	[ExternalCodeLists] [CodeListItems]	[ExternalCodeLists.FK_ CodeLists] [CodeListItems.FK_ CodeLists]	No value in &_cstparm1 can be equal to any value in &_cstparm2.
CRT0112	Content	[ItemGroupLeaf] [DefineDocument]	[ItemGroupLeaf.ID] [DefineDocument.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2.
CRT0112	Content	[ItemGroupLeaf] [MDVLeaf]	[ItemGroupLeaf.ID] [MDVLeaf.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2.

checkid	checktype	tablescope	columnscope	messagetext
CRT0112	Content	[MDVLeaf] [DefineDocument]	[MDVLeaf.ID] [DefineDocument.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2.
CRT0112	Content	[MDVLeaf] [ItemGroupLeaf]	[MDVLeaf.ID] [ItemGroupLeaf.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2.
CRT0113	Content	CodeListItems	[CodedValue] [FK_CodeLists]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0113	Content	FormDefItemGroupRefs	[ItemGroupOID] [FK_FormDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0113	Content	FormDefItemGroupRefs	[OrderNumber] [FK_FormDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0113	Content	ItemGroupDefItemRefs	[ItemOID] [FK_ItemGroupDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0113	Content	ItemGroupDefItemRefs	[OrderNumber] [FK_ItemGroupDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0113	Content	ProtocolEventRefs	[OrderNumber] [FK_MetaDataVersion]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0113	Content	ProtocolEventRefs	[StudyEventOID] [FK_MetaDataVersion]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0113	Content	StudyEventFormRefs	[FormOID] [FK_StudyEventDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0113	Content	StudyEventFormRefs	[OrderNumber] [FK_StudyEventDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.

checkid	checktype	tablescope	columnscope	messagetext
CRT0114	Content	_ALL_		Coded value is either incorrect, missing, or in the wrong case.

Index

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.