

SAS[®] Clinical Standards Toolkit 1.4 User's Guide



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., 2011. *SAS® Clinical Standards Toolkit 1.4: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Clinical Standards Toolkit 1.4: User's Guide

Copyright © 2011, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

For a hardcopy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government Restricted Rights Notice: Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st electronic book, November 2011

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at

support.sas.com/publishing or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Contents

<i>What's New in the SAS Clinical Standards Toolkit</i>	<i>ix</i>
Chapter 1 • Introduction to the SAS Clinical Standards Toolkit	1
What Is the SAS Clinical Standards Toolkit?	1
References	2
Chapter 2 • Framework	5
Overview	6
Global Standards Library	6
What Is a Standard?	8
Common Framework Metadata	9
Common Usage Scenarios for the Framework	10
Maintenance Usage Scenarios	17
Chapter 3 • Metadata File Descriptions	23
Overview	23
Standards	23
StandardSASReferences	25
Standardlookup	26
SASReferences	27
Properties	29
Messages	31
Results	33
Additional Metadata Files	35
Chapter 4 • Supported Standards	37
SAS Representation of Standards	37
CDISC SDTM	40
CDISC ADaM 2.1	46
CDISC CRT-DDS 1.0	51
CDISC ODM 1.3.0	56
CDISC Terminology	62
Chapter 5 • SASReferences File	67
Overview	67
Building a SASReferences File	67
How Is a SASReferences File Used?	75
Chapter 6 • Validation	81
Validation Framework Overview	82
Metadata Requirements	84
Cross-Standard Validation	102
Building a Validation Process	104
Running a Validation Process	111
Validation Checks by Standard	120
Special Topic: Validation Check Macros	144
Special Topic: How the SAS Clinical Standards Toolkit Interprets Validation Check Metadata	150
Special Topic: SAS Implementation of ISO 8601	154

Special Topic: Debugging a Validation Process	159
Special Topic: Validation Customization	164
Special Topic: Using Alternative Controlled Terminologies	170
Special Topic: Performance Considerations	174
Chapter 7 • XML-Based Standards	175
SAS Support of XML-Based Standards	176
Reading XML Files	176
Writing XML Files	190
Validation of XML-Based Standards	202
Special Topic: A Round Trip Exercise Involving the CDISC SDTM and CDISC CRT-DDS Standards	210
Special Topic: A Round Trip Exercise Involving the CDISC CRT-DDS Standard: Importing and Exporting the define.xml File	214
Special Topic: Identifying Unsupported Elements and Attributes in a CDISC ODM File	218
Chapter 8 • Working with CDISC ADaM Data	223
Overview	223
SAS Representation of CDISC ADaM Metadata	224
ADaM Data Set Templates	230
Validation of ADaM Data Sets	231
Sample Reporting Methodology	236
Chapter 9 • Reporting	245
Sample Reports	245
Process Results Reporting	245
Validation Check Metadata Reporting	255
Appendix 1 • Global Macro Variables	261
Overview	261
CST-Framework initialize.properties	262
CDISC ADaM 2.1 initialize.properties	264
CDISC ADaM 2.1 validation.properties	265
CDISC ADaM 2.1 report.properties	267
CDISC CRTDDS 1.0 initialize.properties	267
CDISC CRTDDS 1.0 validation.properties	268
CDISC ODM 1.3.0 initialize.properties	269
CDISC ODM 1.3.0 validation.properties	270
CDISC SDTM (3.1.1 and 3.1.2) initialize.properties	272
CDISC SDTM (3.1.1 and 3.1.2) validation.properties	272
General Purpose (not set in properties files)	274
Appendix 2 • Framework Messages	275
Appendix 3 • Macro Application Programming Interface	279
Module ADaM V2.1 (Run Time)	279
Module CRT-DDS V1.0 (Run Time)	283
Module Framework	292
Module SDTM V3.1.1 (Run Time)	336
Module SDTM V3.1.2 (Run Time)	342
Module ODM V1.3.0 (Run Time)	347
Appendix 4 • CDISC SDTM Validation Checks	351
Validation Checks (with SDTM V3.1.1)	351
Validation Checks (with SDTM V3.1.2)	379

Appendix 5 • CDISC CRT-DDS 1.0 Validation Checks	405
Appendix 6 • CDISC ODM 1.3.0 Validation Checks	415
Appendix 7 • CDISC ADaM 2.1 Validation Checks	433
Appendix 8 • ODM 1.3.0 SAS Data Sets	447
Conventions Used in the Tables	449
ODM SAS Data Set	449
Study SAS Data Set	450
MeasurementUnits SAS Data Set	450
MUTranslatedText SAS Data Set	450
MetaDataVersion SAS Data Set	451
ProtocolEventRefs SAS Data Set	451
ProtocolTranslatedText SAS Data Set	452
StudyEventDefs SAS Data Set	452
StudyEventDefTranslatedText SAS Data Set	452
StudyEventFormRefs SAS Data Set	453
FormDefs SAS Data Set	453
FormDefTranslatedText SAS Data Set	454
FormDefItemGroupRefs SAS Data Set	454
FormDefArchLayouts SAS Data Set	454
ItemGroupDefs SAS Data Set	455
ItemGroupDefTranslatedText SAS Data Set	455
ItemGroupDefItemRefs SAS Data Set	456
ItemGroupAliases SAS Data Set	456
ItemDefs SAS Data Set	457
ItemDefTranslatedText SAS Data Set	457
ItemQuestionTranslatedText SAS Data Set	458
ItemQuestionExternal SAS Data Set	458
ItemMURefs SAS Data Set	459
ItemRangeChecks SAS Data Set	459
ItemRangeCheckValues SAS Data Set	459
RCErrorTranslatedText SAS Data Set	460
ItemRCFormalExpression SAS Data Set	460
ItemRole SAS Data Set	460
ItemAliasesSAS Data Set	461
CodeLists SAS Data Set	461
ExternalCodeLists SAS Data Set	461
EnumeratedItems SAS Data Set	462
CodeListItems SAS Data Set	462
CLItemDecodeTranslatedText SAS Data Set	463
ImputationMethods SAS Data Set	463
Presentation SAS Data Set	463
MethodDefs SAS Data Set	464
MethodDefTranslatedText SAS Data Set	464
MethodDefFormalExpression SAS Data Set	464
ConditionDefs SAS Data Set	465
ConditionDefTranslatedText SAS Data Set	465
ConditionDefFormalExpression SAS Data Set	465
ClinicalData SAS Data Set	466
SubjectDataSAS Data Set	466
StudyEventData SAS Data Set	466
FormData SAS Data Set	467
ItemGroupData SAS Data Set	467
ItemData SAS Data Set	468

ReferenceData SAS Data Set	469
AdminData SAS Data Set	469
User SAS Data Set	469
UserAddress SAS Data Set	470
UserAddressStreetName SAS Data Set	470
UserEmail SAS Data Set	471
UserFax SAS Data Set	471
UserPhone SAS Data Set	471
UserLocationRef SAS Data Set	472
Location SAS Data Set	472
LocationVersion SAS Data Set	472
SignatureDef SAS Data Set	473
AuditRecord SAS Data Set	473
Signature SAS Data Set	474
Annotation SAS Data Set	474
AnnotationFlag SAS Data Set	475
Association SAS Data Set	475
KeySet SAS Data Set	475
Appendix 9 • CRT-DDS 1.0 SAS Data Sets	477
Conventions Used in the Tables	478
DefineDocument SAS Data Set	478
Study SAS Data Set	479
MeasurementUnits SAS Data Set	480
MUTranslatedText SAS Data Set	480
MetaDataVersion SAS Data Set	480
AnnotatedCRFs SAS Data Set	481
SupplementalDocs SAS Data Set	481
MDVLeaf SAS Data Set	482
MDVLeafTitles SAS Data Set	482
ComputationMethods SAS Data Set	482
ValueLists SAS Data Set	483
ValueListItemRefs SAS Data Set	483
ProtocolEventRefs SAS Data Set	483
StudyEventDefs SAS Data Set	484
StudyEventFormRefs SAS Data Set	484
FormDefs SAS Data Set	485
FormDefItemGroupRefs SAS Data Set	485
FormDefArchLayouts SAS Data Set	485
ItemGroupDefs SAS Data Set	486
ItemGroupDefItemRefs SAS Data Set	487
ItemGroupAliases SAS Data Set	487
ItemGroupLeaf SAS Data Set	488
ItemGroupLeafTitles SAS Data Set	488
ItemDefs SAS Data Set	488
ItemQuestionTranslatedText SAS Data Set	489
ItemQuestionExternal SAS Data Set	489
ItemMURefs SAS Data Set	490
ItemRangeChecks SAS Data Set	490
ItemRangeCheckValues SAS Data Set	490
RCErrorsTranslatedText SAS Data Set	491
ItemRole SAS Data Set	491
ItemAliases SAS Data Set	491
ItemValueListRefs SAS Data Set	492
CodeLists SAS Data Set	492
ExternalCodeLists SAS Data Set	492

CodeListItems SAS Data Set	493
CLItemDecodeTranslatedText SAS Data Set	493
ImputationMethods SAS Data Set	493
Presentation SAS Data Set	494
Index	495

What's New in the SAS Clinical Standards Toolkit

Overview

Here are some of the new capabilities in the SAS Clinical Standards Toolkit 1.4:

- The CDISC ADaM 2.1 standard has been registered and contains all of the metadata for the ADSL and BDS data structures, based on the *ADaM Implementation Guide* and *CDISC Controlled Terminology ADaM* from the National Cancer Institute (NCI). Validation checks were created from information from *CDISC ADaM Validation Checks Version 1.1 Maintenance Release*.
- The CDISC ODM 1.3.0 standard, including all metadata and validation checks, has been fully implemented. In addition, ODM Read and Write functionality has been added to the SAS Clinical Standards Toolkit. For a description and limitations of the implementation, see [Chapter 4, “Supported Standards,” on page 37](#) and [Chapter 7, “XML-Based Standards,” on page 175](#).
- Sample reports can output comma-separated values (*.csv) files, enabling you to view the output using Microsoft Excel.
- The SAS Clinical Standards Toolkit global standards library folder structure for controlled terminology has been reconfigured to facilitate the SAS Clinical Standards Toolkit implementation. In previous versions of the SAS Clinical Standards Toolkit, whenever there was a new release of the controlled terminology dictionaries from NCI, an additional folder was required in the Standards directory. In the SAS Clinical Standards Toolkit 1.4, any additional folders needed have been replaced with a single folder named `cdisc-terminology-1.4`. In this folder, there are additional folders for each standard and corresponding subfolders for each NCI release of the controlled terminology dictionaries. In addition, a current folder has been added so that you can save the latest version of the NCI controlled terminology dictionary without changing any references in your SAS Clinical Standards Toolkit code. The SAS Clinical Standards Toolkit provides the current folder populated with the latest version of the controlled terminology dictionary.
- The SAS Clinical Standards Toolkit metadata and code base are updated.
- Two new validation check macros are available. These macros are specifically designed to handle cross-standard validation checks that occur in ADaM 2.1 validation.

Changes to Metadata and Code Base

Framework Changes

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

- `cst_getstandardsubtypes` generates a data set containing the installed Clinical Terminology subtypes (for example, SDTM, CDASH, ADaM, or any user customizations).
- `cstcheck_columnexists` is used in validation processes to determine whether one or more of the columns defined in `columnScope` exist in each of the tables defined in `tableScope`.
- `cstcheck_columnvarlist` is used in validation processes to support comparison of multiple columns within the same data set or across multiple data sets.

Note: As a general rule, this macro expects a check metadata `columnScope` syntax of `{_cstList:var1+var2+var3...varn}` for within-data-set assessments and `{_cstList:var1...varn}{_cstList:var1...varn}` for multi-data set assessments.

Note: This macro requires use of `_cstCodeLogic` at a DATA step level (for example, a full DATA step or PROC SQL invocation). `_cstCodeLogic` creates a work file (`_cstproblems`) containing records in error. `_cstCodeLogic` must handle any data set joins when multiple data sets are involved in the column comparisons.

- `cstcheck_crossstdcomparedomains` is used in validation processes to compare values for one or more columns in one table with those same columns in another domain in another standard, or it compares values against metadata from the comparison standard. This macro allows validation across standards.

Note: This macro requires the use of `_cstCodeLogic` as a full DATA step or PROC SQL invocation. This DATA step or PROC SQL invocation 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.

- `cstcheck_crossstdmetamismatch` is used in validation processes to identify inconsistencies among metadata across registered standards. This macro allows validation across standards.

Note: `cstcheck_crossstdmetamismatch` requires the use of `_cstCodeLogic` as a full DATA step or PROC SQL invocation. This DATA step or PROC SQL invocation 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.

- `cstcheck_java` determines whether any Java issues or related to Java issues exist in the previous DATA step. This macro must be called immediately after the DATA step that declares the Java object. It is called by `crtds_read.sas`, `crtds_write.sas`, `crtds_xmlvalidate.sas`, `odm_read.sas`, `odm_write.sas`, and `odm_xmlvalidate.sas` programs.

- `cstutil_buildformatsfromxml` is designed for use with CDISC XML-based standards such as CRT-DDS and ODM. Those standards capture acceptable values in codelists. This module reads the codelist information to create one or more SAS format catalogs, based on the `xml:lang` language tags. This macro is called by the `odm_read` and `crtds_read` macros.
- `cstutil_createsubdir` creates subdirectories on a computer that is not Microsoft Windows. The SAS Clinical Standards Toolkit sample drivers create output files that need to have Read and Write access to the subdirectories. This macro creates the subdirectories in the specified workspace. If a value is missing, the `StudyOutputPath` points to the Work directory, and any subdirectories are created under it. `StudyOutputPath` is referenced in `SASReferences`.
- `cstutil_createsublists` creates the `work._cstsublists` data set that has interpreted validation check metadata as specified in the `columnScope` column in the expected form of `[var1][var2]`. This macro is used in the Validation Master data set in the check metadata code logic field. This macro is not always called for the derivation of `work._cstsublists`.
- `cstutil_readxmltags` provides a proof-of-concept implementation of a tool to read the element tags and attributes of an XML file to identify those element tags and attributes that the SAS Clinical Standards Toolkit does not currently handle using the CDISC ODM `odm_read` macro. This macro relies on a defined set of XSLT modules, metadata that specifies a SAS representation of ODM, and a SAS XML map file that reads a derived cubxml file. Each of these makes assumptions about the XML content to be read.

Assumptions:

- The XML file has previously been defined with a SAS fileref.
- ODM reference metadata is available as defined in `SASReferences`.

Limitations:

- The code does not work on a continuous-stream (no line returns) XML file.
- The code might not work well on multi-element rows like
`<Study><MetaDataVersionOID=<.><...>`.
- The code might not handle PCDDATA.

`cstutil_readxmltags` is used by `find_unsupported_tags.sas`.

- `cstutil_writeodmcubxml` is used to create an XML file to be used by the `define.xml` process. There is one input to this macro: the MDP SAS data set that contains the member names and library references needed for the define process.

In addition to the new macros, all except one of the preexisting check macros (`cstcheck*.sas`) have been modified. Most of the modifications were minor. However, a few were more significant. You should compare the files, and determine whether the changes affect any of your current processes. The `cstcheck_notimplemented` macro was not modified. Macros are located in the `!sasroot/cstframework/sasmacro` directory.

CDISC SDTM Changes

For SDTM 3.1.1 and 3.1.2, validation check SDTM0452 was modified to check only those variables existing in your AE domain. This change is in the code logic field of the Validation Master data set. In addition, a change was made to the codesource field from `cstcheck_column` to `cstcheck_columnvarlist` with the corresponding values

needed in columnScope. codetype was set to 2 to denote a DATA step or PROC SQL invocation. reportingcolumns was set to blank.

CDISC CRT-DDS Changes

The validation.properties file was added to contain validation-specific settings for the global macro variables. These settings used to be in initialize.properties, which was modified by removing the validation-specific settings, and changing _cstSubjectColumns from **none** to **_none_**. These changes lessen the risk of using a customer variable. These two files are located in the **/standards/cdisc-crtdds-1.0-1.4/programs** folder.

One new CDISC CRT-DDS macro was created, and nine macros were modified. These macros are located in the **/standards/cdisc-crtdds-1.0-1.4/macros** folder.

The crtdds_sdtmtodefine macro replaces the crtdds_sdtm311todefine10 macro. This macro is called by the create_crtdds_from_sdtm.sas driver program. This macro creates the 39 tables for the SAS representation of the CRT-DDS files from the SDTM metadata. This macro, using SDTM table and column metadata as its source, populates a subset of 12 CRT-DDS data sets. The metadata source is specified via the sasreferences data set.

The crtdds_read, crtdds_write, and crtdds_validate macros now use the xmlv2 engine in SAS 9.3, and the xml92 engine in SAS 9.2.

The crtdds_codelistitems macro no longer tries to derive the Rank attribute, which is the numeric order of a CodedValue relative to a CodedValue for other CodeListItems in a CodeList. The reason for this change is that the Rank attribute should be used only where the relative value corresponding to an enumeration cannot or should not be determined by its lexical order. For example, assume that you have a list of enumerated text values including "Low", "Medium", and "High". You want to assign relative numeric values 1, 2, and 3, respectively, to these text values. You should include a Rank attribute for each EnumeratedItem defined. Without the applied Rank attribute, the normal lexical ordering would be "High", "Low", and "Medium".

The following macros were modified. Most of the modifications were minor. However, a few were more significant. You should compare the files, and determine whether the changes affect any of your current processes.

- crtdds_codelistitems
- crtdds_codelists
- crtdds_computationmethods
- crtdds_definedocument
- crtdds_itemgroupdefitemrefs
- crtdds_read
- crtdds_sdtm311todefine10 (this macro is deprecated in the SAS Clinical Standards Toolkit 1.4)
- crtdds_write
- crtddsutil_buildchecktablelist

CDISC-Terminology Changes

The folder structure for CDISC controlled terminology has undergone significant changes. These changes were made to streamline the implementation of the SAS Clinical Standards Toolkit. In previous versions of the SAS Clinical Standards Toolkit, each new release of the NCI EVS controlled terminology dictionary required its own root-level folder in the **cstGlobalLibrary/standards** folder. In the SAS Clinical Standards Toolkit 1.4, there is now one folder in **cstGlobalLibrary/standards** named **cdisc-terminology-1.4**. This folder contains separate folders for each standard, and subfolders within each standard for each new release of the controlled terminology dictionary.

201104 is the default CDISC-Terminology standard version for SDTM 3.1.1 and SDTM 3.1.2 SAS Clinical Standards Toolkit 1.4 standards. For ADaM 2.1 the default CDISC-Terminology is 201101. In addition, for the convenience of our customers, the 201104 release of the CDISC Controlled Terminology for CDASH dictionary is shipped with this release of the SAS Clinical Standards Toolkit.

A new column, **codelist_extensible**, has been added to all of the **cterms.sas7bdat** data sets. This column is provided by NCI and denotes whether the format can have new values added to it by the customer. The value for this new column is Yes or No.

An existing column, **CDISC_PREFERRED_TERM**, has been removed from all **cterms** data sets. This column was removed from the April 2011 (201104) NCI Controlled Terminology spreadsheet. **CDISC_PREFERRED_TERM** was used by the SAS Clinical Standards Toolkit to determine whether duplicate values existed in the spreadsheet before generating the format catalog (**cterms.sas7bcat**). The **NCI_PREFERRED_TERM** column is now used instead.

Chapter 1

Introduction to the SAS Clinical Standards Toolkit

What Is the SAS Clinical Standards Toolkit?	1
References	2

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

The 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

The 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 SAS Clinical Standards Toolkit is not limited to supporting CDISC standards. In time, the SAS Clinical Standards Toolkit will support other evolving industry-standard data models. The SAS Clinical Standards 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. The 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 the SAS Clinical Standards 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/sdtm	Provides access to <i>CDISC SDTM Implementation Guide V3.1.1 Final</i> and <i>CDISC Study Data Tabulation Model Version 1.1 Final</i> .
CDISC SDTM 3.1.2	http://www.cdisc.org/extranet/index.php?a=1209	Provides access to the <i>Study Data Tabulation Model, Version 1.2 (SDTM v1.2)</i> and the <i>SDTM Implementation Guide for Human Clinical Trials (SDTMIG v.3.1.2)</i> .
CDISC CRT-DDS 1.0	http://www.cdisc.org/define-xml	Provides access to <i>Case Report Tabulation Data Definition Specification (CRT-DDS, also called define.xml) Final Version 1.0</i> .
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. <i>Note:</i> 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 <i>Analysis Data Model Version 2.1</i> and the <i>Analysis Data Model Implementation Guide Version 1.0</i> . <i>Note:</i> Registration might be required.
CDISC ADaM 2.1 Validation Checks	http://www.cdisc.org/adam	CDISC ADaM Validation Checks (Version 1.1, released January 21, 2011) prepared by the CDISC Analysis Data Model (ADaM) team. <i>Note:</i> Access to the CDISC members-only site might be required.
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.

Reference	Web Address **	Description
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 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.
SAS Technical Support	Online form: http:// support.sas.com/ctx/ supportform/createForm	Any problems experienced with the product and technical questions should be addressed to SAS Technical Support using the online form or: Phone (North America): 919-677-8008. Otherwise, contact your local SAS office.
SAS Knowledge Base for the SAS Clinical Standards Toolkit	http://support.sas.com/rnd/ base/cdisc/cst/index.html	Find current information, documentation, technical papers, and presentations about the SAS Clinical Standards Toolkit.
<i>SAS Clinical Standards Toolkit 1.4: User's Guide</i>	http://support.sas.com/ documentation/onlinedoc/ clinical/index.html	Link to this document and other documents.
SAS Knowledge Base for the 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 the SAS Clinical Standards Toolkit. (Type <i>Clinical Standards Toolkit</i> in the search field.)
SAS and Clinical Trials Forum	http://communities.sas.com/ community/ sas_and_clinical_trials	Primary public discussion forum for the SAS Clinical Standards Toolkit.

Reference	Web Address **	Description
SAS Training	http://support.sas.com/training/	SAS is pursuing the development of one or more SAS Clinical Standards Toolkit training classes. Some information about the SAS Clinical Standards Toolkit is also provided in the SAS Clinical Data Integration: Essentials training course.
External Vendor Tutorials		Various vendors offer product tutorials, often as a part of an industry-related user conference.

** Accessed on October 5, 2011.

Chapter 2

Framework

Overview	6
Global Standards Library	6
What Is a Standard?	8
Common Framework Metadata	9
Overview	9
Standards Data Set	9
StandardSASReferences	9
Standardlookup	9
SASReferences Data Set	9
Properties Files	10
Messages Data Set	10
Results Data Set	10
Common Usage Scenarios for the Framework	10
Overview	10
Initializing the Framework's Global Macro Variables	10
Referencing the Default Version of a Standard	11
Getting a List of the Standards That Are Installed	11
Determining Which Revision (Release) of a Standard Version Is Installed	12
Getting a List of the Files and Data Sets That Are Associated with a Registered Standard	12
Creating Data Sets Used by the Framework	13
Creating Table Shells Based on a Data Standard	13
Getting a Copy of the Reference Metadata for a Data Standard	14
Inserting Information from Registered Standards into a SASReferences File	15
Maintenance Usage Scenarios	17
Overview	17
Registering a New Version of a Standard	17
Setting the Default Version for a Standard	18
Unregistering a Standard Version	18
Unregistering an Old Version of a Standard, and Then Registering a New Version of a Standard	19

Overview

The Framework module of the 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 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, you are 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 9](#).
- **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.

This display provides the full content of the global standards library Standards data set included with the SAS Clinical Standards Toolkit after a new installation of the application. (The columns are continued in the second image.)

Display 2.1 Global Standards Library: Metadata Standards Data Set

	standard	mnemonic	standardversion	comment	rootpath	isstandarddefault	iscstframework	isdatastandard
1	CDISC-ADAM	ADAM	2.1	CDISC ADAM V2.1	&_cstGRoot./standards/cdisc-adam-2.1-1.4	Y	N	Y
2	CDISC-CRTDDS	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4	Y	N	Y
3	CDISC-ODM	ODM	1.3.0	CDISC ODM V1.3.0	&_cstGRoot./standards/cdisc-odm-1.3.0-1.4	Y	N	Y
4	CDISC-SDTM	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.4	N	N	Y
5	CDISC-SDTM	SDTM	3.1.2	CDISC SDTM V3.1.2	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4	Y	N	Y
6	CDISC-TERMINOLOGY	CT	NCL_THESAURUS	CDISC Terminology 2011-01-07	&_cstGRoot./standards/cdisc-terminology-1.4	Y	N	N
7	CST-FRAMEWORK	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework-1.4	Y	Y	N

	standard	mnemonic	standardversion	supportvalidation	isxmlstandard	importxsl	exportxsl	schema	productrevision
1	CDISC-ADAM	ADAM	2.1	Y	N				1.4
2	CDISC-CRTDDS	CRT	1.0	Y	Y	CRT-DDS/1.0/import/Root.xsl	CRT-DDS/1.0/export/Root.xsl	cdisc-crtdds-1.0.0/define1-0-0.xsd	1.4
3	CDISC-ODM	ODM	1.3.0	Y	Y	ODM/1.3.0/import/Root.xsl	ODM/1.3.0/export/Root.xsl	cdisc-odm-1.3.0/ODM1-3-0.xsd	1.4
4	CDISC-SDTM	SDTM	3.1.1	Y	N				1.4
5	CDISC-SDTM	SDTM	3.1.2	Y	N				1.4
6	CDISC-TERMINOLOGY	CT	NCL_THESAURUS	N	N				1.4
7	CST-FRAMEWORK	CST	1.2	N	N				1.4

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.

This display shows some rows and columns.

Display 2.2 Global Standards Library: Metadata StandardSASReferences Data Set

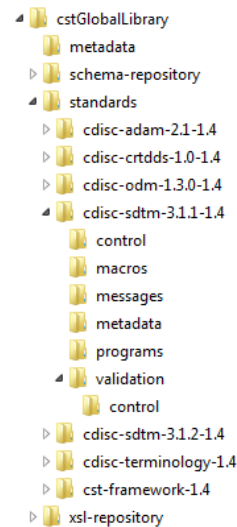
standard	standardversion	type	subtype	SASref	reftype	path	memname
CDISC-ADAM	2.1	autocall		adamauto	fileref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/macros	
CDISC-ADAM	2.1	classmetadata	column	clasmeta	libref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/metadata	class_columns.sas7bdat
CDISC-ADAM	2.1	classmetadata	table	clasmeta	libref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/metadata	class_tables.sas7bdat
CDISC-ADAM	2.1	lookup		lookup	libref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/control	standardlookup.sas7bdat
CDISC-ADAM	2.1	messages		adammsg	libref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/messages	messages.sas7bdat
CDISC-ADAM	2.1	properties	initialize	adamprop	libref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/programs	initialize.properties
CDISC-ADAM	2.1	properties	report	rptprop	fileref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/programs	report.properties
CDISC-ADAM	2.1	properties	validation	adamprop2	fileref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/programs	validation.properties
CDISC-ADAM	2.1	referencecontrol	validation	refcntrl	libref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/validation/contr	validation_master.sas7bdat
CDISC-ADAM	2.1	referencemetadat	column	refmeta	libref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/metadata	reference_columns.sas7bdat
CDISC-ADAM	2.1	referencemetadat	table	refmeta	libref	&_cstGRoot./standards/cdisc-adam-2.1-1.4/metadata	reference_tables.sas7bdat
CDISC-CRTDDS	1.0	autocall		crtauto	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/macros	
CDISC-CRTDDS	1.0	fmtsearch		ctfmt	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/formats	crtddscrt.sas7bcat
CDISC-CRTDDS	1.0	lookup		crtctrl	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/control	standardlookup.sas7bdat
CDISC-CRTDDS	1.0	messages		crtmsg	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/messages	messages.sas7bdat
CDISC-CRTDDS	1.0	properties	initialize	crtprop	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/programs	initialize.properties
CDISC-CRTDDS	1.0	properties	validation	crtprop2	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/programs	validation.properties
CDISC-CRTDDS	1.0	referencemetadat	column	crtref	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/metadata	reference_columns.sas7bdat
CDISC-CRTDDS	1.0	referencemetadat	table	crtref	libref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/metadata	reference_tables.sas7bdat
CDISC-CRTDDS	1.0	referencexml	stylesheet	xslt01	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4/stylesheet	define1-0-0.xsl
CDISC-ODM	1.3.0	autocall		odmauto	fileref	&_cstGRoot./standards/cdisc-odm-1.3.0-1.4/macros	
CDISC-ODM	1.3.0	fmtsearch		odmfmt	libref	&_cstGRoot./standards/cdisc-odm-1.3.0-1.4/formats	odmct.sas7bcat
CDISC-ODM	1.3.0	lookup		odmctrl	libref	&_cstGRoot./standards/cdisc-odm-1.3.0-1.4/control	standardlookup.sas7bdat
CDISC-ODM	1.3.0	messages		odmmsg	libref	&_cstGRoot./standards/cdisc-odm-1.3.0-1.4/messages	messages.sas7bdat
CDISC-ODM	1.3.0	properties	initialize	odmprop	fileref	&_cstGRoot./standards/cdisc-odm-1.3.0-1.4/programs	initialize.properties

The **type** and **subtype** columns can be used to reference information that the 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.

This display shows the directory structure for a Microsoft Windows global standards library with **cdisc-sdtm-3.1.1-1.4** 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 17](#).

For more information about what a SAS Clinical Standards Toolkit standard is, see [Chapter 4, “Supported Standards,” on page 37](#).

Common Framework Metadata

Overview

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 23](#). Use of these metadata files is documented in sections that use the SAS Clinical Standards Toolkit metadata.

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 23](#). They are also discussed elsewhere in this document.

Standards Data Set

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 is in the global standards library metadata folder and within each registered standard folder hierarchy here:

```
<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 is in the global standards library metadata folder and within each registered standard folder hierarchy here:

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

Standardlookup

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

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

SASReferences Data Set

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 Files

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 are within each registered standard folder hierarchy here:

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

Messages Data Set

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 is within each registered standard folder hierarchy here:

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

Results Data Set

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.

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 are in the primary SAS Clinical Data Standards Toolkit autocall path:

- Microsoft Windows

```
!sasroot/cstframework/sasmacro
```
- UNIX

```
!sasroot/sasautos
```

For complete macro documentation, see [Appendix 3, “Macro Application Programming Interface,”](#) on page 279.

Initializing the Framework's Global Macro Variables

The framework requires certain global macro variables to execute properly. You 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** value of **initialize**. By default, this entry is located here:

```

<global standards library directory>/standards/cst-
framework-1.4/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, you do not need to call it again during the SAS session, unless you want 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. This 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. This 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. This 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, this display 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, these 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_GETSTANDARDSSASREFERENCES 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. Here 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. Here 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, you do not have to specify paths and member names in each SASReferences file that you 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 need only to change the contents of the **standardversion** column.

This 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.

This 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 are in the primary SAS Clinical Data Standards Toolkit autocall path:

- Microsoft Windows

```
!sasroot/cstframework/sasmacro
```

- UNIX

```
!sasroot/sasautos
```

Note: All of the maintenance usage scenarios require that you have Write access to the Global Standards Library.

For complete macro documentation, see [Appendix 3, “Macro Application Programming Interface,” on page 279](#).

TIP Best Practice Recommendation: Do not modify SAS-supplied Global Standards Library files, but instead modify copies of these files. Leaving the SAS-supplied files intact enables these files to be updated without concern about overwriting or losing your changes.

Registering a New Version of a Standard

This 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 this information:

- The main path to the directory that contains the standard version's files.
- The path to the registration data sets that are used to populate the global standards library metadata data sets. This is the name of the subfolder in the `_cstRootPath` parameter value.

- 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 9 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 the SAS Clinical Standards Toolkit, see [Chapter 3, “Metadata File Descriptions,”](#) on page 23. 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. This 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.

This 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

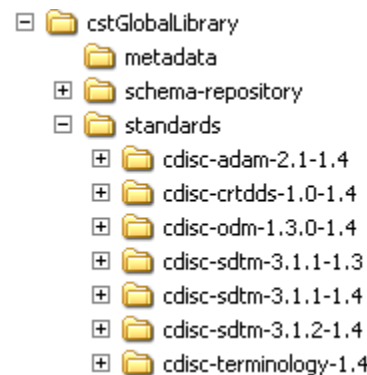
Suppose that the SAS Clinical Standards Toolkit 1.3 is currently installed and used. The SAS Clinical Standards Toolkit 1.4 is released. You want the product updates for a standard version. In the following steps, the CDISC SDTM standard is used as an example. However, the steps apply to all other standard 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 1.3 code base or metadata that you want to preserve. Or, you might want to test the SAS Clinical Standards Toolkit 1.4 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 this example, two subdirectories exist for CDISC SDTM 3.1.1.

Display 2.7 Multiple Versions per Standard in the Global Standards Library



The `cdisc-sdtm-3.1.1-1.3` directory contains files installed with the SAS Clinical Standards Toolkit 1.3. The `cdisc-sdtm-3.1.1-1.4` directory contains files installed with the SAS Clinical Standards Toolkit 1.4.

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.

This display shows that the registered version CDISC SDTM 3.1.1.-1.3 indicates that it is the original version that was shipped with the SAS Clinical Standards Toolkit 1.3. 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-ADAM	ADAM	2.1	1.4	Y	%_cstGRoot/standards/cdisc-adam-2.1-1.4
2	CDISC-CRTDDS	CRT	1.0	1.4	Y	%_cstGRoot/standards/cdisc-crtdds-1.0-1.4
3	CDISC-ODM	ODM	1.3.0	1.4	Y	%_cstGRoot/standards/cdisc-odm-1.3.0-1.4
4	CDISC-SDTM	SDTM	3.1.1	1.3	Y	%_cstGRoot/standards/cdisc-sdtm-3.1.1-1.3
5	CDISC-SDTM	SDTM	3.1.2	1.4	N	%_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4

Step 2: Register the updated CDISC SDTM 3.1.1 metadata in the global standards library to use the SAS Clinical Standards Toolkit 1.4.

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-3.1.1-1.4/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 this 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.

This display shows that the CDISC SDTM 3.1.1 standard is now reregistered, the product revision in use is 1.4, and CDISC SDTM 3.1.2 is registered as the default standard.

Display 2.9 Global Standards Library Metadata Standards Data Set after Updates

	standard	mnemonic	standardversion	productrevision	isstandard	rootpath
1	CDISC-ADAM	ADAM	2.1	1.4	Y	&_cstGRoot./standards/cdisc-adam-2.1-1.4
2	CDISC-CRTDDS	CRT	1.0	1.4	Y	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4
3	CDISC-ODM	ODM	1.3.0	1.4	Y	&_cstGRoot./standards/cdisc-odm-1.3.0-1.4
4	CDISC-SDTM	SDTM	3.1.2	1.4	Y	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4
5	CDISC-SDTM	SDTM	3.1.1	1.4	N	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.4

Chapter 3

Metadata File Descriptions

Overview	23
Standards	23
StandardSASReferences	25
Standardlookup	26
SASReferences	27
Properties	29
Messages	31
Results	33
Additional Metadata Files	35
Overview	35
Validation Master (Validation Control)	35
Reference_Tables(Source_Tables)	35
Reference_Columns(Source_Columns)	35
Validation Metrics	35
CDISC CRT-DDS Style Sheet	36

Overview

The 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.
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.
supportvalidation	(\$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 is here:

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

The global standards library data set contains these records, which are provided with the SAS Clinical Standards Toolkit 1.4 (the columns are continued in the second image):

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

	standard	mnemonic	standardversion	comment	rootpath	isstandarddefault	iscstframework	isdatastandard	supportvalidation
1	CDISC-ADAM	ADAM	2.1	CDISC ADAM V2.1	&_cstGRoot/standards/cdisc-adam-2.1-	Y	N	Y	Y
2	CDISC-CRTDDS	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot/standards/cdisc-crtdds-1.0-	Y	N	Y	Y
3	CDISC-ODM	ODM	1.3.0	CDISC ODM V1.3.0	&_cstGRoot/standards/cdisc-odm-1.3.0-	Y	N	Y	Y
4	CDISC-SDTM	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot/standards/cdisc-sdtm-3.1.1-	N	N	Y	Y
5	CDISC-SDTM	SDTM	3.1.2	CDISC SDTM V3.1.2	&_cstGRoot/standards/cdisc-sdtm-3.1.2-	Y	N	Y	Y
6	CDISC-TERMINOLOGY	CT	200810	CDISC Terminology, Packages 1, 2A, 2B, LABTEST	&_cstGRoot/standards/cdisc-terminolog	N	N	N	N
7	CDISC-TERMINOLOGY	CT	201003	CDISC Terminology 2010-03-05	&_cstGRoot/standards/cdisc-terminolog	Y	N	N	N
8	CDISC-TERMINOLOGY	CT	NCI_THESAURUS	CDISC Terminology 2011-01-07	&_cstGRoot/standards/cdisc-terminolog	N	N	N	N
9	CST-FRAMEWORK	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot/standards/cst-framework-1-	Y	Y	N	N

	standard	mnemonic	standardversion	isxmlstandard	importxsl	exportxsl	schema	productrevision
1	CDISC-ADAM	ADAM	2.1	N				1.4
2	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	1.4
3	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	1.4
4	CDISC-SDTM	SDTM	3.1.1	N				1.4
5	CDISC-SDTM	SDTM	3.1.2	N				1.4
6	CDISC-TERMINOLOGY	CT	200810	N				1.4
7	CDISC-TERMINOLOGY	CT	201003	N				1.4
8	CDISC-TERMINOLOGY	CT	NCI_THESAURUS	N				1.4
9	CST-FRAMEWORK	CST	1.2	N				1.4

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 is here:

```
<global standards library directory>/standards/cdisc-
sdtm-3.1.2-1.4/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 is here:

```
<global standards library directory>/standards/cdisc-
sdtm-3.1.2-1.4/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	sdtmcntrl	libref	validation/control		. validation_stdref.sas7bdat	
2	CDISC-SDTM	3.1.2	referencecontrol	validation	sdtmcntrl	libref	validation/control		. validation_master.sas7bdat	
3	CDISC-SDTM	3.1.2	referencemetadatas	table	sdtmref	libref	metadata		. reference_tables.sas7bdat	
4	CDISC-SDTM	3.1.2	referencemetadatas	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	classmetadatas	table	sdtmcls	libref	metadata		. class_tables.sas7bdat	
7	CDISC-SDTM	3.1.2	classmetadatas	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	sdtmprop2	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. You can

overwrite this value. 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 is here:

```
<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.

This display shows the content for the CDISC SDTM StandardSASReferences data set that is described in [Display 3.2 on page 25](#). 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
42	CDISC-SDTM	3.1.2	autocall		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/macros	1	
43	CDISC-SDTM	3.1.2	classmetadata	column	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/metadata		. class_columns.sas7bdat
44	CDISC-SDTM	3.1.2	classmetadata	table	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/metadata		. class_tables.sas7bdat
45	CDISC-SDTM	3.1.2	lookup		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/control		. standardlookup.sas7bdat
46	CDISC-SDTM	3.1.2	messages		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/messages	1	messages.sas7bdat
47	CDISC-SDTM	3.1.2	properties	initialize	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/programs	1	initialize.properties
48	CDISC-SDTM	3.1.2	properties	validation	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/programs	1	validation.properties
49	CDISC-SDTM	3.1.2	referencecontrol	standardref	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/validation/control		. validation_stdref.sas7bdat
50	CDISC-SDTM	3.1.2	referencecontrol	validation	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/validation/control		. validation_master.sas7bdat
51	CDISC-SDTM	3.1.2	referencemetadata	column	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/metadata		. reference_columns.sas7bdat
52	CDISC-SDTM	3.1.2	referencemetadata	table	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4/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 27.

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 the 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

Column Name	Column Length	Description
refcolumn	(\$32)	Associated SAS Clinical Standards Toolkit column name
refvalue	(\$200)	Associated SAS Clinical Standards Toolkit column value
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 is here:

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

Here is an example of the records in a Standardlookup data set:

Display 3.4 Standardlookup Data Set Content in the Global Standards Library

	SASref	table	column	refcolumn	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 67, describes the content and usage of SASReferences data sets.

This 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-1.4/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-1.4/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 these is true: <ol style="list-style-type: none"> Multiple records exist within these types—autocall, fmtsearch, messages. Library concatenation is wanted (multiple librefs are within the same value of SASref for a type). There is a need to establish precedence within a type (for example, look first in this library and 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.

This 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		sdmcode	libref	&_cstGRoot\standards\cdisc-sdt	1		Standard-specific macro library
CDISC-SDTM	3.1.1	fmtsearch		srcfmt	libref	&studyRootPath\terminology\lor	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		referenceceterm		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. The 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

The 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 1, “Global Macro Variables,”](#) on page 261. These macro variables are derived from properties files provided by SAS.

This 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, “Global Macro Variables,”](#) on page 261.

Table 3.4 Properties File Structure

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

Messages

By default, the 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.

This table describes the structure of all the message files.

Table 3.5 Messages Data Set Structure

Column Name	Column Length	Description	Optional or Required
resultid	(\$8)	The message ID. The 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). You can use any naming convention limited to eight 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
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

Column Name	Column Length	Description	Optional or Required
checkseverity	(\$40)	The severity as assigned by checksource . This value is mapped to these 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 the 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 is here:

```
<global standards library directory>/standards/cst-  
framework-1.4/messages/messages.sas7bdat
```

This 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 2, “Framework Messages,” on page 275](#). 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 this table.

Table 3.6 Results Data Set Structure

Column Name	Column Length	Description
resultid	(\$8)	<p>Result ID. The resultid is a message ID from the standard Messages data set (for example, framework or CDISC SDTM). The 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 <i><global standards library directory>/metadata</i>. This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). You can use any naming convention limited to eight characters.</p> <p>Value should be non-null.</p>
checkid	(\$8)	<p>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 <i><global standards library directory>/metadata</i>. This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). You can use any naming convention limited to eight characters.</p> <p>Value should be non-null for validation processes. Otherwise, this column is optional.</p>
resultseq	(8.)	<p>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.</p> <p>Value should be non-null positive integer.</p>
seqno	(8.)	<p>Sequence number relative to resultseq. This value is a unique sequence number for the Results record in each unique value of resultseq.</p> <p>Value should be non-null positive integer.</p>

Column Name	Column Length	Description												
srcdata	(\$200)	Source data. This string generally specifies: <ul style="list-style-type: none">(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). <table><tr><td>Info</td><td>Informational note</td></tr><tr><td>Note</td><td>Problem detected, low severity</td></tr><tr><td>Warning</td><td>Problem detected, medium severity</td></tr><tr><td>Warning: Check not run</td><td>No assessment able to be made</td></tr><tr><td>Warning: Check not completed</td><td>Full compliance assessment could not be made</td></tr><tr><td>Error</td><td>Problem detected, high severity</td></tr></table> Value should be non-null.	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
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													
resultflag	(8.)	A value that determines whether a problem has been detected. The values are 0=no, otherwise, yes. <table><tr><td>-1</td><td>Validation check not run</td></tr><tr><td>0</td><td>No problem detected (value always 0 when resultseverity=Info)</td></tr><tr><td>1</td><td>Validation check run, error detected</td></tr></table> Value should be non-null.	-1	Validation check not run	0	No problem detected (value always 0 when resultseverity=Info)	1	Validation check run, error detected						
-1	Validation check not run													
0	No problem detected (value always 0 when resultseverity=Info)													
1	Validation check run, error detected													
_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.												
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 117](#) and [Display 6.10 on page 117](#).

Additional Metadata Files

Overview

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 the 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.supportvalidation` field, see [Table 3.1 on page 24](#).) This data set should have the columns as defined in [Table 6.3 on page 89](#), 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 85](#) describes the table metadata for the CDISC SDTM standard. For selected actions, the 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 86](#) describes the column metadata for the CDISC SDTM standard. For selected actions, the 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 6.10 on page 100](#) describes the metrics metadata for the CDISC SDTM standard, and [Display 6.2 on page 101](#) 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	37
Overview	37
CDISC SDTM	40
Purpose	40
Release Dates	40
CDISC SDTM 3.1.1 Reference Standard	41
Description	42
CDISC SDTM 3.1.2 Reference Standard	43
Description	44
CDISC ADaM 2.1	46
Purpose	46
Release Date	47
Regulatory Basis	47
CDISC ADaM 2.1 Reference Standard	47
CDISC CRT-DDS 1.0	51
Purpose	51
Release Date	51
Regulatory Basis	51
CDISC CRT-DDS 1.0 Reference Standard	51
CDISC ODM 1.3.0	56
Purpose	56
Release Date	56
CDISC ODM 1.3.0 Reference Standard	56
CDISC Terminology	62
Purpose	62
CDISC Terminology Reference Standard	63

SAS Representation of Standards

Overview

The SAS Clinical Standards Toolkit is designed to support various clinical standards. The 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 the SAS Clinical Standards Toolkit standards.

- Each supported standard is represented in one or more SAS files. This facilitates these points:
 - 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.

- You are able to define your own customized standards, or you are able to modify existing SAS standards. For more information about how new standards are registered in the SAS Clinical Standards Toolkit, see [“Registering a New Version of a Standard” on page 17](#).

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 your 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 the 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 the SAS Clinical Standards Toolkit.

- A distinct folder hierarchy within the global standards library, comprising the previous definition and any supporting files required by the SAS Clinical Standards Toolkit.

By default, reference standards are specified in the global standards library that is created when the 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.

This global standards library folder hierarchy is provided for CDISC SDTM:

Display 4.1 Global Standards Library Folder Hierarchy



The **metadata** folder contains the data set and column metadata for each supported domain. The 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 the 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 the 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 column definitions used in the SDTM domains.
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 properties 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 the 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.

TIP 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

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

Overview of the CDISC SDTM 3.1.1 Domains

This section lists the 25 domains that are included in the CDISC SDTM 3.1.1 SAS Clinical Standards Toolkit reference standard.

Within these 25 domains, 495 columns have been defined.

Special Purpose Domains

The Special Purpose domains are:

- Demographics-DM
- Comments-CO

Interventions Domains

The Interventions domains are:

- Concomitant Medications-CM
- Exposure-EX
- Substance Use-SU

Findings Domains

The Findings domains are:

- ECG Tests-EG
- Inclusion/Exclusion Exceptions-IE
- Laboratory Tests-LB
- Questionnaires-QS
- Physical Examinations-PE
- Subject Characteristics-SC
- Vital Signs-VS

Events Domains

The Events domains are:

- Adverse Events-AE
- Disposition-DS
- Medical History-MH
- Protocol Deviations-DV

Trial Design Domains

The Trial Design domains are:

- Trial Elements-TE

- Trial Arms-TA
- Trial Visits-TV
- Subject Elements-SE
- Subject Visits-SV
- Trial Inclusion/Exclusion Criteria-TI
- Trial Summary-TS

Special Purpose Relationship Data Sets

The special purpose relationship data sets are:

- Supplemental Qualifiers-SUPPQUAL
- Related Records-RELREC

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 these 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

Overview of the CDISC SDTM 3.1.2 Domains

This section lists the 32 domains that are included in the CDISC SDTM 3.1.2 SAS Clinical Standards Toolkit reference standard.

Within these 32 domains, 723 columns have been defined.

Special Purpose Domains

The Special Purpose domains are:

- Demographics-DM
- Comments-CO
- Subject Elements-SE
- Subject Visits-SV

Findings Domains

The Findings domains are:

- Drug Accountability-DA
- ECG Tests-EG
- Inclusion/Exclusion Criterion Not Met-IE
- Laboratory Tests-LB
- Laboratory Test Results-LB
- Microbiology Specimen-MB
- Microbiology Susceptibility Test-MS
- PK Concentrations-PC
- Physical Examinations-PE
- PK Parameters-PP
- Questionnaires-QS
- Subject Characteristics-SC
- Vital Signs-VS

Findings about Domains

The Findings About domain is:

- Findings About-FA

Events Domains

The Events domains are:

- Adverse Events-AE
- Clinical Events-CE
- Disposition-DS
- Protocol Deviations-DV

- Medical History-MH

Interventions Domains

The Interventions domains are:

- Concomitant Medications-CM
- Exposure-EX
- Substance Use-SU

Trial Design Domains

The Trial Design domains are:

- Trial Arms-TA
- Trial Elements-TE
- Trial Inclusion/Exclusion Criteria-TI
- Trial Summary-TS
- Trial Visits-TV

Relationship Data Sets

The relationship data sets are:

- Supplemental Qualifiers-SUPPQUAL
- Related Records-RELREC

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 these 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 [Table 4.1 on page 44](#)])
- as column metadata for each domain (`reference_columns` in the standard metadata folder [see the example in [Table 4.2 on page 45](#)])

Table 4.1 Sample `Reference_Tables` Record (CDISC SDTM 3.1.2)

Column Name	Column Value
<code>sasref</code>	REFDATA
<code>table</code>	CE
<code>label</code>	Clinical Events

Column Name	Column Value
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 4.2 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	
xmldatatype	text
xmlodelist	STENRF
core	Perm
origin	Derived

Column Name	Column Value
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 these 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 ADaM 2.1

Purpose

The Analysis Data Model (ADaM) specifies the fundamental principles and standards to follow when creating analysis data sets and associated metadata. ADaM supports efficient generation, replication, and review of analysis results. The design of analysis data sets is generally driven by the scientific and medical objectives of the clinical trial. A fundamental principle is that the structure and content of the analysis data sets must

support clear, unambiguous communication of the scientific and statistical aspects of the clinical trial.

The purpose of ADaM is to provide a framework that enables analysis of the data. At the same time, ADaM enables reviewers and other recipients of the data to have a clear understanding of the data's lineage from collection to analysis to results. Whereas ADaM is optimized to support data derivation and analysis, CDISC Study Data Tabulation Model (SDTM) is optimized to support data tabulation.

Release Date

CDISC ADaM Analysis Data Model, Final Version 2.1, December 17, 2009

Regulatory Basis

(Source: Submission of Data in CDISC Format to CBER, <http://www.fda.gov/BiologicsBloodVaccines/DevelopmentApprovalProcess/ucm209137.htm>, page updated: January 6, 2011)

Effective December 15, 2010, SDTM and ADaM are being accepted for all BLA submissions. (Source: Study Data Specifications, Version 1.5.1, January 4, 2010)

Before submission, sponsors should contact the appropriate center's review division to determine the division's analysis data set needs. The CDISC ADaM standard for analysis data sets (<http://www.cdisc.org/adam>) can be used if it is acceptable to the review division.

(Source: *CDER Common Data Standards Issues Document*, Version 1.0/May 2011, <http://www.fda.gov/downloads/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/UCM254113.pdf>)

To determine how to create ADaM analysis data sets for submission to CDER, sponsors should refer to the following three documents:

- the Analysis Data Model (<http://www.CDISC.org/adam>)
- the *Analysis Data Model Implementation Guide* (<http://www.CDISC.org/adam>)
- the *FDA Study Data Specifications* (<http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM199599.pdf>)

You should comply with the *Analysis Data Model Implementation Guide*. Any specific questions about these documents should be discussed with the review division.

CDISC ADaM 2.1 Reference Standard

Section 2.1 of the *Analysis Data Model Implementation Guide* provides the fundamental principles of the CDISC ADaM model.

- Analysis data sets and associated metadata must clearly and unambiguously communicate the content and source of the data sets supporting the statistical analyses performed in a clinical study.
- Analysis data sets and associated metadata must provide traceability to enable an understanding of where an analysis value came from.
- Analysis data sets must be readily usable with commonly available software tools.

- Analysis data sets must be associated with metadata to facilitate clear and unambiguous communication. Ideally, the metadata is machine-readable.
- Analysis data sets should have a structure and content that enable statistical analyses to be performed with minimal programming. Such data sets are described as analysis-ready.

Implementation of the CDISC ADaM 2.1 reference standard in the SAS Clinical Standards Toolkit supports each of these principles.

The number and structure of analysis data sets are highly dependent on the type of study, the study objectives as defined in the statistical analysis plan, and discussions with the reviewing authority. ADaM data sets incorporate derived and collected data that permit analysis with little or no additional programming. Data can be from various SDTM domains, other ADaM data sets, or any combination thereof.

As initially released, the CDISC ADaM 2.1 reference standard supports two types of analysis data set structures:

- The subject-level analysis data set (ADSL) provides descriptive information about subjects, such as study disposition, demographic, and baseline characteristics. The ADSL is the primary source for subject-level variables included in other analysis data sets, such as population flags and treatment variables. There is only one ADSL per study, and the ADSL and its related metadata are required in each CDISC-based submission of data from a clinical trial, even if no other analysis data sets are submitted.
- The ADaM Basic Data Structure (BDS) is used for the majority of ADaM data sets, regardless of the therapeutic area or type of analysis. Each BDS data set contains one or more records per subject and analysis parameter. The structure of some BDS data sets might include an analysis time point. A record in a BDS analysis data set can represent an observed, derived, or imputed value required for analysis. Each BDS data set contains a core set of variables that describe the analysis parameter and the value being analyzed. A data value can be derived from any source file, including any combination of SDTM and ADaM data sets.

The Analysis Data Model identifies four types of metadata that are captured and supported by the SAS Clinical Standards Toolkit.

Table 4.3 ADaM Metadata Types and SAS Clinical Standards Toolkit Locations

ADaM Metadata Type	SAS Clinical Standards Toolkit Location
Analysis data set metadata	reference_tables.sas7bdat
Analysis variable metadata	reference_columns.sas7bdat
Analysis parameter-value-level metadata	reference_columns.sas7bdat (parameterid column)
Analysis results metadata	Derivable from reference_columns.sas7bdat (where table='RESULTS')

Version 1.0 of the *Analysis Data Model Implementation Guide* (ADaMIG) defines a common set of ADSL and BDS columns that can be used as templates for any ADaM analysis data set. These columns are defined in the SAS Clinical Standards Toolkit reference_tables and reference_columns data sets. These SAS Clinical Standards Toolkit data sets contain a metadata representation of the analysis results metadata. Empty

ADSL, BDS, and analysis results data sets can be derived from the SAS Clinical Standards Toolkit global standards library using the utility macro `cst_createTablesForDataStandard`.

To see the SAS Clinical Standards Toolkit metadata that is available, [Table 4.4 on page 49](#) and [Table 4.5 on page 49](#) provide a sample record from the `reference_tables` and `reference_columns` data sets, respectively. These data sets are found in the standard metadata folder.

Table 4.4 Sample Reference_Tables Record (CDISC ADaM 2.1)

Column Name	Column Value
sasref	REFDATA
table	ADSL
label	Subject-Level Analysis Dataset
class	ADSL
xmlpath	adsl.xpt
xmltitle	
structure	One record per subject
purpose	Analysis
keys	USUBJID
state	Final
date	2009-12-17
standard	CDISC-ADAM
standardversion	2.1
standardref	
comment	Subject disposition, demographic, and baseline characteristics
documentation	SAP, <deriving SAS code>

Table 4.5 Sample Reference_Columns Record (CDISC ADaM 2.1)

Column Name	Column Value
sasref	REFDATA
table	BDS
column	TRTP

Column Name	Column Value
label	Planned Treatment
order	5
type	C
length	40
displayformat	
xmldatatype	text
xmlcodelist	
core	Req
role	TreatmentBDS
term	
parameter	
sourcederivation	
qualifiers	
standard	CDISC-ADAM
standardversion	2.1
standardref	
comment	TRTP is a record-level identifier that represents the planned treatment attributed to a record for analysis purposes. TRTP indicates how treatment varies by record within a subject and enables analysis of crossover and other designs. TRT:xxxxP (copied from ADSL) may also be needed for some analysis purposes, and may be useful for traceability and to provide context.

The SAS Clinical Standards Toolkit CDISC ADaM reference standard also provides metadata and code to validate the structure and content of the ADaM analysis data sets.

To enable validation, supplemental files supporting ADaM validation processes include these SAS Clinical Standards Toolkit global standards library files:

- The Validation Master data set in the **validation/control** folder contains the superset of checks validating the structure and content of each analysis data set. These checks are based on version 1.1 of the CDISC ADaM Validation Checks as prepared by the CDISC ADaM team.
- The Messages data set in the **messages** folder provides error messaging for all Validation Master checks.

- SAS code in the **macros** folder provides code that is specific to ADaM that augments code that is provided in the primary SAS Clinical Standards Toolkit autocall library (`!sasroot/cstframework/sasmacro`).

These supplemental files, in whole or in part, define the SAS Clinical Standards Toolkit CDISC ADaM 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

Overview

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 [Table 4.6 on page 51](#)])
- as column metadata for 176 columns in the 39 data sets (`reference_columns` in the standard metadata folder [see the example in [Table 4.7 on page 52](#)])

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

Column Name	Column Value
<code>sasref</code>	REFDATA

Column Name	Column Value
table	ItemGroupDefs
label	
keys	OID
standard	CDISC-CRTDDS
standardversion	1.0
standardref	
comment	
xmlelementname	ItemGroupDefs
class	ItemGroupDefs
qualifiers	

Table 4.7 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

Column Name	Column Value
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. The TE domain metadata is shown in this code:

```
<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>
```

The TE domain metadata is shown in this table.

Table 4.8 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

Column	Value
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.

This 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 4.9 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	RCErrorsTranslatedText
ItemGroupAliases	Study
ItemGroupDefItemRefs	StudyEventDefs
ItemGroupDefs	StudyEventFormRefs

Table	Table
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 this 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 4.10 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

CDISC CRT-DDS SAS Data Set Construction

Appendix 9, “CRT-DDS 1.0 SAS Data Sets,” on page 477 lists the data sets with member columns that form the CDISC CRT-DDS 1.0 data in the SAS Clinical Standards Toolkit.

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

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, supplementary files include these global standards library files:

- A SAS format catalog (crtddset.sas7bcat) in the **formats** folder provides valid values for selected columns in the 39 data sets 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 data sets.
- 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.xml` file. The style sheet is copied from http://www.cdisc.org/stuff/contentmgr/files/0/464923b10ea16b477151fcaa9f465166/misc/define1_0_0.xml. 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.

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

The SAS Clinical Standards Toolkit 1.4 supports this CDISC ODM 1.3.0 functionality:

- reading and representing in SAS a complete `odm.xml` file (specific limitations are noted below)
- building an `odm.xml` file from a SAS representation of the ODM standard
- schema-level validating of an `odm.xml` file
- validating the structure and content of the SAS representation of an `odm.xml` file
- identifying unsupported (unrecognized) ODM elements and attributes by using a sample tool

The SAS Clinical Standards Toolkit 1.4 does not support this CDISC ODM 1.3.0 functionality:

- reading or writing the DigitalSignatures section of the ODM
- vendor or customer extensions of the ODM
- any functionality added with ODM 1.3.1

- processing is limited to a single ODM file (for example, the use of PriorFileOID to reference another file is ignored)
- Full file metadata is expected in each file.
- Effective support only for ODM FileType=Snapshot. The SAS Clinical Standards Toolkit 1.4 makes no attempt to process multiple transactions per data point; multiple transactions are saved in the SAS ODM representation for subsequent processing

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 66 data sets (reference `_tables` in the standard metadata folder [see the example in [Table 4.11 on page 57](#)])
- as column metadata for 315 columns in the 66 data sets (reference `_columns` in the standard metadata folder [see the example in [Table 4.12 on page 57](#)])

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	FK_FORMDATA FK_REFERENCEDATA ITEMGROUPOID ITEMGROUPREPEATKEY
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	SUBJECTKEY

Column Name	Column Value
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
xmlodelist	
xmlattributename	SubjectKey
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 this 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>
```

This 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	"P2006-101"
		MetaDataVersionOID	"101.01"
<SubjectData SubjectKey="1000" TransactionType="Insert">	SubjectData	SubjectKey	"1000"
		TransactionType	"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	"101.USUBJID"
		ItemDataType	"ItemDataString"
		Value	"101-01-01"
<ItemDataString ItemOID="101.SEX">F</ ItemDataString>	ItemData	ItemOID	"101.SEX"
		ItemDataType	"ItemDataString"
		Value	"F"

This table lists the complete set of 66 tables that form the SAS Clinical Standards Toolkit 1.4 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
AdminData	ItemRangeCheckValues
Annotation	ItemRCFormalExpression
AnnotationFlag	ItemRole
Association	KeySet
AuditRecord	Location
ClinicalData	LocationVersion
CLItemDecodeTranslatedText	MeasurementUnits
CodeListItems	MetaDataVersion

Table	Table
CodeLists	MethodDefFormalExpression
ConditionDefFormalExpression	MethodDefs
ConditionDefs	MethodDefTranslatedText
ConditionDefTranslatedText	MUTranslatedText
EnumeratedItems	ODM
ExternalCodeLists	Presentation
FormData	ProtocolEventRefs
FormDefArchLayouts	ProtocolTranslatedText
FormDefItemGroupRefs	RCErrorsTranslatedText
FormDefs	ReferenceData
FormDefTranslatedText	Signature
ImputationMethods	SignatureDef
ItemAliases	Study
ItemData	StudyEventData
ItemDefs	StudyEventDefs
ItemDefTranslatedText	StudyEventDefTranslatedText
ItemGroupAliases	StudyEventFormRefs
ItemGroupData	SubjectData
ItemGroupDefItemRefs	User
ItemGroupDefs	UserAddress
ItemGroupDefTranslatedText	UserAddressStreetName
ItemMURRefs	UserEmail
ItemQuestionExternal	UserFax
ItemQuestionTranslatedText	UserLocationRef
ItemRangeChecks	UserPhone

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 this 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	64	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 table metadata for the 66 data sets and the column metadata for the 315 columns in those data sets that comprise the SAS representation of the CDISC ODM 1.3.0 standard are in this folder:

`<global standards library directory>/standards/cdisc-
odm-1.3.0-1.4/metadata.`

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

[Appendix 8, “ODM 1.3.0 SAS Data Sets,” on page 447](#) shows the CDISC ODM 1.3.0 data model that is the foundation for implementation 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 key variable name is appended with two asterisks (**). Some data sets do not have a key. A foreign key variable name is appended with two carat characters (^). A foreign key variable name references the name of the data set for which it is a foreign key. The data set name is referenced within brackets. A required field is marked with an X between brackets, [X]. A required field is a field for which a non-nil and non-whitespace-only value must be provided for any observation in that data set.

Only the ODM data set, which contains valid values for the FileOID, CreationDateTime, and FileType variables, is needed to create a minimal, but valid, CDISC ODM-compliant XML document. This is based on the CDISC ODM standard, which is flexible. All table and column names are case sensitive. They must be specified exactly as shown.

In the SAS implementation of the relational data model, the keys are extended to define a unique record in every SAS data set. For example, a unique record in the EnumeratedItems data set is defined by the variables FK_CODELISTS and CODEDVALUE. These SAS data set keys are in the table metadata in the SAS Reference_Tables data set.

Starting in ODM 1.3.0, there are two forms of the ItemData element, which is the element used by ODM for transmitting clinical data item values. These two forms are untyped and typed. Here is an example of a typed ItemData element:

```
<ItemDataFloat ItemOID="ItemDef.OID.VS.VSSTRESN"
TransactionType="Insert">76</ItemDataFloat>
```

Here is an example of an untyped ItemData element:

```
<ItemData ItemOID="ID.AETERM" Value="HEADACHE" />
```

Both of these data values are stored in the Value variable in the ItemData SAS data set. In the case of typed data, the ItemDataType variable in the ItemData SAS data set has the data type (for example, Float). In the case of untyped data, the ItemDataType variable in the ItemData SAS data set is null.

Typed and untyped data transmission should not be mixed within a single ODM file. However, in the example provided by the SAS Clinical Standards Toolkit, both types are part of the same example for demonstration purposes.

With the production release of the SAS Clinical Standards Toolkit 1.4, the CDISC ODM standard supports 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 validates the structural integrity of the odm.xml file.

To support all of this functionality, supplemental files include the following global standards library files:

- A SAS format catalog (crtdsct.sas7bcat) in the **formats** folder provides valid values for selected columns in the 66 tables of the SAS representation.
- The Messages data set in the **messages** folder provides error messaging for all Validation Master checks.
- The Validation Master data set in the **validation/control** folder contains the superset of checks validating the structure and content of the 66 tables.
- SAS code in the **macros** folder provides CDISC ODM-specific code that augments the code 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.

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.

The 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. Several snapshots are currently supported across several standards.

For SDTM, these snapshots are supplied:

- 200810 snapshot was taken from the NCI EVS Controlled Terminology for SDTM released October 2008 in support of the SAS Clinical Standards Toolkit 1.2. This snapshot supports CDISC SDTM 3.1.1.
- 201003 snapshot was taken from the NCI EVS Controlled Terminology for SDTM released March 2010 in support of the SAS Clinical Standards Toolkit 1.3. This snapshot supports CDISC SDTM 3.1.2.
- 201104 snapshot was taken from the NCI EVS Controlled Terminology for SDTM released April 2011 in support of the SAS Clinical Standards Toolkit 1.4. This snapshot supports CDISC SDTM 3.1.2.

For ADaM, this snapshot is supplied:

- The 201101 snapshot was taken from the NCI EVS Controlled Terminology for SDTM released January 2011 in support of the SAS Clinical Standards Toolkit 1.4. This snapshot supports CDISC ADaM 2.1.

For CDASH, this snapshot is supplied:

- The 201104 snapshot was taken from the NCI EVS Controlled Terminology for SDTM released April 2011 in support of the SAS Clinical Standards Toolkit 1.4. This snapshot supports CDISC CDASH. Although the SAS Clinical Standards Toolkit is not shipped with this standard, the terminology is provided as a convenience for our users.

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 this global standards library folder (where **xxxx** is the specific standard (adam, cdash, or sdtm) and **YYYYYY** is the specific snapshot (200810, 201101, and so on):

```
<global standards library directory>/standards/cdisc-terminology1.4/cdisc-xxxx/<current OR YYYYYY>/formats
```

Here is an example for SDTM:

```
C:/cstGlobalLibrary/standards/cdisc-terminology-1.4/cdisc-sdtm/201104/formats
```

The **current** folder is a convenience and enables you to place the current terminology that you are using within the SAS Clinical Standards Toolkit. The default for this folder is the most recent release of the Controlled Terminology as shipped with the SAS

Clinical Standards Toolkit. This table lists the 62 code lists (SAS formats) that are in the cumulative CDISC-Terminology-201104 SDTM 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
EGMETHOD	ECG Test Method	22
EGSTRESC	ECG Result	119
EGTEST	ECG Test Name	46
EGTESTCD	ECG Test Code	46
ETHNIC	Ethnic Group	4
EVAL	Evaluator	15
FREQ	Frequency	51
FRM	Pharmaceutical Dosage Form	168
IECAT	Category for Inclusion/Exclusion	2
LBTEST	Laboratory Test Name	649
LBTESTCD	Laboratory Test Code	649
LOC	Anatomical Location	303
MARISTAT	Marital Status	9
METHOD	Method	72

Code List/Format Name	Description	Unique Values
MICROORG	Microorganism	871
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
PKPARAM	PK Parameters	187
PKPARAMCD	PK Parameters Code	187
PKUNIT	PK Parameter Units of Measure	207
POSITION	Position	10
RACE	Race	5
RELTYPE	Relationship Type	2
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

Code List/Format Name	Description	Unique Values
TCNTRL	Control Type	3
TDIGRP	Diagnosis Group	1
TINDTP	Trial Indication Type	5
TOXGR	Common Terminology Criteria for Adverse Events V4.0	6
TPHASE	Trial Phase	12
TSPARM	Trial Summary Parameter Test Name	30
TSPARMCD	Trial Summary Parameter Test Code	30
TTYPE	Trial Type	8
UNIT	Unit	310
VSRESU	Units for Vital Signs Results	14
VSTEST	Vital Signs Test Name	15
VSTESTCD	Vital Signs Test Code	15

Chapter 5

SASReferences File

Overview	67
Building a SASReferences File	67
How Is a SASReferences File Used?	75
Overview	75
Communicating the Filename and Location to the SAS Clinical Standards Toolkit	75
Assessing Structural Integrity and Content	76
Translating Content for a SAS Session	78

Overview

The 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 23](#). 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, the 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 28](#) and example content is provided in [Display 3.5 on page 29](#).

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 in this location in SAS 9.3:

```
!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.4/  
sample/cdisc-sdtm-3.1.2/sascstdemodata/control
```

An excerpt of this sample SASReferences file is provided in [Display 3.5 on page 29](#).

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 is here:

```
<global standards library directory>/standards/cst-  
framework-1.4/templates
```

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-3.1.2-1.4/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 23](#) 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 the SAS Clinical Standards Toolkit to values itemized in the framework Standardlookup data set found in:

```
<global standards library directory>/standards/cst-  
framework-1.4/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.

This 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.
codemodule		Identifies an “external” code segment, which is identified using a SAS fileref, that might be included (%include) in a SAS Clinical Standards Toolkit process. Examples include code that derives a CDISC ADaM data set or that generates an ADaM report.
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 the 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 or tlfxml	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. TLFXML refers to the tables, listings, and figures XML file that is used in ADaM 2.1.
fintsearch		Provides a way to build the format search path for a validation process. The SAS Clinical Standards Toolkit sets the SAS fintsearch 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 you must specify a value. The type=fintsearch value is optional unless one or more checks are to be run that assess value compliance against a SAS format.

Type	Subtype	Comments
lookup		Identifies a data set (Standardlookup) that is associated with each The 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. You must specify value only with user customizations that require new or modified messages. The 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	initialize, validation, or report	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.
referencecontrol	validation or standardref	<p>If subtype=validation, then the value identifies the standard-supplied master super-set of supported validation checks. Although 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.</p> <p>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.</p>
referenceceterm		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=referenceceterm 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 you must specify a value only to override the default for the standard. Records for both subtypes are required.

Type	Subtype	Comments
referencexml	stylesheet, map, or tlfxml	<p>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.</p> <p>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 the 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).</p>
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 the 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 the SAS Clinical Standards Toolkit 1.4. 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 you must specify a value. 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.
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.

Type	Subtype	Comments
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 the SAS Clinical Standards Toolkit 1.4. 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 this macro was described:

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

This macro call produces this SASReferences file:

Display 5.1 Standard SASReferences File for CST-FRAMEWORK

	standard	standardversion	type	subtype	SASref	reltype	path	order	memname	comment
1	CST-FRAMEWORK	1.2	control	reference	csttmp	libref	&_cstGRoot/standards/cst-framework-1.4/templates		.sasreferences	
2	CST-FRAMEWORK	1.2	control	validation	csttmp	libref	&_cstGRoot/standards/cst-framework-1.4/templates		.validation_master	
3	CST-FRAMEWORK	1.2	lookup		control	libref	&_cstGRoot/standards/cst-framework-1.4/control		.standardlookup	
4	CST-FRAMEWORK	1.2	messages		cstmsg	libref	&_cstGRoot/standards/cst-framework-1.4/messages	1	messages	
5	CST-FRAMEWORK	1.2	properties	initialize	cstprop	fileref	&_cstGRoot/standards/cst-framework-1.4/programs	1	initialize.properties	
6	CST-FRAMEWORK	1.2	results	metrics	csttmp	libref	&_cstGRoot/standards/cst-framework-1.4/templates		.metrics	
7	CST-FRAMEWORK	1.2	results	results	csttmp	libref	&_cstGRoot/standards/cst-framework-1.4/templates		.results	
8	CST-FRAMEWORK	1.2	standards	registeredsasreferences	csttmp	libref	&_cstGRoot/standards/cst-framework-1.4/templates		.sasreferences	
9	CST-FRAMEWORK	1.2	standards	registeredstandards	csttmp	libref	&_cstGRoot/standards/cst-framework-1.4/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-1.4/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).

Display 5.2 on page 72 shows the information returned by this call to %cst_getstandardsasreferences for the CDISC SDTM standard.

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

Display 5.2 Standard SASReferences for CDISC SDTM

	standard	standardversion	type	subtype	SASref	reltype	path	order	memname
1	CDISC-SDTM	3.1.2	autocall		sdmauto	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/macros	1	
2	CDISC-SDTM	3.1.2	classmetadata	column	sdmccls	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/metadata		.class_columns.sas7bdat
3	CDISC-SDTM	3.1.2	classmetadata	table	sdmccls	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/metadata		.class_tables.sas7bdat
4	CDISC-SDTM	3.1.2	lookup		sdmctl	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/control		.standardlookup.sas7bdat
5	CDISC-SDTM	3.1.2	messages		sdmmsg	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/messages	1	messages.sas7bdat
6	CDISC-SDTM	3.1.2	properties	initialize	sdmprop	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/programs	1	initialize.properties
7	CDISC-SDTM	3.1.2	properties	validation	sdmprp2	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/programs	1	validation.properties
8	CDISC-SDTM	3.1.2	referencecontrol	standardref	sdmctl	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/validation/control		.validation_stdref.sas7bdat
9	CDISC-SDTM	3.1.2	referencecontrol	validation	sdmctl	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/validation/control		.validation_master.sas7bdat
10	CDISC-SDTM	3.1.2	referencemetadata	column	sdmref	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/metadata		.reference_columns.sas7bdat
11	CDISC-SDTM	3.1.2	referencemetadata	table	sdmref	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/metadata		.reference_tables.sas7bdat

A comparison of [Display 5.1 on page 72](#) and [Display 5.2 on page 72](#) 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). The 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 in this location in SAS 9.3:

```
!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.4/sample/
cdisc-sdtm-3.1.2/sascstdemodata/control
```

This display 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		sdtmcd	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-3.1.2-1.4/messages	1	messages.sas7bdat
6	CDISC-SDTM	3.1.2	properties	initialize	inprop	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.4/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	referencecclerm		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-201003-1.4/formats	2	cterms.sas7bcat
19	CST-FRAMEWORK	1.2	messages		cstmsg	libref	&_cstGRoot/standards/cst-framework-1.4/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.
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.

Lines	Comment
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 72 , path and memname are blank. Leaving them blank tells the 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 72 .
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 the 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';
```

This table lists the parameters that are supported by the `cstutil_processsetup` macro.

Table 5.3 Parameters Supported by `cstutil_processsetup`

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

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

Note: The 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/
    1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata');
  otherwise call symput('studyRootPath',
    '!sasroot/../../SASClinicalStandardsToolkitSDTM312/
    1.4/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

Overview

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 73](#). 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 the SAS Clinical Standards Toolkit, including SASReferences. This macro validates that SASReferences has the structure and content defined by the StandardSASReferences and Standardlookup data sets.

Here 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.

Common Errors and Solutions

This list describes the most common errors detected by the `cstutil_checkds` macro. It suggests solutions as well.

- A standard-specific macro cannot be found. For example:

```
%odm_write;
WARNING: Apparent invocation of macro ODM_WRITE not resolved
```

Reported in the SAS log.

The autocall path has not been set correctly.

Common causes:

- SASReferences does not contain a `type=autocall` for that standard.
- In the same SAS session, you change the standard you are working with and fail to issue these statements first:

```
%let _cstReallocatesASRefs=1;
%include "&_cstGRoot/standards/cst-framework-1.4/
        programs/resetautocallpath.sas";
```

- Input parameters to macro insufficient for cstutil_checkds macro to run.
Reported in the Results data set.
Cause: One of the required macro variable options is missing.
- Location for Results data set is undefined.
Reported in the SAS log.
Solution: Define the Results data set in the macro variable _cstResultsDS.
- Data set could not be opened.
Reported in the Results data set.
Cause: The data set that is passed in via the _cstdsname parameter cannot be opened.
Solution: 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.
Reported in the Results data set.
Cause: The data set that is passed in via the _cstdsname parameter has a different structure than the template data set.
Solution: 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.
Reported in the Results data set.
Cause: Some columns are required to be non-null. If you receive this error, then you are also informed which column must contain a value.
Solution: Enter a non-null value for this column.
- Invalid value for column column_name, row ## in data set.
Reported in the Results data set.
Cause: 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 is in the Standardlookup data set that is registered with each data standard.
Solution: 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, these events happen:

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 the 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.

CAUTION:

SASReferences is key to the process, and any errors will cause the process to fail. 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. For tips on debugging problems with the SASReferences file, see [“Common Errors and Solutions” on page 77](#).

TIP 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	82
Metadata Requirements	84
Overview	84
Reference Metadata	85
Source Metadata	88
Validation Check Metadata: Validation Master	88
Supplemental Validation Check Metadata: Validation Standard References	95
Supplemental Validation Check Metadata: Domains by Check	97
Validation.Properties	97
Messages	99
Validation Metrics	100
Cross-Standard Validation	102
Overview	102
The cstcheck_crossstdcomparedomains Macro	102
The cstcheck_crossstdmetamismatch Macro	103
Building a Validation Process	104
Overview	104
SASReferences Customizations	104
Validation Control: Specification of Run-Time Checks	105
Setting Properties for the Validation Process	110
Running a Validation Process	111
Sample CDISC SDTM 3.1.1 Driver Program: validate_data.sas	111
Validation Results and Metrics	116
Sample CDISC CRT-DDS 1.0 Driver Program: validate_crtds_data.sas	120
Validation Checks by Standard	120
ADaM 2.1	120
CDISC ODM 1.3.0	121
CDISC SDTM 3.1.1	128
CDISC SDTM 3.1.2	130
CDISC CRT-DDS 1.0	132
Special Topic: Validation Check Macros	144
Special Topic: How the SAS Clinical Standards Toolkit	
Interprets Validation Check Metadata	150
Overview	150
Case Study 1: CDISC SDTM Check SDTM0604	151
Case Study 2: CDISC SDTM Check SDTM0623	152
Special Topic: SAS Implementation of ISO 8601	154

Overview	154
Example ISO 8601 Values	155
SAS ISO 8601 References	158
Special Topic: Debugging a Validation Process	159
Overview	159
Errors in Setting Up the SAS Clinical Standards Toolkit Environment	160
Errors in Performing Some Primary SAS Clinical Standards Toolkit Action	161
Other Debugging Tips	163
Special Topic: Validation Customization	164
Overview	164
Case Study 1: Modifying an Existing Standard or Defining a New Reference Standard	164
Case Study 2: Using Any Set of Source Data and Metadata	165
Case Study 3: Modifying the SAS Validation Checks for Supported Standards ..	165
Case Study 4: Adding New Validation Checks for Supported Standards	166
Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros	167
Case Study 6: Modifying the SAS Clinical Standards Toolkit Messaging, Including Internationalization	168
Case Study 7: Validation of Multiple Studies	169
Special Topic: Using Alternative Controlled Terminologies	170
Special Topic: Performance Considerations	174

Validation Framework Overview

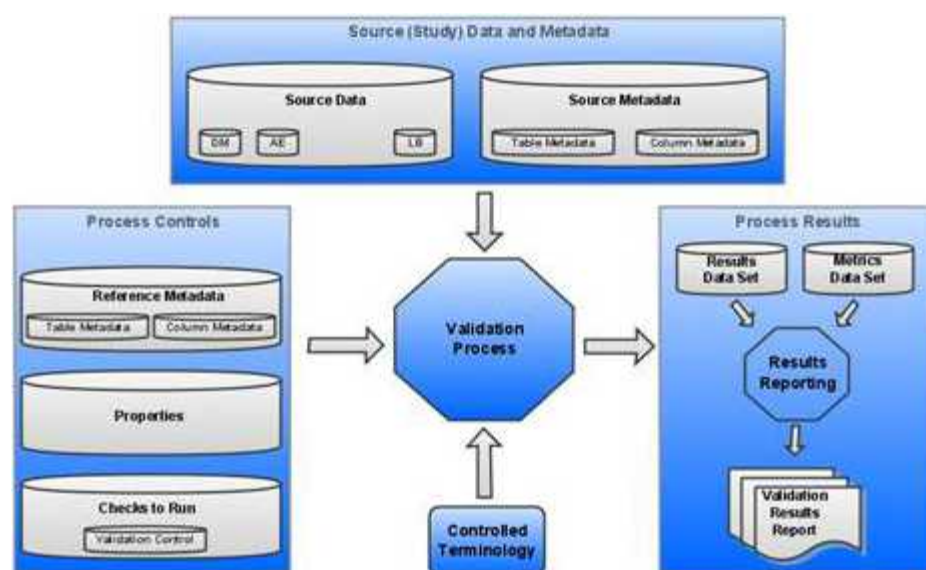
The SAS Clinical Standards Toolkit validation assesses the compliance of data, and the metadata describing the data, with an accepted reference standard. It assesses the consistency of values in a specific column, between columns, across records in a specific data set, and across data sets. The primary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

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`).

This 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:
 - *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 4.1 on page 44](#) and [Table 4.2 on page 45](#).
 - *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 Metadata: Validation Master” on page 88](#). For information about the SAS Clinical Standards Toolkit global macro variables, see [Appendix 1, “Global Macro Variables,” on page 261](#).

- *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 these basic assumptions:

1. There is some combination of source data and metadata available as SAS files that you want 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 37](#), 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.supportvalidation` 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 is in the `cstGlobalLibrary` folder hierarchy.

For example, validation files that define the CDISC SDTM 3.1.1 standard are in this folder hierarchy:

```
<global standards library directory>/standards/cdisc-  
sdm-3.1.1-1.4
```

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 23](#).

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 4.1 on page 44](#) and an example of a `reference_columns` record is provided in [Table 4.2 on page 45](#). 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, “Supported Standards,” on page 37](#), 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
<code>sasref</code>	\$8	The SAS libref that refers to the table in the SAS Clinical Standards Toolkit process. This value should match the value of the <code>SASReferences.sasref</code> field, where <code>type=referencemetadata</code> and <code>subtype=table</code> . This column is required.
<code>table</code>	\$32	The name of the domain being defined in the standard. The value must conform to SAS naming conventions. This column is required.
<code>label</code>	\$40	The label of the domain being defined in the standard. The value must conform to SAS naming conventions. This column is optional.
<code>class</code>	\$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.
<code>xmlpath</code>	\$200	The path to the SAS transport file. This path can be specified as a relative path. The value can be used when creating <code>define.xml</code> to populate the value for the <code>def:leaf xlink:href</code> link to the domain file. The value should be the pathname and filename of the SAS transport file relative to the location of <code>define.xml</code> file. This column is optional and not relevant for all standards.
<code>xmltitle</code>	\$200	The title of the SAS transport file. The value can be used when creating a <code>define.xml</code> file to populate the value for the <code>def:leaf def:title</code> value. It can provide a meaningful description, label, or location of the domain leaf (for example, <code>crt/datasets/Protocol 1234/AE.xpt</code>). This column is optional and not relevant for all standards.
<code>structure</code>	\$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.
<code>purpose</code>	\$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.

Column Name	Column Length	Description
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, such as the release date, the creation date, or the 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 6. 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.

Column Name	Column Length	Description
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.
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 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 6 . 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.

Column Name	Column Length	Description
comment	\$1000	Any character string that provides comments relevant to the column. This column is optional.

The standard reference metadata provided by SAS is in the SAS Clinical Standards Toolkit global standards library. By default, this library is here:

```
<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-
sdm-3.1.1-1.4/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, you 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. Although 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 Metadata: Validation Master

The Validation Master data set contains all validation checks defined for a standard. By default, this data set is deployed to this 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.

This 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 <i><global standards library directory>/metadata</i> . This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). You can use any naming convention limited to eight 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 6 . 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. 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 these 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.

Column Name	Column Length	Description
checktype	\$20	<p>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:</p> <p>Metadata-structural—Checks some metadata-only property (no data access required).</p> <p>ColumnValue-content: Checks a column value or compares two column values.</p> <p>Date-content: Checks ISO 8601 compliance or compares two date values.</p> <p>Multirecord-content: Looks across multiple records in a single domain.</p> <p>Multitable-content: Looks across multiple domains.</p> <p>Controlterm-content: Assesses whether column value is consistent with controlled terminology.</p> <p>This column is optional.</p>
codesource	\$32	<p>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.</p>
usesourcemetadata	\$1	<p>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.</p>
tablescope	\$200	<p>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:</p> <p><u>_ALL_</u>-DM-DS: Multiple domains that exclude one or more specific domains that are delimited with a -.</p> <p>DM: Any single domain; can be specified as libref.domain.</p> <p>DM+AE: Multiple domains delimited with a +.</p> <p><u>_ALL_</u>: Multiple DM domains that exclude specific domains delimited with a -.</p> <p>SUPP*: Wildcard to include multiple domains.</p> <p>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.)</p> <p>[<u>_ALL_</u>-DM][DM]: Bracket syntax to define sublists for comparative purposes. In this example, all non-DM domains are compared with the DM domain.</p> <p>See the Validation Master data set for a full set of values.</p> <p>This column is required.</p>

Column Name	Column Length	Description
columnscope	\$200	<p>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:</p> <p><code>_ALL_</code>: All columns (equivalent to <code>**</code> or a null value).</p> <p><code>_NA_</code>: Not applicable (that is, domain-level check).</p> <p>AGE: Any single column. This value can be specified as <code>libref.domain.column</code> or <code>domain.column</code>.</p> <p>ARM+ARMCD: Multiple columns delimited with a <code>+</code>.</p> <p><code>**BLFL-LBBLFL</code>: Multiple columns that exclude specific columns delimited with a <code>-</code>.</p> <p><code>**DTC</code>: Wildcard to include multiple columns with <code>**</code> representing the domain name.</p> <p><code>xxx**</code>: (For example, <code>AE**</code>, where <code>**</code> is a column wildcard).</p> <p><code>[**STDTC][**ENDTC]</code>: 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.</p> <p>See the Validation Master data set for a full set of values.</p> <p>This column is optional. (If null, the value is equivalent to <code>_ALL_</code>.)</p>
codelogic	\$2000	<p>Check-specific code segment that is inserted into the check macro defined in <code>codesource</code> and consistent with <code>codetype</code>. The <code>codelogic</code> 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 <code>%include</code> expands this capability. The <code>codelogic</code> 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:</p> <pre> 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 <code>filerefs</code> set outside of the SAS Clinical Standards Toolkit or in the SASReferences control data set. %sdmcheckutil_recordlookup data _cstProblems; set&_cstDSName; if <some condition>; run; </pre> <p>This column is optional.</p>

Column Name	Column Length	Description
codetype	8.	<p>This value defines whether to use code logic and what type of code logic can be used in the validation code. Values include:</p> <p>0: No code logic used.</p> <p>1: DATA step statement level. (For example, if <code>&_cstColumn < 0</code> then <code>_cstError=1.</code>)</p> <p>2: Full DATA step, PROC SQL step, or multiple steps.</p> <p>3: Calls a SAS macro or <code>%include</code> that can contain only DATA step statement level code. (For example, <code>codetype=1.</code>)</p> <p>4: Calls a SAS macro or <code>%include</code> that can contain only full DATA step or PROC SQL step code. (For example, <code>codetype=2.</code>)</p> <p>This column is required.</p>
lookuptype	\$20	<p>This value defines the type of information to use for value comparison to some standard. Values include:</p> <p>Metadata: Use the SAS Clinical Standards Toolkit metadata. Specifically, use the value of the column metadata field <code>xmlcodelist</code> to identify the codelist (rendered as a SAS format).</p> <p>Format: Use a SAS format from the SAS format search path.</p> <p>Dataset: Use a reference SAS data set (for example, <code>medDRA</code>). There are no SAS Clinical Standards Toolkit requirements for the structure and content of the reference SAS data set.</p> <p><extensible>: Other user-defined values can be used if there are explicitly referenced in user-written code.</p> <p>This column is optional.</p>
lookupsource	\$32	<p>The specific SAS format or file associated with <code>lookuptype</code>. For example:</p> <p>If <code>lookuptype</code> is metadata, then <code>lookupsource</code> should be blank. The code gets the value from the <code>source_columns.xmlcodelist</code> field.</p> <p>If <code>lookuptype</code> is format, then <code>lookupsource</code> 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 <code>source_columns.xmlcodelist</code> for the columns specified in <code>columnscope</code>. This field allows a run-time validation check against another format.</p> <p>If <code>lookuptype</code> is dataset, then <code>lookupsource</code> should be the name of a SAS data set. This value is specified as the data set name (for example, <code>meddra</code>) or <code>libref.dataset</code>. If a value is provided without a <code>libref</code>, then the SAS Clinical Standards Toolkit looks for any SASReferences <code>type=referenceceterm</code> records for the <code>sasref</code> value.</p> <p>This column is optional.</p>
standardref	\$200	<p>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.</p>
reportingcolumns	\$200	<p>This value includes columns not included in <code>columnscope</code> 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 <code>tablescope</code> field. The values of these columns can be reported in the Results data set. This column is optional.</p>

Column Name	Column Length	Description
checkstatus	8.	<p>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:</p> <p>0: (inactive, default)</p> <p>>0: (active)</p> <p>-1: (deprecated, archived)</p> <p>-2: (not implemented in this SAS Clinical Standards Toolkit release)</p> <p>This column is optional, although it is expected.</p>
reportall	\$1	<p>This value enables more concise reporting of errors. Values include:</p> <p>Y: (yes, report all records, default)</p> <p>N: (no)</p> <p>This column is required although not all check macro modules support abbreviated (N) reporting.</p>
uniqueid	\$48	<p>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.</p> <p>Legend:</p> <p>characters 1-8: checkid</p> <p>characters 9-10: checkid repeat indicator (00 unless multiple invocations of checkid are included)</p> <p>characters 11-16: the version of the SAS Clinical Standards Toolkit where the check metadata was last materially modified</p> <p>characters 17-23: standard version</p> <p>characters 24-42: implementation datetime of the last metadata update</p> <p>characters 43-48: assigning authority</p> <p>This column is optional, although it is expected.</p>
comment	\$200	<p>Any character string that provides comments relevant to the check. This column is optional.</p>

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

This 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.

Column Name	Column Value	Comment
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.
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.
code logic	<pre>_vnum=strip(put(&_ cstColumn,best.));_ dot=indexc(_vnum,"."); if _dot then if length(substr(_vnum,_dot+1))>3 then _cstError=1;</pre>	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	SDTM020700CST120SDTM3112 009-05-13T15:57:59CST	
codetype	1	This code logic is used in the DATA step.
comment		

The Validation Master data set contains all validation checks for a standard, whereas 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 the SAS Clinical Standards Toolkit Interprets Validation Check Metadata”](#) on page 150.

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 you about the origin of the check. It also provides any supporting documentation about the check. By default, this data set is deployed to this 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 89 .)
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.

This table describes information about a specific check in the Validation_StdRef data set (record 1) for the CDISC SDTM 3.1.2 standard.

Table 6.6 Sample CDISC SDTM 3.1.2 Validation_StdRef Data Set for Check SDTM0207 — Record 1

Column Name	Column Value	Comment
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	<i>SDTM 3.1.2 Implementation Guide</i>	This reference information originated from the <i>SDTM 3.1.2 Implementation Guide</i> .
sourcelocation	5.3.2, page 72	Section 5.3.2, page 72 of the <i>SDTM 3.1.2 Implementation Guide</i> .
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 <i>SDTM 3.1.2 Implementation Guide</i> .

This table describes information about a specific check in the Validation_StdRef data set (record 2) for the CDISC SDTM 3.1.2 standard.

Table 6.7 Sample CDISC SDTM 3.1.2 Validation_StdRef Data Set for Check SDTM0207 — Record 2

Column Name	Column Value	Comment
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 the SAS Clinical Standards Toolkit Interprets Validation Check Metadata” on page 150](#). 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 9, “Reporting,” on page 245](#). 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 6.8 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 89 .)
table	\$32	This value captures the domain or table name. This column is required.
standardversion	\$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 will use, the metrics global macro variables.

This table describes the properties in the Validation.Properties file.

Table 6.9 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 <code>_DATA_</code> is used, then the data set order is assumed. Or, <code>_cstCheckSortOrder</code> can be set to sort the Validation Control data set at run time by any fields in that data set (for example, <code>CHECKSOURCE CHECKID</code>).
_cstMetrics	This property determines whether to calculate and report metrics. An example value is <code>1=Yes</code> .
_cstMetricsDS	This property sets the SAS data set name to use to accumulate metrics during the process. The default value is <code>work._cstmetrics</code> .
_cstMetricsNumSubj _cstMetricsCntNumSubj	This property determines whether to calculate and report subject-level counts. An example value is <code>1=Yes</code> , initialize <code>_cstMetricsCntNumSubj</code> 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 <code>1=Yes</code> , initialize <code>cstMetricsCntNumRecs</code> to 0.
_cstMetricsNumChecks _cstMetricsCntNumChecks	This property determines whether to summarize and report the number of checks run. An example value is <code>1=Yes</code> , initialize <code>cstMetricsCntNumChecks</code> to 0.
_cstMetricsNumBadChecks _cstMetricsCntNumBadChecks	This property determines whether to summarize and report the number of check invocations that failed. An example is <code>1=Yes</code> , initialize <code>cstMetricsCntNumBadChecks</code> to 0.
_cstMetricsNumErrors _cstMetricsCntNumErrors	This property determines whether to summarize and report the total number of errors (<code>resultseverity=Error</code>) found. An example is <code>1=Yes</code> , initialize <code>cstMetricsCntNumErrors</code> to 0.
_cstMetricsNumWarnings _cstMetricsCntNumWarnings	This property determines whether to summarize and report the total number of warnings (<code>resultseverity=Warning</code>) found. An example is <code>1=Yes</code> , initialize <code>cstMetricsCntNumWarnings</code> to 0.
_cstMetricsNumNotes _cstMetricsCntNumNotes	This property determines whether to summarize and report the total number of notes (<code>resultseverity=Note</code>) found. An example value is <code>1=Yes</code> , initialize <code>cstMetricsCntNumNotes</code> to 0.
_cstMetricsNumStructural _cstMetricsCntNumStructural	This property determines whether to summarize and report the total number of structural (metadata) errors found. An example value is <code>1=Yes</code> , initialize <code>cstMetricsCntNumStructural</code> to 0.

Property Name	Description
<code>_cstMetricsNumContent</code> <code>_cstMetricsCntNumContent</code>	This property determines whether to summarize and report the total number of content (data) errors found. An example value is 1=Yes, initialize <code>cstMetricsCntNumContent</code> to 0.
<code>_cstMetricsTimer</code>	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` is here:

```
<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 is in a `!sasroot` subdirectory similar to `/sample/cdisc-sdtm-3.1.1-1.4/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 this 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 23.

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 these appendixes:

- [Appendix 2, “Framework Messages,”](#) on page 275
- [Appendix 4, “CDISC SDTM Validation Checks,”](#) on page 351
- [Appendix 5, “CDISC CRT-DDS 1.0 Validation Checks,”](#) on page 405
- [Appendix 6, “CDISC ODM 1.3.0 Validation Checks,”](#) on page 415

Validation Metrics

Generating the 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.

This 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 Validation 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 6.9 on page 98](#).

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

Table 6.10 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 <i><global standards library directory>/metadata</i> . This prefix is followed by a 4-digit numeric that is unique within the standard (for example, SDTM1234). You can use any naming convention limited to eight 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.

This 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

VIEWTABLE: Results.Validation_metrics					
	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 6.10 on page 100](#).

Cross-Standard Validation

Overview

The implementation of the ADaM 2.1 standard in the SAS Clinical Standards Toolkit 1.4 introduces six cross-standard validation checks that require metadata checks and data checks between two different standards. These two standards are ADaM 2.1 and SDTM 3.1.2.

Earlier releases of the SAS Clinical Standards Toolkit did not need to address cross-standard validation. Therefore, macros to handle cross validation were not developed. With the release of the SAS Clinical Standards Toolkit 1.4, two macros were developed: `cstcheck_crossstdcomparedomains.sas` and `cstcheck_crossstdmetamismatch.sas`. These macros are located in `!sasroot/cstframework/sasmacro`.

The `cstcheck_crossstdcomparedomains` Macro

The `cstcheck_crossstdcomparedomains` macro compares values for one or more columns in one table with those same columns in another domain in another standard. Or, it compares the values against metadata from the comparison standard. The macro requires use of `_cstCodeLogic` as a full DATA step or PROC SQL invocation. This DATA or 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.

Here are example validation checks that use the `cstcheck_crossstdcomparedomains` macro:

- ADaM subject not found in the SDTM DM domain
- ADaM SDTM domain reference (for traceability), but the SDTM domain is unknown

An ADaM 2.1 validation check that uses this macro is ADAM0053. Here is the rule description for this check, taken from the CDISC ADaM Validation document:

```
Invalid STUDYID/USUBJID combination not found in the SDTM Demographics domain.
```

Here is the message text for this check:

```
The values of USUBJID are not present in SDTM.DM
```

Here is sample code from the `codeLogic` field from the ADaM 2.1 Validation Master data set for validation check ADAM0053. In this example, `&_cstSQLColList` and `&_cstCrossDataLib` are generated by the macro prior to execution of `codeLogic`.

```
proc sql noprint;
create table work._cstproblems as
  select &_cstSQLColList from &_cstDSName
  except select &_cstSQLColList from &_cstCrossDataLib..dm;
quit;
```


The cstcheck_crossstdmetamismatch Macro

The `cstcheck_crossstdmetamismatch` macro identifies inconsistencies in metadata across registered standards. The macro requires use of `_cstCodeLogic` as a full DATA step or PROC SQL invocation. This DATA step or 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:

1. No data content is accessed for this check.
2. Both study and reference metadata are available to assess compliance.
3. The `_cstProblems` macro includes at least two columns. The mnemonics are from the global standards library data set:
 - `&_cstStMnemonic._value` (for example, `ADAM_value` containing the value of the column of interest from the primary standard)
 - `&_cstCrMnemonic._value` (for example, `SDTM_value` containing the value of the column of interest from the comparison standard)

Required global macro variables:

- `_cstcrossstd`: The name of the comparison standard. It is also used as a parameter to initialize `_cstCrMnemonic`.
- `_cstcrossstdver`: The version of the comparison standard.
- `_cstrunstd`: The primary standard. It is also used as a parameter to initialize `_cstStMnemonic`.
- `_cstrunstdver`: The version of the primary standard.

An ADaM 2.1 validation check that uses this macro is ADAM0002. Here is the rule description for this check, taken from the CDISC ADaM Validation document:

Any ADaM variable whose name is the same as an SDTM variable must be a copy of the SDTM variable, and its label and values must not be modified.

Here is the message text for this check:

```
A variable is present in ADaM with the same name as a variable present in
SDTM but the variables do not have identical labels
```

Here is sample code from the `codeLogic` field from the ADaM 2.1 Validation Master data set for validation check ADAM0002. In this example, `&_cstStMnemonic=ADAM` and `&_cstCrMnemonic=SDTM` are generated by the macro prior to execution of `codeLogic`.

```
%let _cstAttr=label;
proc sql noprint;
create table work._cstProblems as
  select &_cstStMnemonic..table, &_cstStMnemonic..column,
         &_cstStMnemonic..&_cstAttr as &_cstStMnemonic._value,
         &_cstCrMnemonic..&_cstAttr as &_cstCrMnemonic._value
from work._cstcolumnmetadata &_cstStMnemonic left join
     work._cstcrosscolumnmetadata &_cstCrMnemonic on
     upcase(&_cstStMnemonic..column)=upcase(&_cstCrMnemonic..column)
where &_cstCrMnemonic..column ne "" and
     (&_cstStMnemonic..&_cstAttr ne &_cstCrMnemonic..&_cstAttr);
quit;
```

Building a Validation Process

Overview

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. These 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.

The SAS Clinical Standards Toolkit validation can include references to these 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 97](#). For information about the derived global macro variables, see [Appendix 1, “Global Macro Variables,” on page 261](#). The Validation.Properties file is a required file to support the SAS Clinical Standards Toolkit validation.

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 100](#) and [“Validation Results and Metrics” on page 116](#). For information about the global macro variables that govern metrics output, see [Appendix 1, “Global Macro Variables,” on page 261](#). 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 the Location of Controlled Terminology in the SASReferences Data Set

standard	standardversion	type	subtype	SASref	reltype	path	order	memname
CDISC-SDTM	3.1.1	fmtsearch		sctfmt	libref	&studyRootPath\terminology\formats	1	formats.sas7bcat
CDISC-TERMINOLOGY	200810	fmtsearch		ctfmt	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 **codeologic** 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	reltype	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 Metadata: Validation Master” on page 88](#).) 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 this SAS 9.3 directory:

```
!sasroot/../../SASClinicalStandardsToolkitSDTM311/1.4/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. (See [Display 6.7 on page 105](#).)

The &studyRootPath value is assumed to have been set to !sasroot/../../SASClinicalStandardsToolkitSDTM311/1.4/sample/cdisc-sdtm-3.1.1/sascstdemodata.

The Validation Master data set (illustrated in [Display 6.3 on page 104](#) and in this display) serves as the source for Validation Control content. Note that in this display, the

path and **memname** information have been derived from the StandardSASReferences data set and points to the Global Standards Library.

Display 6.8 Defining Validation Control Data Set Location

type	subtype	SASref	reftype	path	order	memname
referencecontrol	validation	refcntl	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.1-1.4/validation/control		validation_master.sas7bdat

This 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 6.11 Sample Code to Create Validation Control Data Set

Check Subset	Sample Code
All checks provided with the SAS Clinical Standards Toolkit.	<pre>data control.validation_control; set refcntl.validation_master; run;</pre>
Structural checks (metadata-only checks that do not require access to the domain data).	<pre>data control.validation_control; set refcntl.validation_master (where=(upcase(checktype)="METADATA"));run;</pre>
Content checks (checks that require access to the domain data).	<pre>data control.validation_control; set refcntl.validation_master (where=(upcase(checktype) ne "METADATA")); run;</pre>
Checks with a production status.	<pre>data control.validation_control; set refcntl.validation_master (where=(checkstatus>0)); run;</pre>
WebSDM checks (CDISC SDTM only).	<pre>data control.validation_control; set refcntl.validation_master (where=(upcase(checksource)= "WEBSDM")); run;</pre>

Check Subset	Sample Code
Sampling of checks, one for each check macro.	<pre> 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; </pre>
CDISC SDTM 3.1.1 checks.	<pre> data control.validation_control; set refcntl.validation_master (where=(standardVersion = "3.1.1" or standardVersion = "****")); run; </pre>
All codelist-related checks (checks that use the cstcheck_notincodelist macro).	<pre> data control.validation_control; set refcntl.validation_master (where=(upcase(checksource) = "CSTCHECK_ NOTINCODELIST")); run; </pre>

Check Subset	Sample Code
All checks applicable to a specific domain.	<pre> %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); </pre>

Check Subset	Sample Code
--------------	-------------

```

stop;
run;
%if &_amp;cstCheckID=&_amp;cstOldCheckID %then %do;
%let _amp;cstCheckSeqCount=%eval(&_amp;cstCheckSeqCount+1) ;
%end;
%else %let _amp;cstCheckSeqCount=1;

/* Call macro to interpret tableScope and columnScope to build
work._amp;cstcolumnmetadata for each check */
/* _amp;cstDomSubOverride=Y parameter allows us to also look at check
records with unequal sublist lengths */
%cstutil_buildcollist(_amp;cstFormatType=DATASET,_
cstDomSubOverride=Y);
proc sql noprint;
create table work._amp;csttempds as
select distinct table, "&_amp;cstCheckID" as checkid length=8,
&_amp;cstCheckSeqCount as resultseq,
"&_amp;cstStandardVersion" as standardversion length=20,
"&_amp;cstChecksSource" as checksSource length=40
from work._amp;cstcolumnmetadata;
quit;

proc append base=&_amp;cstOutputDSdata=work._amp;csttempds force;
run;
%let _amp;cstOldCheckID=&_amp;cstCheckID;

* Clear contents for next loop, in case of problems *;
data work._amp;csttempds;
set work._amp;csttempds;
if _amp;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 ;

/* Note: Be aware that the code returns all records for any
checkid even if only one matches a specified domain. */

%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.

TIP Best Practice Recommendation: You 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 1, “Global Macro Variables,” on page 261](#).) 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 provides a space-delimited list of the columns that identify a subject.
- `_cstReallocateSASRefs`, which reallocates SAS librefs and filerefs in the same SAS session, which is important 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).

TIP 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`

Overview

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 the SAS Clinical Standards Toolkit validation. For example, in a SAS 9.3 deployment, the CDISC SDTM 3.1.2 validation driver module is in: `!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata/programs/validate_data.sas`

Step 1: Define the locations of the study data and the metadata.

`/* 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
      /1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata');
    call symput('studyOutputPath','!sasroot/../../SASClinicalStandardsToolkitSDTM312
      /1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata');
  end;
otherwise do;
  call symput('studyRootPath','!sasroot/../../SASClinicalStandardsToolkitSDTM312
    /1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata');
  call symput('studyOutputPath','!sasroot/../../SASClinicalStandardsToolkitSDTM312
    /1.4/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 *;
****let studyOutputPath=users/myname/mystudy; *<--- example user override *;
****cstutil_createsubdir(_cstSubDir=results); *<--- example user 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 hierarchy. 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_createsubdir` 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 the global macro variables.

```
* 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: Set the version of the controlled terminology.

The following macro selects the version of the controlled terminology required by the validation process. In the following example, the standard is CDISC-Terminology. The %cst_getstandardsubtypes macro builds a work data set named work._cstStdSubTypes, which contains all records for the CDISC-Terminology standard. A DATA_NULL_step subsets the values to a specific &_cstStandard (in this case, CDISC-SDTM). The value of the variable isstandarddefault is equal to Y. Values from this single observation are used to create the macro values for &_cstCTRoot and &_cstCTDescription. You can override the version of the controlled terminology by setting &_cstCTRoot to any directory. (This is shown in the commented line.)

```
* Set Controlled Terminology version for this process *;
%cst_getstandardsubtypes(_cstStandard=CDISC-TERMINOLOGY, _cstOutputDS=work._cstStdSubTypes);
data _null_;
set work._cstStdSubTypes (where=(standardversion="&_cstStandard" and isstandarddefault='Y'));
call symputx('_cstCTRoot', cats(lowercase(standardversion), '/',
lowercase(standardsubtypeversion)));
call symputx('_cstCTDescription', description);
run;
***let _cstCTRoot=cdisc-sdtm/201003; * <----- User can override
CT version of interest *;
```

The &_cstCTRoot macro variable is used to set the controlled terminology directory in the work.sasreferences data set in the next step.

Step 4: Create an empty work.sasreferences data set to be populated in the validation process, and build the work.sasreferences data set.

Create work.sasreferences.

```
%cst_createds(_cstStandard=CST-FRAMEWORK, _cstType=control,
_cstSubType=reference, _cstOutputDS=work.sasreferences);
```

The validate_data.sas module initializes the SASReferences data set that is required for SDTM validation. The SASReferences 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 program, sets the path of the Validation Control data set to &studyRootPath/control and name to validation_control.sas7bdat. In SAS 9.3, this translates to !sasroot/../../SASClinicalStandardsToolkitSDTM312/1.4/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 67](#). For a fully initialized SASReferences data set for SDTM validation, see [Display 5.3 on page 73](#).

Step 5: 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 these 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. These 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 15.
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.

CAUTION:

The SASReferences data set is key to the process, and any errors will cause the process to fail. This is a common process failure point because of the importance of the SASReferences data set. For tips on debugging problems with the SASReferences data set, see [“Special Topic: Debugging a Validation Process” on page 159](#).

Step 6: Run validation tasks.

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

The `%sdm_validate` macro performs these 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, these events happen:
 - 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 159](#).

Step 7: 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);
```

This step 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.

Parameter Details

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

Table 6.12 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.
_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=(lowercase(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.

[Display 6.9 on page 117](#) summarizes a sample validation process. Here are a few facts about the sample validation process:

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 these displays, some rows have been hidden to reduce redundant examples.

Display 6.9 Example of a 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 this display, some rows have been hidden to reduce redundant examples.

Display 6.10 Example of a 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=SASCSTDDEMODATA,USUBJID=S002P034,VISITNUM=1,SUTRT=CIGARETTES	
89	CST0100	SDTM0851	1	1			
90	CST0100	SDTM0851	2	1			

Table 6.13 Comments about the Validation Results Data Sets in Displays 6.9 and 6.10

Lines	Comment
1,4,5	Informational notes about processing the properties files.
2	Informational note saying that the creation of work.sasreferences was successful.

Lines	Comment
3	Informational note from cstutil_processsetup that informs you 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 you 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.
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 this display, some rows have been hidden to reduce redundant examples.

Display 6.11 Example of a 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_RECMMISMATCH	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 6.14 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.
43-44	Check SDTM0851 evaluated 28 records in the SRCDATA.CO domain. The check took one second to run using cstcheck_recmmismatch.
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).

Lines	Comment
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 117](#) 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](#).

Here 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 9, “Reporting,” on page 245](#).

Sample CDISC CRT-DDS 1.0 Driver Program:

`validate_crtds_data.sas`

The 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_crtds_data.sas`, is provided to perform process setup steps and to call the `crtds_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 175](#). In this chapter, the use of the `validate_crtds_data` driver module is described.

Validation Checks by Standard

ADaM 2.1

The SAS Clinical Standards Toolkit 1.4 provides 159 unique ADaM 2.1 validation checks. These validation checks are derived from the SAS interpretation of the CDISC ADaM Validation Checks Version 1.0 (final production version dated September 20, 2010), and the CDISC ADaM Validation Checks Version 1.1 Maintenance Release (dated and released January 21, 2011 to correct errors and remove duplicate checks). These documents are available in the members only area at <http://www.cdisc.org/adam>.

Information about the 159 records in the CDISC ADaM 2.1 Validation Master data set is in [Appendix 7, “CDISC ADaM 2.1 Validation Checks,” on page 433](#). Only selected columns are listed in the appendix.

Consider the interrelationships among the 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 99](#).

By default, the Validation Master data set is located in the `<global standards library directory>/standards/cdisc-adam-2.1-1.4/validation/control` folder.

CDISC ODM 1.3.0

The SAS Clinical Standards Toolkit provides check macros that validate the data in the SAS data sets representing CDISC ODM 1.3.0 data. The structure of this data is similar to CDISC CRT-DDS. Therefore, the process for validating the data is similar. 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.

As in CRT-DDS, the validity of ODM 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 formats.

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 SAS Clinical Standards Toolkit 1.4 provides 179 unique ODM 1.3.0 validation checks. These validation checks were developed by SAS and are based on ODM implementation experience and careful review of the *CDISC ODM Implementation Guide*, with special emphasis on the occurrence of “should” within the Implementation Guide.

Information about the 179 records in the CDISC ODM 1.3.0 Validation Master data set is in [Appendix 6, “CDISC ODM 1.3.0 Validation Checks,” on page 415](#). Only selected columns are listed in the appendix.

Consider the interrelationships among the 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 99](#).

By default, the Validation Master data set is located in the `<global standards library directory>/standards/cdisc-odm-1.3.0-1.4/validation/control` folder.

Table 6.15 on page 122 lists the types of checks for ODM 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 a 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 and trailing white space.

Table 6.15 ODM Validation Check Types

Check Type	Check ID	Category	Description
Unique in data set	ODM0100	Structural	No two values for the source column can be equivalent within the same source data set.
	ODM0211	Structural	Duplicate OrderNumber element. The OrderNumber attribute must be unique within the same source data set when not null.
Required character value	ODM0101	Data	The trimmed (white space removed) value of the character data must consist of one or more characters.
Required numeric value	ODM0101	Data	The numeric value of the column cannot be missing.
Enumeration(s0,s1,...)	ODM0114	Data	If character data exists, its value must match one of the given enumerated character strings. All string comparisons are case sensitive.
Foreign key(targetColumn)	ODM0110	Structural	Each existing value in this column must have an equivalent value in the given target column.
Foreign key required(targetColumn)	(1)	Structural	A value is required for this column in every row and each value must have an equivalent value in the given target column. This check is the equivalent of running the required character value check, and failing if that check fails. If required character value passes, the foreign key() check is run.
Character format: language	ODM0106	Data	The character data must consist of 1-8 alphabetical characters of either case, followed optionally by a hyphen character and any sequence of 1-8 alphabetical characters of either case or numeric after that hyphendigits. For example, e is a legal value, as are en-us and english and english-d842 . Illegal values include 1en , mumblespeak , and en_us . The hyphen character sequence can be repeated any number of times also 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})*”.

Check Type	Check ID	Category	Description
Character format: fileName	ODM0107	Data	The character data must not contain any characters other than upper- and lower-case letters of the alphabet, numeric digits, the underscore (_) character, or a period. Regular expression: [A-Za-z0-9_.] ⁺ .
Character format: sasName	ODM0108	Data	The first character must be either a lower- or upper-case letter or an underscore (_). Any subsequent character must be either an upper- or lowercase letter, a numeric digit, or the underscore (_). Regular expression: [A-Za-z_][A-Za-z0-9_]*.
Character format: sasFormat	ODM0109	Data	The first character must be either a lower- or upper-case letter, an underscore (_), or the dollar sign (\$). Any subsequent character must be either an upper- or lowercase letter, a numeric digit, the underscore (_), or a period. Regular expression: [A-Za-z_ \$][A-Za-z0-9_.] [*] .
Must Have Corresponding Value(targetColumn)	ODM0111	Structural	For each distinct value in this column, there must be at least one equivalent value in the supplied target column.
Unique across data sets(targetcolumn0,...)	ODM0112	Structural	No value in this column can be equal to any value in any of the given data set columns.
Primary key	(2)	Data	Must satisfy the Unique in data set check type and the required character value check type.

Check Type	Check ID	Category	Description
Invalid Value	ODM0200	Data	Documents based on ODM 1.3 should have ODM version set to 1.3.
	ODM0217	Data	An invalid SAS format name. In case the data type is character, the format name needs to start with the \$ character.
	ODM0219	Data	An invalid integer value. The attribute is defined as an integer, but the text string does not match the named data format. The allowed string pattern for an integer is: -?digit+.
	ODM0221	Data	An invalid float value. The attribute is defined a float, but the text string does not match the named data format. The allowed string pattern for a float is: -?digit+(.digit+)?.
	ODM0222	Data	An invalid date value. The attribute is defined as a date, but the text string does not match the named data format. The allowed string pattern for a date is: YYYY-MM-DD.
	ODM0223	Data	An invalid time value. The attribute is defined a time, but the text string does not match the named data format. The allowed string pattern for a time is: hh:mm:ss(.n+)?((+ -)hh:mm)?.
	ODM0224	Data	An invalid datetime value. The attribute is defined as a datetime, but the text string does not match the named data format. The allowed string pattern for a datetime is: YYYY-MMM-DD T hh:mm:ss(.n+)?((+ -)hh:mm)?.
External File Reference Found	ODM0201	Data	External file reference found because the prior file OID is not missing (for example, ODM.PriorFileOID ne '')
Referenced OID Not Found	ODM0202	Data	If Metadata version IncludedOID is non-null, the referenced OID must be found in this XML file.
	ODM0203	Data	If Metadata version IncludedStudyOID is non-null, the referenced OID must be found in this XML file.

Check Type	Check ID	Category	Description
Attribute is Required	ODM0216	Column	The ItemDef length attribute is required when data type is text, string, integer, or float and can be ignored for the other types.
	ODM0220	Column	The required attribute SignificantDigits cannot be empty or missing when Data type is Float.
	ODM0228	Column	Only numeric (integer or float) items should have measurement units. The MeasurementUnitRefs list the acceptable measurement units for this type of item. If only one MeasurementUnitRef is present, all items of this type carry this measurement unit by default. If no MeasurementUnitRef is present, the item's value is scalar (for example., a pure number).
Data Set Does Not Exist	ODM0218	Metadata	Invalid root element. The ODM file must contain a root element called ODM. In other words, the ODM data set must exist.
Mixed Data Exists	ODM0226	Multirecord	Typed and Untyped data transmission should not be mixed within a single ODM file.
Multiple Records Exists	ODM0227	Column	To avoid ambiguity, a particular language tag should not occur more than once in a series of TranslatedText elements

(1) This validation is a combination of checks ODM0101 and ODM0110.

(2) This validation is a combination of checks ODM0100 and ODM0101.

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. An example is foreign key enforcement. Structural conditions must be met for the successful generation of an ODM XML file. You might want to defer structural checks until later in the process when populating the ODM data sets. This is because foreign key relationships require that the data is made available in a particular order (that is, a referenced key must be available before the foreign key to it can exist).

The CDISC ODM validation checks that are listed in [Table 6.16 on page 126](#) are performed by comparing the data against a set of expected values. The expected values are stored in a format catalog (odmct.sas7bcat) and a data set (odmct.sas7bdat). They are located in the **<global standards library directory>/standards/cdisc-odm-1.3.0-1.4/formats** folder.

The methodology ensures case-sensitivity compliance, which is required by the XML schema validation. For example, ItemRangeChecks has a Comparator variable whose values are controlled by the Comp Enumeration ("EQ", "GE", "GT", "IN", "LE", "LT", "NE", and "NOTIN"). If mixed case or lowercase values are detected, then this validation check reports an error. The validation check is ODM0114 (see [Table 6.15 on page 122](#)), and it uses the Comp SAS format to report this as an error.

The SASReferences data set needs to contain a row for fmtsearch, with SAS libref set to **odmfmt** and the Filename set to **odmct.sas7bcat**.

Table 6.16 ODM Validation Checks Using Expected Values

CheckID	Table	Variable Checked	XML Codelist	Check
ODM0114	Annotation	TransactionType	TType	Enumeration ("Context", "Insert", "Remove", "Update", "Upsert")
ODM0114	Annotation	CommentSponsorOrSite	SPSite	Enumeration ("Site", "Sponsor")
ODM0114	Annotation	ParentType	PType	Enumeration ("Annotations", "Association", "AuditRecords", "FormData", "ItemData", "ItemGroupData", "Signatures", "StudyEventData", "SubjectData")
ODM0114	AuditRecord	EditPoint	EditPT	Enumeration ("DBAudit", "DataManagement", "Monitoring")
ODM0114	AuditRecord	UsedImputationMethod	NY	Enumeration ("No", "Yes")
ODM0114	AuditRecord	ParentType	PType	Enumeration ("Annotations", "Association", "AuditRecords", "FormData", "ItemData", "ItemGroupData", "Signatures", "StudyEventData", "SubjectData")
ODM0114	CodeLists	DataType	CLType	Enumeration ("float", "integer", "string", "text")
ODM0114	FormData	TransactionType	TType	Enumeration ("Context", "Insert", "Remove", "Update", "Upsert")
ODM0114	FormDefItemGroupRefs	Mandatory	NY	Enumeration ("No", "Yes")
ODM0114	FormDefs	Repeating	NY	Enumeration ("No", "Yes")
ODM0114	ItemData	TransactionType	TType	Enumeration ("Context", "Insert", "Remove", "Update", "Upsert")
ODM0114	ItemData	IsNull	NY	Enumeration ("No", "Yes")

CheckID	Table	Variable Checked	XML Codelist	Check
ODM0114	ItemData	ItemDataType	DType	Enumeration ("Any", "Base64Binary", "Base64Float", "Boolean", "Date", "Datetime", "DurationDatetime", "Float", "HexBinary", "HexFloat", "IncompleteDatetime", "Integer", "IntervalDatetime", "PartialDate", "PartialTime", "String", "Time", "URI")
ODM0114	ItemDefs	DataType	IDType	Enumeration ("URI", "base64Binary", "base64Float", "boolean", "date", "datetime", "double", "durationDatetime", "float", "hexBinary", "hexFloat", "incompleteDatetime", "integer", "intervalDatetime", "partialDate", "partialTime", "string", "text", "time")
ODM0114	ItemGroupData	TransactionType	TType	Enumeration ("Context", "Insert", "Remove", "Update", "Upsert")
ODM0114	ItemGroupDefItemRefs	Mandatory	NY	Enumeration ("No", "Yes")
ODM0114	ItemGroupDefs	Repeating	NY	Enumeration ("No", "Yes")
ODM0114	ItemGroupDefs	IsReferenceData	NY	Enumeration ("No", "Yes")
ODM0114	ItemRangeChecks	Comparator	Comp	Enumeration ("EQ", "GE", "GT", "IN", "LE", "LT", "NE", "NOTIN")
ODM0114	ItemRangeChecks	SoftHard	Soft	Enumeration ("Hard", "Soft")
ODM0114	Location	LocationType	LocType	Enumeration ("CRO", "Lab", "Other", "Site", "Sponsor")
ODM0114	MethodDefs	Type	MType	Enumeration ("Computation", "Imputation", "Other", "Transpose")
ODM0114	ODM	Archival	YesNo	Enumeration ("Yes")
ODM0114	ODM	FileType	FileType	Enumeration ("Snapshot", "Transactional")

CheckID	Table	Variable Checked	XML Codelist	Check
ODM0114	ODM	Granularity	Gran	Enumeration ("AdminData", "All", "AllClinicalData", "Metadata", "ReferenceData", "SingleSite", "SingleSubject")
ODM0114	Signature	ParentType	PType	Enumeration ("Annotations", "Association", "AuditRecords", "FormData", "ItemData", "ItemGroupData", "Signatures", "StudyEventData", "SubjectData")
ODM0114	SignatureDef	Methodology	SMethod	Enumeration ("Digital", "Electronic")
ODM0114	StudyEventData	TransactionType	TType	Enumeration ("Context", "Insert", "Remove", "Update", "Upsert")
ODM0114	StudyEventDefs	Repeating	NY	Enumeration ("No", "Yes")
ODM0114	StudyEventDefs	Type	Type	Enumeration ("Common", "Scheduled", "Unscheduled")
ODM0114	StudyEventForm Refs	Mandatory	NY	Enumeration ("No", "Yes")
ODM0114	SubjectData	TransactionType	TType	Enumeration ("Context", "Insert", "Remove", "Update", "Upsert")
ODM0114	User	UserType	UType	Enumeration ("Investigator", "Lab", "Other", "Sponsor")
ODM0114	UserAddress	Country	C3166ISO	Controlled by the two-letter codes from the ISO 3166-1 standard for country codes. The values are available from this Web site: http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm

CDISC SDTM 3.1.1

The SAS Clinical Standards Toolkit 1.4 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/ValidationChecks 2.6/WebSDM V2.6 Validation Checks FINAL.pdf](http://phaseforward.com/resource/whitepapers/ValidationChecks2.6/WebSDM%20V2.6%20Validation%20Checks%20FINAL.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 1.4, 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 your 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, the SAS Clinical Standards Toolkit attempts to run the check as defined, regardless of the value of the checkstatus column.

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

Table 6.17 Distribution of CDISC SDTM 3.1.1 Validation Checks

Check Source	Count	Deprecated	Inactive
WebSDM	114	5	1
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 1.4 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 4, “CDISC SDTM Validation Checks,” on page 351](#). 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 93](#).

Consider the interrelationships among the 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 99](#).

By default, the Validation_StdRef data set is found in the `<global standards library directory>/standards/cdisc-sdtm-3.1.1-1.4/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 1.4 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

The SAS Clinical Standards Toolkit 1.4 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 1.4, 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

the SAS Clinical Standards Toolkit. In the SAS Clinical Standards Toolkit 1.4, the Janus and JanusFR checks were dropped for SDTM 3.1.2.

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

Table 6.18 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 4, “CDISC SDTM Validation Checks,” on page 351](#). Only selected columns are listed in the appendix.

Consider the interrelationships among the 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 99](#).

By default, the Validation_StdRef data set is found in the `<global standards library directory>/standards/cdisc-sdtm-3.1.2-1.4/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 1.4 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

[Table 6.19 on page 132](#) 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 6.19 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.
Foreign key(targetColumn)	CRT0110	Structural	Each existing value in this column must have an equivalent value in the target column.

Check Type	Check ID	Category	Description
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. You 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 6.20 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	ItemQuestionTranslated Text	CRT0106	lang	Character format: language
0106	ItemQuestionTranslated Text	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	RCErrorsTranslatedText	CRT0106	lang	Character format: language
0120	RCErrorsTranslatedText	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 6.20 on page 134](#) are performed by comparing the data against a set of expected values. The expected values have been stored in a format catalog (crtdsct.sas7bcat) and a data set (crtdsct.sas7bdat). They are in the **<global standards library directory>/standards/cdisc-crtdds-1.0-1.4/formats** folder.

This 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 6.20 on page 134](#)) and it uses the Comp format to report this as an error.

Table 6.21 Enumeration Validation Format Values*

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

Format	Value 1	Value 2
CLType	integer	integer
	float	float
	text	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 **crtdsdct.sas7bcat**.

Display 6.12 Example SASReferences File

	standard	standardversion	type	subtype	SASref	reftype	path	order	memname
1	CDISC-CRTDDS	1.0	autocall		auto1	libref	&_cstGRoot/standards/cdisc-crtdds-1.0-1.4/macros	1	
2	CDISC-CRTDDS	1.0	control	reference	crtf_s	libref	C:\DOCUME~1\gelgh\LOCALS~1\Temp\SAS Temporary Files\TD2732_WILLIS2_		.sasreferences
3	CDISC-CRTDDS	1.0	control	validation	crtf_v	libref	tsasroot/././SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/control		.validation_control
4	CDISC-CRTDDS	1.0	fmtsearch		crtffmt	libref	&_cstGRoot/standards/cdisc-crtdds-1.0-1.4/formats	1	crtdsdct.sas7bcat
5	CST-FRAMEWORK	1.2	messages		messages	libref	&_cstGRoot/standards/cst-framework-1.4/messages	1	messages
6	CDISC-CRTDDS	1.0	messages		crtmsg	libref	&_cstGRoot/standards/cdisc-crtdds-1.0-1.4/messages	2	messages.sas7bdat
7	CDISC-CRTDDS	1.0	properties	initialize	inprop	libref	&_cstGRoot/standards/cdisc-crtdds-1.0-1.4/programs	1	initialize.properties
8	CDISC-CRTDDS	1.0	properties	validation	valprop	libref	&_cstGRoot/standards/cdisc-crtdds-1.0-1.4/programs	2	validation.properties
9	CDISC-CRTDDS	1.0	referencemetadata	column	crtfref	libref	&_cstGRoot/standards/cdisc-crtdds-1.0-1.4/metadata		.reference_columns.sas7bdat
10	CDISC-CRTDDS	1.0	referencemetadata	table	crtfref	libref	&_cstGRoot/standards/cdisc-crtdds-1.0-1.4/metadata		.reference_tables.sas7bdat
11	CDISC-CRTDDS	1.0	results	validationmetrics	results	libref	tsasroot/././SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/results		.validation_metrics
12	CDISC-CRTDDS	1.0	results	validationresults	results	libref	tsasroot/././SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/results		.validation_results
13	CDISC-CRTDDS	1.0	sourcedata		srcdata	libref	tsasroot/././SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/data		
14	CDISC-CRTDDS	1.0	sourcemetadata	column	srcmeta	libref	tsasroot/././SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/metadata		.source_columns
15	CDISC-CRTDDS	1.0	sourcemetadata	table	srcmeta	libref	tsasroot/././SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/metadata		.source_tables

Special Topic: Validation Check Macros

These SAS Clinical Standards Toolkit design requirements shape the implementation of the 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 9.3.
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 SAS macros 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, 11 unique CDISC CRT-DDS validation checks, and multiple standards. The SAS Clinical Standards Toolkit check macros support validation of CDISC SDTM, CDISC ADaM, CDISC CRT-DDS, and CDISC ODM files.

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. This table identifies and describes the purpose of each of the check macros provided with the SAS Clinical Standards Toolkit.

Table 6.22 SAS Clinical Standards Toolkit Validation Check Macros

Check Macro	Code Logic Style	Description of Purpose
cstcheck_column	Statement	Identifies any invalid column values or attributes.
cstcheck_columncompare	Step	Supports comparison of column values.
cstcheck_columnexists	By default, this check does not require the use of codeLogic. If the check metadata includes a non-null value of codeLogic, then DATA step code logic is required.	Determines whether one or more of the columns defined in columnScope exist in each of the tables defined in tableScope.
cstcheck_columnvarlist	Step	Supports comparison of multiple columns within the same data set or across multiple data sets.
cstcheck_comparedomains	Step	Compares values for one or more columns in one domain with values for those same columns in another domain.
cstcheck_crossstdcomparedomains	Step	Generally compares values for 1+ columns in one table against either those same columns in another domain in another standard, or compares values against metadata from the comparison standard.
cstcheck_crossstdmetamismatch	Step	Identifies inconsistencies between metadata across registered standards.
cstcheck_dsmismatch	Step	Identifies any data set mismatches between study and template metadata and the source data library.
cstcheck_metamismatch	Step	Identifies inconsistencies between study and reference column metadata.
cstcheck_notconsistent	Step	Identifies any inconsistent column values across records.
cstcheck_notimplemented	(not used)	Placeholder to report that a check is not yet implemented.

Check Macro	Code Logic Style	Description of Purpose
cstcheck_notinodelist	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_notsorted	(not used)	Identifies any domain that is not sorted by the keys defined in the metadata.
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)?
cstcheck_recismatch	Step	Identifies any record mismatches across domains (domain as referenced in another domain).
cstcheck_recnofound	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_violatesstd	Statement	Identifies any invalid column values defined in a reference standard.
cstcheck_zeroobs	(not used)	Identifies any data set with zero observations.

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

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.

This 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 6.24 on page 149](#).

Table 6.23 CDISC SDTM Validation Checks

Check Macro (codesource)	Type of Check (checktype)	Unique Check Identifier (checkid) (add SDTM prefix)
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, 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_columnvarlist	ColumnValue	0452
cstcheck_comparedomains	Multitable	0645, 0801, 0804, 0812, 0842, 0844, 0845, 0846
cstcheck_dsmismatch	Metadata	0004, 0005, 0006, 0017
cstcheck_metamismatch	Metadata	0011, 0012, 0013, 0014, 0015, 0019, 0020, 0022, 0023, 0030, 0031, 0032, 0033
cstcheck_notconsistent	Multirecord	0604, 0605, 0607, 0621, 0643, 0644
	Multitable	0807

Check Macro (codesource)	Type of Check (checktype)	Unique Check Identifier (checkid) (add SDTM prefix)
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
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_notsorted	Multirecord	0601
cstcheck_notunique	Multirecord	0602, 0603, 0622, 0623, 0631, 0641, 0642, 0651, 0661, 0662, 0671, 0808, 0809
cstcheck_recmismatch	Multitable	0851, 0861, 0862, 0863, 0864, 0865, 0866, 0871, 0872
cstcheck_recnofound	Multitable	0802, 0803, 0805, 0806, 0811, 0821, 0822, 0823, 0831, 0836, 0841
cstcheck_violatesstd	Column	0201, 0202, 0203, 0606
cstcheck_zeroobs	Metadata	0001, 0002, 0003

Table 6.24 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
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

Check Macro (codesource)	Check Type (checktype)	Unique Check Identifier (checkid)	Corresponding CRT-DDS Validation Number*
cstcheck_recnofound	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
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 5, “CDISC CRT-DDS 1.0 Validation Checks,”](#) on page 405.

More complete documentation is provided for each check macro in [Appendix 3, “Macro Application Programming Interface,”](#) on page 279. This information is derived from the code header. See [“Special Topic: Validation Customization”](#) on page 164.

Special Topic: How the 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: `usesourcemetadate`, `tablescope`, `columnscope`, and `codeologic`.

The SAS Clinical Standards Toolkit uses `usesourcemetadate` to point to the correct metadata. If `usesourcemetadate` 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 code logic, 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".

This 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;
```

These 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 6.25 Multiple Validation Check Invocations for a Specific CheckID

checkid	tablescope	columnscope	code logic
SDTM0604	_ALL_-TS	**SEQ	<pre>%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;</pre>
SDTM0604	TS	TSSEQ	<pre>data work._cstproblems; set &_cstDSName (keep=&_cstDSKeys &_cstColumn); if &_cstcolumn ne _n_ then output; run;</pre>

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 89](#).)

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 show how to solve the problem. The generated Results data set contains this excerpt:

Display 6.13 Example of a 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 IESTESTCD 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 shown here:

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

Consider this 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.

Special Topic: SAS Implementation of ISO 8601

Overview

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 is in the references.

This 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.

SAS provides capabilities for processing ISO 8601 text strings that are far beyond those capabilities 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.

Example ISO 8601 Values

Overview

The tables in this section provide an overview of some commonly used values. It groups the comments based on the ISO 8601 string type.

Dates and Times: Template

Table 6.26 Example ISO 8601 Values for Dates and Times: Template

String	Interpretation	Comment
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

Table 6.27 Example ISO 8601 Values for Dates and Times: Full Datetime Examples

String	Interpretation	Comment
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.
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 (:).

String	Interpretation	Comment
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	<p>If provided, the time zone must be in HH:MM format. It cannot be truncated or a partial value.</p> <p>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 (,).</p> <p>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.</p>
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.

Table 6.28 Example ISO 8601 Values for Dates and Times: Partial Datetime Examples

String	Interpretation	Comment
-----T22:29	The time 10:29 p.m. No value for the date is provided.	<p>A time value must always be prefixed by a date value.</p> <p>In this example, the date value is completely missing, which would be appropriate for time-only fields.</p>
2009	Year 2009.	Trailing values can be truncated when the values are missing.
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.	<p>Trailing partial values can be omitted (truncated).</p> <p>If time is omitted, then T must also be omitted.</p>
2009-03--T12	The 12th hour of an unknown day in March 2009.	Missing day of month.

Durations: Template**Table 6.29** Example ISO 8601 Values for Durations: Template

String	Interpretation	Comment
PnYnMnDTnHnMnS	Duration	<p>A span of time where n is the number of the unit that follows the unit.</p> <p>P: indicates that the value is a duration (period)</p> <p>nY: n elapsed years</p> <p>nM: n elapsed months</p> <p>nD: n elapsed days</p> <p>T: the elapsed time in hours, minutes, and seconds</p> <p>nH: n elapsed hours</p> <p>nM: n elapsed minutes</p> <p>nS: n elapsed seconds</p> <p>Typically, only the units with actual values are given. For example, P0Y1M would be P1M.</p>

Durations: Examples**Table 6.30** Example ISO 8601 Values for Durations: Examples

String	Interpretation	Comment
P1D	The span of one day.	<p>Durations always start with P for a period of time.</p> <p>Units of time that are not known are usually omitted. If time is omitted, then T must also be omitted.</p>
P0000-00-01	The span of zero years + zero months + one day.	<p>Durations can be expressed in an alternative format.</p> <p>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.</p>
P1Y2M3DT4H5M6S	The span of 1 year, 2 months, 3 days, 4 hours, 5 minutes, and 6 seconds.	<p>The units must be in the correct order.</p> <p>The T is required for all time values, but it should not be specified if no time value is given.</p>

Intervals: Template**Table 6.31** Example ISO 8601 Values for Intervals: Template

String	Interpretation	Comment
PnYnMnDTnHnMnS/YYYY-MM-DDTHH:MM:SS	Intervals	This is a duration that is anchored to a specific point in time.
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: Examples**Table 6.32** Example ISO 8601 Values for Intervals: Examples

String	Interpretation	Comment
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.
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.

SAS ISO 8601 References

This table lists additional references for SAS ISO 8601.

Table 6.33 SAS ISO 8601 References

Topic	Link
SAS 9.3 Language Reference: Concepts	http://support.sas.com/documentation/cdl/en/lrcon/62753/HTML/default/viewer.htm#titlepage.htm

Topic	Link
Working with Dates and Times Using the ISO 8601 Basic and Extended Notations	http://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#p1a0qt18rxydrkn1b0rtdfh2t8zs.htm
CALL IS8601_CONVERT Routine	http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p0bhy7ndmdivmmn10b2okmbgiqmj.htm
\$N8601Bw.d Informat	http://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#n1mqdr981wjxx3n1lkqndfer2ei5.htm
\$N8601Ew.d Informat	http://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#p17xoiovjnngrnlp8yw1r0xyyep.htm
Reading Dates and Times Using the ISO 860 Basic and Extended Notations	http://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#n09mk4hlba9wpln1tc3e7x0eow8q.htm

Special Topic: Debugging a Validation Process

Overview

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 this excerpt from the sample driver module (`validate_data.sas` for SDTM 3.1.1) documented in “Running a Validation Process” on page 111, additional messaging to the SAS log can be enabled.

```
%let _cstDebug=0;
data _null_;
    _cstDebug = input(symget('_cstDebug'),8.);
    if _cstDebug then
        call execute("options &_cstDebugOptions;");
    else
        call execute("options nomprint nomlogic 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. Your debugging focus is different for these two tasks.

Errors in Setting Up the SAS Clinical Standards Toolkit Environment

In the SAS Clinical Standards Toolkit environment setup, errors most often occur because of problems with the SASReferences data set. For recommendations on configuring the SASReferences data set appropriately, see [“Building a SASReferences File” on page 67](#).

This table lists some common setup errors and possible causes.

Table 6.34 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	<p>(1) An invalid physical name for the libref has been used.</p> <p>Is the libref a valid SAS name?</p> <p>A SAS name can contain one to 32 characters.</p> <p>It must start with a letter or an underscore (_), not a number.</p> <p>Subsequent characters must be letters, numbers, or underscores.</p> <p>Blanks cannot appear in SAS names.</p> <p>Is the libref a reserved SAS libref name? You should not use Work, Sasuser, or Sashelp.</p> <p>(2) The path specified for the libref is invalid; it points to a nonexistent directory. Check the path in your SASReferences data set.</p>
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	<p>There is a problem with the SASReferences data set being used. Check for these potential problems:</p> <p>The SASReferences data set does not exist.</p> <p>The SASReferences data set exists but it is empty.</p> <p>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.</p> <p>A column type might be incorrect. For example, the Order column might be character instead of numeric.</p> <p>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 <i><global standards library directory>/standards/cst-framework-1.4/control</i>.</p> <p>A TYPE value is missing.</p> <p>A SASREF value is missing or invalid.</p> <p>A REFTYPE value is missing or is not equal to libref or fileref (case insensitive).</p>
Error: Physical file does not exist.	SAS Log	<p>(1) The SASReferences data set references a file that does not exist.</p> <p>(2) The filename is not a valid SAS name.</p>
WARNING: Apparent invocation of macro SDTM_VALIDATE not resolved.	SAS Log	<p>(1) The macro is misnamed or has not been added to the expected autocall library.</p> <p>Does the macros folder for this standard exist in the cstGlobalLibrary, in the !sasroot hierarchy, or in some correctly designated custom location?</p> <p>(2) The expected autocall path was not created correctly in the call to %cstutil_allocatesasreferences.</p> <p>Check that the SASReferences data set contains a type=autocall record, defined as a fileref, and points to the correct folder location.</p> <p>Check for an error occurring earlier in the SAS log suggesting that %cstutil_allocatesasreferences failed before setting the autocall path.</p>

Errors in Performing Some Primary SAS Clinical Standards Toolkit Action

If the task to perform the primary SAS Clinical Standards Toolkit action begins (that is, the standard-specific validation macro, such as %sdm_validate or %crtdds_validate, is found and begins processing), then setup has completed successfully. The 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, these 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.

This table lists some common causes for premature process failure or the failure of specific checks to run.

Table 6.35 Debugging Validation Process Errors

Error	Resultid in Results Data Set	Possible Cause or Corrective Action
No tables evaluated-check validation control data set.	CST0002	No tables interpreted from the <code>tablescope</code> value could be found in the <code>work._csttablemetadata</code> data set.
<Data set> could not be found	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 <code>columnscope</code> value could be found in the <code>work._cstcolumnmetadata</code> data set. The SAS Clinical Standards Toolkit looks at the union of both <code>tablescope</code> and <code>columnscope</code> to build <code>work._cstcolumnmetadata</code> . The specified column might exist in a domain, but not in any column specified in a <code>tablescope</code> domain.
Lookup to SASReferences control data set failed.	CST0006	The SAS Clinical Standards Toolkit code has a call to the <code>estutil_getsasreference</code> 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.

Error	Resultid in Results Data Set	Possible Cause or Corrective Action
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 <code>_ALL_-DM</code>).
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 <code>usesourcemetadate</code> value, or no <code>lookuptype</code> or <code>lookupsources</code> 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

Here are some debugging tips that you might find useful:

- Review available Work files for information about the errors (for example, `_cstresults`, `_csttablemetadate`, and `_cstcolumnmetadate`). 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 ends.
- When debugging, avoid setting the parameter flags in `cstutil_cleanupstsession` to 1 (if that cleanup macro is called).

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

- Use `work._cstcolumnmetadate` and `work._csttablemetadate` 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 you can customize the solution to meet your 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 the SAS Clinical Standards Toolkit messaging, including internationalization
- attempting to validate multiple studies in a single validation process

Each of these customizations is described in these case studies:

- [“Case Study 1: Modifying an Existing Standard or Defining a New Reference Standard” on page 164](#)
- [“Case Study 2: Using Any Set of Source Data and Metadata” on page 165](#)
- [“Case Study 3: Modifying the SAS Validation Checks for Supported Standards” on page 165](#)
- [“Case Study 4: Adding New Validation Checks for Supported Standards” on page 166](#)
- [“Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros” on page 167](#)
- [“Case Study 6: Modifying the SAS Clinical Standards Toolkit Messaging, Including Internationalization” on page 168](#)
- [“Case Study 7: Validation of Multiple Studies” on page 169](#)

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 17](#).

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.

TIP Best Practice Recommendation: 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 9.3 (`!sasroot/../../../../SASClinicalStandardsToolkitSDTM312/1.4/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. 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. As you find it useful to modify a reference standard (for example, to include other domains you consistently use) or you have one or more studies that have new domains, changes are likely to involve alterations to the Validation Master and Validation Control (run time) tablescope or columnscope fields.
- Changing the Validation Control code logic 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 167.) 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 89](#).
- 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 this 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 you must abide by the 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 99.
- Use a similar validation check as a template to build the check metadata required by the SAS Clinical Standards Toolkit. Ask yourself the following types of questions:
 - What category or type of check is it?
Look at the Validation Master data set checktype column. Does it look only at table or column metadata, and not at data values (Metadata)? Does it require a specific raw column value (ColumnValue), or a value that complies with some controlled terminology (Cntlterm)? Must the assessment look across multiple records (Multirecord) or multiple tables (Multitable)?
 - Does the check compare columns within a single table?

Consider Validation Master records where the codesource column is `estcheck_columncompare`, `estcheck_columnvarlist`, or `estcheck_notunique`.

- Does the check compare tables?

Consider Validation Master records where the codesource column is `estcheck_comparedomains` or `estcheck_recnotfound`.

- Does the check look across multiple standards?

Consider Validation Master records where the codesource column is `estcheck_crossstdcomparedomains` or `estcheck_crossstdmetamismatch`.

- What `tablescope` and `columnscope` values are appropriate?

- **Tablescope**

Does the check apply to a specific class of tables (for example, `Class:Findings`)? Does the check apply to all tables for the standard (`_ALL_`)? Does the check apply only to one or more specific tables (for example, `DM+TA`)? Does the check apply to all tables except one (for example, `_ALL_-DM`)? Does the check compare the same column in two tables (for example, `[DM][TA]`)?

- **Columnscope**

Does the check apply to all columns in the selected tables (`_ALL_`)? Does the check apply only to one column (for example, `USUBJID`)? Does the check compare two columns in the same table (for example, `[AESDTH][AEOUT]`)? Does the check apply to all column names that end in a particular suffix (for example, `**DTC`)?

- If column values are to be compared against an external source (coding dictionary or specificodelist), how are these values referenced for other checks in the `lookuptype` and `lookupsource` Validation Master columns?

Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros

The SAS Clinical Standards Toolkit provides 18 validation check macros. These macros, located in the primary SAS Clinical Standards Toolkit autocall library, offer a variety of code examples that are available to all standards supporting validation. For information about the purpose and use of each check macro, see [“Special Topic: Validation Check Macros” on page 144](#) and [Appendix 3, “Macro Application Programming Interface,” on page 279](#).

Some validation scenarios might require modifications to the SAS Clinical Standards Toolkit check macros or the derivations of new macros. If so, these 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 144](#).

- 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 the SAS Clinical Standards Toolkit Messaging, Including Internationalization

This case study considers these three issues related to the support of the SAS Clinical Standards Toolkit messaging:

1. Maintain the relationship between the 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 this CDISC SDTM 3.1.1 Validation Master record excerpt:

Display 6.14 Validation Master Data Set Excerpt for Check SDTM0013

checkid	standard	standardversion	checksource	sourceid	checkseverity	tablescape
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 6.15 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 this CDISC SDTM Validation Master record excerpt:

Display 6.16 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 6.17 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 be needed only 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.

Case Study 7: Validation of Multiple Studies

Most illustrations and discussions in this chapter assume a reference to a single clinical study. But, what if you need to validate multiple clinical studies at one time? A key consideration is the information that source data libraries and source metadata files

contain, and how they should be referenced in the SASReferences data set used by the validation process.

Consider the following four methodologies, which are ordered based on estimated rates of adoption. Other candidate methodologies are possible.

- A common methodology is to build single source data and metadata libraries that contain pooled data sets where metadata reconciliation has already occurred. (This is frequently done in integrated summaries of efficacy and safety.) In this case, the SASReferences data set will contain a single type=sourcedata record pointing to the pooled integrated data library. The SASReferences SAS librefs (where type=sourcemetadate) must match the source metadata library references in the sasref column of the table and column metadata data sets.
- A second methodology is to build a SAS Clinical Standards Toolkit process that daisy-chains multiple job streams, where each study is defined in a unique SASReferences data set and validated independently. Within the same SAS session, unless your validation process deletes work files, the results and metrics files are appended. The files at the end of the process contain results for all studies.
- An alternative approach defines a single SASReferences libref for multiple type=sourcedata records, each pointing to a different study source library. The SAS Clinical Standards Toolkit supports library concatenation, but SAS only reads data sets from the first defined library when the same data set name occurs in multiple libraries. Because standard domain names are expected, this approach does not work unless a unique domain-naming convention across studies is used. A similar approach is required for source metadata. These constraints make this approach less tenable.
- Another alternative methodology is to use multiple SASReferences librefs (multiple type=sourcedata records). You have one for each study source library, and a single source metadata library (with one table and one column metadata data set, setting the SASRef column to each libref used in SASReferences). This methodology works for any validation check that does not compare columns across domains or compares domains. Because of the way tablescope and columnscope parsing occurs in the SAS Clinical Standards Toolkit, source data libraries are not considered. This makes it possible that unintended comparisons of multiple columns or multiple domains from different studies can occur. As a result, this methodology is not recommended.

Special Topic: Using Alternative Controlled Terminologies

The SAS Clinical Standards Toolkit supports using any set of controlled terminology or any coding dictionaries such as MedDRA or WHO Drug.

Generally, controlled terminology is defined to the SAS Clinical Standards Toolkit as SAS format catalogs, and coding dictionaries as SAS data sets, although either format is allowed. A SASReferences data set documents all of these, and facilitates run-time references to the input sources. In the SAS Clinical Standards Toolkit sample drivers, a SASReferences type=fmtsearch record points to each SAS format catalog (and allows specification of a reference order for like-named formats). And, a type=referenceceterm record points to each specific coding dictionary to be referenced. The format search path is set with a call to the cstutil_processsetup utility macro.

Consider the following scenarios and how each one can be handled using the SAS Clinical Standards Toolkit:

- Scenario 1: You want to create and manage codelists (SAS formats) independent of the CDISC-Terminology standard provided with SAS Clinical Standards Toolkit.

This scenario assumes you have one or more user-defined SAS format catalogs that contain valid values associated with your data columns. These user-defined format catalogs might include extensions to existing CDISC-Terminology codelists or to new formats associated with columns in custom domains. The SAS Clinical Standards Toolkit SASReferences data set enables you to specify references to multiple catalogs and to manage the order in which these appear in the format search path. For example, if you have a catalog named MYTERMS that contains all formats of interest for your study, your SASReferences data set can contain a single type=fmtsearch record:

Display 6.18 Single type=fmtsearch Record Example

standard	standardversion	type	subtype	SASref	reftype	path	order	memname
MY_STD	MY_VERSION	fmtsearch		myfmt	libref	C:/temp/formats	1	myterms.sas7bcat

However, if you prefer to keep your customizations in a separate format catalog, but you want to use the CDISC-Terminology codelists provided by SAS, your SASReferences data set will have multiple type=fmtsearch records, with the order column value set to establish the format search path precedence:

Display 6.19 Multiple type=fmtsearch Records Example

standard	standardversion	type	subtype	SASref	reftype	path	order	memname
MY_STD	MY_VERSION	fmtsearch		myfmt	libref	C:/temp/formats	1	myterms.sas7bcat
CDISC-TERMINOLOGY	NCI_THESAURUS	fmtsearch		cstfmt	libref	&_cstGRoot/standards/cdisc-terminology-&_cstVersion/&_cstCTRoot/formats	2	ctems.sas7bcat

In this case, any extended, like-named formats in MYTERMS will be used instead of the original formats in CTERMS provided by SAS.

- Scenario 2: You want to manage codelist (SAS format) customizations as a registered standard in the global standards library of the SAS Clinical Standards Toolkit.

SAS provides snapshots of the CDISC Terminology standard, as provided by the National Cancer Institute (NCI) Enterprise Vocabulary Services (EVS). These snapshots are defined in the global standards library. In the SAS Clinical Standards Toolkit 1.4, these are provided (by CDISC model and snapshot date) in the following location:

<global standards library>/standards/cdisc-terminology-1.4/.

Consider whether you want to add a new version (such as a dated snapshot) or a completely new set of terminology to the global standards library. To add a new version, follow the snapshot folder hierarchy in the global standards library, and register your new standard in the standardssubtypes data set located in the **<global standards library>/standards/cdisc-terminology-1.4/control** folder.

For example, suppose you want to add a new CDISC ADaM controlled terminology snapshot released on 02February2012. A new 201202 folder hierarchy is created in the global standards library, a new record is added to the standardssubtypes data set, and the format catalog in the Current subfolder is replaced with the 201202 catalog.

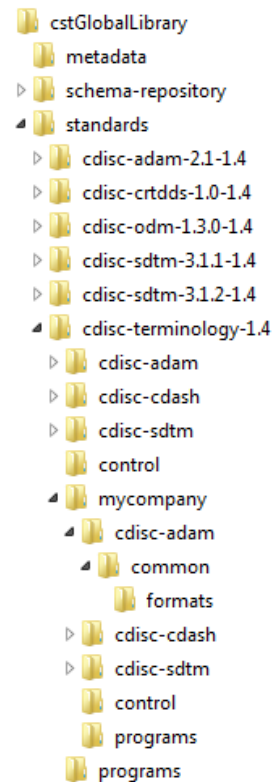
Display 6.20 New Controlled Terminology Record Example

standard	standardversion	standardsubtype	standardsubtypeversion	rootpath	isstandarddefault	description
CDISC-TERMINOLOGY	CDISC-ADAM	NCI_THESAURUS	201101	&_cstGRoot./standards/cdisc-terminology-1.4/cdisc-adam/201101	N	Controlled Terminology released by NCI on 2011-01-07
CDISC-TERMINOLOGY	CDISC-ADAM	NCI_THESAURUS	201202	&_cstGRoot./standards/cdisc-terminology-1.4/cdisc-adam/201202	Y	Controlled Terminology released by NCI on 2012-02-24
CDISC-TERMINOLOGY	CDISC-ADAM	NCI_THESAURUS	Current	&_cstGRoot./standards/cdisc-terminology-1.4/cdisc-adam/current	N	Current Controlled Terminology (copy of the latest version: 2012-02-24)

Note: SAS Clinical Standards Toolkit 1.4 does not provide utility macros to perform any of these tasks. It is assumed that you have Write access to the global standards library.

If you want to add a completely new set of terminology to the global standards library, you must follow the information in “[Maintenance Usage Scenarios](#)” on page 17.

Assume that your organization has created its own comprehensive set of CDISC controlled terminology, and you have created the following global standards library subfolder hierarchy (with CDISC ADaM fully expanded):

Display 6.21 Global Standards Library Subfolder Hierarchy Example

After the registration process, your global standards library data set might look like this (using the folder hierarchy above):

Display 6.22 Global Standards Library Standards Data Set Example

standard	mnemonic	standardversion	comment	rootpath	isstandarddefault
CDISC-ADAM	ADAM	2.1	CDISC ADAM V2.1	&_cstGRoot./standards/cdisc-adam-2.1-1.4	Y
CDISC-CRTDDS	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1.0-1.4	Y
CDISC-ODM	ODM	1.3.0	CDISC ODM V1.3.0	&_cstGRoot./standards/cdisc-odm-1.3.0-1.4	Y
CDISC-SDTM	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.4	N
CDISC-SDTM	SDTM	3.1.2	CDISC SDTM V3.1.2	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.4	Y
CDISC-TERMINOLOGY	CT	COMPANY_STD	CDISC terminology used by our company	&_cstGRoot./standards/cdisc-terminology-1.4/mycompany	Y
CDISC-TERMINOLOGY	CT	NCI_THESAURUS	CDISC Terminology 2011-01-07	&_cstGRoot./standards/cdisc-terminology-1.4	N
CST-FRAMEWORK	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework-1.4	Y

The standardsubtypes data set located in the `<global standards library>/standards/cdisc-terminology-1.4/mycompany/control` folder now contains this CDISC ADaM record:

Display 6.23 CDISC ADaM Record Example

standard	standardversion	standardsubtype	standardsubtypeversion	rootpath	isstandarddefault	description
CDISC-TERMINOLOGY	CDISC-ADAM	COMPANY_STD	common	&_cstGRoot./standards/cdisc-terminology-1.4/ mycompany/cdisc-adam/common	Y	Controlled Terminology (company standard as of 2011-07-31)

- Scenario 3: You use multiple versions of the MedDRA dictionary to code Adverse Events across multiple studies within a submission.

The SAS Clinical Standards Toolkit does not provide copies of the MedDRA coding dictionary as maintained and distributed by the Maintenance and Support Services Organization. Your organization more than likely maintains the multiple updates to MedDRA, and you might need to reference multiple versions of MedDRA in a single SAS Clinical Standards Toolkit process.

Although it is possible to create and use SAS format catalogs for MedDRA lookups (and similar coding dictionary lookups), the SAS Clinical Standards Toolkit provides a mechanism to reference and use a data set lookup methodology in the SASReferences data set using one or more `type=referenceceterm` records. Each record points to a specific MedDRA version using a unique SAS libref, with the resulting `libref.dataset` available for use, as needed.

- Scenario 4: You use the WHO Drug dictionary to ensure that your coding of Concomitant Medications in CMDECOD and CMCLASCD includes valid terms and class codes.

The SAS Clinical Standards Toolkit does not provide copies of the WHO Drug dictionary as created by the World Health Organization and managed by the Uppsala Monitoring Centre. As in Scenario 3, the SAS Clinical Standards Toolkit provides a mechanism to reference and use a data set lookup methodology in the SASReferences data set using one or more `type=referenceceterm` records. Your WHO Drug reference might look like this:

Display 6.24 WHO Drug Reference Example

standard	standardversion	type	subtype	SASref	reftype	path	order	memname
CDISC-SDTM	3.1.2	referenceceterm		ctref	libref	C:/coding-dictionaries/whodrug/01june2009		whodrug.sas7bdat

The SAS Clinical Standards Toolkit provides several CDISC SDTM validation checks that involve lookups to coding dictionaries. Relevant metadata columns from the validation check data set are listed:

Display 6.25 Metadata Columns Example

checkid	codesource	tablescode	columnscode	codelogic	lookuptype	lookupsource
SDTM0450	cstcheck_notincodelist	_ALL_	DECODE	%let _cstDictCol=<dictionary column placeholder>;proc sql noprint;create table work._cstproblems as select ds.dict _cstDictCol from &_cstDSName ds left join &_cstLookupSource dict on upcase(ds._cstColumn) = upcase(dict._cstDictCol) where dict._cstDictCol="";quit;	DATASET	/* dictionary name goes here */
SDTM0451	cstcheck_notincodelist	AE	AEDECOD	proc sql noprint;create table work._cstproblems as select ds._dict_pt_name from &_cstDSName ds left join &_cstLookupSource dict on upcase(ds._cstColumn) = upcase(dict.pt_name) where dict.pt_name="";quit;	DATASET	meddra

The codelogic value is specific to the coding dictionary. In a WHO Drug lookup, `drugname` and `atc_code` (or their equivalents) are used. The `cstcheck_notincodelist` check macro retrieves and uses the lookup data set named in the `lookupsource` metadata column based on information stored in the SASReferences data set records where `type=referenceceterm`.

Special Topic: Performance Considerations

Here are some best practice recommendations:

- You should first run the SAS Clinical Standards Toolkit validation on a subset of source data to identify general process problems, missing or inconsistent process control metadata, and common (and perhaps correctable) data errors.
- You should subset the SAS Clinical Standards Toolkit standard-specific Validation Master data set to remove duplicate checks. For example, CDISC SDTM 3.1.1 Janus checks are generally duplicates of WebSDM checks with occasionally different resultseverity values.
- You should be toggled off the `_cstDebug` option, 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.
- You should run in batch or using PROC PRINTTO any SAS Clinical Standards Toolkit validation process that involves a large number of checks. 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	176
Reading XML Files	176
Overview of Basic Workflow	176
Reading CDISC ODM XML Files: odm_read Macro	176
Sample Driver Program: create_sasodm_fromxml.sas	179
Reading CDISC CRT-DDS define.xml Files: crtdds_read Macro	183
Sample Driver Program: create_sascrtdds_fromxml.sas	185
Writing XML Files	190
Overview	190
Basic Workflow	190
Creating the CDISC CRT-DDS 1.0 define.xml File	191
Sample Driver Program: create_crtdds_from_sdtm.sas	192
Sample Driver Program: create_crtdds_define.sas	195
Creating the CDISC ODM 1.3.0 XML File	198
Sample Driver Program: create_odmxml.sas	199
Validation of XML-Based Standards	202
XML Validation	202
Validating CDISC CRT-DDS 1.0 Files	202
Validating CDISC ODM 1.3.0 Files	206
Special Topic: A Round Trip Exercise Involving the CDISC	
SDTM and CDISC CRT-DDS Standards	210
Overview	210
The Workflow	211
Running Multiple Driver Programs	213
Special Topic: A Round Trip Exercise Involving the CDISC	
CRT-DDS Standard: Importing and Exporting the define.xml File	214
Overview	214
Sample Driver Program: import_sascrtdds_fromxml_export_toxml.sas	215
Special Topic: Identifying Unsupported Elements and	
Attributes in a CDISC ODM File	218
Overview	218
Sample Utility Program: find_unsupported_tags.sas	220

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. The SAS Clinical Standards Toolkit reads a CDISC ODM 1.3.0 or a CDISC CRT-DDS 1.0 XML file and converts the information into a SAS 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. For CDISC ODM 1.3.0, there are 66 data sets containing 315 columns in the SAS representation of the model. 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. The SAS Clinical Standards Toolkit can create both a CDISC CRT-DDS 1.0 XML file and a CDISC ODM 1.3.0 XML file.

Reading XML Files

Overview of Basic Workflow

Here 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

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. This folder is located here:

```
<global standards library directory>/standards/cdisc-  
odm-1.3.0-1.4/ macros
```

This macro is referenced from the *create_sasodm_fromxml.sas* driver program (described more fully below).

File references and other metadata that are required by the macro are set as global macro variable values. Currently, these 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 defined in the SASReferences data set, the odm_read macro accesses the ODM XML file.

Here is a partial listing of a 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>
    <MeasurementUnit OID="MeasurementUnits.OID.MMHG" Name="MMHG"
      <Symbol>
        <TranslatedText xml:lang="en">mmHG</TranslatedText>
        <TranslatedText xml:lang="fr-CA">mmHG</TranslatedText>
      </Symbol>
    </MeasurementUnit>
    <MeasurementUnit OID="MeasurementUnits.OID.YRS" Name="YEARS">
      <Symbol>
        <TranslatedText xml:lang="de">Jahren</TranslatedText>
        <TranslatedText xml:lang="en">Years of age</TranslatedText>
        <TranslatedText xml:lang="fr-CA">Ans</TranslatedText>
      </Symbol>
    </BasicDefinitions>
    <MetaDataVersion 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. 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 **/reference.xml** 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.

Here is a partial listing of the **odm.map** file.

```
<?xml version="1.0" encoding="windows-1252"?>
<SXLEMAP name="ODM130" version="1.2">

<TABLE name="ItemDefs">
  <TABLE-PATH syntax="XPath">/LIBRARY/ItemDefs</TABLE-PATH>
  <TABLE-DESCRIPTION>Item metadata</TABLE-DESCRIPTION>

  <COLUMN name="OID">
    <PATH syntax="XPath">/LIBRARY/ItemDefs/OID</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>character</DATATYPE>
    <DESCRIPTION>Unique identifier for this item</DESCRIPTION>
    <LENGTH>64</LENGTH>
  </COLUMN>
  <COLUMN name="Name">
    <PATH syntax="XPath">/LIBRARY/ItemDefs/Name</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>character</DATATYPE>
    <DESCRIPTION>Item (variable) name</DESCRIPTION>
    <LENGTH>128</LENGTH>
  </COLUMN>
  <COLUMN name="DataType">
    <PATH syntax="XPath">/LIBRARY/ItemDefs/DataType</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>character</DATATYPE>
    <DESCRIPTION>Item (variable) data type (text, integer, float)</DESCRIPTION>
    <LENGTH>18</LENGTH>
  </COLUMN>
  <COLUMN name="Length">
    <PATH syntax="XPath">/LIBRARY/ItemDefs/Length</PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>numeric</DATATYPE>
    <DESCRIPTION>Item (variable) length</DESCRIPTION>
    <LENGTH>8</LENGTH>
  </COLUMN>
```

When the **cubeXML** is processed, each of the 66 data sets (such as **ItemDefs**) that are included in the SAS representation of the CDISC ODM model is derived.

A number of input parameters can be specified in the call to the **odm_read** macro. These parameters offer the options of building source metadata files and SAS format catalogs for codelist translated text. These parameters are itemized in this table.

Table 7.1 ODM_read Macro Parameters

Parameter	Description
_cstBuildSrcMetadata	Create the source metadata files (for example, source_tables and source_columns) as a part of the Read operation. Default=Y (yes), otherwise leave blank. Optional.

Parameter	Description
<code>_cstBuildFmtCat</code>	Build format catalog(s), representing language-specific codelist TranslatedText, as a part of the Read operation. Default=Y (yes), otherwise leave blank. Optional.
<code>_cstFmtLib</code>	Where catalog(s) are to be written. Optional. If not specified, default first to value derived from SASReferences, then Work.
<code>_cstReplaceFmtCat</code>	Should an existing format catalog by that name in <code>_cstFmtLib</code> be replaced? Optional. Values: N Y Default behavior: Y (overwrite existing catalog)
<code>_cstFmtCatPrefix</code>	Use this prefix for catalog names. Optional. If not specified, default is <standard mnemonic>FmtCat (such as ODMFmtCat). This default will produce an English format catalog name of ODMFmtCat_en.
<code>_cstFmtCatLang</code>	If specified, create a format catalog ONLY for the specified language. Optional. Example: <code>_cstFmtCatLang=en</code> . If no records exist for the specified language, an empty catalog is created.
<code>_cstFmtCatLangOption</code>	If no language tag is provided in the XML, what action should be taken with these records? Optional. Values: Ignore English Use_cstFmtCatLang. If Ignore, records are ignored (but reported in the SAS log). If English, records are added to the English catalog (default). If Use_cstFmtCatLang, records are added to the language catalog specified in the <code>_cstFmtCatLang</code> parameter.

By default, if a null-parameter `%odm_read()` macro call is made, source metadata files and SAS format catalogs for each language found in the `clitemdecodetranslatedtext` data set are created after the SAS data sets representing the ODM XML metadata and data content are derived. The target location of the derived metadata files is defined in the SASReferences data set. The target location of any derived SAS format catalogs is the SAS Work library unless 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`.

The driver program is located at:

```
!sasroot/../../../../SASClinicalStandardsToolkitODM130/1.4/sample/
cdisc-odm-1.3.0/programs
```

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, 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 67](#).

In the SASReferences data set, there are two input file references and five output references that are key to the successful completion of the driver program. [Table 7.2 on page 180](#) lists these files and data sets, and they are discussed in separate sections. In the

sample create_sasodm_fromxml.sas driver module, these values are set for &studyRootPath and &studyOutputPath:

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitODM130/
&_cstVersion/sample/cdisc-odm-1.3.0

&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitODM130/&cstVersion/sample/cdisc-
odm-1.3.0
```

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	odmxml	fileref	&studyRootPath/ sourcexml	odm_sample.xml
Input	referencexml	odmmmap	fileref	&studyRootPath/ referencexml	odm.map
Output	sourcedata	srcdata	libref	&studyOutputPath /derived/data	*.*
Output	sourcemetadata	srcmeta	libref	&studyOutputPath /derived/metadata	source_ tables.sas7bdat
Output	sourcemetadata	srcmeta	libref	&studyOutputPath /derived/metadata	source_ columns.sas7bdat
Output	targetdata	trgdata	libref	&studyOutputPath /derived/formats	
Output	results	results	libref	&studyOutputPath /results	read_ results.sas7bdat

Process Inputs

The metadata type externalxml refers to the ODM XML file that is being read. The filename reference odmxml is defined in the SASReferences data set. This filename reference is used in the submitted SAS code when referring to the ODM XML 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 reference odmmmap is defined in the SASReferences data set. This filename reference 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 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.

This 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

VIEWTABLE: Results.Read_results								
	Result 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)
1	CST0108	1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cst-framework-1.4/programs/initialize.properties	Info	0	0
2	CST0102	1	1	CST_CREATEDS	work.sasreferences was created as requested	Info	0	0
3	CST0200	1	1	CSTUTIL_PROCESSETUP	Process setup is using this SASReferences: C:\Users\vjans\AppData\Local\Temp\SAS Temporary Files\TD7372_L72371\sasreferences	Info	0	0
4	CST0108	1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-odm-1.3.0-1.4/programs/initialize.properties	Info	0	0
5	CST0200	1	1	ODM_XMLVALIDATE	PROCESS STANDARD: CDISC-ODM	Info	0	0
6	CST0200	1	2	ODM_XMLVALIDATE	PROCESS STANDARDVERSION: 1.3.0	Info	0	0
32	ODM0001	1	18	XML TRANSFORMER	The document validated successfully	Info	0	0
33	ODM0115	1	1	ODM_XMLVALIDATE	No errors were found in the ODM file.	Info	0	0
34	CST0200	1	1	ODM_READ	PROCESS STANDARD: CDISC-ODM	Info	0	0
35	CST0200	1	2	ODM_READ	PROCESS STANDARDVERSION: 1.3.0	Info	0	0
36	CST0200	1	3	ODM_READ	PROCESS DRIVER: CREATE_SASODM_FROMXML	Info	0	0
37	CST0200	1	4	ODM_READ	PROCESS DATE: 2011-06-21T00:22:53	Info	0	0
38	CST0200	1	5	ODM_READ	PROCESS TYPE: FILEIO	Info	0	0
39	CST0200	1	6	ODM_READ	PROCESS SASREFERENCES: C:\Users\vjans\AppData\Local\Temp\SAS Temporary Files\TD7372_L72371_cstsasrefs.sas7bdat	Info	0	0
40	CST0200	1	7	ODM_READ	PROCESS STUDYROOTPATH: lsasroot/././SASClinicalStandardsToolkitODM130/1.4/sample/cdisc-odm-1.3	Info	0	0
41	CST0200	1	8	ODM_READ	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info	0	0
42	CST0200	1	9	ODM_READ	PROCESS CSTVERSION: 1.4	Info	0	0
43	CST0200	1	1	JAVA CHECK	No Java issues	Info	0	0
44	ODM0013	1	1	ODM_READ	The ODM map file was read from the following location: C:\Program Files\SASHome\SASClinicalStandardsToolkitODM130\1.4\sample\cdisc-odm-	Info	0	0
45	CST0200	1	2	ODM_READ	Destination library for format catalogs set to trgdata	Info	0	0
46	CST0200	1	3	CSTUTIL_BUILDFORMATSFROMXML	trgdata.ODMfmtcat_de catalog and data set created	Info	0	0
47	CST0200	1	4	CSTUTIL_BUILDFORMATSFROMXML	trgdata.ODMfmtcat_en catalog and data set created	Info	0	0
48	CST0200	1	5	CSTUTIL_BUILDFORMATSFROMXML	trgdata.ODMfmtcat_fr_CA catalog and data set created	Info	0	0
49	ODM0012	1	6	ODM_READ	The ODM file C:\Program Files\SASHome\SASClinicalStandardsToolkitODM130\1.4\sample\cdisc-odm- 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 are derived from the ODM XML file.

Display 7.2 Example of Partial Source_Tables Data Set Derived during odm_read

VIEWTABLE: Srcmeta.Source_tables					
	SASreferences sourcedata	Table Name	Name of Standard	Version of Standard	Case-sensitive XML element name
1	SRCDATA	AdminData	CDISC-ODM	1.3.0	AdminData
2	SRCDATA	Annotation	CDISC-ODM	1.3.0	Annotation
3	SRCDATA	AnnotationFlag	CDISC-ODM	1.3.0	AnnotationFlag
4	SRCDATA	Association	CDISC-ODM	1.3.0	Association
5	SRCDATA	AuditRecord	CDISC-ODM	1.3.0	AuditRecord
6	SRCDATA	CLItemDecodeTranslatedText	CDISC-ODM	1.3.0	CLItemDecodeTranslatedText
7	SRCDATA	ClinicalData	CDISC-ODM	1.3.0	ClinicalData
8	SRCDATA	CodeListItems	CDISC-ODM	1.3.0	CodeListItems
9	SRCDATA	CodeLists	CDISC-ODM	1.3.0	CodeLists
10	SRCDATA	ConditionDefFormalExpression	CDISC-ODM	1.3.0	ConditionDefFormalExpression
11	SRCDATA	ConditionDefTranslatedText	CDISC-ODM	1.3.0	ConditionDefTranslatedText
12	SRCDATA	ConditionDefs	CDISC-ODM	1.3.0	ConditionDefs
13	SRCDATA	EnumeratedItems	CDISC-ODM	1.3.0	EnumeratedItems
14	SRCDATA	ExternalCodeLists	CDISC-ODM	1.3.0	ExternalCodeLists
15	SRCDATA	FormData	CDISC-ODM	1.3.0	FormData
16	SRCDATA	FormDefArchLayouts	CDISC-ODM	1.3.0	FormDefArchLayouts
17	SRCDATA	FormDefItemGroupRefs	CDISC-ODM	1.3.0	FormDefItemGroupRefs
18	SRCDATA	FormDefTranslatedText	CDISC-ODM	1.3.0	FormDefTranslatedText
19	SRCDATA	FormDefs	CDISC-ODM	1.3.0	FormDefs
20	SRCDATA	ImputationMethods	CDISC-ODM	1.3.0	ImputationMethods
21	SRCDATA	ItemAliases	CDISC-ODM	1.3.0	ItemAliases
22	SRCDATA	ItemData	CDISC-ODM	1.3.0	ItemData
23	SRCDATA	ItemDefTranslatedText	CDISC-ODM	1.3.0	ItemDefTranslatedText
24	SRCDATA	ItemDefs	CDISC-ODM	1.3.0	ItemDefs
25	SRCDATA	ItemGroupAliases	CDISC-ODM	1.3.0	ItemGroupAliases
26	SRCDATA	ItemGroupData	CDISC-ODM	1.3.0	ItemGroupData
27	SRCDATA	ItemGroupDefItemRefs	CDISC-ODM	1.3.0	ItemGroupDefItemRefs

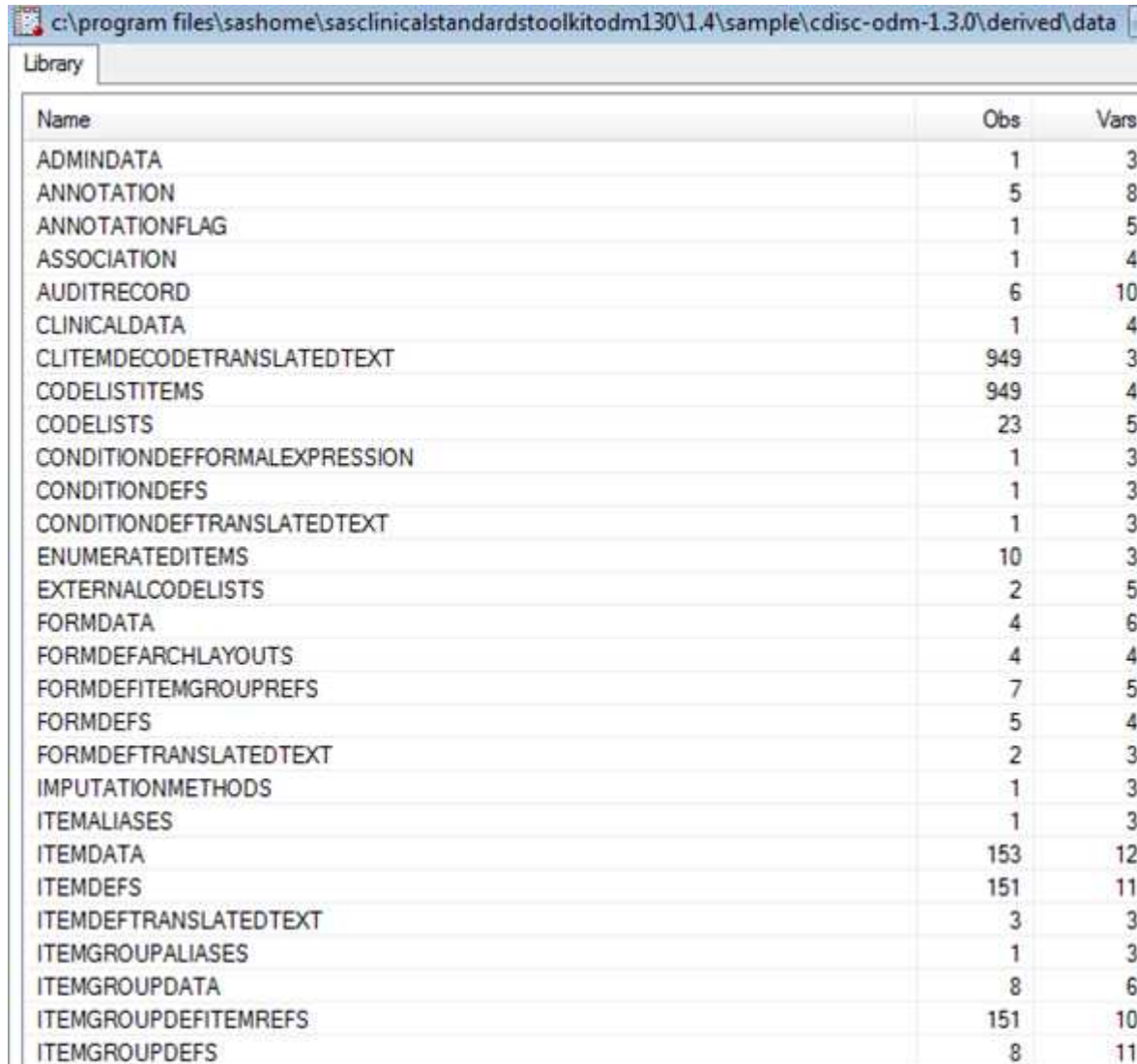
Display 7.3 Example of Partial Source_Columns Data Set Derived during odm_read

VIEWTABLE: Srcmeta.Source_columns							
	SASreferences sourcedata libref	Table Name	Column Name	Column Description	Column Order	Column Type	Column Length
1	SRCDATA	AdminData	GeneratedID	CST generated unique ID	1	C	64
2	SRCDATA	AdminData	StudyOID	Associated unique study identifier	2	C	64
3	SRCDATA	AdminData	FK_ODM	Foreign key: ODM.FileOID	3	C	64
4	SRCDATA	Annotation	GeneratedID	CST generated unique ID	1	C	64
5	SRCDATA	Annotation	ID	Unique ID for a specific Annotation element	2	C	2000
6	SRCDATA	Annotation	SeqNum	Uniquely identifies the annotation within its parent entity	3	N	8
7	SRCDATA	Annotation	TransactionType	Transaction type (Insert Update Remove Upsert Context)	4	C	7
8	SRCDATA	Annotation	CommentSponsorOrSite	Comment source (Sponsor Site)	5	C	7
9	SRCDATA	Annotation	Comment	Free-text (uninterpreted) comment about clinical data	6	C	2000
10	SRCDATA	Annotation	ParentType	Parent element type	7	C	14
11	SRCDATA	Annotation	ParentKey	Associated OID or ID in ParentType table	8	C	64
12	SRCDATA	AnnotationFlag	FlagValue	Value of flag	1	C	2000
13	SRCDATA	AnnotationFlag	FlagValueCodeListOID	Foreign key: CodeLists.OID	2	C	64
14	SRCDATA	AnnotationFlag	FlagType	Type of flag	3	C	128
15	SRCDATA	AnnotationFlag	FlagTypeCodeListOID	Foreign key: CodeLists.OID	4	C	64
16	SRCDATA	AnnotationFlag	FK_Annotation	Foreign key: Annotation.GeneratedID	5	C	64
17	SRCDATA	Association	GeneratedID	CST generated unique ID	1	C	64
18	SRCDATA	Association	StudyOID	Foreign key: Study.OID	2	C	64
19	SRCDATA	Association	MetaDataVersionOID	Foreign key: MetaDataVersion.OID	3	C	64
20	SRCDATA	Association	FK_ODM	Foreign key: ODM.FileOID	4	C	64
21	SRCDATA	AuditRecord	ID	Unique ID for a specific AuditRecord element	1	C	2000

The Srcdata library contains the SAS data sets that represent the ODM file metadata and content. By default, the odm_read macro creates 66 unique data sets in the SAS Clinical

Standards Toolkit. 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



Name	Obs	Vars
ADMINDATA	1	3
ANNOTATION	5	8
ANNOTATIONFLAG	1	5
ASSOCIATION	1	4
AUDITRECORD	6	10
CLINICALDATA	1	4
CLITEMDECODETRANSLATEDTEXT	949	3
CODELISTITEMS	949	4
CODELISTS	23	5
CONDITIONDEFFORMALEXPRESSION	1	3
CONDITIONDEFS	1	3
CONDITIONDEFTRANSLATEDTEXT	1	3
ENUMERATEDITEMS	10	3
EXTERNALCODELISTS	2	5
FORMDATA	4	6
FORMDEFARCHLAYOUTS	4	4
FORMDEFITEMGROUPPREFS	7	5
FORMDEFS	5	4
FORMDEFTRANSLATEDTEXT	2	3
IMPUTATIONMETHODS	1	3
ITEMALIASES	1	3
ITEMDATA	153	12
ITEMDEFS	151	11
ITEMDEFTRANSLATEDTEXT	3	3
ITEMGROUPALIASES	1	3
ITEMGROUPDATA	8	6
ITEMGROUPDEFITEMREFS	151	10
ITEMGROUPDEFS	8	11

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. The SAS Clinical Standards Toolkit 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 is available in the CRT-DDS 1.0 standards macro folder, located in *<global standards library directory>/standards/cdisc-crtdds-1.0-1.4/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 `crtds_read` macro, the Results data set contains all framework and CRT-DDS 1.0 specific messages generated during run time.

Based on file references defined in the `SASReferences` data set, the `crtds_read` macro accesses the `define.xml` file.

Here is a partial listing of a `define.xml` file.

```
<ODM xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:def="http://www.cdisc.org/ns/def/v1.0"
  xmlns="http://www.cdisc.org/ns/odm/v1.2" FileOID="1"
  CreationDateTime="2011-07-13T17:15:43-04:00"
  AsOfDateTime="2011-07-13T17:12:42"
  Description="define1" FileType="Snapshot" Id="define1"
  ODMVersion="1.0">
<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="1.0.0"
    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"/>
```

After the `crtds_read` macro confirms that the `define.xml` file exists, a call is made to the SAS DATA step component `JavaObj`. 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 `_cube#####.xml`, where `#####` 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.

Here 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>Annotated CRF metadata</TABLE-DESCRIPTION>

  <COLUMN name="DocumentRef">
    <PATH syntax="XPath">/LIBRARY/AnnotatedCRFs/DocumentRef</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>character</DATATYPE>
    <DESCRIPTION>The referenced Annotated CRF document</DESCRIPTION>
    <LENGTH>2000</LENGTH>
  </COLUMN>
  <COLUMN name="leafID">
    <PATH syntax="XPath">/LIBRARY/AnnotatedCRFs/leafID</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>character</DATATYPE>
    <DESCRIPTION>The unique ID of the referenced Annotated CRF</DESCRIPTION>
    <LENGTH>128</LENGTH>
  </COLUMN>
  <COLUMN name="FK_MetaDataVersion">
    <PATH syntax="XPath">/LIBRARY/AnnotatedCRFs/FK_MetaDataVersion</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>character</DATATYPE>
    <DESCRIPTION>Foreign key: MetaDataVersion.OID</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.

The driver program is located in:

```
!sasroot/..SASClinicalStandardsToolkitCRTDDS10/1.4/sample/
cdisc-crtdds-1.0/programs
```

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, 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 67](#).

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. [Table 7.3 on page 186](#) lists these files and data sets, and they are discussed in separate sections. In the sample create_sasrctdds_fromxml.sas driver program, these values are set for &studyRootPath and &studyOutputPath and are specific to a SAS release.

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/
&_cstVersion/sample/cdisc-crtdds-1.0

&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/
&_cstVersion/sample/cdisc-crtdds-1.0
```

Table 7.3 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	libref	&studyOutputPat h/deriveddata	*.*
Output	sourcemetadata	srcmeta	libref	&studyOutputPat h/ derivedmetadata	source_ tables.sas7bdat
Output	sourcemetadata	srcmeta	libref	&studyOutputPat h/ derivedmetadata	source_ columns.sas7bdat
Output	sourcemetadata	srcmeta	libref	&studyOutputPat h/ derivedmetadata	source_ study.sas7bdat
Output	results	results	libref	&studyOutputPat h/results	read_ results.sas7bdat

Process Inputs

Process Inputs The metadata type externalxml refers to the define.xml file that is being read. The filename reference crtxml is defined in the SASReferences data set. This filename reference 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 reference `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 `crtds_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.

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. Both of the data sets are written to the `Srcmeta` library. The `sourcemetadata` also refers to a data set `source_study`. The `source_study` data set is also created in the `Srcmeta` library and contains study metadata.

The `results` type refers to the `Results` data set that contains information from running the CRT-DDS process. This information is written to the `read_results` data set in the `Results` library.

Process Results

When the driver program finishes running, the `read_results` 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.

This 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	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)
1	CST0108	1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:\cstGlobalLibrary\standards\cst-framework-1.4\programs\initialize.properties	Info	0	0
2	CST0102	1	1	CST_CREATEDS	work.sasreferences was created as requested	Info	0	0
3	CST0200	1	1	CSTUTIL_PROCESSETUP	Process setup is using this SASReferences: C:\Users\vfjans\AppData\Local\Temp\SAS Temporary Files_TD7552_L72371_cstsasrefs.sas7bdat	Info	0	0
4	CST0108	1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:\cstGlobalLibrary\standards\cdisc-crtdds-1.0-1.4\programs\initialize.properties	Info	0	0
5	CST0200	1	1	CRTDDS_XMLVALIDATE	PROCESS STANDARD: CDISC-CRTDDS	Info	0	0
6	CST0200	1	2	CRTDDS_XMLVALIDATE	PROCESS STANDARDVERSION: 1.0	Info	0	0
7	CST0200	1	3	CRTDDS_XMLVALIDATE	PROCESS DRIVER: CREATE_CRTDDS_DEFINE	Info	0	0
8	CST0200	1	4	CRTDDS_XMLVALIDATE	PROCESS DATE: 2011-08-07T15:32:23	Info	0	0
9	CST0200	1	5	CRTDDS_XMLVALIDATE	PROCESS TYPE: VALIDATE CRTDDS DEFINE XML	Info	0	0
10	CST0200	1	6	CRTDDS_XMLVALIDATE	PROCESS SASREFERENCES: C:\Users\vfjans\AppData\Local\Temp\SAS Temporary Files_TD7552_L72371_cstsasrefs.sas7bdat	Info	0	0
11	CST0200	1	7	CRTDDS_XMLVALIDATE	PROCESS STUDYROOTPATH: %ssroot%\..\SASClinicalStandards\Toolkit\CRTDDS10\1.4\sample\cdisc-crtdds	Info	0	0
12	CST0200	1	8	CRTDDS_XMLVALIDATE	PROCESS GLOBALLIBRARY: c:\cstGlobalLibrary	Info	0	0
13	CST0200	1	9	CRTDDS_XMLVALIDATE	PROCESS CSTVERSION: 1.4	Info	0	0
14	CST0200	1	1	JAVA CHECK	No Java issues	Info	0	0
15	CRT0001	1	1	XML TRANSFORMER	Transform starting.	Info	0	0
16	CRT0001	1	2	XML TRANSFORMER	Using JRE: C:\PROGRAM~2\Java\jre6	Info	0	0
32	CRT0001	1	18	XML TRANSFORMER	The document validated successfully	Info	0	0
33	CRT0115	1	1	CRTDDS_XMLVALIDATE	No errors were found in the CRT-DDS file.	Info	0	0
34	CST0200	1	1	CRTDDS_READ	PROCESS STANDARD: CDISC-CRTDDS	Info	0	0
35	CST0200	1	2	CRTDDS_READ	PROCESS STANDARDVERSION: 1.0	Info	0	0
36	CST0200	1	3	CRTDDS_READ	PROCESS DRIVER: CREATE_SASCRTDDS_FROMXML	Info	0	0
37	CST0200	1	4	CRTDDS_READ	PROCESS DATE: 2011-08-07T15:32:27	Info	0	0
38	CST0200	1	5	CRTDDS_READ	PROCESS TYPE: FILEIO	Info	0	0
39	CST0200	1	6	CRTDDS_READ	PROCESS SASREFERENCES: C:\Users\vfjans\AppData\Local\Temp\SAS Temporary Files_TD7552_L72371_cstsasrefs.sas7bdat	Info	0	0
40	CST0200	1	7	CRTDDS_READ	PROCESS STUDYROOTPATH: %ssroot%\..\SASClinicalStandards\Toolkit\CRTDDS10\1.4\sample\cdisc-crtdds	Info	0	0
41	CST0200	1	8	CRTDDS_READ	PROCESS GLOBALLIBRARY: c:\cstGlobalLibrary	Info	0	0
42	CST0200	1	9	CRTDDS_READ	PROCESS CSTVERSION: 1.4	Info	0	0
43	CST0200	1	1	JAVA CHECK	No Java issues	Info	0	0
44	CRT0013	1	1	CRTDDS_READ	The CRT-DDS map file was read from the following location: C:\Program Files\SASHome\SASClinicalStandards\Toolkit\CRTDDS10\1.4\sample\cdisc-crt-dds-map.xml	Info	0	0
45	CRT0012	1	2	CRTDDS_READ	The CRT-DDS file C:\Program Files\SASHome\SASClinicalStandards\Toolkit\CRTDDS10\1.4\sample\cdisc-crt-dds 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

VIEWTABLE: Srcmeta.Source_tables					
	SASReferences sourcedata libref	Table Name	Name of Standard	Version of Standard	Case-sensitive XML element name
1	SRCDATA	AnnotatedCRFs	CDISC-CRTDDS	1.0	AnnotatedCRFs
2	SRCDATA	CLItemDecodeTranslatedText	CDISC-CRTDDS	1.0	CLItemDecodeTranslatedText
3	SRCDATA	CodeListItems	CDISC-CRTDDS	1.0	CodeListItems
4	SRCDATA	CodeLists	CDISC-CRTDDS	1.0	CodeLists
5	SRCDATA	ComputationMethods	CDISC-CRTDDS	1.0	ComputationMethods
6	SRCDATA	DefineDocument	CDISC-CRTDDS	1.0	DefineDocument
7	SRCDATA	ExternalCodeLists	CDISC-CRTDDS	1.0	ExternalCodeLists
8	SRCDATA	FormDefArchLayouts	CDISC-CRTDDS	1.0	FormDefArchLayouts
9	SRCDATA	FormDefItemGroupRefs	CDISC-CRTDDS	1.0	FormDefItemGroupRefs
10	SRCDATA	FormDefs	CDISC-CRTDDS	1.0	FormDefs
11	SRCDATA	ImputationMethods	CDISC-CRTDDS	1.0	ImputationMethods
12	SRCDATA	ItemAliases	CDISC-CRTDDS	1.0	ItemAliases
13	SRCDATA	ItemDefs	CDISC-CRTDDS	1.0	ItemDefs
14	SRCDATA	ItemGroupAliases	CDISC-CRTDDS	1.0	ItemGroupAliases
15	SRCDATA	ItemGroupDefItemRefs	CDISC-CRTDDS	1.0	ItemGroupDefItemRefs
16	SRCDATA	ItemGroupDefs	CDISC-CRTDDS	1.0	ItemGroupDefs
17	SRCDATA	ItemGroupLeaf	CDISC-CRTDDS	1.0	ItemGroupLeaf
18	SRCDATA	ItemGroupLeafTitles	CDISC-CRTDDS	1.0	ItemGroupLeafTitles
19	SRCDATA	ItemMURefs	CDISC-CRTDDS	1.0	ItemMURefs
20	SRCDATA	ItemQuestionExternal	CDISC-CRTDDS	1.0	ItemQuestionExternal
21	SRCDATA	ItemQuestionTranslatedText	CDISC-CRTDDS	1.0	ItemQuestionTranslatedText

Display 7.7 Example of Partial Source_Columns Data Set Derived during crtdds_read

VIEWTABLE: Srcmeta.Source_columns									
	SASReferences sourcedata libref	Table Name	Column Name	Column Description	Column Order	Column Type	Column Length	Name of Standard	Version of Standard
1	SRCDATA	AnnotatedCRFs	DocumentRef	The referenced Annotated CRF document	1	C	2000	CDISC-CRTDDS	1.0
2	SRCDATA	AnnotatedCRFs	leafID	The unique ID of the referenced Annotated CRF	2	C	128	CDISC-CRTDDS	1.0
3	SRCDATA	AnnotatedCRFs	FK_MetaDataVersion	Foreign key: MetaDataVersion.OID	3	C	128	CDISC-CRTDDS	1.0
4	SRCDATA	CLItemDecodeTranslatedText	TranslatedText	Human-readable text appropriate for a particular language	1	C	2000	CDISC-CRTDDS	1.0
5	SRCDATA	CLItemDecodeTranslatedText	lang	Natural language or country-specific language variant	2	C	17	CDISC-CRTDDS	1.0
6	SRCDATA	CLItemDecodeTranslatedText	FK_CodeListItems	Foreign key: CodeListItems.OID	3	C	128	CDISC-CRTDDS	1.0
7	SRCDATA	CodeListItems	OID	Unique identifier for this codelist item	1	C	128	CDISC-CRTDDS	1.0
8	SRCDATA	CodeListItems	CodedValue	Value of the codelist item	2	C	512	CDISC-CRTDDS	1.0
9	SRCDATA	CodeListItems	FK_CodeLists	Foreign key: CodeLists.OID	3	C	128	CDISC-CRTDDS	1.0
10	SRCDATA	CodeListItems	Rank	CodedValue order relative to other coded item values	4	N	8	CDISC-CRTDDS	1.0
11	SRCDATA	CodeLists	OID	Unique identifier for this codelist	1	C	128	CDISC-CRTDDS	1.0
12	SRCDATA	CodeLists	Name	CodeList name	2	C	128	CDISC-CRTDDS	1.0
13	SRCDATA	CodeLists	DataType	CodeList item value data type (integer float text string)	3	C	7	CDISC-CRTDDS	1.0
14	SRCDATA	CodeLists	SASFormatName	SAS format name	4	C	8	CDISC-CRTDDS	1.0
15	SRCDATA	CodeLists	FK_MetaDataVersion	Foreign key: MetaDataVersion.OID	5	C	128	CDISC-CRTDDS	1.0
16	SRCDATA	ComputationMethods	OID	Unique identifier for this computation method	1	C	128	CDISC-CRTDDS	1.0
17	SRCDATA	ComputationMethods	method	Rule for deriving data value	2	C	2000	CDISC-CRTDDS	1.0
18	SRCDATA	ComputationMethods	FK_MetaDataVersion	Foreign key: MetaDataVersion.OID	3	C	128	CDISC-CRTDDS	1.0
19	SRCDATA	DefineDocument	FileOID	Unique identifier for this file	1	C	128	CDISC-CRTDDS	1.0
20	SRCDATA	DefineDocument	Archival	File meets requirements of an electronic	2	C	3	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

Library					
Name	Obs	Vars	Name	Obs	Vars
ANNOTATEDCRFS	0	3	ITEMQUESTIONTRANSLATEDTEXT	0	3
CLITEMDECODETRANSLATEDTEXT	2909	3	ITEMRANGECHECKS	0	5
CODELISTITEMS	2909	4	ITEMRANGECHECKVALUES	0	2
CODELISTS	32	5	ITEMROLE	0	2
COMPUTATIONMETHODS	0	3	ITEMVALUELISTREFS	0	2
DEFINEDOCUMENT	1	12	MDVLEAF	0	3
EXTERNALCODELISTS	0	3	MDVLEAFTITLES	0	2
FORMDEFARCHLAYOUTS	0	4	MEASUREMENTUNITS	0	3
FORMDEFITEMGROUPPREFS	0	4	METADATAVERSION	1	9
FORMDEFS	0	4	MUTRANSLATEDTEXT	0	3
IMPUTATIONMETHODS	0	3	PRESENTATION	0	4
ITEMALIASES	0	3	PROTOCOLEVENTREFS	0	4
ITEMDEFS	733	14	RCERRORTRANSLATEDTEXT	0	3
ITEMGROUPALIASES	0	3	STUDY	1	5
ITEMGROUPDEFITEMREFS	733	8	STUDYEVENTDEFS	0	6
ITEMGROUPDEFS	33	16	STUDYEVENTFORMREFS	0	4
ITEMGROUPLEAF	33	3	SUPPLEMENTALDOCS	0	3
ITEMGROUPLEAFTITLES	33	2	VALUELISTITEMREFS	0	8
ITEMMUREFS	0	2	VALUELISTS	0	2
ITEMQUESTIONEXTERNAL	0	4			

c:\program files\sashome\sasclinicalstandardstoolkitcrtdds10\1.4\sample\cdisc-crtdds-1.0\deriveddata

When running the driver programs against non-sample data, you must 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 67.

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 the SAS Clinical Standards Toolkit, 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. You can also create a CDISC ODM 1.3.0 XML file.

The next section outlines the basic workflow for the creation of model-specific XML files.

Basic Workflow

Here 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 in the standard-specific SAS format.

2. (Optional) 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).
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. (Optional) Validate the structure and syntax of the XML file that was created.

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:

1. The `crtds_sdtmtodefine` 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.
Note: This macro replaces the `crtds_sdtm311todefine10` macro, which will be deprecated in future releases of the SAS Clinical Standards Toolkit.
2. The `crtds_validate` 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.
3. The `crtds_write` macro creates the define.xml file from the SAS representation of the CRT-DDS files.
4. The `crtds_xmlvalidate` 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 related to the creation of the define.xml file.

Here is the purpose of each of these driver programs:

- The `create_crtds_from_sdtm.sas` driver program sets up the required metadata and SASReferences data set for the sample study. It runs the `crtds_sdtmtodefine` macro. It creates the SAS representation of the CRT-DDS define data sets from the sample study SDTM data sets.
- The `validate_crtds_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.
- The `create_crtds_define.sas` driver program creates the define.xml file. It runs the `crtds_write` and `crtds_xmlvalidate` macros. This driver program creates and validates the XML syntax for the define.xml file. It should be noted that the `create_crtds10_define311.sas` driver program has been replaced by the `create_crtds_define.sas` driver program.

These 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_crtds_from_sdtm.sas`

Overview

The `create_crtds_from_sdtm.sas` driver program sets up the required environment variables and library references to initiate the `crtds_sdtmtodefine` 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 88](#).) 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.

These parameters must be set before submitting the macro:

Table 7.4 Parameters for the `create_crtds_from_sdtm.sas` Macro

Parameter	Required	Description
<code>_cstOutLib</code>	Yes	Identifies the library reference (LIBNAME) where the tables are created.
<code>_cstSourceTables</code>	Yes	A data set that contains the SDTM metadata for the domains to be included in the CRT-DDS file.
<code>_cstSourceColumns</code>	Yes	A data set that contains the SDTM metadata for the domain columns to be included in the CRT-DDS file.
<code>_cstSourceStudy</code>	Yes	A data set that contains the SDTM metadata for the studies to be included in the CRT-DDS file.

Here is an example of a call to the `crtds_sdtmtodefine` macro:

```
%crtds_sdtmtodefine(
  _cstOutLib=srcdata,
  _cstSourceTables=sampdata.source_tables,
  _cstSourceColumns=sampdata.source_columns,
  _cstSourceStudy=sampdata.source_study
);
```

In the example, the `crtds_sdtmtodefine` 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_crtds_from_sdtm.sas` driver program is provided with SAS, and it is ready to run on any of the SDTM sample studies. 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.

The driver program is located in:

```
!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/
cdisc-crtdds-1.0/programs
```

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, 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 67](#).

In the SASReferences data set, there are three input file references and one output reference that are key to successful completion of the create_crtdds_from_sdtm.sas driver program. [Table 7.5 on page 193](#) lists these files and data sets, and they are discussed in separate sections. In the sample create_crtdds_from_sdtm.sas driver program, these values are set for &studyRootPath and &studyOutputPath:

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitSDTM312/
&_cstVersion/sample/cdisc-sdtm-3.1.2/sascstdemodata

&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/
&_cstVersion/sample/cdisc-crtdds-1.0
```

Table 7.5 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	libref	&studyRootPath/ metadata	source_ tables.sas7bdat
Input	sourcemetadata	sampdata	libref	&studyRootPath/ metadata	source_ columns.sas7bdat
Input	sourcemetadata	sampdata	libref	&studyRootPath/ metadata	source_ study.sas7bdat
Output	sourcedata	srcdata	libref	&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_crtdds_from_sdtm.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 this location:

```
!sasroot/../../SASClinicalStandardsToolkitSDTM312/1.4/sample/  
cdisc-sdtm-3.1.2/sascstdemodata/metadata
```

This location is represented in the driver program by the Srcmeta library name.

A source study data set (source_study) is required by this macro. These variables are required in this data set:

Table 7.6 Variables Required in the Source Study Data Set (source_study.sas)

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.
StudyDescription	Yes	Description of the study. This value is used to populate the srcdata.study.studydescription column. <i>Note:</i> You cannot 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_crtds_from_sdtm.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/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtds-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 sdtmtoresults data set is created. This data set contains informational, warning, and any error messages that were generated by the submitted driver program.

Display 7.9 Example of a Partial Results Data Set from CRT-DDS Sample Study

VIEWTABLE: Results.Sdtmdefine_results							
	Result identifier	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)
11	CST0200	7	CREATE_CRTDDS_FROM_SDTM	PROCESS STUDYROOTPATH: %sasroot%\..\SASClinicalStandardsToolkitSDTM312\1.4\sample\cdisc-sdtm-	Info	0	0
12	CST0200	8	CREATE_CRTDDS_FROM_SDTM	PROCESS GLOBALLIBRARY: c:\cstGlobalLibrary	Info	0	0
13	CST0200	9	CREATE_CRTDDS_FROM_SDTM	PROCESS CSTVERSION: 1.4	Info	0	0
14	CST0200	10	CREATE_CRTDDS_FROM_SDTM	PROCESS CONTROLLED TERMINOLOGY SOURCE: c:\cstGlobalLibrary\standards\cdisc-terminology-1.4\cdisc-sdtm\201104\for (Controlled Terminology released by NCI on 2011-04-08)	Info	0	0
15	CST0122	1	CST_CREATETABLESFORDATASTAN	The tables were created for CDISC-CRTDDS 1.0 in library srodata	Info	0	0
16	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.definestructure was created as requested	Info	0	0
17	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.study was created as requested	Info	0	0
18	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.metadaversion was created as requested	Info	0	0
19	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.itemgroupdefs was created as requested	Info	0	0
20	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.codelist was created as requested	Info	0	0
21	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.codelistitems was created as requested	Info	0	0
22	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.citemdecodetranslatedtext was created as requested	Info	0	0
23	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.itemdefs was created as requested	Info	0	0
24	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.itemgroupdefitemrefs was created as requested	Info	0	0
25	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.itemgroupleaf was created as requested	Info	0	0
26	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.itemgroupleaftitles was created as requested	Info	0	0
27	CST0102	1	CRTDDS_SDTMTODEFINE	srodata.computationmethods was created as requested	Info	0	0

Sample Driver Program: create_crtds_define.sas**Overview**

The create_crtds_define.sas driver program sets up the required environment variables and library references to initiate the crtds_write 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 SASReferences data set.

This table lists the optional parameters that can be set when submitting the macro.

Table 7.7 Parameters for the crtds_write.sas Macro

Parameter	Required	Description
_cstCreateDisplayStyleSheet	Optional	Specifies whether the macro creates 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 of referencexml and a subtype of stylesheet, and then 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 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 the SAS Clinical Standards Toolkit.

Parameter	Required	Description
<code>_cstResultsOverrideDS</code>	Optional	Designates [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 <code>&_cstResultsDS</code> global macro variable is used.
<code>_cstLogLevel</code>	Optional	Specifies the level of error reporting. Valid values are Info, Warning, Error, and Fatal Error. The default setting is Info.

Here is an example of a call to the `crtds_write.sas` macro:

```
%crtds_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.

Here is the call to the macro from the sample `create_crtds_define.sas` driver program:

```
%crtds_write(_cstCreateDisplayStyleSheet=1);
```

The call creates a display style sheet and uses default values for the parameters.

The `create_crtds_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.

The driver program is located in:

```
!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/
cdisc-crtds-1.0/programs
```

Multiple tasks can be executed in any SAS Clinical Standards Toolkit driver program. The `create_crtds_define.sas` driver program calls both the `crtds_write` macro to create the `define.xml` file, and the `crtds_xmlvalidate` macro to validate the syntax of the generated `define.xml` file. For more information about the `crtds_xmlvalidate` macro, see [“Validation of XML-Based Standards” on page 202](#).

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, 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 67](#).

In the `SASReferences` data set, there are two input file references and three output references that are key to successful completion of the `create_crtds_define.sas` driver program. [Table 7.8 on page 197](#) lists these files and data sets, and they are discussed in separate sections. In the sample `create_crtds_define.sas` driver program, these values are set for `&studyRootPath` and `&studyOutputPath`:

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/&_cstVersion/sample/cdisc-
crtds-1.0

&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/
&_cstVersion/sample/cdisc-crtds-1.0
```

Table 7.8 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	libref	&workpath	sasreferences.sas7bdat
Input	sourcedata	srcdata	libref	&studyRootPath/data	
Output	referencexml	xslt01	filename	&studyOutputPath/sourcexml	
Output	results	results	LIBNAME	&studyOutputPath/results	write_results.sas7bdat
Output	externalxml	extxml	filename	&studyOutputPath/sourcexml	define.xml
Input	referencexml	odmmmap	fileref	&studyRootPath/referencexml	define.map

Process Inputs

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 this 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_crtds10_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/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtds-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/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtds-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 crtds_write macro interprets the `_cstCreateDisplayStyleSheet=1` parameter to use the default style sheet that is provided by the 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 crtds_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/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtds-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	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)
28	CRT0001	1	13	XML TRANSFORMER PARAMETER	Is Validating XML: true	Info	0	0
29	CRT0001	1	14	XML TRANSFORMER PARAMETER	Creating Display Stylesheet: true	Info	0	0
30	CRT0001	1	15	XML TRANSFORMER PARAMETER	Custom Stylesheet: c:/cstGlobalLibrary/standards/cdisc-crtdd	Info	0	0
31	CRT0001	1	16	XML TRANSFORMER PARAMETER	Custom Stylesheet Output Shortname: define1-0-0.xml	Info	0	0
32	CRT0001	1	17	XML TRANSFORMER PARAMETER	Creating Output Folders: true	Info	0	0
33	CRT0001	1	18	XML TRANSFORMER	Transform complete.	Info	0	0
34	CRT0001	1	19	XML TRANSFORMER	Transform time: 9719 ms.	Info	0	0
35	CRT0001	1	20	XML TRANSFORMER	The document validated successfully	Info	0	0
36	CRT0010	1	1	CRTDDS_WRITE	The CRT-DDS file was created at C:\SAS-Lab\SASClinicalStandardsToolkit	Info	0	0
37	CST0200	1	1	CRTDDS_XMLVALIDATE	Starting XML Validation	Info	0	0

Creating the CDISC ODM 1.3.0 XML File

There are several key macros that are provided with the SAS Clinical Standards Toolkit that support creation of the ODM XML file. The macros are listed in the order in which they are executed:

1. The odm_validate 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 each ODM XML file.
2. The odm_write macro creates the ODM XML file from the SAS representation of the ODM files and validates that the XML file is syntactically correct. This macro is important if you customize the XML file outside of the workflow.
3. The odm_xmlvalidate macro validates that the XML file is syntactically correct. This macro is important if you customize the ODM XML file outside of the workflow.

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. Two sample driver modules are provided with the SAS Clinical Standards Toolkit CDISC ODM standard to support creation of XML files. Here is the purpose of each of these drivers:

1. The validate_odm_data.sas driver program validates the SAS representation of the ODM data sets based on the selected ODM validation checks. This driver program can be run multiple times until data validation has been reconciled.
2. The create_odmxml.sas driver program calls the odm_write macro to create the XML file. This driver program creates and validates the syntax for the XML file.

These 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_odmxml.sas

Overview

The create_odmxml.sas driver program sets up the required environment variables and library references to initiate the odm_write macro. This macro reads the 66 data sets that comprise the default SAS representation of the CDISC ODM 1.3.0 model, and then converts that information to the required ODM XML structure. If source metadata or data are missing, then empty elements and attributes are not created in the ODM XML file. The inputs and outputs are specified in the SASReferences data set.

This table lists the optional parameters that can be set when submitting the macro.

Table 7.9 Parameters for the odm_write.sas Macro

Parameter	Required	Description
_cstCreateDisplayStyleSheet	Optional	Specifies 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 0.
_cstOutputEncoding	Optional	XML encoding to use for the ODM XML file that is created. By default, UTF-8 is used.
_cstHeaderComment	Optional	A short comment is added at the top of the ODM XML file. If no comment is provided, then a default comment is used. The default comment notes that the file was produced by the 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	Specifies the level of error reporting. Valid values are Info, Warning, Error, and Fatal Error. The default setting is Info.

Here is an example of a call to the odm_write macro:

```
%odm_write(_cstOutputEncoding=UTF-16, _cstResultsOverrideDS=&_cstResultsDS);
```

In this example, no default style sheet is generated for the XML output, XML encoding is set to UTF-16, and process results are written to the default &_cstResultsDS data set.

This is the call to the macro from the sample create_odmxml.sas driver program, using default values for all parameters:

```
%odm_write();
```

The create_odmxml.sas driver program is ready to run on the sample CDISC ODM provided with the SAS Clinical Standards Toolkit.

The driver program is located in:

```
!sasroot/../../ SASClinicalStandardsToolkiODM130/1.4/sample/
cdisc-odm-1.3.0/programs
```

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, 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 67](#).

In the SASReferences data set, one input file reference and two output references are key to successful completion of the create_odmxml.sas driver program. [Table 7.10 on page 200](#) lists these files and data sets, and they are discussed in separate sections. In the sample create_odmxml.sas driver program, these values are set for &studyRootPath and &studyOutputPath:

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkiODM130/
&_cstVersion/sample/cdisc-odm-1.3.0

&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkiODM130/
&_cstVersion/sample/cdisc-odm-1.3.0
```

Table 7.10 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	sourcedata	srcdata	libref	&studyRootPath/ data	
Output	results	results	libref	&studyOutputPat h/results	write_ results.sas7bdat
Output	externalxml	extxml	filename	&studyOutputPat h/sourcexml	odm_sample_ out.xml

Process Inputs

The sourcedata type is the library that contains the default 66 data sets that comprise the SAS representation of an ODM XML file. These data sets might have been populated by a previous odm_read task, or you might have processes in place that build these files from some set of source files. In the SAS Clinical Standards Toolkit sample data, these data sets are read from the !sasroot/../../
SASClinicalStandardsToolkiODM130/1.4/sample/cdisc-odm-1.3.0/
data directory. This location is represented in the driver program by the Srcdata library name.

Process Outputs

The externalxml type refers to the ODM XML file that is to be derived by the process. This file is accessed in the driver program using the extxml filename statement and is written to the !sasroot/../../SASClinicalStandardsToolkiODM130/1.4/
sample/cdisc-odm-1.3.0/sourcexml directory.

Note: Unlike CDISC CRT-DDS, CDISC does not supply a default style sheet for ODM, nor is one provided as a part of the SAS Clinical Standards Toolkit. However, if you want to do so, the `odm_write` macro provides the `_cstCreateDisplayStyleSheet` parameter to make use of information that you can provide in the Metadata Type `referencexml` record of the `SASReferences` file.

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 this location:

```
!sasroot/../../SASClinicalStandardsToolkiODM130/1.4/sample/
cdisc-odm-1.3.0/results
```

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 7.11 Example of a Partial Results Data Set from the ODM Sample Data Hierarchy

VIEWTABLE: Results.Write_results								
	Result 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)
11	CST0200	1	7	ODM_WRITE	PROCESS STUDYROOTPATH: !sasroot/../../SASClinicalStandardsToolkitODM1	Info	0	0
12	CST0200	1	8	ODM_WRITE	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info	0	0
13	CST0200	1	9	ODM_WRITE	PROCESS CSTVERSION: 1.4	Info	0	0
14	CST0122	1	1	CST_CREATETABLESFORDATASTANDA	The tables were created for CDISC-ODM 1.3.0 in library _cst0920	Info	0	0
15	CST0200	1	1	JAVA CHECK	No Java issues	Info	0	0
16	ODM0001	1	1	XML TRANSFORMER	Transform starting.	Info	0	0
17	ODM0001	1	2	XML TRANSFORMER	Using JRE: C:\PROGRA~2\Java\jre6	Info	0	0
18	ODM0001	1	3	XML TRANSFORMER PARAMETER	Import Or Export: EXPORT	Info	0	0
19	ODM0001	1	4	XML TRANSFORMER PARAMETER	Standards XML Path: C:/Program Files/SASHome/SASClinicalStandardsToolkitOD	Info	0	0
20	ODM0001	1	5	XML TRANSFORMER PARAMETER	Fail on Validation Error: false	Info	0	0
21	ODM0001	1	6	XML TRANSFORMER PARAMETER	Standard Name: CDISC-ODM	Info	0	0
22	ODM0001	1	7	XML TRANSFORMER PARAMETER	Standard Version: 1.3.0	Info	0	0
23	ODM0001	1	8	XML TRANSFORMER PARAMETER	Schema Repository Location: c:/cstGlobalLibrary/schema-repository	Info	0	0
24	ODM0001	1	9	XML TRANSFORMER PARAMETER	XSL Repository Location: c:/cstGlobalLibrary/xsl-repository	Info	0	0
25	ODM0001	1	10	XML TRANSFORMER PARAMETER	Output Encoding: UTF-8	Info	0	0
26	ODM0001	1	11	XML TRANSFORMER PARAMETER	Log File Location: C:/Users/frjans/AppData/Local/Temp/SAS Temporary Files/_TD6804_L72371/_log5834	Info	0	0
27	ODM0001	1	12	XML TRANSFORMER PARAMETER	Header Comment Text: Produced from SAS data using the SAS Clinical Standards Toolkit	Info	0	0
28	ODM0001	1	13	XML TRANSFORMER PARAMETER	Is Validating XML: true	Info	0	0
29	ODM0001	1	14	XML TRANSFORMER PARAMETER	Creating Display Stylesheet: false	Info	0	0
30	ODM0001	1	15	XML TRANSFORMER PARAMETER	Custom Stylesheet: null	Info	0	0
31	ODM0001	1	16	XML TRANSFORMER PARAMETER	Custom Stylesheet Output Shortname: null	Info	0	0
32	ODM0001	1	17	XML TRANSFORMER PARAMETER	Creating Output Folders: true	Info	0	0
33	ODM0001	1	18	XML TRANSFORMER	Transform complete.	Info	0	0
34	ODM0001	1	19	XML TRANSFORMER	Transform time: 4072 ms.	Info	0	0
35	ODM0001	1	20	XML TRANSFORMER	The document validated successfully	Info	0	0
36	ODM0010	1	1	ODM_WRITE	The ODM file was created at C:\Program Files\SASHome\SASClinicalStandardsToolkitOD	Info	0	0

Validation of XML-Based Standards

XML Validation

When validating XML-based standards (such as CDISC ODM and CDISC CRT-DDS), the SAS Clinical Standards Toolkit offers two complementary methodologies.

The first methodology is described in [Chapter 6, “Validation,” on page 81](#). 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.

The SAS Clinical Standards Toolkit provides both methodologies to support the validation of CDISC CRT-DDS 1.0 and CDISC ODM 1.3.0 files.

Validating CDISC CRT-DDS 1.0 Files

The crtdds_xmlvalidate Macro

The crtdds_xmlvalidate macro validates the structure and syntax of the define.xml file against the XML schema for the CRT-DDS standard. It can be run at any time. The SAS Clinical Standards Toolkit includes a call to the crtdds_xmlvalidate macro immediately following the call to the crtdds_write 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 SAS Clinical Standards Toolkit also includes a call to the crtdds_xmlvalidate macro immediately before the call to the crtdds_read macro in the create_crtdds_fromxml.sas sample driver program.

Here 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 7.11 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	Designates [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 7.12 Log Levels for the *crtds_xmlvalidate.sas* Macro

Log Level	Description
Info	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 the Warning level, then all Warning messages and anything more severe, such as Error and Fatal error messages, are generated. If you choose the Error level, then only Error and Fatal Error messages are generated.

Validation of the SAS Representation: *crtds_validate* Macro

The *crtds_validate* 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 the 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 81](#) and [Appendix 5, “CDISC CRT-DDS 1.0 Validation Checks,” on page 405](#). Metadata about each check is provided in the Validation Master data set in `<global standards library directory>/standards/cdisc-crtdds-1.0-1.4/validation/control`.

The *crtds_validate* 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 metadata `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, you will probably 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 macro.

The driver program is located in:

```
!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/
cdisc-crtdds-1.0/programs
```

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, 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 67](#).

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. [Table 7.13 on page 204](#) lists these libraries and data sets, and they are discussed in separate sections. In the sample validate_crtdds_data.sas driver program, these values are set for &studyRootPath and &studyOutputPath.

Note: The &studyRootPath and &studyOutputPath paths are the same for this driver.

Two macro variables have been retained to maintain consistency across the SAS Clinical Standards Toolkit driver programs.

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/
&_cstVersion/sample/cdisc-crtdds-1.0

&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitCRTDDS10/
&_cstVersion/sample/cdisc-crtdds-1.0
```

Table 7.13 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	libref	&workpath	sasreferences.sas7bdat
Input	control	cntl_v	libref	&studyRootPath/ control	validation_ control.sas7bdat
Input	sourcemetadata	srcmeta	libref	&studyRootPath/ metadata	source_tables.sas7bdat
Input	sourcemetadata	srcmeta	libref	&studyRootPath/ metadata	source_ columns.sas7bdat

Input or Output	Metadata Type	LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	sourcedata	srcdata	libref	&studyRootPath/ data	
Output	results	results	libref	&studyOutputPath/ /results	validation_ results.sas7bdatt

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 this 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 (run-time validation check) data set, and is accessed by the cntl_v LIBNAME statement. This LIBNAME is assigned to the **!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/control** directory.

The sourcedata 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/../../SASClinicalStandardsToolkitCRTDDS10/1.4/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/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/data** directory. This location is represented in the driver program by the Srcdata library name.

Process Outputs

For the 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 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 7.12 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)	Actual value observed
14	CST0200		1	9	CRTDDS_VALIDATE	PROCESS CSTVERSION: 1.4	Info	0	0	
15	CST0022	CRT0100	1	1	CSTCHECK_NOTUNIQUE	SRCDATA.AnnotatedCRFs keys could not be found	Warning: Check not run	-1	0	
16	CST0022	CRT0100	1	2	CSTCHECK_NOTUNIQUE	SRCDATA.CLItemDecodeTranslatedTex keys could not be found	Warning: Check not run	-1	0	
17	CST0100	CRT0100	1	3	SRCDATA.CodeListItems	No errors detected in SRCDATA.CodeListItems	Info	0	0	keys=OID
18	CST0100	CRT0100	1	4	SRCDATA.CodeLists	No errors detected in SRCDATA.CodeLists	Info	0	0	keys=OID
19	CST0100	CRT0100	1	5	SRCDATA.ComputationMethods	No errors detected in SRCDATA.ComputationMethods	Info	0	0	keys=OID
20	CST0100	CRT0100	1	6	SRCDATA.DefineDocument	No errors detected in SRCDATA.DefineDocument	Info	0	0	keys=FileOID
21	CST0022	CRT0100	1	7	CSTCHECK_NOTUNIQUE	SRCDATA.ExternalCodeLists keys could not be found	Warning: Check not run	-1	0	

Validating CDISC ODM 1.3.0 Files

XML Schema Validation

When an ODM XML is created using the `create_odmxml` driver (and the `odm_write` macro), the structure and syntax of the XML file are validated against the XML schema for the ODM standard. The results of this validation are written to the Results data set. Here is a sample of the validation results.

Display 7.13 Example of Schema Validation Reported in a CDISC ODM Results Data Set

	Result 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)
16	ODM0001	1	1	XML TRANSFORMER	Transform starting.	Info	0	0
17	ODM0001	1	2	XML TRANSFORMER	Using JRE: C:\PROGRA~2\Java\jre6	Info	0	0
18	ODM0001	1	3	XML TRANSFORMER PARAMETER	Import Or Export: EXPORT	Info	0	0
19	ODM0001	1	4	XML TRANSFORMER PARAMETER	Standards XML Path: C:/Program Files/SASHome/SASClinicalStandardsToolkitODM130/1.4/sample/cdisc-odm-1.3.0/sourc	Info	0	0
20	ODM0001	1	5	XML TRANSFORMER PARAMETER	Fail on Validation Error: false	Info	0	0
21	ODM0001	1	6	XML TRANSFORMER PARAMETER	Standard Name: CDISC-ODM	Info	0	0
22	ODM0001	1	7	XML TRANSFORMER PARAMETER	Standard Version: 1.3.0	Info	0	0
23	ODM0001	1	8	XML TRANSFORMER PARAMETER	Schema Repository Location: c:/cstGlobalLibrary/schema-repository	Info	0	0
24	ODM0001	1	9	XML TRANSFORMER PARAMETER	XSL Repository Location: c:/cstGlobalLibrary/xsl-repository	Info	0	0
25	ODM0001	1	10	XML TRANSFORMER PARAMETER	Output Encoding: UTF-8	Info	0	0
26	ODM0001	1	11	XML TRANSFORMER PARAMETER	Log File Location: C:/Users/frjans/AppData/Local/Temp/SAS Temporary Files/_TD6804_L72371_/_log5834	Info	0	0
27	ODM0001	1	12	XML TRANSFORMER PARAMETER	Header Comment Text: Produced from SAS data using the SAS Clinical Standards Toolkit	Info	0	0
28	ODM0001	1	13	XML TRANSFORMER PARAMETER	Is Validating XML: true	Info	0	0
29	ODM0001	1	14	XML TRANSFORMER PARAMETER	Creating Display Stylesheet: false	Info	0	0
30	ODM0001	1	15	XML TRANSFORMER PARAMETER	Custom Stylesheet: null	Info	0	0
31	ODM0001	1	16	XML TRANSFORMER PARAMETER	Custom Stylesheet Output Shortname: null	Info	0	0
32	ODM0001	1	17	XML TRANSFORMER PARAMETER	Creating Output Folders: true	Info	0	0
33	ODM0001	1	18	XML TRANSFORMER	Transform complete.	Info	0	0
34	ODM0001	1	19	XML TRANSFORMER	Transform time: 4072 ms.	Info	0	0
35	ODM0001	1	20	XML TRANSFORMER	The document validated successfully	Info	0	0

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 7.14 Log Levels for Schema Validation

Log Level	Description
Info	Messages such as the system properties of the current Java environment and progress messages. This is the default value.

Log Level	Description
Warning	Messages that indicate that there might be an issue with the ODM document or with the execution of the validation process.
Error	Messages that indicate that something in the ODM document is invalid with respect to the normal XML schema for ODM. 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 specified determines what other messages are generated. For example, if the Warning level is specified, then all Warning messages and anything more severe, such as Error and Fatal Error messages, are generated. In the SAS Clinical Standards Toolkit, the Log Level is set to Info by default when using the create_odmxml driver (and the odm_write macro).

It is also possible to use the odm_xmlvalidate macro to validate the structure and syntax of an ODM 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 odm_xmlvalidate macro immediately following the call to the odm_write macro as the last step of the create_odmxml.sas sample driver program. If you customize the ODM XML file after it is generated, then this macro can be used to validate the changes. The SAS Clinical Standards Toolkit also includes a call to the odm_xmlvalidate macro immediately before the call to the odm_read macro in the create_sasodm_fromxml.sas sample driver program.

Here is an example of a call to the odm_xmlvalidate macro:

```
%odm_xmlvalidate(_cstLogLevel=info,_cstResultsOverrideDS=work.xmlvalidate);
```

In this example, the odm_xmlvalidate macro is being submitted with a log level of Info. The Results data set is named XMLVALIDATE and resides in the Work library.

Validation of the SAS Representation: odm_validate Macro

The odm_validate macro supports the XML validation methodology described above that 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 is based on SAS and validates the SAS representation of the XML-based standard.

In the SAS Clinical Standards Toolkit, CDISC ODM 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 ODM that are designed to verify the metadata definitions and values of the default 66 data sets that comprise the SAS representation of the ODM model. These checks were created by SAS. For more information about these checks, see [Chapter 6, “Validation,” on page 81](#) and [Appendix 6, “CDISC ODM 1.3.0 Validation Checks,” on page 415](#). Metadata about each check is provided in the Validation Master data set in *<global standards library directory>/standards/cdisc-odm-1.3.0-1.4/validation/control*. The odm_validate macro controls the validation workflow for ODM. 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 default 66 data sets that comprise the SAS

representation for CDISC ODM. 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 metadata reference_tables and reference_columns data sets. If all 66 ODM tables contribute information to the ODM 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, you can elect to run validation against a subset of the 66 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 odm_validate macro.

Sample Driver Program: validate_odm_data.sas

The validate_odm_data.sas driver program sets up the required environment variables and library references before a call is made to the odm_validate macro.

The driver program is located in:

```
!sasroot/../../SASClinicalStandardsToolkiODM130/1.4/sample/
cdisc-odm-1.3.0/programs
```

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, 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 67](#).

In the SASReferences data set, there are three input file references, one input library reference, and one output file reference that are key to successful completion of the validation process. These libraries and data sets are listed in [Table 7.15 on page 208](#), and they are addressed in separate sections. In the sample validate_odm_data.sas driver program, these values are set for &studyRootPath and &studyOutputPath.

Note: The &studyRootPath and &studyOutputPath paths are the same for this driver. These two macro variables have been retained to maintain consistency across the SAS Clinical Standards Toolkit driver programs.

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkiODM130/
&_cstVersion/sample/cdisc-odm-1.3.0

&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkiODM130/
&_cstVersion/sample/cdisc-odm-1.3.0
```

Table 7.15 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_v	libref	&studyRootPath/ control	validation_ control.sas7bdat

Input or Output	Metadata Type	LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	sourcemetadata	srcmeta	libref	&studyRootPath/ metadata	source_tables.sas7bdat
Input	sourcemetadata	srcmeta	libref	&studyRootPath/ metadata	source_ columns.sas7bdat
Input	sourcedata	srcdata	libref	&studyRootPath/ data	
Output	results	results	libref	&studyOutputPath /results	validation_ results.sas7bdat

Process Inputs

The control record points to the validation_control (run-time validation check) data set and is accessed by the cntl_v LIBNAME statement. This LIBNAME is assigned to the `!sasroot/../../SASClinicalStandardsToolkitODM130/1.4/sample/cdisc-odm-1.3.0/control` directory.

The sourcemetadata type references two metadata data sets that describe the table (source_tables) and column (source_columns) metadata for the default 66 data sets that comprise the SAS representation of the ODM model. Both data sets are stored in the same library. In the SAS Clinical Standards Toolkit, this source metadata is read from the `!sasroot/../../SASClinicalStandardsToolkitODM130/1.4/sample/cdisc-odm-1.3.0/metadata` directory. This location is represented in the driver program using the Srcmeta library name.

The sourcedata type is the library where the default 66 data sets that comprise the SAS representation of the ODM 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/../../SASClinicalStandardsToolkitODM130/1.4/sample/cdisc-odm-1.3.0/data` directory. This location is represented in the driver program by the Srcdata library name.

Process Outputs

For the 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_odm_data driver program finishes running, the validation_results 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 also supported for CDISC ODM validation.

Display 7.14 Example of a CDISC ODM Validation Results Data Set

VIEWTABLE: Results.Validation_results

	Result identifier	Validation check identifier	Unique invocation of resultid	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
194	CST0100	ODM0110	32	SRCDATA.ITEMALIASES (SRCDATA.ITEMDEFS)	No errors detected in source data	Info	0	0	
195	CST0100	ODM0110	33	SRCDATA.ITEMDATA (SRCDATA.ANNOTATION)	No errors detected in source data	Info	0	0	
196	CST0100	ODM0110	34	SRCDATA.ITEMDATA (SRCDATA.AUDITRECORD)	No errors detected in source data	Info	0	0	
197	CST0100	ODM0110	35	SRCDATA.ITEMDATA (SRCDATA.ITEMDEFS)	No errors detected in source data	Info	0	0	
198	CST0100	ODM0110	36	SRCDATA.ITEMDATA (SRCDATA.ITEMGROUPDATA)	No errors detected in source data	Info	0	0	
199	CST0100	ODM0110	37	SRCDATA.ITEMDATA (SRCDATA.MEASUREMENTUNITS)	No errors detected in source data	Info	0	0	
200	CST0100	ODM0110	38	SRCDATA.ITEMDATA (SRCDATA.SIGNATURE)	No errors detected in source data	Info	0	0	
201	ODM0110	ODM0110	39	SRCDATA.ITEMDEFS (SRCDATA.CODELISTS)	The foreign key OID does not have a corresponding value in the target data set SRCDATA.ITEMDEFS	Error	1	0	CODELISTREF=CodeLists.OID.LBTEST
202	CST0100	ODM0110	40	SRCDATA.ITEMDEFS (SRCDATA.METADATAVERSION)	No errors detected in source data	Info	0	0	
203	CST0100	ODM0110	41	SRCDATA.ITEMDEFTRANSLATED (SRCDATA.ITEMDEFS)	No errors detected in source data	Info	0	0	

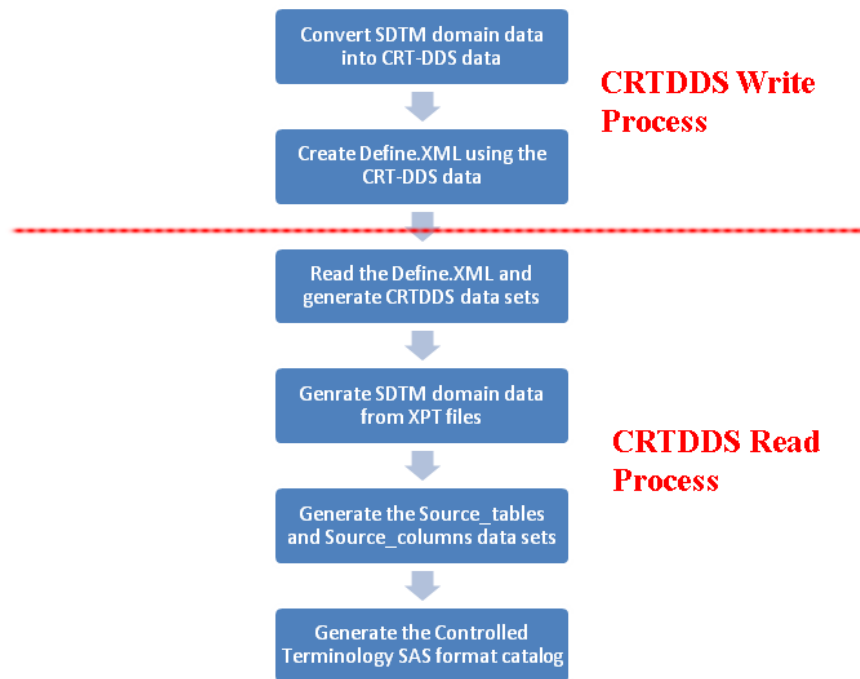
Special Topic: A Round Trip Exercise Involving the CDISC SDTM and CDISC CRT-DDS Standards

Overview

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 stand-alone 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. This display details the high-level workflow for this exercise.

Figure 7.1 Round Trip Process



The Workflow

These 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 the SAS Clinical Standards Toolkit metadata about that SDTM study, such as source_tables and source_columns. The 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_crtds_fromsdm.sas driver program to access the crtds_sdtmtodefine 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/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtds-1.0/data` directory.
4. Validate the CRT-DDS data sets by submitting the validate_crtds_data.sas driver program. This step is optional.
5. Create the define.xml file by submitting the create_crtds_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 crtds_xmlvalidate macro to validate the XML file structure. The define.xml file is written to the `!sasroot/../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtds-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 is 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/../../../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/deriveddata` directory. This driver program generates the `source_tables` and `source_columns` data sets in the `!sasroot/../../../../SASClinicalStandardsToolkitCRTDDS10/1.4/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/../../../../SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata/programs` directory. This driver program accesses the `sdtmutil_createsasdatafromxpt` macro to generate the SDTM domain data sets from the SAS transport files. Creation of the SAS transport files is not performed by the 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` 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/../../../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/cdisc-crtdds-1.0/transport` directory. The generated data sets are written to the `!sasroot/../../../../SASClinicalStandardsToolkitSDTM312/1.4/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.
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/../../../../SASClinicalStandardsToolkitSDTM312/1.4/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/../../../../SASClinicalStandardsToolkitSDTM312/1.4/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.
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/../../../../SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata/programs` directory. The driver program accesses the `sdtmutil_createformatsfromcrtdds` 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/../../../../SASClinicalStandardsToolkitSDTM312/1.4/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.

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.

Running Multiple Driver Programs

CAUTION:

When running multiple driver programs, be aware that 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 this 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, by default the SAS Clinical Standards Toolkit does not attempt to reallocate SAS librefs and filerefs. Records will be written to the process results data set noting (for example):

```
SAS libref from SASref=refmeta sasreferences record not allocated
```

Generally, if you are resubmitting the same process code again, without changing the `&_cststandard` or `&_cststandardversion` global macro variables, or pointers to different data or metadata libraries, this is of no consequence. However, if you are attempting to change the standard or version in the same SAS session, or you are attempting to reference different studies, code libraries or terminology libraries, it is imperative that you use this code between each code submission:

```
%let _cstReallocateSASRefs=1;  
%include "&_cstGRoot/standards/cst-framework-1.4/programs/resetautocallpath.sas";
```

In the driver programs provided with the SAS Clinical Standards Toolkit, the previous code is commented so that it is not submitted during run time.

Special Topic: A Round Trip Exercise Involving the CDISC CRT-DDS Standard: Importing and Exporting the define.xml File

Overview

In general, when representing an XML-based standard in SAS, an XML element is mapped to a SAS data set and its associated attributes are mapped to the columns of the SAS data set. When the SAS Clinical Standards Toolkit creates a CDISC CRT-DDS 1.0 XML file, it converts the information from a SAS data set representation of the CRT-DDS model into XML. For CDISC CRT-DDS 1.0, this means that 39 data sets (such as ItemDefs) containing 176 columns are the source for creating the define.xml element and attribute structure. The SAS representation of the CRT-DDS standard can be derived in part from other standards (such as CDISC SDTM) and can include supporting metadata from other sources.

The first step in creating a define.xml file with the SAS Clinical Standards Toolkit is populating the SAS data set representation of the CRT-DDS model from the SDTM domain metadata (source_tables and source_columns data sets) and the study metadata (source_study data set) by running the crtdds_sdtmtodefine macro. Depending on the completeness of this source data, the crtdds_sdtmtodefine macro can (partially) populate these 12 of the 39 CRT-DDS SAS tables:

clitemdecodetranslatedtext	itemgroupdefitemrefs
codelistitems	itemgroupdefs
codelists	itemgroupleaf
computationmethods	itemgroupleaftitles
definedocument	metadaversion
itemdefs	study

The remainder of the tables will not be automatically populated by the SAS Clinical Standards Toolkit.

To create a CRT-DDS define.xml file that can be considered more complete, these tables typically need be populated by a process not included in the SAS Clinical Standards Toolkit:

- annotatedcrfs (contains document references to the annotated case report form)
- supplementaldocs (contains document references to the supplemental documentation)
- mdvleaf (contains the href link for each of the documents listed in the AnnotatedCRF and SupplementalDocs tables)
- mdvleaftitles (contains a descriptive title for each MDVLeaf item)
- itemvaluelistrefs (associates each value list item to a row in the ItemDefs table)

- valuelists (contains the ID of value lists)
- valuelistitemrefs (contains the ID of each item in a value list)
- externalcodelists (contains name and versions of external code lists)

The SAS Clinical Standards Toolkit provides a sample program to demonstrate that it supports importing and exporting of CRT-DDS metadata beyond the metadata that can be created from the SDTM domain metadata (source_tables and source_columns data sets) and the study metadata (source_study data set).

Sample Driver Program:

import_sascrtdds_fromxml_export_toxml.sas

Overview

The SAS Clinical Standards Toolkit provides a driver program, `import_sascrtdds_fromxml_export_toxml.sas`, to demonstrate import and export of extensive CRT-DDS metadata.

This program is located in:

```
!sasroot/../../../../SASClinicalStandardsToolkitCRTDDS10/1.4/sample/  
cdisc-crtdds-1.0/programs
```

This program provides the same process setup function supported in most SAS Clinical Standards Toolkit driver modules, using a SASReferences data set that defines process inputs and outputs, and allocating all SAS librefs and filerefs. In this sample driver program, the SASReferences data sets are not created in the program, but rather read from a permanent SAS data set.

Here is the general workflow of this sample driver program:

1. Call the `cstutil_processsetup` macro to set process paths and perform required library and file allocations.

The `cstutil_processsetup` macro is called with these parameters:

- `_cstSASReferencesLocation=&studyRootPath/control`
- `_cstSASReferencesName=import_sasreferences`

2. Call the `crtdds_xmlvalidate` macro to validate the CRT-DDS file (`define_import.xml`) to be imported.
3. Call the `crtdds_read` macro to import the CRT-DDS file (`define_import.xml`) that was validated in step 2 to the CRT-DDS SAS data sets in the WORK library.
4. Call the `cstutil_processsetup` macro to set process paths and perform required library and file allocations.

The `cstutil_processsetup` macro is called with these parameters:

- `_cstSASReferencesLocation=&studyRootPath/control`
- `_cstSASReferencesName=export_sasreferences`

5. Call the `crtdds_write` macro to export the CRT-DDS SAS data sets in the WORK library to the CRT-DDS file (`define_export.xml`).
6. Call the `crtdds_xmlvalidate` macro to validate the CRT-DDS file (`define_export.xml`) that was exported in step 5.

The CRT-DDS file `define_export.xml` will be identical to the CRT-DDS file `define_import.xml`, apart from a time stamp.

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, 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 67](#).

The driver program initiates the macro variable &workpath with this SAS code:

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

[Table 7.16 on page 216](#) and [Table 7.17 on page 216](#) list the files and data sets that are key components in the SASReference files that are used in the sample driver program `import_sascrtdds_fromxml_export_toxml.sas`. In this driver program, these values are set for &studyRootPath and &studyOutputPath:

```
&studyRootPath=!sasroot/../../  
SASClinicalStandardsToolkitCRTDDS10/  
&_cstVersion/sample/cdisc-crtdds-1.0  
  
&studyOutputPath=!sasroot/../../  
SASClinicalStandardsToolkitCRTDDS10/  
&_cstVersion/sample/cdisc-crtdds-1.0
```

Table 7.16 Key Components of the SASReferences Data Set `import_sasreference`

Input or Output	Metadata Type	SAS LIBNAME of Fileref to Use	Reference Type	Path	Name of File
Input	externalxml	crtxml	fileref	&studyRootPath/ sourcexml	define_ import.xml
Input	referencexml	crtmap	fileref	&studyRootPath/ referencexml	define.map
Output	sourcedata	srcdata	libref	&workpath	*.*
Output	sourcemetadate	srcmeta	libref	&workpath	source_ tables.sas7bdat
Output	sourcemetadate	srcmeta	libref	&workpath	source_ columns.sas7bdat
Output	sourcemetadate	srcmeta	libref	&workpath	souce_study
Output	results	results	libref	&studyOutputPat h/results	import_ results.sas7bdat

Table 7.17 Key Components of the SASReferences Data Set `export_sasreferences`

Input or Output	Metadata Type	SAS LIBNAME of Fileref to Use	Reference Type	Path	Name of File
Output	externalxml	crtxml	fileref	&studyRootPath/ sourcexml	define_ export.xml

Input or Output	Metadata Type	SAS LIBNAME of Fileref to Use	Reference Type	Path	Name of File
Output	referencexml	xslt01	fileref	&studyRootPath/ referencexml	
Input	sourcedata	srcdata	libref	&workpath	*.*
Output	results	results	libref	&studyOutputPat h/results	export_ results.sas7bdat

Process Outputs

When running the sample driver program interactively, you can verify in the Work library the SAS representation of the CRT-DDS model contains observations for these CRT-DDS data sets.

clitemdecodetranslatedtext	metadataaversion
codelistitems	study
codelists	annotatedcrfs
computationmethods	supplementaldocs
definedocument	mdvleaf
itemdefs	mdvleaftitles
itemgroupdefitemrefs	itemvaluelistrefs
itemgroupdefs	valuelists
itemgroupleaf	valuelistitemrefs
itemgroupleaftitles	externalcodelists

This example shows how the XML code from the CRT-DDS file define_import.xml has been imported in to four SAS CRT-DDS data sets (itemdefs, valuelists, valuelistitemrefs, and itemvaluelistrefs) in the Work library:

```
<def:ValueListDef OID="ValueList.SC.SCTESTCD">
  <ItemRef ItemOID="SC.SCTESTCD.EDLEVEL" OrderNumber="19" Mandatory="No"/>
  <ItemRef ItemOID="SC.SCTESTCD.MARISTAT" OrderNumber="20" Mandatory="No"/>
  <ItemRef ItemOID="SC.SCTESTCD.SUBJINIT" OrderNumber="21" Mandatory="No"/>
</def:ValueListDef>
<ItemDef OID="SC.SCTESTCD" Name="SCTESTCD" DataType="text" Length="8"
  Origin="Assigned" def:Label="Subject Characteristic Short Name">
  <def:ValueListRef ValueListOID="ValueList.SC.SCTESTCD"/>
</ItemDef>
<ItemDef OID="SC.SCTESTCD.EDLEVEL" Name="EDLEVEL" DataType="text"
  Length="24" Origin="CRF Page 6" def:Label="Education Level"/>
<ItemDef OID="SC.SCTESTCD.MARISTAT" Name="MARISTAT" DataType="text"
  Length="8" Origin="CRF Page 6" def:Label="Marital Status"/>
```

```
<ItemDef OID="SC.SCTESTCD.SUBJINIT" Name="SUBJINIT" DataType="text"
      Length="3" Origin="CRF Page 3" def:Label="Subject Initials"/>
```

Display 7.15 SAS CRT-DDS Data Sets Imported from *define_import.xml*

VIEWTABLE: Work.Itemdefs						
	OID	Name	DataType	Length	SDSVarName	Origin
359	SC.SCTESTCD.EDLEVEL	EDLEVEL	text	24		CRF Page 6
360	SC.SCTESTCD.MARISTAT	MARISTAT	text	8		CRF Page 6
361	SC.SCTESTCD.SUBJINIT	SUBJINIT	text	3		CRF Page 3
362	IF.IFCAT.FXCJ.USION.IFTEST	FXCJ.01	text	200		CRF Page 5
						Is pregnant, nursing, or planning to become pregnant within 6 months of last

VIEWTABLE: Work.ValueLists		
	OID	FK_MetaDataVersion
1	ValueList.DA.DATESTCD	CDISC.SDTMIG.3.1.2
2	ValueList.EG.EGTESTCD	CDISC.SDTMIG.3.1.2
3	ValueList.PE.PETESTCD	CDISC.SDTMIG.3.1.2
4	ValueList.SC.SCTESTCD	CDISC.SDTMIG.3.1.2
5	ValueList.VS.VSTESTCD	CDISC.SDTMIG.3.1.2
6	ValueList.IR.IRCAT	CDISC.SDTMIG.3.1.2

VIEWTABLE: Work.ValueListitemrefs				
	ItemOID	OrderNumber	Mandatory	FK_ValueLists
17	PE.PETESTCD.PE08	17	No	ValueList.PE.PETESTCD
18	PE.PETESTCD.PE09	18	No	ValueList.PE.PETESTCD
19	SC.SCTESTCD.EDLEVEL	19	No	ValueList.SC.SCTESTCD
20	SC.SCTESTCD.MARISTAT	20	No	ValueList.SC.SCTESTCD
21	SC.SCTESTCD.SUBJINIT	21	No	ValueList.SC.SCTESTCD
22	VS.VSTESTCD.DIABP	22	No	ValueList.VS.VSTESTCD
23	VS.VSTESTCD.FRMSIZE	23	No	ValueList.VS.VSTESTCD
24	VS.VSTESTCD.HEIGHT	24	No	ValueList.VS.VSTESTCD
25	VS.VSTESTCD.WEIGHT	25	No	ValueList.VS.VSTESTCD

VIEWTABLE: Work.Itemvaluelistrefs		
	ValueListOID	FK_ItemDefs
5	ValueList.PE.PETESTCD	PE.PETESTCD
6	ValueList.QS.QSCAT	QS.QSCAT
7	ValueList.SC.SCTESTCD	SC.SCTESTCD
8	ValueList.SUPPAE.QNAM	SUPPAE.QNAM
9	ValueList.SUPPCM.QNAM	SUPPCM.QNAM
10	ValueList.SUPPDM.QNAM	SUPPDM.QNAM

Special Topic: Identifying Unsupported Elements and Attributes in a CDISC ODM File

Overview

In practice, vendor and custom extensions to ODM are common. For example, Electronic Data Capture (EDC) vendors use data management features and flags that might be exported using ODM XML extensions. By default, such extensions are ignored by the SAS Clinical Standards Toolkit. Recall that the SAS Clinical Standards Toolkit uses XSL style sheets for each of the default, supported 66 ODM data sets (such as *ItemDefs.xsl*). These style sheets look for specifically named tags and hierarchical paths based on the CDISC ODM 1.3.0 published specification. If elements or attributes exist in the XML file but not in the specification, they are ignored.

For example, in this XML code fragment, note the Vendor: *<name>* syntax. This represents a hypothetical extension to the ODM XML, presumably accompanied by a namespace reference supporting the Vendor naming convention.

```

<FormData FormOID=" FormDefs.OID.Death" FormRepeatKey="00-01"
  TransactionType="Remove" Vendor:Revised="No">
  <Vendor:DataQuery DQOID="DQ.OID.001"
    QueryText="Premature report of patients demise?">
    <Flag>Y</Flag>
    <AuditRecord>
      <UserRef UserOID="User.OID.I024" />
      <LocationRef LocationOID="Location.OID.S001" />
      <DateTimeStamp>2011-01-24T15:13:22</DateTimeStamp>
    </AuditRecord>
  </Vendor:DataQuery>
</FormData>

```

In this code fragment, the Vendor:DataQuery syntax specifies a new element with several new attributes and references to other existing (supported) elements. Note also the additional Vendor:Revised attribute for FormData.

The SAS Clinical Standards Toolkit provides a utility macro to parse the ODM XML file to identify currently unsupported elements and tags. This macro, `cstutil_readxmltags`, is located in the primary SAS Clinical Standards Toolkit autocall library (`!sasroot/cstframework/sasmacro`).

Here is an example of a call to the `cstutil_readxmltags` macro:

```

%cstutil_readxmltags(
  _cstxmlfilename=inxml
  ,_cstxmlreporting=Dataset
  ,_cstxmlelementds=work.cstodmelements
  ,_cstxmlattrds=work.cstodmattributes);

```

In this call, the XML file to be parsed is specified with the `inxml` fileref. The results of the parsing are to be written to two data sets, `work.cstodmelements` for all unique elements found in the XML file and `work.cstodmattributes` for all unique attributes found associated each unique element.

The `cstutil_readxmltags` macro parameters are described in this table.

Table 7.18 Parameters for the `cstutil_readxmltags.sas` Macro

Parameter	Required	Description
<code>_cstxmlfilename</code>	Yes	Fileref for input XML file.
<code>_cstxmlreporting</code>	Yes	How results are to be reported. Valid values: Dataset or Results. If Dataset is specified, these two parameters are referenced. If Results is specified, differences detected are reported in the process results data set (as defined by the <code>&_cstResultsDS</code> global macro variable).
<code>_cstxmlelementds</code>	No	Libref.dataset for file elements. Default= <code>work.cstodmelements</code>
<code>_cstxmlattrds</code>	No	Libref.dataset for file attributes. Default= <code>work.cstodmattributes</code>

See the macro header for more details about current assumptions and limitations.

Sample Utility Program: find_unsupported_tags.sas**Overview**

The SAS Clinical Standards Toolkit provides a utility program, `find_unsupported_tags.sas`, to demonstrate assessment of the ODM XML file elements and attributes. This program is located in:

```
!sasroot/../../SASClinicalStandardsToolkitODM130/1.4/sample/
cdisc-odm-1.3.0/programs
```

This program provides the same process setup function supported in most SAS Clinical Standards Toolkit driver modules, building a SASReferences data set that defines process inputs and outputs, and allocating all SAS librefs and filerefs.

Here is the general workflow of this utility program:

1. Build a process-specific SASReferences data set.
2. Call the `%cstutil_processsetup()` macro to set process paths and perform required library and file allocations.
3. Call the `cstutil_readxmltags` macro to create a data set of element names and a data set of attribute names.
4. Compare elements and attributes to a set of known (for example, supported) elements and attributes.
5. Report discrepancies.

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, 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 67](#).

In the SASReferences data set, three input references and one output reference are key to successful completion of the `find_unsupported_tags.sas` utility program. [Table 7.19 on page 220](#) lists these files and data sets, and they are discussed in separate sections.

In the sample `find_unsupported_tags.sas` utility program, these values are set for `&studyRootPath` and `&studyOutputPath`:

```
&studyRootPath=!sasroot/../../
SASClinicalStandardsToolkitODM130/&cstVersion/sample/cdisc-
odm-1.3.0

&studyOutputPath=!sasroot/../../
SASClinicalStandardsToolkitODM130/&cstVersion/sample/cdisc-
odm-1.3.0
```

Table 7.19 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_extended.xml

Input or Output	Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input	standardmetadata(element)	odmmeta	libref		
Input	standardmetadata(attribute)	odmmeta	libref		
Output	results	results	libref	&studyOutputPath/ results	readxmltags_ 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 XML file. The ODM XML file odm_extended.xml contains sample extensions to the core ODM 1.3.0 model.

The metadata type standardmetadata, referenced by the odmmeta SAS libref, references the `<global standards library directory>/standards/cdisc-odm-1.3.0-1.4/metadata` folder. This folder includes the two data sets valid_elements and valid_attributes, which contain the full list of ODM core elements and attributes supported by the SAS Clinical Standards Toolkit. The valid_elements data set contains a single column element itemizing the ODM core elements. The valid_attributes data set contains each attribute within the context of its parent tag and containing element.

This display provides a partial listing of the valid_attributes data set.

Display 7.16 Partial Listing of the valid_attributes Data Set

VIEWTABLE: Odmmeta.Valid_attributes			
	element	parent	attribute
1	AdminData	ODM	StudyOID
2	Alias	ItemDef	Context
3	Alias	ItemDef	Name
4	Alias	ItemGroupDef	Context
5	Alias	ItemGroupDef	Name
6	Annotation	Association	ID
7	Annotation	Association	SeqNum
8	Annotation	Association	TransactionType
9	Annotation	ClinicalData	ID
10	Annotation	ClinicalData	SeqNum
11	Annotation	ClinicalData	TransactionType
12	Annotation	FormData	ID
13	Annotation	FormData	SeqNum
14	Annotation	FormData	TransactionType
15	Annotation	ItemData	ID

Process Outputs

The results type refers to the Results data set that contains information from running the process. In the SAS Clinical Standards Toolkit sample code hierarchy, this information

is written to the `!sasroot/../../SASClinicalStandardsToolkitODM130/1.4/sample/cdisc-odm-1.3.0/results` directory. This location is represented in the utility program by the Results library name.

Depending on the parameter values associated with the call to the `cstutil_readxmltags` macro, two additional process outputs might be persisted at the conclusion of the process. If the `_cstxmlreporting` parameter is set to Dataset, any unsupported elements are documented in the data set referenced by the `_cstxmlelementds` parameter and any unsupported attributes are documented in the data set referenced by the `_cstxmlattrds` parameter.

Process Results

When the utility program finishes running, the `readxmltags_results` data set is created in the Results library. This data set contains informational, warning, and error messages that were generated by the submitted utility program.

This display shows an example of the contents of a Results data set run against the customized `odm_extended.xml` input file (with the `_cstxmlreporting` parameter set to Results).

Display 7.17 Example of a Partial Results Data Set Created by `find_unsupported_tags.sas`

VIEWTABLE: Results.Readxmltags_results									
	resultid	resultseq	seqno	srcdata	message	resultseverity	resultflag	_cst_rc	actual
4	CST0108	1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-	Info	0	0	
5	ODM0900	1	1	C:\Program Files\SAS\SASClinicalStandardsTo	Element found in XML file that is not present in CDISC ODM Model.	Info	1	0	Element = PassCriteria
6	ODM0900	1	2	C:\Program Files\SAS\SASClinicalStandardsTo	Element found in XML file that is not present in CDISC ODM Model.	Info	1	0	Element = SubjectEligibility
7	ODM0900	1	3	C:\Program Files\SAS\SASClinicalStandardsTo	Element found in XML file that is not present in CDISC ODM Model.	Info	1	0	Element = nsdemo:Subject
8	ODM0900	1	4	C:\Program Files\SAS\SASClinicalStandardsTo	Element found in XML file that is not present in CDISC ODM Model.	Info	1	0	Element = nsdemo:SubjectStatus
9	ODM0901	1	1	C:\Program Files\SAS\SASClinicalStandardsTo	Attribute found in XML file that is not present in CDISC ODM Model.	Info	1	0	Parent = Element = ODM Attribute = nsdemo:frameworkversion
10	ODM0901	1	2	C:\Program Files\SAS\SASClinicalStandardsTo	Attribute found in XML file that is not present in CDISC ODM Model.	Info	1	0	Parent = Element = ODM Attribute = nsdemo:productrevision

Chapter 8

Working with CDISC ADaM Data

Overview	223
SAS Representation of CDISC ADaM Metadata	224
ADaM Data Set Templates	230
Validation of ADaM Data Sets	231
Overview	231
Specific Check Implementation Details	232
Unique Validation Properties	232
Validation Check Macros	233
Cross-Standard Validation Checks	233
Sample Data for Validation and Reporting	234
Validation Results	234
Sample Reporting Methodology	236
Overview	236
TLF Metadata	238
Analysis Programs	240
Analysis Results (Tables, Listings, and Figures)	242
Analysis Results Metadata	243

Overview

The SAS Clinical Standards Toolkit provides the following support for the CDISC ADaM 2.1 standard:

- A metadata representation of the CDISC ADaM standard in a set of SAS data sets.
For more information, see [“SAS Representation of CDISC ADaM Metadata” on page 224](#).
- The ability to derive template (zero-observation) data sets for the ADaM subject-level Analysis (ADSL) data set, a representative Basic Data Structure (BDS) data set, and an analysis results data set.
Note: Templates for additional ADaM data structures (such as ADAE) will be provided in future releases after the CDISC ADaM team approves them for use.
- Implementation of version 1.1 of the CDISC ADaM Validation Checks as prepared by the CDISC ADaM team.

- A sample reporting methodology that combines the analysis results metadata with a sample set of tables, listings, and figures (TLF) metadata to create example clinical study reports.

SAS Representation of CDISC ADaM Metadata

The SAS Clinical Standards Toolkit provides a SAS metadata representation of each supported standard. The SAS Clinical Standards Toolkit implementation of the CDISC ADaM 2.1 standard provides an interpretation of Analysis Data Model, Version 2.1, ADaM document and the *Analysis Data Model Implementation Guide, Version 1.0*. The Analysis Data Model identifies four types of ADaM metadata that are captured and supported by the SAS Clinical Standards Toolkit.

The specific sources from the ADaM document for each metadata type are listed:

Table 8.1 ADaM Document Sources for Each Metadata Type

Metadata Type	ADaM Document Source
Analysis Data Set	Section 5.1, Analysis Data Set Metadata, Table 5.1.1
Analysis Variable	Section 5.2, Analysis Variable Metadata, Table 5.2.1
Analysis Parameter	Section 5.2.1, Analysis Parameter Value-Level Metadata
Analysis Results	Section 5.3, Analysis Results Metadata, Table 5.3.1

In the SAS Clinical Standards Toolkit, the Analysis data set metadata is captured in the `reference_tables` and `class_tables` data sets, which are located here:

```
<global standards library directory>/standards/cdisc-  
adam-2.1-1.4/metadata
```

The SAS Clinical Standards Toolkit captures more metadata than might be specified for a standard. This helps support SAS Clinical Standards Toolkit functionality and provides greater consistency across supported standards.

This table provides the mapping of the Analysis data set metadata defined by the CDISC ADaM team to the SAS metadata representation in the `reference_tables` data set:

Table 8.2 Analysis Data Set Metadata

Analysis Data Set Metadata Field**	Description**	reference_tables Column Mapping
DATASET NAME	The file name of the dataset, hyperlinked to the corresponding analysis dataset variable descriptions (i.e., the data definition table) within the define file.	table
DATASET DESCRIPTION	A short descriptive summary of the contents of the dataset	label

Analysis Data Set Metadata Field**	Description**	reference_tables Column Mapping
DATASET LOCATION	The folder and filename where the dataset can be found, ideally hyperlinked to the actual dataset (i.e., XPT file)	xmlpath
DATASET STRUCTURE	The level of detail represented by individual records in the dataset (e.g., “One record per subject,” “One record per subject per visit,” “One record per subject per event”).	structure
KEY VARIABLES OF DATASET	A list of variable names that parallels the structure, ideally uniquely identifies and indexes each record in the dataset.	keys
CLASS OF DATASET	Identification of the general class of the dataset using the name of the ADaM structure (i.e., “ADSL,” “BDS”) or “OTHER” if not an ADaM-specified structure	class
DOCUMENTATION	Description of the source data, processing steps, and analysis decisions pertaining to the creation of the dataset. Software code of various levels of functionality and complexity, such as pseudo-code or actual code fragments may be provided. Links or references to external documents (e.g., protocol, statistical analysis plan, software code) may be used.	documentation

**Source: Version 2.1 of the Analysis Data Model Document, Section 5.1, Analysis Dataset Metadata, Table 5.1.1

The reference_tables data set provided with the SAS Clinical Standards Toolkit 1.4 contains three records for the ADaM ADSL Analysis data set, a representative ADaM BDS data set, and an ADaM analysis results (RESULTS) data set. CDISC ADaM specifies that only the ADSL data set is required. Any number of BDS data sets can be defined as required for each study. A single, optional analysis results data set can be used for each study. [Table 4.4 on page 49](#) lists the column contents for the ADSL data set record in the reference_tables data set.

In the SAS Clinical Standards Toolkit, Analysis Variable metadata is captured in the reference_columns and class_columns data sets in the global standards library folder:

```
<global standards library directory>/standards/cdisc-  
adam-2.1-1.4/metadata
```

This table provides the mapping of Analysis Variable metadata defined by the CDISC ADaM team to the SAS metadata representation in the reference_columns data set:

Table 8.3 Analysis Variable Metadata

Analysis Variable Metadata Field**	Description**	reference_columns Column Mapping
DATASET NAME	The file name of the analysis dataset	table
VARIABLE NAME	The name of the variable	column
VARIABLE LABEL	A brief description of the variable	label

Analysis Variable Metadata Field**	Description**	reference_columns Column Mapping
VARIABLE TYPE	The variable type. Valid values are as defined in the Case Report Tabulation Data Definition Specification Standard (e.g., in version 1.0.0 they include “text,” “integer,” and “float”)	xmldata type
DISPLAY FORMAT	The variable display information (i.e., the format used for the variable in a tabular or graphical presentation of results). It is suggested that the syntax be consistent with the format terminology incorporated in the software package used for analysis (e.g., \$16 or 3.1 if using SAS).	displayformat
CODELIST / CONTROLLED TERMS	A list of valid values or allowable codes and their corresponding decodes for the variable. The field can include a reference to an external codelist (identified by name and version) or a hyperlink to a list of the values in the codelist/controlled terms section of the define file.	xmlcodelist
SOURCE / DERIVATION	Provides details about the variable’s lineage – what was the predecessor, where the variable came from in the source data (SDTM or other analysis dataset) or how the variable was derived. This field is used to identify the immediate predecessor source and/or a brief description of the algorithm or process applied to that source and can contain hyperlinked text that refers readers to additional information. The source / derivation can be as simple as a two level name (e.g., ADSL.AGEGR) identifying the data file and variable that is the source of the variable (i.e., a variable copied with no change). It can be a simple description of a derivation and the variable used in the derivation (e.g., “categorization of ADSL.BMI”). It can also be a complex algorithm, where the element contains a complete description of the derivation algorithm and/or a link to a document containing it and/or a link to the analysis dataset creation program.	sourcederivation

**Source: Analysis Data Model, Version 2.1, ADaM Document, Section 5.2, Analysis Variable Metadata, Table 5.2.1

The reference_columns data set provided with the SAS Clinical Standards Toolkit 1.4 contains one record for each column in each of the three data sets (ADSL, BDS, and RESULTS) in the reference_tables data set. This results in 63 records (columns) for ADSL, 142 records (columns) for BDS, and 13 records (columns) for the RESULTS data set.

Core reference_columns metadata for each column is in the *Analysis Data Model Implementation Guide, Version 1.0*. [Figure 8.1 on page 227](#) provides an excerpt of ADSL column metadata as itemized in Table 3.1.1 of the *Analysis Data Model Implementation Guide, Version 1.0*. This metadata has been translated into the SAS representation of ADSL as shown in [Figure 8.2 on page 227](#). (See also [Table 4.5 on page 49](#), which provides the full set of column metadata for the BDS TRTP column in the reference_columns data set.)

Figure 8.1 ADSL Columns as Specified in the Analysis Data Model Implementation Guide

Variable Name	Variable Label	Type	CodeList / Controlled Term	Core	CDISC Notes
Study Identifiers					
STUDYID	Study Identifier	Char		Req	Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.
USUBJID	Unique Subject Identifier	Char		Req	
SUBJID	Subject Identifier for the Study	Char		Req	
SITEID	Study Site Identifier	Char		Req	
SITEGRy	Pooled Site Group y	Char		Perm	Character description of a grouping or pooling of clinical sites for analysis purposes. For example, SITEGR3 is the name of a variable containing site group (pooled site) names, where the grouping has been done according to the third site grouping algorithm, defined in variable metadata; SITEGR3 does not mean the third group of sites.
SITEGRyN	Pooled Site Group y (N)	Num		Perm	The numeric code for SITEGRy. One-to-one map to SITEGRy.
Subject Demographics					
AGE	Age	Num		Req	The age of the subject is a required variable in ADSL. If the variable is not a copy of DM.AGE, then an additional differently named variable must be added.
AGEU	Age Units	Char	(AGEU)	Req	The units for the subject's age is a required variable in ADSL. If the variable is not a copy of DM.AGEU, then an additional differently named variable must be added.

Figure 8.2 ADSL Columns as Defined in reference_columns Data Set

sasref	table	column	label	order	type	length	displayformat	xmldatatype	xmlcodelist	core	role	term
REFDATA	ADSL	STUDYID	Study Identifier	1	C	40		text		Req	StudyIdentifier	
REFDATA	ADSL	USUBJID	Unique Subject Identifier	2	C	40		text		Req	StudyIdentifier	
REFDATA	ADSL	SUBJID	Subject Identifier for the Study	3	C	40		text		Req	StudyIdentifier	
REFDATA	ADSL	SITEID	Study Site Identifier	4	C	40		text		Req	StudyIdentifier	
REFDATA	ADSL	SITEGRy	Pooled Site Group y	5	C	80		text		Perm	StudyIdentifier	
REFDATA	ADSL	SITEGRyN	Pooled Site Group y (N)	6	N	8		integer		Perm	StudyIdentifier	
REFDATA	ADSL	AGE	Age	7	N	8 8.1		float		Req	SubjectDemographic	
REFDATA	ADSL	AGEU	Age Units	8	C	10		text	AGEU	Req	SubjectDemographic (AGEU)	
sasref	table	column	parameterid	sourcederivation	qualifiers	standard	standardversion	standardref	comment			
REFDATA	ADSL	STUDYID				CDISC-ADAM	2.1		Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.			
REFDATA	ADSL	USUBJID				CDISC-ADAM	2.1		Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.			
REFDATA	ADSL	SUBJID				CDISC-ADAM	2.1		Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.			
REFDATA	ADSL	SITEID				CDISC-ADAM	2.1		Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.			
REFDATA	ADSL	SITEGRy				CDISC-ADAM	2.1		Character description of a grouping or pooling of clinical sites for analysis purposes. For example, SITEGR3 is the name of a variable containing site group (pooled site) names, where the grouping has been done according to the third site grouping algorithm, defined in variable metadata; SITEGR3 does not mean the third group of sites.			
REFDATA	ADSL	SITEGRyN				CDISC-ADAM	2.1		The numeric code for SITEGRy. One-to-one map to SITEGRy.			
REFDATA	ADSL	AGE				CDISC-ADAM	2.1		The age of the subject is a required variable in ADSL. If the variable is not a copy of DM.AGE, then an additional differently named variable must be added.			
REFDATA	ADSL	AGEU				CDISC-ADAM	2.1		The units for the subject's age is a required variable in ADSL. If the variable is not a copy of DM.AGEU, then an additional differently named variable must be added.			

The SAS representation of ADaM analysis metadata in reference_tables and reference_columns provides a study template based on the Data Model Document and the *Analysis Data Model Implementation Guide, Version 1.0*. Each specific study implementation of ADaM creates multiple BDS data sets. The number of data sets is determined by the study design, the statistical analysis plan, and the available source data (for example, SDTM). Each analysis data set (including ADSL) might contain a different subset of columns defined by the CDISC ADaM model.

The SAS implementation makes assumptions about the data type and length of each column. These assumptions represent a typical implementation consistent with SDTM metadata and conventions for specific types of columns. For example, most identifiers have a default length of 40, most flags have a length of 1, and columns using controlled terminology are defined with a length that is long enough to capture the longest controlled term.

A third type of metadata identified in the Analysis Data Model, Version 2.1, ADaM Document (see [Table 8.1 on page 224](#)) is analysis parameter value-level metadata. As noted in the ADaM document:

“Each BDS data set can contain multiple analysis parameters. In a BDS analysis dataset, the variable PARAM contains a unique description for every analysis parameter included in that dataset. Each value of PARAM identifies a set of one or more rows in the dataset. To describe how variable metadata vary by PARAM/PARAMCD, the metadata element PARAMETER IDENTIFIER is required in variable-level metadata for a BDS analysis dataset. This PARAMETER IDENTIFIER metadata element identifies which variables have metadata that vary depending on PARAM/PARAMCD, and links the metadata for a variable to the appropriate value of PARAM/PARAMCD.”

The reference_columns data set contains a column named parameterid that can be used to capture the value-level metadata for BDS data sets. For more information about analysis parameter value-level metadata, see sections 5.2.1 and 5.2.2 of the Analysis Data Model, Version 2.1, ADaM Document.

The final set of metadata prescribed by the Analysis Data Model, Version 2.1, ADaM Document is analysis results metadata. Analysis results metadata is described in the ADaM document:

“These metadata provide traceability from a result used in a statistical display to the data in the analysis data sets. Analysis results metadata are not required. Analysis results metadata describe the major attributes of a specified analysis result found in a clinical study report or submission.”

The metadata fields used to describe an analysis result are listed in [Table 8.4 on page 228](#). In the SAS Clinical Standards Toolkit, these metadata fields are captured in the reference_columns data set (where table='RESULTS'), and serve as a template to initialize an analysis results data set. For more information, see “[ADaM Data Set Templates](#)” on page 230.

Table 8.4 Analysis Results Metadata

Analysis Results Metadata Field**	Description**	reference_columns (value of column where table='RESULTS')
DISPLAY IDENTIFIER	A unique identifier for the specific analysis display (such as a table or figure number)	dispid
DISPLAY NAME	Title of display, including additional information if needed to describe and identify the display (e.g., analysis population)	dispname
RESULT IDENTIFIER	Identifies the specific analysis result within a display. For example, if there are multiple p-values on a display and the analysis results metadata specifically refers to one of them, this field identifies the p-value of interest. When combined with the display identifier provides a unique identification of a specific analysis result.	resultid
PARAM	The analysis parameter in the BDS analysis dataset that is the focus of the analysis result. Does not apply if the result is not based on a BDS analysis dataset.	param

Analysis Results Metadata Field**	Description**	reference_columns (value of column where table='RESULTS')
PARAMCD	Corresponds to PARAM in the BDS analysis dataset. Does not apply if the result is not based on a BDS analysis dataset.	paramcd
ANALYSIS VARIABLE	The analysis variable being analyzed	analvar
REASON	The rationale for performing this analysis. It indicates when the analysis was planned (e.g., "Pre-specified in Protocol," "Pre-specified in SAP," "Data Driven," "Requested by Regulatory Agency") and the purpose of the analysis within the body of evidence (e.g., "Primary Efficacy," "Key Secondary Efficacy," "Safety"). The terminology used is sponsor defined. An example of a reason is "Primary Efficacy Analysis as Pre-specified in Protocol."	reason
DATASET	The name of the dataset used to generate the analysis result. In most cases, this is a single dataset. However, if multiple datasets are used, they are all listed here.	data sets
SELECTION CRITERIA	Specific and sufficient selection criteria for analysis subset and / or numerator– a complete list of the variables and their values used to identify the records selected for the analysis. Though the syntax is not ADaM-specified, the expectation is that the information could easily be included in a WHERE clause or something equivalent to ensure selecting the exact set of records appropriate for an analysis. This information is required if the analysis does not include every record in the analysis dataset.	selcrit
DOCUMENTATION	Textual description of the analysis performed. This information could be a text description, pseudo code, or a link to another document such as the protocol or statistical analysis plan, or a link to an analysis generation program (i.e., a statistical software program used to generate the analysis result). The contents of the documentation metadata element contains depends on the level of detail required to describe the analysis itself, whether or not the sponsor is providing a corresponding analysis generation program, and sponsor-specific requirements and standards. This documentation metadata element will remain free form, meaning it will not become subject to a rigid structure or controlled terminology.	document

Analysis Results Metadata Field**	Description**	reference_columns (value of column where table='RESULTS')
PROGRAMMING STATEMENTS	The software programming code used to perform the specific analysis. This includes, for example, the model statement (using the specific variable names) and all technical specifications needed for reproducing the analysis (e.g., covariance structure). The name and version of the applicable software package should be specified either as part of this metadata element or in another document, such as a Reviewer's Guide (see Appendix B for more information about a Reviewer's Guide).	progstmt

**Source: Analysis Data Model, Version 2.1, ADaM Document, Section 5.3, Analysis Results Metadata, Table 5.3.1

ADaM Data Set Templates

The SAS Clinical Standards Toolkit implementation of the CDISC ADaM 2.1 standard provides metadata templates for creating analysis data sets that conform to the structure prescribed in the *Analysis Data Model Implementation Guide, Version 1.0*. You can use the SAS Clinical Standards Toolkit metadata in the `reference_tables` and `reference_columns` data sets to create these templates.

A framework utility macro, `cst_createTablesForDataStandard`, builds empty ADSL, BDS, and analysis results data sets using the `reference_tables` and `reference_columns` metadata.

Submit this code to create the three data sets:

```
%cst_setstandardproperties(_cstStandard=CST-FRAMEWORK,
_cstSubType=initialize);
%cst_createtablesfordatastandard(_cstStandard=CDISC-ADAM,
_cstStandardVersion=2.1, _cstOutputLibrary=work);
```

The successful creation of the data sets is reported in the SAS log:

```
NOTE: The data set WORK.ADSL has 0 observations and 63 variables.
NOTE: The data set WORK.BDS has 0 observations and 142 variables.
NOTE: The data set WORK.RESULTS has 0 observations and 13 variables.
```

Specifying additional data sets or columns in the global standards library folder results in the macro `cst_createTablesForDataStandard` building a different set of zero-observation data sets. The global standards library folder is located in:

```
<global standards library directory>/standards/cdisc-
adam-2.1-1.4/metadata
```

Validation of ADaM Data Sets

Overview

Validation of CDISC ADaM data sets in the SAS Clinical Standards Toolkit uses the same validation methodology used for other standards. Within the global standards library, registering each standard includes setting the flag `supportvalidation` in the metadata standards data set. All standards that support validation, including ADaM, use the same validation framework and processes described in [Chapter 6, “Validation,”](#) on [page 81](#).

ADaM validation of ADSL and BDS data sets is based on the *CDISC ADaM Validation Checks Version 1.1 Maintenance Release* (dated and released January 21, 2011 to correct errors and remove duplicate checks). This documentation was prepared by the CDISC ADaM team. The version 1.1 release identifies 173 validation checks to be performed. The SAS Clinical Standards Toolkit defines validation checks using a combination of two files:

- the Validation Master data set (located at `<global standards library directory>/standards/cdisc-adam-2.1-1.4/validation/control`)
- the Messages data set (located at `<global standards library directory>/standards/cdisc-adam-2.1-1.4/messages`)

Each of these ADaM data sets has only 159 records. Fourteen validation checks have been combined with other checks by the SAS Clinical Standards Toolkit.

Consider checks 92 and 93:

- 092: There is more than one value of TRTPN for a given value of TRTP.
- 093: There is more than one value of TRTP for a given value of TRTPN.

Checks 92 and 93 are defined and run together as check ADAM0092 because the check macro that is used (`cstcheck_notunique`) checks both conditions by default. The SAS Clinical Standards Toolkit supports all of the checks specified in the version 1.1 release. These checks are listed in [Appendix 7, “CDISC ADaM 2.1 Validation Checks,”](#) on [page 433](#).

The following sections highlight certain aspects of CDISC ADaM validation that are unique or noteworthy.

Specific Check Implementation Details

Implementation details for specific checks are listed in this table:

Table 8.5 CDISC ADaM Validation Check Implementation Details

Check	Details
ADAM0041- ADAM0043	<p>A variable with a suffix of DT, TM, or DTM does not have a SAS Date format.</p> <p>Check metadata codelogic relies on the presence of a nonmissing displayformat value in the column metadata data set. Alternative assessments, such as relying on whether each analysis data set column has an acceptable SAS date-and-time format, or evaluating the values against predetermined formats such as ddmmyy8., are possible.</p>
ADAM0132	<p>R2BASE is not equal to AVAL divided by BASE</p> <p>Implementation uses the round() function with a precision of .001. Changes in the check metadata codelogic might be required if your values are of greater precision.</p>
ADAM0133	<p>R2AyLO is not equal to AVAL divided by AyLO</p> <p>Implementation uses the round() function with a precision of .001. Changes in the check metadata codelogic might be required if your values are of greater precision.</p>
ADAM0134	<p>R2AyHI is not equal to AVAL divided by AyHI</p> <p>Implementation uses the round() function with a precision of .001. Changes in the check metadata codelogic might be required if your values are of greater precision.</p>

Unique Validation Properties

Two validation properties have been added to the SAS Clinical Standards Toolkit to support ADaM validation:

- `_cstParseLengthOverride`

By default, the value is set to **1** and is used only by the SAS Clinical Standards Toolkit framework macro `cstutil_parsescopesegment()` when evaluating the validation check data set fields `tablescope` and `columnscope`. For ADaM validation, it is recommended that this value always be set to **1**.

- `_cstCaseMgmt`

By default, the value is set to **<blank>**. A value of **UPCASE** is also allowed. This property (global macro variable) is used only in the validation check data set field codelogic. For example, consider this codelogic:

```
if (&_cstCaseMgmt(&_cstColumn) not in ("","Y")) then _cstError=1;
```

When `_cstCaseMgmt=UPCASE`, the column value is case insensitive, and the values “y” and “Y” are equivalent. When `_cstCaseMgmt=`, the value “y” is reported as an error.

Validation Check Macros

ADaM validation uses these check macros from the autocall library in the 159 defined checks:

cstcheck_column	cstcheck_crossstdmetamismatch*
cstcheck_columncompare	cstcheck_metamismatch
cstcheck_columnexists*	cstcheck_notincodelist
cstcheck_columnvarlist*	cstcheck_notunique
cstcheck_crossstdcomparedomains*	cstcheck_zeroobs

* These macros are new to the SAS Clinical Standards Toolkit 1.4. They are used only for CDISC ADaM validation, although they are available to all standards.

Note: This list represents a subset of check macros that are available to all standards to be validated.

For information about the purpose and use of each check macro, see [Appendix 3](#), “Macro Application Programming Interface,” on page 279.

Cross-Standard Validation Checks

Six ADaM validation checks require a comparison of ADaM data or metadata with SDTM data or metadata. These checks require the availability of table and column metadata from two different standards. To support this comparison, two new check macros (cstcheck_crossstdcomparedomains and cstcheck_crossstdmetamismatch) are available in the SAS Clinical Standards Toolkit 1.4. Part of the metadata available in the Validation Master data set for the six ADaM cross-standard validation checks is shown in [Figure 8.3 on page 233](#).

Figure 8.3 Partial Metadata for the CDISC ADaM Cross-Standard Validation Checks

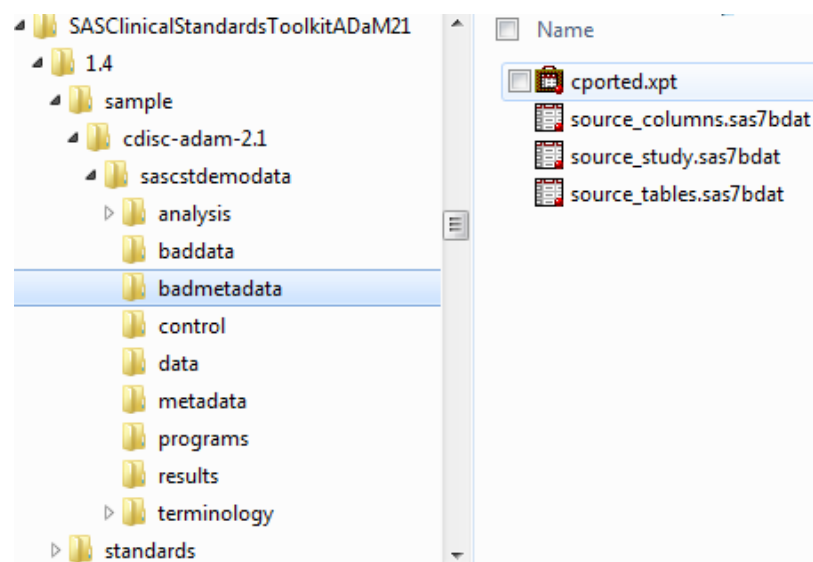
	checkid	codesource	tablescope	columnscope	code logic
2	ADAM0002	cstcheck_crossstdmetamismatch	_ALL_		%let _cstAttr=label;proc sql noprint;create table work._cstProblems as select &_cstStMnemonic.table, &_cstStMnemonic.column, &_cstStMnemonic._cstAttr as &_cstStMnemonic._value, &_cstCrMnemonic._cstAttr as &_cstCrMnemonic._value from work._cstcolumnmetadata &_cstStMnemonic left join work._cstcrosscolumnmetadata &_cstCrMnemonic on upcase(&_cstStMnemonic.column)=upcase(&_cstCrMnemonic.column) where &_cstCrMnemonic.column ne "" and (&_cstStMnemonic._cstAttr ne &_cstCrMnemonic._cstAttr);quit;
3	ADAM0003	cstcheck_crossstdmetamismatch	_ALL_		%let _cstAttr=displayformat;proc sql noprint;create table work._cstProblems as select &_cstStMnemonic.table, &_cstStMnemonic.column, &_cstStMnemonic._cstAttr as &_cstStMnemonic._value, &_cstCrMnemonic._cstAttr as &_cstCrMnemonic._value from work._cstcolumnmetadata &_cstStMnemonic left join work._cstcrosscolumnmetadata &_cstCrMnemonic on upcase(&_cstStMnemonic.column)=upcase(&_cstCrMnemonic.column) where &_cstCrMnemonic.column ne "" and (&_cstStMnemonic._cstAttr ne &_cstCrMnemonic._cstAttr);quit;
4	ADAM0004	cstcheck_crossstdmetamismatch	_ALL_		%let _cstAttr=length;proc sql noprint;create table work._cstProblems as select &_cstStMnemonic.table, &_cstStMnemonic.column, &_cstStMnemonic._cstAttr as &_cstStMnemonic._value, &_cstCrMnemonic._cstAttr as &_cstCrMnemonic._value from work._cstcolumnmetadata &_cstStMnemonic left join work._cstcrosscolumnmetadata &_cstCrMnemonic on upcase(&_cstStMnemonic.column)=upcase(&_cstCrMnemonic.column) where &_cstCrMnemonic.column ne "" and (&_cstStMnemonic._cstAttr ne &_cstCrMnemonic._cstAttr);quit;
51	ADAM0053	cstcheck_crossstdcomparedomains	_ALL_	STUDYID+USUBJID	proc sql noprint;create table work._cstproblems as select &_cstSQLColList from &_cstDSName except select &_cstSQLColList from &_cstCrossDataLib.dn; quit;
56	ADAM0061	cstcheck_crossstdmetamismatch	ADSL	TRTSDT+TRTSDTM	proc sql noprint;select count(distinct column) into _cstColFound from work._cstcolumnmetadata;create table work._cstProblems as select count(table) as exFound, catx(" ", &_cstCrMnemonic.table, 'table found') as &_cstCrMnemonic._value, "&_cstColFound column(s) found" as &_cstStMnemonic._value, "&_cstTableScope" as table, "&_cstColumnScope" as column from work._cstcrosscolumnmetadata (where=table="EX") having exFound=1 and &_cstColFound=0;quit;
156	ADAM0180	cstcheck_crossstdcomparedomains	_ALL_ADSL	SRCDOM	proc sql noprint;create table work._cstproblems as select * from &_cstDSName where SRCDOM not in (select table from work._cstcrosscolumnmetadata);quit;

Sample Data for Validation and Reporting

The SAS Clinical Standards Toolkit implementation of ADaM includes two sets of data and metadata. One set supports the SAS Clinical Standards Toolkit ADaM reporting. In this set, few, if any, data errors and anomalies are included, and this set is considered a clean, analysis-ready set of data. A second set includes illustrative data and metadata errors to demonstrate ADaM validation functionality.

The following figure shows some of the installed SAS files for ADaM, the data and metadata folders that support reporting, and the baddata and badmetadata folders that support validation. The corresponding sample driver modules (analyze_data.sas and validate_data.sas, respectively), which are located in the programs folder (as shown in [Figure 8.4 on page 234](#)) point to the correct source data and metadata folders.

Figure 8.4 Example Folder Hierarchy for a CDISC ADaM Sample Study



Validation Results

The results of an ADaM validation process, as documented in the validation_results data set, are shown in [Figure 8.5 on page 235](#) and [Figure 8.6 on page 235](#). The first 15 records of the data set have been excluded from the display because they report generic process setup and metadata information common to all validation processes.

Records 22 through 24 report the results of one of the cross-standard validation checks. This validation check finds a subject (USUBJID) in the ADaM data sets that was not found in the SDTM DM domain.

Figure 8.5 Results from an ADaM Validation Process (Partial Listing)

	resultid	checkid	resultseq	seqno	srcdata	message	resultseverity	resultflag
16	CST0100	ADAM0001	1	1	SRCDATA.ADSL	No errors detected in SRCDATA.ADSL	Info	0
17	CST0100	ADAM0002	1	1	WORK_CSTCOLUMNMETADATA	No errors detected in source data	Info	0
18	CST0100	ADAM0003	1	1	WORK_CSTCOLUMNMETADATA	No errors detected in source data	Info	0
19	CST0100	ADAM0007	1	1	WORK_CSTCOLUMNMETADATA	No errors detected in source data	Info	0
20	CST0100	ADAM0047	1	1	SRCDATA.ADSL	No errors detected in source data	Info	0
21	CST0100	ADAM0048	1	1	SRCDATA.ADSL	No errors detected in source data	Info	0
22	ADAM0053	ADAM0053	1	1	SRCDATA.ADAE	The values of USUBJID are not present in SDTM.DM	Error	1
23	ADAM0053	ADAM0053	1	2	SRCDATA.ADQS	The values of USUBJID are not present in SDTM.DM	Error	1
24	ADAM0053	ADAM0053	1	3	SRCDATA.ADSL	The values of USUBJID are not present in SDTM.DM	Error	1
25	ADAM0054	ADAM0054	1	1	SRCDATA.ADSL	Within ADSL there is more than one record for a unique value of USUBJID	Error	1
26	CST0100	ADAM0055	1	1	SRCDATA.ADSL	No errors detected in source data	Info	0
27	ADAM0061	ADAM0061	1	1	ADSL TRTSDT+TRTSDTM	SDTM.EX is present and neither TRTSDT or TRTSDTM are present	Error	1
28	ADAM0069	ADAM0069	1	1	SRCDATA.ADSL	A variable with a prefix of TR and containing AG is present and a variable with the same root with a suffix of N is not present	Error	1
29	CST0100	ADAM0070	1	1	WORK_CSTCOLUMNMETADATA	No errors detected in source data	Info	0
30	CST0100	ADAM0090	1	1	SRCDATA.ADAE	No errors detected in source data	Info	0
31	ADAM0090	ADAM0090	1	2	SRCDATA.ADQS	TRTP is not present	Error	0
32	CST0021	ADAM0102	1	1	CSTCHECK_COLUMNVARLIST	Table SRCDATA.ADQS does not contain APERIOD column(s)	Warning: Check not run	-1
33	ADAM0102	ADAM0102	1	108	SRCDATA.ADAE	For every unique xx value of APERIOD in BDS datasets, there is not a ADSL variable TRTxP	Error	1
34	ADAM0138	ADAM0138	1	1	SRCDATA.ADAE	CRITy is populated and CRITyFL is not populated	Error	1
35	ADAM0143	ADAM0143	1	1	SRCDATA.ADAE	PARAMCD has more than 8 characters in length	Error	1
36	ADAM0143	ADAM0143	1	2	SRCDATA.ADQS	PARAMCD has more than 8 characters in length	Error	1
37	CST0021	ADAM0151	1	1	CSTCHECK_COLUMNVARLIST	Table SRCDATA.ADQS does not contain CRIT1 column(s)	Warning: Check not run	-1
38	CST0100	ADAM0151	1	2	SRCDATA.ADAE	No errors detected in source data	Info	0

Figure 8.6 Results from an ADaM Validation Process (Partial Listing—Continued)

	resultid	checkid	_cst_rc	actual	keyvalues	resultdetails
16	CST0100	ADAM0001	0			
17	CST0100	ADAM0002	0			
18	CST0100	ADAM0003	0			
19	CST0100	ADAM0007	0	tableScope=_ALL_columnScope="FN+FL"		
20	CST0100	ADAM0047	0			
21	CST0100	ADAM0048	0			
22	ADAM0053	ADAM0053	0	STUDYID=SASCSTDEMODATA,USUBJID=S999P999	USUBJID=S999P999,AETERM=,AESTDY=,	ADaM IG 1.0 section number: S3.1
23	ADAM0053	ADAM0053	0	STUDYID=SASCSTDEMODATA,USUBJID=S999P999	USUBJID=S999P999,PARAM=,	ADaM IG 1.0 section number: S3.1
24	ADAM0053	ADAM0053	0	STUDYID=SASCSTDEMODATA,USUBJID=S999P999	USUBJID=S999P999	ADaM IG 1.0 section number: S3.1
25	ADAM0054	ADAM0054	0	keys=USUBJID	USUBJID=S999P999	ADaM IG 1.0 section number: S3.1
26	CST0100	ADAM0055	0			
27	ADAM0061	ADAM0061	0	ADAM=0 column(s) found, SDTM=SDTM EX table found		
28	ADAM0069	ADAM0069	0	TRAG1N		
29	CST0100	ADAM0070	0	tableScope=ADSL,columnScope=TR**		
30	CST0100	ADAM0090	0			
31	ADAM0090	ADAM0090	0			
32	CST0021	ADAM0102	0			
33	ADAM0102	ADAM0102	0			All results may not be reported because reportAll=N
34	ADAM0138	ADAM0138	0	CRIT1=2,CRIT1FL=	USUBJID=S999P999,AETERM=HEARTBURN-LIKE DYSPEPSIA,AESTDY=3	ADaM IG 1.0 section number: S3.2.4
35	ADAM0143	ADAM0143	0	PARAMCD=leucocytes	USUBJID=S999P999,AETERM=HEARTBURN-LIKE DYSPEPSIA,AESTDY=3	ADaM IG 1.0 section number: S3.2.4
36	ADAM0143	ADAM0143	0	PARAMCD=leucocytes	USUBJID=S999P999,PARAM=icount	ADaM IG 1.0 section number: S3.2.4
37	CST0021	ADAM0151	0			
38	CST0100	ADAM0151	0			

A partial report of the validation_metrics data set (including a process summary noting that 17 checks were attempted, two could not be run, and 11 errors were detected) is shown in [Figure 8.7 on page 236](#). The two checks that could not be run referenced columns in the check metadata that could not be found or assessed in the source data sets.

Figure 8.7 Metrics from an ADaM Validation Process (Partial Listing)

	metricparameter	reccount	resultid	srcdata	resultseq
32	# of subjects	48	ADAM0138	SRCDATA.ADAE	1
33	# of records tested	106	ADAM0138	SRCDATA.ADAE	1
34	Elapsed time to run check: 0:00:01		ADAM0138	CSTCHECK_COLUMNCOMPARE	1
35	# of subjects	48	ADAM0143	SRCDATA.ADAE	1
36	# of records tested	106	ADAM0143	SRCDATA.ADAE	1
37	# of subjects	1	ADAM0143	SRCDATA.ADQS	1
38	# of records tested	2	ADAM0143	SRCDATA.ADQS	1
39	Elapsed time to run check: 0:00:01		ADAM0143	CSTCHECK_COLUMN	1
40	# of records tested	106	ADAM0151	SRCDATA.ADAE	1
41	Elapsed time to run check: 0:00:01		ADAM0151	CSTCHECK_COLUMNVARLIST	1
42	# of distinct check invocations	17	METRICS	ADAM_VALIDATE	1
43	# check invocations not run	2	METRICS	ADAM_VALIDATE	1
44	Errors (severity=High) reported	11	METRICS	ADAM_VALIDATE	1
45	Warnings (severity=Medium) reported	0	METRICS	ADAM_VALIDATE	1
46	Notes (severity=Low) reported	0	METRICS	ADAM_VALIDATE	1
47	Structural errors, warnings and notes	3	METRICS	ADAM_VALIDATE	1
48	Content errors, warnings and notes	10	METRICS	ADAM_VALIDATE	1

Sample Reporting Methodology

Overview

The primary purpose of the CDISC ADaM standard is to build analysis data sets that support analysis and reporting of clinical research. This purpose, in turn, supports the greater goal of submitting clinical research results to regulatory authorities. These regulatory authorities determine the efficacy and safety of a medical device or product.

The Analysis Data Model, Version 2.1, ADaM Document provides specifications for the structure and content of analysis data sets, and a suggested metadata format for documenting the analysis results generated. Analysis results metadata describe the major attributes of a specified analysis result found in a clinical study report or submission. Analysis results metadata support traceability from an analysis result used in a statistical display to the data in the analysis data sets.

The SAS Clinical Standards Toolkit representation of the ADaM standard includes a sample implementation of an analysis reporting methodology.

Note: This methodology is for illustrative purposes only. Each organization has its own set of processes and workflows that support the generation of a clinical study report or submission. The sample reporting methodology provided with the SAS Clinical Standards Toolkit is intended to be representative of similar industry reporting methodologies. The intent is not to provide a definitive reporting methodology, but to illustrate the interaction of reporting components through the adoption of the ADaM standard.

Key clinical trial reporting components are described in this table.

Table 8.6 Key Clinical Trial Reporting Components

Reporting Component	Comments
Clinical Protocol, Statistical Analysis Plan	Used to identify and define data to be collected, analysis methods and algorithms to be used, and efficacy endpoints and safety measures that determine report output.
Source Data	Source data for analysis data sets, often SDTM. Traceability back to source data is a key ADaM requirement.
Source Metadata	Metadata about the source data.
Controlled Terminology	Set of allowable terms used in any source or analysis data set. For CDISC, NCI EVS serves as the primary source of terms.
Analysis Data Sets	ADaM data sets, typically including the ADSL data set and any number of BDS data sets (for example, ADAE and ADLB) required to support analyses.
Analysis Data Set Metadata	Metadata about the analysis data sets.
Analysis Results (Tables, Listings, and Figures)*	The set of statistical displays (for example, text, tabular, or graphical presentation of results) or inferential statements (such as p-values or estimates of treatment effect).
TLF Metadata (to include table shells)*	Commonly provided as <i>table shells</i> , which provide templates for the statistical displays. Can also include display-specific metadata (often as Microsoft Excel files) used by the analysis programs to generate the displays.
Analysis Results Metadata*	Defined by the Analysis Data Model, Version 2.1, ADaM Document, Section 5.3. For more information, see Table 8.4 on page 228 .
Analysis Programs*	Programming code that uses the analysis data sets (and, optionally, TLF metadata) to create the analysis results.
Submission Package (for example, eCTD)	The structured submission used to package data, metadata, code, and results in a standard form to facilitate review.
Define.xml	A metadata format that documents each tabulation (SDTM) or analysis (ADaM) data set, ancillary documents, and controlled terminology for a study or submission.
CSR/ISS/ISE	The focus of each ADaM implementation. Most commonly a Clinical Study Report (CSR) for a single clinical study. Can be an Integrated Summary of Safety (ISS) or Integrated Summary of Efficacy (ISE) across multiple clinical studies.

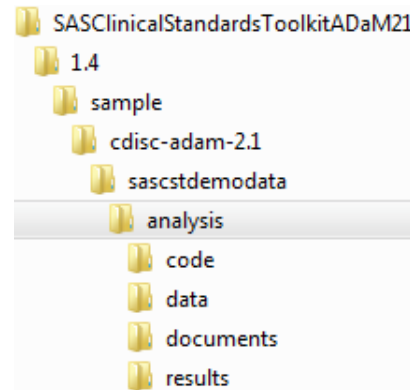
*Component that is discussed later in this section.

The majority of the files supporting the ADaM sample reporting methodology provided with the SAS Clinical Standards Toolkit are located in the ADaM analysis folder:

```
!sasroot/./SASClinicalStandardsToolkitADaM21/1.4/sample/  
cdisc-adam-2.1/sascstdemodata/analysis
```

Here is an illustration of the ADaM analysis folder hierarchy:

Figure 8.8 SAS Clinical Standards Toolkit ADaM Analysis Folder Hierarchy



Here are noteworthy folders:

- The code folder contains the code to create each statistical display. This corresponds to the Analysis Results component described in [Table 8.6 on page 237](#).
- The data folder contains the display-specific metadata noted in the TLF Metadata component of [Table 8.6 on page 237](#).
- The documents folder contains table shells for the TLF Metadata component. For more information about table shells, see “[TLF Metadata](#)” on page 238.
- The results folder contains several sample statistical displays, which correspond to the Analysis Results component.

TLF Metadata

A common industry reporting strategy is to create *table shells* (templates) which specify the output for each statistical display. The SAS Clinical Standards Toolkit provides sample table shells in this file:

```
!sasroot/../../SASClinicalStandardsToolkitADaM21/1.4/sample/
cdisc-adam-2.1/sascstdemodata/analysis/documents/
Mock_tables_shells.pdf.
```

One of these displays, a table reporting patient demographics (Table 14.2.01), follows:

Figure 8.9 SAS Clinical Standards Toolkit Sample Table Shell

Table 14.2.01
Summary of Demographic and Baseline Characteristics
Intent to Treat

		Placebo (N=xxx)	Low Dose (N=xxx)	High Dose (N=xxx)	Total (N = xxx)
		n (%)	n (%)	n (%)	n (%)
Age (Years)	n	xxx	xxx	xxx	xxx
	Mean	xx.x	xx.x	xx.x	xx.x
	STD	x.xx	x.xx	x.xx	x.xx
	Median	xx.x	xx.x	xx.x	xx.x
	Min	xx.x	xx.x	xx.x	xx.x
	Max	xx.x	xx.x	xx.x	xx.x
Age	<30 years	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
	30 – 45 years	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
	>45 years	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
Sex	Female	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
	Male	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
Race	Asian	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
	Black	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
	Caucasian	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
	Hispanic	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)
	Other	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)	xxx (yy.y%)

Produced by SAS Clinical Standards Toolkit at YYYY-MM-DDThh:mm:ss
<program name.sas>

The elements of each table shell (for example, titles, footnotes, headings, column and row labels, cell formatting, and so on) are sometimes captured in a metadata format, often in Microsoft Excel files. The usual intent is to create reporting macros that can generate analysis reports based on this metadata, so that changes in metadata are all that is required to modify and rerun any report.

For the SAS Clinical Standards Toolkit 1.4, sample metadata is included that illustrates the use of such metadata within the ADaM reporting environment.

Note: The sample metadata provided does not represent a full implementation. All metadata fields used in the report examples are not provided.

Supplemental metadata is provided in this file:

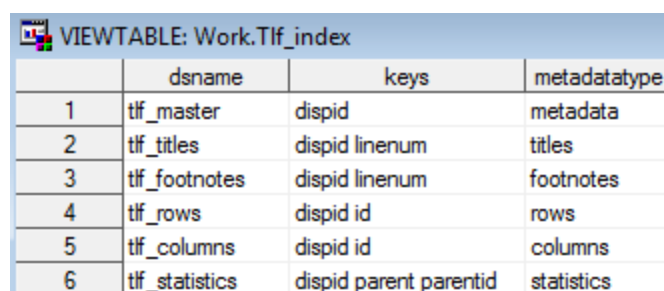
```
!sasroot/./SASClinicalStandardsToolkitADaM21/1.4/sample/
cdisc-adam-2.1/sascstdemodata/metadata/tlfdtdt.xml
```

To interpret this metadata, a sample SAS XML map file (tlfdtdt.map) is provided in the same folder. SAS data sets, representing this XML metadata, are provided in the library of SAS files located in this folder:

```
!sasroot/./SASClinicalStandardsToolkitADaM21/1.4/sample/
cdisc-adam-2.1/sascstdemodata/analysis/data
```

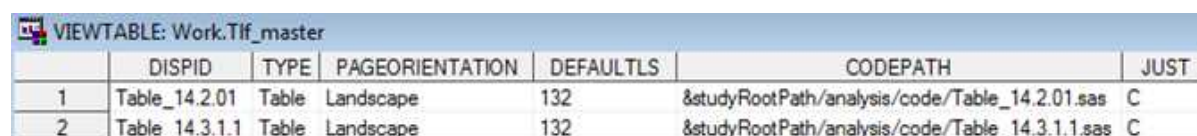
The following figures provide examples of some of the metadata available in the source XML file. This metadata has been extracted into SAS data sets.

Figure 8.10 Sample TLF Metadata: Tlf_index



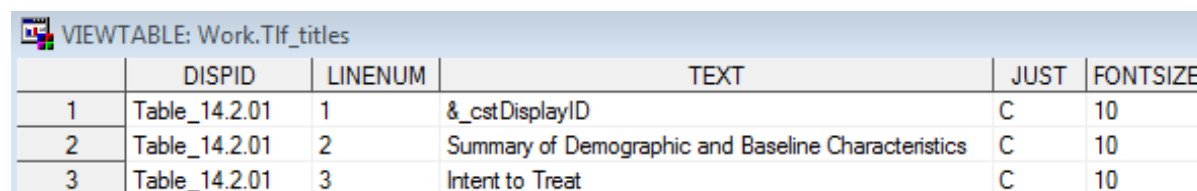
	dsname	keys	metadatatype
1	tlf_master	dispid	metadata
2	tlf_titles	dispid linenum	titles
3	tlf_footnotes	dispid linenum	footnotes
4	tlf_rows	dispid id	rows
5	tlf_columns	dispid id	columns
6	tlf_statistics	dispid parent parentid	statistics

Figure 8.11 Sample TLF Metadata: Tlf_master



	DISPID	TYPE	PAGEORIENTATION	DEFAULTLS	CODEPATH	JUST
1	Table_14.2.01	Table	Landscape	132	&studyRootPath/analysis/code/Table_14.2.01.sas	C
2	Table_14.3.1.1	Table	Landscape	132	&studyRootPath/analysis/code/Table_14.3.1.1.sas	C

Figure 8.12 Sample TLF Metadata: Tlf_titles



	DISPID	LINENUM	TEXT	JUST	FONTSIZE
1	Table_14.2.01	1	&_cstDisplayID	C	10
2	Table_14.2.01	2	Summary of Demographic and Baseline Characteristics	C	10
3	Table_14.2.01	3	Intent to Treat	C	10

Row 1 of the Tlf_master data set describes a centered landscape table and shows where the generating code can be found. The title for that table is provided in the Tlf_titles file. These tables correspond to the table shell titles specified in [Figure 8.9 on page 239](#).

Analysis Programs

The analysis program to generate sample Table 14.2.01 is located in this folder:

```
!sasroot/./SASClinicalStandardsToolkitADaM21/1.4/sample/
cdisc-adam-2.1/sascstdemodata/analysis/code
```

Two versions are provided:

- Table_14.2.01.sas uses the TLF metadata described previously.
- Table_14.2.01_nomd.sas does not rely on TLF metadata to generate the report output.

As noted above, these sample analysis programs do not fully use the sample TLF metadata provided with the SAS Clinical Standards Toolkit. The basic coding strategy adopted with each SAS Clinical Standards Toolkit sample analysis program is to build each section (one or more row combinations) and to concatenate these sections into a single input file used by PROC REPORT.

A sample driver module is provided to perform the process setup, to define (or reference) the sasreferences data set, to perform any required report setup, and to call the

generic ADaM reporting macro `adam_createdisplay()`. This sample driver module is located in this folder:

```
!sasroot/./SASClinicalStandardsToolkitADaM21/1.4/sample/  
cdisc-adam-2.1/sascstdemodata/programs/analyze_data.sas
```

In the sample driver module, a call is made to `adam_createdisplay()` for each analysis report to be produced:

```
%adam_createdisplay (displaysrc=Metadata,useanalysisresults=N,usetlfdtd=Y,  
displayid=%str(Table_14.2.01));
```

To automate this process of creating all analysis reports for a study, it would be necessary to cycle through any available metadata (such as that described in [Figure 8.11 on page 240](#)) to construct multiple calls to the `adam_createdisplay` macro. The `adam_createdisplay` macro header provides an overview of the macro functionality and a summary of the defined macro parameters:

```
adam_createdisplay
```

```
Generate an analysis result from ADaM analysis data sets
```

The basic function of this code module is to create an analysis result display. The path to the code to create the display is provided either directly in the macro parameters or is derived from a metadata source such as the analysis results metadata or Tables, Listings and Figures data definition metadata (TLFDDT) you maintain and reference in the SASReferences data set.

Two primary paths (parameter settings) are supported:

- (1) A code source is specified. A fully qualified path is required. The expectation is that this module will be `%included` below to generate an analysis result (display).
- (2) Metadata is to be used to provide the information necessary to generate an analysis result (display). This metadata will be in the form of the CDISC-ADaM Analysis Results metadata, supplemental Tables, Listings and Figures data definition metadata (TLFDDT), or both.

```
@param displaysrc - Where will information come from to generate result? Values: Code |  
Metadata (default). Required.  
@param displaycode - Either a valid filename or the fully qualified path to code to produce an  
analysis result. Required and used only if displaysrc=Code. All of the remaining parameters  
below are ignored.  
@param useanalysisresults - Should the study-specific analysis results metadata be used to  
provide report metadata? Values: N | Y (default). Either this parameter or usetlfdtd must  
be set to Y if displaysrc=Metadata. If both the useanalysisresults and usetlfdtd parameters  
are set to Y, useanalysisresults will take precedence.  
@param usetlfdtd - Should the study-specific mock tables shells metadata (known here as Tables,  
Listings and Figures data definition metadata (TLFDDT)) be used to provide report metadata?  
Values: N | Y (default). Either this parameter or useanalysisresults must be set to Y if  
displaysrc=Metadata. If both the useanalysisresults and usetlfdtd parameters are set to Y,  
useanalysisresults will take precedence.  
@param displayid - The ID of the display from the designated metadata source. Required if  
displaysrc=Metadata.  
@param displaypath - Either a valid filename or the fully qualified path to the generated  
display. Optional. If not provided, code looks in SASReferences for type=report.
```

The SAS Clinical Standards Toolkit ADaM reporting methodology uses a `report.properties` file to specify the default report format. By default, the property (and global macro variable) `_cstDefaultReportFormat` is set to PDF. Submitting the `analyze_data.sas` driver module produces the specified statistical displays and generates a process results data set. Here is a sample results data set:

Figure 8.13 Sample Results Data Set Generated by the `analyze_data.sas` Driver Module

	resultid	seqno	srcdata	message
1	CST0108	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cst-framework-1.4/programs/initialize.properties
2	CST0102	1	CST_CREATEDS	work.sasreferences was created as requested
3	CST0200	1	CSTUTIL_PROCESSETUP	Process setup is using this SASReferences: C:\Local\Temp\SAS Temporary Files\TD8876_D72672_\sasreferences
4	CST0108	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-adam-2.1-1.4/programs/initialize.properties
5	CST0108	1	CST_SETPROPERTIES	The properties were processed from the PATH Isasroot/./././SASClinicalStandardsToolkitADaM21/1.4/sample/cdisc-adam-2.1/sascstdemodata/programs/report.properties
6	CST0200	1	ADAM_CREATEDISPLAY	PROCESS STANDARD: CDISC-ADAM
7	CST0200	2	ADAM_CREATEDISPLAY	PROCESS STANDARDVERSION: 2.1
8	CST0200	3	ADAM_CREATEDISPLAY	PROCESS DRIVER: ADAM_CREATEDISPLAY
9	CST0200	4	ADAM_CREATEDISPLAY	PROCESS DATE: 2011-07-22T15:04:48
10	CST0200	5	ADAM_CREATEDISPLAY	PROCESS TYPE: REPORTING
11	CST0200	6	ADAM_CREATEDISPLAY	PROCESS SASREFERENCES: work_cstsasrefs
12	CST0200	7	ADAM_CREATEDISPLAY	PROCESS STUDYROOTPATH: Isasroot/./././SASClinicalStandardsToolkitADaM21/1.4/sample/cdisc-adam-2.1/sascstdemodata
13	CST0200	8	ADAM_CREATEDISPLAY	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary
14	CST0200	9	ADAM_CREATEDISPLAY	PROCESS CSTVERSION: 1.4
15	CST0200	10	ADAM_CREATEDISPLAY	PROCESS CONTROLLED TERMINOLOGY SOURCE: c:/cstGlobalLibrary/standards/cdisc-terminology-1.4/cdisc-adam/201101/formats/cterns (Controlled Terminology released by NCI on 2011-01-07)
16	CST0200	11	CSTUTIL_WRITERESULT	No report destination specified so the default report output location has been set to C:\Program Files\SASHome\SASClinicalStandardsToolkitADaM21\1.4\sample\cdisc-adam-2.1\sascstdemodata\analysis\results\Table_14.3.1.1.pdf
17	CST0200	12	ADAM_CREATEDISPLAY	Display location - C:\Program Files\SASHome\SASClinicalStandardsToolkitADaM21\1.4\sample\cdisc-adam-2.1\sascstdemodata\analysis\results\Table_14.3.1.1.pdf

Analysis Results (Tables, Listings, and Figures)

Each generated statistical display should correspond to a table shell, as described in the TLF Metadata section. (See [Figure 8.9](#) on page 239.)

For example, the Summary of Demographic and Baseline Characteristics provided in Table 14.2.01 is shown in this figure:

Figure 8.14 Sample Analysis Report: Table 14.2.01

Table_14.2.01
Summary of Demographic and Baseline Characteristics
Intent to Treat

		Placebo (N=21)	Low Dose (N=23)	High Dose (N=22)	Total (N=66)
		n(%)	n(%)	n(%)	n(%)
Age (Years)	n	21	23	22	66
	Mean	34.9	38.6	37.6	37.1
	STD	10.84	8.52	10.88	10.07
	Median	33	40	37	38
	Min	20	21	18	18
	Max	53	48	54	54
Age	<30 years	8 (38.1%)	5 (21.7%)	5 (22.7%)	18 (27.3%)
	30-45 years	7 (33.3%)	11 (47.8%)	10 (45.5%)	28 (42.4%)
	>45 years	6 (28.6%)	7 (30.4%)	7 (31.8%)	20 (30.3%)
Sex	Female	15 (71.4%)	11 (47.8%)	12 (54.5%)	38 (57.6%)
	Male	6 (28.6%)	12 (52.2%)	10 (45.5%)	28 (42.4%)
Race	Asian	4 (19.0%)	4 (17.4%)	1 (4.5%)	9 (13.6%)
	Black	2 (9.5%)	6 (26.1%)	4 (18.2%)	12 (18.2%)
	Caucasian	9 (42.9%)	9 (39.1%)	11 (50.0%)	29 (43.9%)
	Hispanic	2 (9.5%)	4 (17.4%)	5 (22.7%)	11 (16.7%)
	Other	4 (19.0%)		1 (4.5%)	5 (7.6%)

Produced by SAS Clinical Standards Toolkit at 2011-07-22T15:06:01

!sasroot/.../SASClinicalStandardsToolkitADaM21/1.4/sample/cdisc-adam-2.1/sascstdemodata/analysis/code/Table_14.2.01.sas

Analysis Results Metadata

The Analysis Data Model, Version 2.1, ADaM Document provides specifications for capturing analysis results. As a result, traceability back to the contributing source data is possible. [Table 8.4 on page 228](#) identifies the columns to be included in the analysis results data set. All analysis results metadata for the two statistical displays provided with the SAS Clinical Standards Toolkit is shown in this figure:

Figure 8.15 Analysis Results Metadata

VIEWTABLE: Srcmeta.Analysis_results (ADaM Analysis Results)									
	DISPID	DISPNAME	RESULTID	PARAM	PARAMCD	ANALVAR	REASON	DATASETS	SELCRIT
1	Table_14.2.01	Summary of Demographics and Baseline Characteristics, ITT Population	Age, Sex and Race summaries			multiple	Pre-specified in SAP	ADSL	ITTFL="Y"
2	Table_14.3.1.1	Incidence of Treatment-Emergent Adverse Events by System Organ Class and Preferred Term, Safety Population	Incidence of Treatment-Emergent AEs			multiple	Pre-specified in SAP	ADSL, ADAE	SAFFL="Y"

VIEWTABLE: Srcmeta.Analysis_results (ADaM Analysis Results)				
	DISPID	DOCUMENT	PROGSTMT	XMLTITLE
1	Table_14.2.01	SAP Section 10.1.1. Subject demographics will be summarized for each treatment population. Summary descriptive statistics and column frequencies will be provided. No inferential statistics are planned.	!sasroot/.../SASClinicalStandardsToolkitADaM21/1.4/sample/cdisc-adam-2.1/sascstdemodata/analysis/code/Table_14.2.01_nomd.sas	Table 14.2.01 - Summary of Demographics and Baseline Characteristics, ITT Population
2	Table_14.3.1.1	SAP Section 10.4.2. Treatment emergent adverse events and serious adverse events were summarized by system organ class (SOC) and preferred term (PT). The incidence of treatment emergent events grouped under preferred terms for each active treatment were compared to placebo using Fisher's exact test.	!sasroot/.../SASClinicalStandardsToolkitADaM21/1.4/sample/cdisc-adam-2.1/sascstdemodata/analysis/code/Table_14.3.1.1_nomd.sas	Table 14.3.1.1 - Incidence of Treatment-Emergent Adverse Events by System Organ Class and Preferred Term, Safety Population

The analysis results data set is located at:

```
!sasroot/.../SASClinicalStandardsToolkitADaM21/1.4/sample/
cdisc-adam-2.1/sascstdemodata/metadata/analysis_
results.sas7bdat
```

Chapter 9

Reporting

Sample Reports	245
Overview	245
Process Results Reporting	245
Validation Check Metadata Reporting	255

Sample Reports

Overview

To show how the 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 the 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 the 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). This excerpt from the driver program header provides a brief overview:

```
cst_report.sas
```

```
Sample driver program to perform a primary Toolkit action, in this case,
reporting of process results. This code performs any needed set-up 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, within 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 being 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 file 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 file must exist, and must 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 these 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 9.1 on page 247](#).

TIP 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 this 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.

CAUTION:

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. In the example above, the referenced SASReferences data set points to a !sasroot folder hierarchy that was used for a SAS Clinical Standards Toolkit 1.2 process. This folder hierarchy no longer exists in the SAS Clinical Standards Toolkit 1.4, so the results data set would not be found. This scenario or technique is most appropriate for sites that adopt a consistent means of building and populating SASReferences data sets.

Table 9.1 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: <ol style="list-style-type: none"> 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.

Data or Metadata Source	Scenario 1: Continuation of an Active SAS Session	Scenario 2: Using a Results Data Set from a Previous SAS Session
Metrics	Precedence: <ol style="list-style-type: none"> 1. The data set referenced in <code>&_cstSASRefs</code> with <code>type=results</code> and <code>subtype</code> is either <code>metrics</code> or <code>validationmetrics</code>. 2. The data set referenced by <code>&_cstMetricsDS</code>. 	The data set referenced in <code>&_cstSASRefs</code> with <code>type=results</code> and <code>subtype</code> is either <code>metrics</code> or <code>validationmetrics</code> .
Validation_Control	The data set referenced in <code>&_cstSASRefs</code> with <code>type=control</code> and <code>subtype=validation</code> .	The data set referenced in <code>&_cstSASRefs</code> with <code>type=control</code> and <code>subtype=validation</code> .
Messages	<code>&_cstMessages</code> used by the prior task.	<code>&_cstMessages</code> built by a call to <code>%cstutil_allocatesasreferences</code> .

Note: Beginning 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](#).

This code is 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 is 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 246](#), would include these 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 report 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 this code:

```
%cstutil_createreport(_cstsasreferencesdsset=&_cstSASRefs,_cstreportbydomain=Y,
_cstreportoutput=&studyrootpath/results/cstchecktablereport.pdf);
```

This table describes all supported parameters in the sample `%cstutil_createreport` macro.

Table 9.2 Supported Parameters for the `%cstutil_createreport` Macro

Parameter	Description
<code>_cstsasreferencesdsset</code>	The libref.dataset of SASReferences data set used for a specific process. This parameter is optional. If it is specified, then <code>_cstresultsdsset</code> and <code>_cstmetricsdsset</code> parameters are ignored. Either <code>_cstsasreferencesdsset</code> or <code>_cstresultsdsset</code> must be provided.
<code>_cstresultsdsset</code>	The libref.dataset of the SAS Clinical Standards Toolkit process Results data set. This parameter is optional. Either <code>_cstsasreferencesdsset</code> or <code>_cstresultsdsset</code> must be provided. This parameter is ignored if <code>_cstsasreferencesdsset</code> is specified.
<code>_cstmetricsdsset</code>	The libref.dataset of the SAS Clinical Standards Toolkit process Metrics data set. This parameter is optional. This parameter is ignored if <code>_cstsasreferencesdsset</code> is specified.
<code>_cstreporterrorsonly</code>	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 <code>results.resultflag=1</code>).
<code>_cstreportobs</code>	If null (default), then this parameter reports all records in error (where <code>results.resultflag=1</code>) in the Results data set. Otherwise, set this parameter to any integer value > 0 , signifying the number of records to print per checkid (where <code>results.checkid</code> is non-null). If <code>_cstreportobs > 0</code> excludes any records, then a footnote is printed, noting that not all records were printed.

Parameter	Description
<code>_cstreportbytable</code>	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).
<code>_cstablechecksdsset</code>	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 97 . This parameter is optional. If this parameter is not used, then the data set is created.
<code>_cstablecheckscodes</code>	Report 2 parameter. The code module (macro) to build <code>_cstablechecksdsset</code> if it does not exist, or is not passed as a parameter. This parameter is required only if <code>_cstreportbytable=Y</code> and <code>_cstablechecksdsset</code> is not provided.
<code>_cstkeeptablechecklist</code>	Report 2 parameter. The value is Y or N (default). If running report 2, then keep the derived list of tables (<code>_cstablechecklist</code>) to reuse in subsequent report requests. Building this file takes awhile.
<code>_csttablesubset</code>	Report 2 parameter. This parameter is optional. It produces a report based on a specific table, indicated by <code>libref.data set</code> . If the value is blank or the keyword <code>_ALL_</code> is specified, then all tables are included in the report. This parameter is ignored if <code>_cstreportbytable=N</code> .
<code>_cstreportoutput</code>	The path and filename where report output is to be written. File types HTML, RTF, and PDF are supported. This parameter is required.
<code>_cstsummaryReport</code>	The value is Y (default) or N. If set to Y, then generate the report summary panel.
<code>_cstioReport</code>	The value is Y (default) or N. If set to Y, then generate the process inputs and outputs panel.
<code>_cstmetricsReport</code>	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.
<code>_cstgeneralResultsReport</code>	The value is Y (default) or N. If set to Y, then generate the general process reporting panel.
<code>_cstcheckIdResultsReport</code>	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 this macro call:

```
%cstutil_createreport(
  _cstsasreferencesdsset=&_cstSASRefs,
  _cstresultsdsset=&_cstRptResultsDS,
```

```

_cstmetricsdsset=&_cstRptMetricsDS,
_cstreportbytable=N,
_cstreporterroronly=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.

These displays show report content. The displays apply to report 1 (by checkid) unless otherwise indicated.

Display 9.1 Example of Report Summary

SAS Clinical Standards Toolkit 1.4 CDISC-SDTM 3.1.2 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	2011-07-25T12:01:25
Report only errors, warnings & notes?	Yes
# records to report	50
Report results by table	No
Report output file	Isasroot/././SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata/results/cstreport.pdf

Display 9.2 Example of Process Inputs and Outputs

SAS Clinical Standards Toolkit 1.4 CDISC-SDTM 3.1.2 VALIDATION

Process Inputs/Outputs

Type	Path
Autocall Libraries	(sdtmcode sasautos)
	sdtmcode: c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.2-1.4/macros
Format Search Path Libraries	(srcfmt cstfmt)
	srcfmt: Isasroot/././SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata/terminology/formats
	cstfmt: c:/cstGlobalLibrary/standards/cdisc-terminology-201003-1.4/formats
Reference Metadata	c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.2-1.4/metadata
Source Data	Isasroot/././SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata/data
Source Metadata	Isasroot/././SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata/metadata

Process Metrics

Summary Metrics			Check Metrics			
Metric	#	Check ID	# Check Invocations	# Recs (if available)	# Errors	# Check Invocations Not Run
# of distinct check invocations	13	SDTM0011	1	1	1	0
# check invocations not run	0	SDTM0012	1	1	1	0
Errors (severity=High) reported	0	SDTM0013	1	1	1	0
Warnings (severity=Medium) reported	10	SDTM0014	1	1	1	0
Notes (severity=Low) reported	0	SDTM0015	1	10	10	0
Structural errors, warnings and notes	85	SDTM0019	1	10	10	0
Content errors, warnings and notes	0	SDTM0020	1	10	10	0
		SDTM0022	1	10	10	0
		SDTM0023	1	10	10	0
		SDTM0030	1	10	10	0
		SDTM0031	1	10	10	0
		SDTM0032	1	10	10	0
		SDTM0033	1	1	1	0
Note: "# Check Invocations Not Run" includes both checks that did not run and checks that failed to complete successfully.						

Display 9.4 Example of Process Metrics by Domain (Report 2)

SAS Clinical Standards Toolkit 1.4
CDISC-SDTM 3.1.2 VALIDATION

Process Metrics

Summary Metrics		Table Metrics				
Metric	#	Table	# Check Invocations	# Recs (if available)	# Errors	# Check Invocations Not Run
# of distinct check invocations	13	AE	13	13	0	0
# check invocations not run	0	CE	13	13	0	0
Errors (severity=High) reported	0	CM	13	13	0	0
Warnings (severity=Medium) reported	10	CO	13	13	0	0
Notes (severity=Low) reported	0	DA	13	13	0	0
Structural errors, warnings and notes	85	DM	13	13	0	0
Content errors, warnings and notes	0	DS	13	13	0	0
# of distinct check invocations	13	DV	13	13	0	0
# check invocations not run	0	EG	13	13	0	0
Errors (severity=High) reported	0	EX	13	13	0	0
Warnings (severity=Medium) reported	20	FA	13	13	0	0
Notes (severity=Low) reported	0	IE	13	13	0	0
Structural errors, warnings and notes	170	LB	13	13	0	0
Content errors, warnings and notes	0	MB	13	13	0	0
		MH	13	13	0	0
		MS	13	13	0	0
		PC	13	13	0	0
		PE	13	13	0	0
		PP	13	13	0	0
		QS	13	13	0	0
		RELREC	13	13	0	0
		SC	13	13	0	0
		SE	13	13	0	0
		SU	13	13	0	0

Report generated 2011-07-26T11:35:59 on process run 2011-07-26T11:31:56

Display 9.5 Example of General Process Reporting

**SAS Clinical Standards Toolkit 1.4
CDISC-SDTM 3.1.2 VALIDATION**

General Process Reporting

Seq #	Source Data	Result Identifier	Severity	Problem Detected?	Message
1	CST_SETPROPERTIES	CST0108	Info	No	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cst-framework-1.4/programs/initialize.properties
1	CST_CREATEDS	CST0102	Info	No	work.sasreferences was created as requested
1	CSTUTIL_PROCESSETUP	CST0200	Info	No	Process setup is using this SASReferences: C:\DOCUME~1\geligh\LOCALS~1\Temp\SAS Temporary Files_TD5228_WILLIS2_sasreferences
1	CST_SETPROPERTIES	CST0108	Info	No	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.2-1.4/programs/initialize.properties
1	CST_SETPROPERTIES	CST0108	Info	No	The properties were processed from the PATH Isasroot/.J./SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-3.1.2/sascstdemodata/programs/validation.properties
1	SDTM_VALIDATE	CST0200	Info	No	PROCESS STANDARD: CDISC-SDTM
2	SDTM_VALIDATE	CST0200	Info	No	PROCESS STANDARDVERSION: 3.1.2
3	SDTM_VALIDATE	CST0200	Info	No	PROCESS DRIVER: SDTM_VALIDATE
4	SDTM_VALIDATE	CST0200	Info	No	PROCESS DATE: 2011-07-25T12:01:25
5	SDTM_VALIDATE	CST0200	Info	No	PROCESS TYPE: VALIDATION
6	SDTM_VALIDATE	CST0200	Info	No	PROCESS SASREFERENCES: C:\DOCUME~1\geligh\LOCALS~1\Temp\SAS Temporary Files_TD5228_WILLIS2_sasreferences.sas7bdat

Display 9.6 Example of Validation Results by CheckID (Report 1)

**SAS Clinical Standards Toolkit 1.4
CDISC-SDTM 3.1.2 VALIDATION**

Process Results, CheckID: SDTM0015

Description: Identifies a column that appears in the SAS dataset but is not listed in the domain description

Check scope: (Tables) _ALL_, (Columns)

Source: WebSDM (IR5254)

Validation check macro: cstcheck_metamismatch, using source metadata

Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
1	1	SRCDATA.SUPPAE	SDTM0015	Variable IDVAR appears in dataset but is not in SDTM standard	Warning	Yes		
1	2	SRCDATA.SUPPAE	SDTM0015	Variable IDVARVAL appears in dataset but is not in SDTM standard	Warning	Yes		
1	3	SRCDATA.SUPPAE	SDTM0015	Variable QEVAL appears in dataset but is not in SDTM standard	Warning	Yes		

Display 9.7 Example of Validation Results by Domain (Report 2)

**SAS Clinical Standards Toolkit 1.4
CDISC-SDTM 3.1.2 VALIDATION**

Process Results, Table: SUPPAE

Check ID	Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
SDTM0011	1	1	SRCDATA.SUPPAE	CST0025	Data set not found in reference standard - compliance not assessed	Warning: Check incomplete	Yes		

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`). This excerpt from the driver program header provides a brief overview:

```
cst_metadatareport.sas
```

Sample driver program to perform reporting of validation check metadata. This code performs any needed set-up and data management tasks, followed by one or more calls to the `%cstutil_createmetadatareport()` macro to generate report output.

Two scenarios for invoking this routine are addressed in this driver module:

- (1) This code is run as a natural continuation of a CST process, within the same SAS session, with all required files available. The working assumption is that the SASReferences data set (`&_cstSASRefs`) exists and contains information on all files required for reporting.
- (2) This code is being 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 found, it will be used. If 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 will be written to `cstcheckmetadatareport.pdf` in the SAS WORK library unless another location is specified in SASReferences or in the set-up code below.
- (7) The report macro `cstutil_createmetadatareport` will only produce 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 this code:

```

%cstutil_createmetadatareport (
    _cstValidationDS=&_cstRptControl
    ,_cstMessagesDS=&_cstMessages
    ,_cstReportOutput=%bquote (&_cstRptOutput)
) ;

```

This table describes all supported parameters in the sample %cstutil_createmetadatareport macro:

Table 9.3 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 you.
_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 this 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 on page 256](#), produces a validation check metadata report (cstcheckmetadatareport.pdf) that contains all four report sections for the CDISC SDTM 3.1.1 validation checks.

Display 9.8 Example of Check Overview

**SAS Clinical Standards Toolkit 1.4
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	***	Janus	IR4250	Identifies a column that was described in the domain description but not included in the SAS dataset for that domain	Note	_ALL_	
	***	WebSDM	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 9.9 Example of Additional Check Details (Panel 2) [*cstCheckMDReport=Y*]

SAS Clinical Standards Toolkit 1.4
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	Janus	Metadata	cstcheck_metamismatch	Yes	proc sql noprint; create table _cstNonMatch as select sasref, b1 table from (select distinct table from work._cstsrccolumnmetadata except select distinct table from work._cstrefcolumnmetadata) b1 left join (select distinct sasref, table from work._cstsrccolumnmetadata) b2 on b1.table=b2.table; create table _csttemps1 as select ref.table, ref.column from work._cstrefcolumnmetadata ref left join (select distinct table from work._cstsrccolumnmetadata) src on ref.table=src.table where ref.table=src.table; create table _cstMatch as select base.table, base.column from _csttemps1 base except select upcase(comp.table) as table, upcase(comp.column) as column from work._cstcolumnmetadata comp; quit; %let _cstRptLevel=TABLE;			Active	Yes
	WebSDM	Metadata	cstcheck_metamismatch	Yes	proc sql noprint; create table _cstNonMatch as select sasref, b1 table from (select distinct table from work._cstsrccolumnmetadata except select distinct table from work._cstrefcolumnmetadata) b1 left join (select distinct sasref, table from work._cstsrccolumnmetadata) b2 on b1.table=b2.table; create table _csttemps1 as select ref.table, ref.column from work._cstrefcolumnmetadata ref left join (select distinct table from work._cstsrccolumnmetadata) src on ref.table=src.table where ref.table=src.table; create table _cstMatch as select base.table, base.column from _csttemps1 base except select upcase(comp.table) as table, upcase(comp.column) as column from work._cstcolumnmetadata comp; quit; %let _cstRptLevel=TABLE;			Active	Yes

Report generated: 2019-03-29 11:28:29
 Report source: (Folder) c:\sas\Global\Library\standards\cdisc-sdtm-3.1.1\validation\control (data set) validation_master

Display 9.10 Example of Message Details (Panel 3) [*cstMessageReport=Y*]

SAS Clinical Standards Toolkit 1.4
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			
	WebSDM	SDTM expected variable &_cstparm1 not found			

Display 9.11 Example of Reference Information (Panel 4) [_cstSTDRefReport=Y]

SAS Clinical Standards Toolkit 1.4
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 Rules	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.

Display 9.12 Example of Using WHERE Clause [_cstValidationDSWhClause=checkid='SDTM0801']

SAS Clinical Standards Toolkit 1.4
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 9.13 Example of by Record View [_cstRecordView=Y]

SAS Clinical Standards Toolkit 1.4
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

Overview	261
CST-Framework initialize.properties	262
CDISC ADaM 2.1 initialize.properties	264
CDISC ADaM 2.1 validation.properties	265
CDISC ADaM 2.1 report.properties	267
CDISC CRTDDS 1.0 initialize.properties	267
CDISC CRTDDS 1.0 validation.properties	268
CDISC ODM 1.3.0 initialize.properties	269
CDISC ODM 1.3.0 validation.properties	270
CDISC SDTM (3.1.1 and 3.1.2) initialize.properties	272
CDISC SDTM (3.1.1 and 3.1.2) validation.properties	272
General Purpose (not set in properties files)	274

Overview

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_cleanupepstsession 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).

CST-Framework initialize.properties

Global Macro Variable	Values	Comments
	* default value	
_cstDebug	0 (off)* 1 (on)	If on, then _cstDebugOptions are set. Many files remain in the Work library at process conclusion. <i>Note:</i> When _cstDebug=1, the size of the SAS log is significantly larger.
_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.

Global Macro Variable	Values * default value	Comments
_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.
_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. If you switch standards in the same SAS session, it is important that you set this variable to 1.
_cstFMTLibraries	<blank>* ** work.fmt (example)	This variable enables you 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.
_cstCTRoot	<blank>	The root directory where the CTERMS.SAS7BCAT resides. This value is set based on which CT is the default.
_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 SASReferences data set is located in the SAS Work library (signified by &workpath). Use of &workpath is not required.

Global Macro Variable	Values	Comments
	* default value	
_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.
_cstCTVersion	NCI_THESAURUS*	The version of the Controlled Terminology (CT) dictionary. Currently, this is a unique value.

CDISC ADaM 2.1 initialize.properties

Global Macro Variable	Values	Comments
	* default value	
_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.

* default value

CDISC ADaM 2.1 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. You 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.
_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.

Global Macro Variable	Values * default value	Comments
<code>_cstMetricsNumStructural</code>	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 <code>checktype="Metadata"</code> . This excludes informational records in the Results data set. The value is ignored if <code>_cstMetrics=0</code> .
<code>_cstMetricsNumContent</code>	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 <code>checktype ^= "Metadata"</code> . This excludes informational records in the Results data set. The value is ignored if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumSubj</code>	0*	Actual count of the number of subjects that were tested. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumRecs</code>	0*	Actual count of the number of records that were tested. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumChecks</code>	0*	Actual count of the number of validation checks that were run. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumBadChecks</code>	0*	Actual count of the number of check invocations that were not run. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumErrors</code>	0*	Actual count of the number of errors that were reported. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumWarnings</code>	0*	Actual count of the number of warnings that were reported. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumNotes</code>	0*	Actual count of the number of notes that were reported. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumStructural</code>	0*	Actual count of the number of structural errors that were reported. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstMetricsCntNumContent</code>	0*	Actual count of the number of content errors that were reported. The value is not calculated if <code>_cstMetrics=0</code> .
<code>_cstParseLengthOverride</code>	0 (off) 1 (on)*	Allows the SAS Clinical Standards Toolkit validation process to interpret a <code>tableScope</code> value like <code>**DATA</code> as ANY table ending in DATA, regardless of table name length. If 1, this signals the code to ignore the length of the <code>_cstVarName</code> when building the WHERE clause against the work._csttablemetadata data set (this parameter is currently only used by <code>cstutil_parsetablescope</code>).

Global Macro Variable	Values	Comments
	* default value	
_cstCaseMgmt	<blank> UPCASE	Used to assess the value of a variable when case is important in the code logic field of the validation data set. A blank _cstCaseMgmt property value tells the SAS Clinical Standards Toolkit to do nothing to values when making value comparisons. The check would report the use of lowercase values as errors. A _cstCaseMgmt property value=UPCASE tells the SAS Clinical Standards Toolkit that both lowercase and uppercase values are considered equivalent, and that values should be uppercase in any validation logic. Lowercase values would not be considered to be in error.

* default value

CDISC ADaM 2.1 report.properties

Global Macro Variable	Values	Comments
	* default value	
_cstDefaultReportFormat	pdf*, rtf, html	Sets the report destination, if the report destination is not provided as a report parameter.

* default value

CDISC CRTDDS 1.0 initialize.properties

Global Macro Variable	Values	Comments
	* default value	
_cstSubjectColumns	_none_*	The standard-specific set of columns that identify a subject. Since CDISC CRTDDS does not have subject columns this value is set to _NONE_.
_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.

* default value

CDISC CRTDDS 1.0 validation.properties

Global Macro Variable	Values * default value	Comments
_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. cstmtrics*	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.
_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.

Global Macro Variable	Values * default value	Comments
_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.
_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.
* default value		

CDISC ODM 1.3.0 initialize.properties

Global Macro Variable	Values * default value	Comments
_cstSubjectColumns	_none_*	The standard-specific set of columns that identify a subject. Since CDISC ODM does not have subject columns this value is set to _NONE_.
_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.

Global Macro Variable	Values * default value	Comments
_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.

* default value

CDISC ODM 1.3.0 validation.properties

Global Macro Variable	Values * default value	Comments
_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.
_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.

Global Macro Variable	Values * default value	Comments
_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.
_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.

* default value

CDISC SDTM (3.1.1 and 3.1.2) initialize.properties

Global Macro Variable	Values * default value	Comments
_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.

* default value

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. You 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.

Global Macro Variable	Values * default value	Comments
_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.
_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. <i>Note:</i> This variable might be deprecated in a future release.
* default value		

General Purpose (not set in properties files)

Global Macro Variable	Values * default value	Comments
_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.
studyOutputPath	Example: C:\Study1\output	This variable is optional. It defines the location of generated output. 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 output locations, which facilitates portability.

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
CST0020	Info	Check run but nonmissing codeLogic is not used
CST0021	Warning: Check not run	Table &_cstparm1 does not contain &_cstparm2 column(s)
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

Result ID	Check Severity	Message Text
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
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

Result ID	Check Severity	Message Text
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 or subtype &_cstParm1 is not defined for &_cstParm2
CST0088	Error	These columns of &_cstParm1 cannot be empty: &_cstParm2
CST0089	Error	Only libraries are supported for this operation
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 before 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
CST0117	Error	No template data set found for type=&_cstParm1, subtype=&_cstParm2 in the registered data standards

Result ID	Check Severity	Message Text
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
CST0201	Warning	&_cstParm1
CST0202	Error	&_cstParm1

Note: Not all message data set fields are displayed.

Appendix 3

Macro Application Programming Interface

Module ADaM V2.1 (Run Time)	279
Overview	279
Macro Detail	280
Module CRT-DDS V1.0 (Run Time)	283
Overview	283
Macro Detail	284
Module Framework	292
Overview	292
Macro Detail	298
Module SDTM V3.1.1 (Run Time)	336
Overview	336
Macro Summary	336
Macro Detail	337
Module SDTM V3.1.2 (Run Time)	342
Overview	342
Macro Summary	342
Macro Detail	343
Module ODM V1.3.0 (Run Time)	347
Overview	347
Macro Summary	347
Macro Detail	348

Module ADaM V2.1 (Run Time)

Overview

This is the CDISC ADaM 2.1 run-time macro library.
 Since: V1.4

Table A3.1 Module ADaM V2.1 (Run Time) Macro Summary

Exposure	Macro
External ADAM Reporting	<pre>%macro adam_createdisplay (displaysrc=Metadata,displaycode=, useanalysisresults=,usetlfdt=,displayid=,displaypath=) /des='CST: Create ADaM analysis result';</pre> <p>Check: See “%adam_createdisplay” on page 280</p>
External ADAM Validation Process	<pre>%macro adam_validate /des='CST: Validate CDISC ADAM model files';</pre> <p>Check: See “%adam_validate” on page 281</p>
Internal ADAM Utility	<pre>%macro adamutil_buildcheckdomainlist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=) / des="CST: Build table list by checkid";</pre> <p>Check: See “%adamutil_buildcheckdomainlist” on page 281</p>
External ADAM Tool	<pre>%macro adamutil_createsrcmetafromsaslib /des='CST: Create ADAM metadata from SAS library';</pre> <p>Check: See “%adamutil_createsrcmetafromsaslib” on page 282</p>
External ADAM Reporting	<pre>%macro adamutil_gettlfmetadata(_cstOutLib= WORK, _cstMetaDataType=);</pre> <p>Check: See “%adamutil_gettlfmetadata” on page 282</p>

Macro Detail

%adam_createdisplay

```
%adam_createdisplay (displaysrc=Metadata,displaycode=, useanalysisresults=,
usetlfdt=, displayid=, displaypath=) /des='CST: Create ADaM analysis result';
```

[Exposure: external] [Macro Type: ADAM Reporting]

Generate an analysis result from ADaM analysis data sets.

The basic function of this code module is to create an analysis result display. The path to the code to create the display is provided either directly in the macro parameters or is derived from a metadata source. Examples of metadata sources are analysis results metadata or tables, listings, and figures data definition metadata (TLFDDT) that you maintain and reference in the SASReferences data set.

Two primary paths (parameter settings) are supported:

1. A code source is specified. A fully qualified path is required. The expectation is that this module will be %included below to generate an analysis result (display).
2. Metadata provides the information necessary to generate an analysis result (display). This metadata is in the form of the CDISC ADaM analysis results metadata, supplemental tables, listings, and figures data definition metadata (TLFDDT), or both.

Parameters:

- (Required) **displaysrc**: Specifies where information comes from to generate the result. Valid values are: Code | Metadata (default).

- **displaycode:** Specifies either a valid filename or the fully qualified path to the code that produces an analysis result. If **displaysrc=Code**, this parameter is used and is required. All of the remaining parameters below are ignored.
- **useanalysisresults:** Specifies whether the study-specific analysis results metadata are used to provide report metadata. Valid values are: N | Y (default). Either this parameter or **usetlfdtdt** must be set to Y if **displaysrc=Metadata**. If both **useanalysisresults** and **usetlfdtdt** parameters are set to Y, **useanalysisresults** takes precedence.
- **usetlfdtdt:** Specifies whether the study-specific mock table shell's metadata (known as tables, listings, and figures data definition metadata (TLFDDT)) are used to provide report metadata. Valid values are: N | Y (default). Either this parameter or **useanalysisresults** must be set to Y if **displaysrc=Metadata**. If both **useanalysisresults** and **usetlfdtdt** parameters are set to Y, **useanalysisresults** takes precedence.
- **displayid:** The ID of the display from the designated metadata source. This parameter is required if **displaysrc=Metadata**.
- (Optional) **displaypath:** Either a valid filename or the fully qualified path to the generated display. If not provided, the code looks in SASReferences for **type=report**.

File: adam_createdisplay.sas

%adam_validate

%adam_validate /des='CST: Validate CDISC ADAM model files';

[Exposure: external] [Macro Type: ADAM Validation Process]

Validate CDISC ADAM model files.

The basic function of this code module is to cycle through the validation checks to be run and to write validation results to the process results and metrics data sets. These results are persisted to any permanent location based on **type=results** records in SASReferences. 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: adam_validate.sas

%adamutil_buildcheckdomainlist

%adamutil_buildcheckdomainlist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=) /
des="CST: Build table list by checkid";

[Exposure: internal] [Macro Type: ADAM 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 containing the checks for any 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: **VAR1=1**, **uppercase(var2)="Y"**, or **checkstatus>0**.

- `_cstOutputDS`: The output data set returned to the calling program. This data set contains a record for each domain referenced by any checkid, standardversion, and checksource.

File: adamutil_buildcheckdomainlist.sas

%adamutil_createsrcmetafromsaslib

`%adamutil_createsrcmetafromsaslib /des='CST: Create ADAM metadata from SAS library';`

[Exposure: external] [Macro Type: ADAM Tool]

This sample utility macro attempts to derive source metadata files from a SAS library for a CDISC ADAM study.

These source metadata files are used by the SAS Clinical Standards Toolkit to support CDISC ADaM validation and derivation of CDISC CRT-DDS (define.xml) files: `source_tables`, `source_columns`, and `source_study`.

This is the general strategy:

1. Use PROC CONTENTS output as the primary source of information.
2. Use `reference_tables` and `reference_columns` for matching columns.

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). If it is not set, data set keys are estimated based on columns that SAS uses to define keys in the reference standard.
- 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 values are marked as `<--- HARDCODE`.

Limitations: Only those listed above.

File: adamutil_createsrcmetafromsaslib.sas

%adamutil_gettlfmetadata

`%adamutil_gettlfmetadata(_cstOutLib=WORK,_cstMetaDataType=);`

[Exposure: external] [Macro Type: ADAM Reporting]

Module to read TLF metadata from the sources defined in SASReferences that point to the TLF XML file and a SAS XML map file to render that XML metadata in SAS.

Assumptions:

- This module references a specific XML representation of the mock table shell's metadata. Alternative representations of the metadata require modifications to this macro to read that metadata content.
- The implementation of the sample ADaM reports provided with the SAS Clinical Standards Toolkit is intended to provide a sample workflow of report generation within the context of the SAS Clinical Standards Toolkit framework. You should use alternative means to capture and use metadata to support the reports to be submitted.

Parameters:

- (Optional) `_cstOutLib`: Library for SAS representation of TLF metadata. The default is `Work`.
- (Optional) `_cstMetaDataType`: Identifies a specific type of metadata. Example: `tlf_master`. If this parameter is not specified, all available metadata is created in `_cstOutLib`.

File: `adamutil_gettlfmetadata.sas`

Module CRT-DDS V1.0 (Run Time)

Overview

This is the CDISC CRT-DDS 1.0 run-time macro library.

Since: V1.2

Table A3.2 Module CRT-DDS V1.0 (Run Time) Macro Summary

Exposure	Macro
	<code>%crtdds_clitemdecodetrans(_cstsourcestudy=, _cstsourcecolumns=, _cstcodelistitemsds=, _cstmdvDS=, _cststudyds=, _cstcodelistsds=, _cstCLlang=, _cstoutclitemdecodetransds=);</code>
	<code>%crtdds_codelistitems(_cstsourcecolumns=, _cstcodelistsds=, _cstoutcodelistitemsds=);</code>
	<code>%crtdds_codelists(_cstsourcecolumns=, _cstmdvds=, _cstmdvname=, _cstoutcodelistsds=);</code>
	<code>%crtdds_computationmethods(_cstsourcecolumns=, _cstsourcestudy=, _cstmdvds=, _cstitemdefsds=, _cststudyds=, _cstoutcomputationmethodsds=);</code>
	<code>%crtdds_defineddocument(_cstname=, _cstdescr=, _cstoutdefinedocds=);</code>
	<code>%crtdds_getstatic(_cstName=, _cstVar=);</code>
	<code>%crtdds_itemdefs(_cstsourcecolumns=, _cstsourcestudy=, _cststudyds=, _cstmdvds=, _cstcodelistsDS=, _cstoutitemdefsds=, _cstoutitemdefsds2=);</code>
	<code>%crtdds_itemgroupdefitemrefs(_cstsourcecolumns=, _cstsourcetables=, _cstsourcestudy=, _cstitemdefsds2=, _cstmdvds=, _cstitemgroupdefsds=, _cststudyds=, _cstoutitemgroupdefitemrefsds=);</code>
	<code>%crtdds_itemgroupdefs(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupdefsds=);</code>
	<code>%crtdds_itemgroupleaf(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupleafds=);</code>
	<code>%crtdds_itemgroupleaftitles(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupleaftitlesds=);</code>

Exposure	Macro
	<code>%crtdds_metadataversion(_cstname=, _cstdescr=, _cststandard=, _cstversion=, _cstdefineversion=, _cststudyds=, _cststudyname=, _cstoutmdvds=);</code>
External CRTDDS	<code>%crtdds_read;</code>
External CRTDDS	<code>%crtdds_sdtm311todefne10(_cstOutLib=, _cstSourceTables=, _cstSourceColumns=, _cstSourceStudy=);</code> <i>Note: Deprecated.</i>
External CRTDDS	<code>%crtdds_sdtmtodefne(_cstOutLib=, _cstSourceTables=, _cstSourceColumns=, _cstSourceStudy=);</code>
	<code>%crtdds_study(_cstname=, _cstdescr=, _cstprotocol=, _cstdefineds=, _cstdefinename=, _cstoutstudyds=);</code>
External CRTDDS Validation Process	<code>%crtdds_validate /des='CST: Validate CDISC CRTDDS model files';</code>
External CRT-DDS	<code>%crtdds_write(_cstCreateDisplayStyleSheet=1, _cstOutputEncoding=, _cstHeaderComment=, _cstResultsOverrideDS=, _cstLogLevel=info);</code>
External CRTDDS	<code>%crtdds_xmlvalidate(_cstLogLevel=info, _cstResultsOverrideDS=);</code>
Internal Framework Utility	<code>%crtddsutil_buildchecktablelist(_cstCheckDS=, _cstWhereClause=, _cstOutputDS=);</code>

Macro Detail

`%crtdds_clitemdecodetrans`

```
%crtdds_clitemdecodetrans(_cstsourcestudy=, _cstsourcecolumns=,
_cstcodelistitemsds=, _cstmdvDS=, _cststudyds=, _cstcodelistds=, _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: `crtds_definedocument.sas`

`%crtds_getstatic`

`%crtds_getstatic(_cstName=, _cstVar=);`

[Exposure: Not specified] [Macro type: Not specified]

Parameters:

- `_cstName`
- `_cstVar`

File: `crtds_getstatic.sas`

`%crtds_itemdefs`

`%crtds_itemdefs(_cstsourcecolumns=, _cstsourcestudy=, _cststudyds=, _cstmdvds=, _cstcodelistsDS=, _cstoutitemdefsds=, _cstoutitemdefsds2=);`

[Exposure: Not specified] [Macro type: Not specified]

Parameters:

- `_cstsourcecolumns`
- `_cstsourcestudy`
- `_cststudyds`
- `_cstmdvds`
- `_cstcodelistsDS`
- `_cstoutitemdefsds`
- `_cstoutitemdefsds2`

File: `crtds_itemdefs.sas`

`%crtds_itemgroupdefitemrefs`

`%crtds_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
- _cstoutitemgroupleafds

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
- _cstoutitemgroupleaftitlesds

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: The location of an existing Results data set. If not specified, work._cstResults is used.

File: crtdds_metadataversion.sas

%crtdds_sdtmtodefine

```
%crtdds_sdtmtodefine(_cstOutLib=, _cstSourceTables=, _cstSourceColumns=,
_cstSourceStudy=);
```

This macro replaces %crtdds_sdtm311todefine10 in the SAS Clinical Standards Toolkit 1.4.

[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.

These 12 CRT-DDS tables are created:

- clitemdecodetranslatedtext
- codelistitems

- codelists
- computationmethods
- definedocument
- itemdefs
- itemgroupdefitemrefs
- itemgroupdefs
- itemgroupleaf
- itemgroupleaftitles
- metadataversion
- study

The metadata source is specified in SASReferences.

Required global macro variables:

- framework initialization properties
- CRT-DDS 1.0 initialization properties
- _cstresultsds: The location of an existing results data set.

Parameters:

- (Required) _cstOutLib: The library reference where the resulting tables should be written to.
- (Required) _cstSourceTables: A data set that contains the SDTM metadata for the domains to be included in the CRT-DDS file.
- (Required) _cstSourceColumns: A data set that contains the SDTM metadata for the domain columns to be included in the CRT-DDS file.
- (Required) _cstSourceStudy: A data set that contains the metadata for the studies to be included in the CRT-DDS file.

File: crtdds_sdtmtodefine.sas

%crtdds_sdtm311todefine10

Note: Deprecated.

```
%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. These 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: The location of an existing Results data set.

Parameters:

- (Required) _cstOutLib: Identifies library reference where the resulting tables should be written to.
- (Required) _cstSourceTables: A data set that contains the SDTM metadata for the domains to be included in the CRT-DDS file.
- (Required) _cstSourceColumns: A data set that contains the SDTM metadata for the domain columns to be included in the CRT-DDS file.
- (Required) _cstSourceStudy: 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: The location of an existing Results data set, or it should override this value using the _cstResultsOverrideDS parameter to this macro

Parameters:

- (Optional) _cstCreateDisplayStyleSheet: Specifies 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.
- (Optional) _cstOutputEncoding: The XML encoding to use for the CRT-DDS file that is created.
- (Optional) _cstHeaderComment: 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.
- (Optional) _cstResultsOverrideDS: 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.
- (Optional) _cstLogLevel: 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:

- (Optional) `_cstLogLevel`: The level of error reporting. Valid values are Info, Warning, Error, and Fatal Error.
- (Optional) `_cstResultsOverrideDS`: 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.
- (Optional) `_cstWhereClause`: A WHERE clause to subset `_cstCheckDS`. The syntax should comply with a SAS statement argument, such as: `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 primary SAS Clinical Standards Toolkit Framework run-time autocall macro library.

Table A3.3 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 Framework	%cst_getstandardsubtypes(_cstStandard=CDISC-TERMINOLOGY, _cstOutputDS=, _cstResultsDS=);
External standard_name	%cst_getstandardmetadata(_cstSASReferences=, _cstResultsOverrideDS=); Deprecated
External Framework	%cst_getstandardsasreferences(_cstStandard=, _cstStandardVersion=, _cstOutputDS=, _cstResultsOverrideDS=);
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=);

Exposure	Macro
External Framework	<code>%cst_unregisterstandard(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);</code>
External Framework	<code>%cst_unsetproperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);</code>
External Validation Check	<code>%cstcheck_column(_cstControl=);</code>
External Validation Check	<code>%cstcheck_columncompare(_cstControl=);</code>
External Validation Check	<code>%cstcheck_columnexists (_cstControl=);</code>
External Validation Check	<code>%cstcheck_columnvarlist (_cstControl=);</code>
External Validation Check	<code>%cstcheck_comparedomains(_cstControl=);</code>
External Validation Check	<code>%cstcheck_crossstdcomparedomains(_cstControl=);</code>
External Validation Check	<code>%cstcheck_crossstdmetamismatch(_cstControl=);</code>
Internal System Check	<code>%cstcheck_java();</code>
External Validation Check	<code>%cstcheck_dsmismatch(_cstControl=);</code>
External Validation Check	<code>%cstcheck_metamismatch(_cstControl=);</code>
External Validation Check	<code>%cstcheck_notconsistent(_cstControl=);</code>
External Validation Check	<code>%cstcheck_notimplemented(_cstControl=);</code>
External Validation Check	<code>%cstcheck_notincodelist(_cstControl=);</code>
External Validation Check	<code>%cstcheck_notsorted(_cstControl=);</code>

Exposure	Macro
External Validation Check	%cstcheck_notunique(_cstControl=);
External Validation Check	%cstcheck_recmismatch(_cstControl=);
External Validation Check	%cstcheck_recnofound(_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_allocclobalmetadatalib(_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_buildddomlist(_cstFormatType=DATASET, _cstDomWhere=, _cstDomDSName=&_cstTableMetadata, _cstSubOverride=N);
External Utility	%cstutil_buildformatsfromxml(_cstFmtLib=, _cstReplaceFmtCat=Y, _cstFmtCatPrefix=, _cstFmtCatLang=, _cstFmtCatLangOption=English);
Internal Framework Check	%cstutil_checkds(_cstdsname=, _csttype=, _cstsubtype=, _cststandard=*, _cststandardversion=*);
Internal Framework Utility	%cstutil_cleanupcstsession(_cstClearCompiledMacros=0, _cstClearLibRefs=0, _cstResetSASAutos=0, _cstResetFmtSearch=0, _cstResetSASOptions=1, _cstDeleteFiles=1, _cstDeleteGlobalMacroVars=0);

Exposure	Macro
External Framework Utility	%cstutil_createmetadatatereport(_cstStandardTitle=, _cstValidationDS=, _cstValidationDSWhClause=, _cstMessagesDS=, _cstStdRefDS=, _cstReportOutput=, _cstCheckMDReport=N, _cstMessageReport=N, _cstStdRefReport=N, _cstRecordView=N);
External Framework Utility	%cstutil_createreport(_cstsasreferencesdset=, _cstresultsdset=&_cstRptResultsDS, _cstmetricsdset=&_cstRptMetricsDS, _cstreporterroronly=N, _cstreportobs=, _cstreportbytable=N, _csttablechecksds=, _csttablecheckscodes=, _cstkeepablechecklist=N, _csttablesubset=, _cstreportoutput=, _cstsummaryReport=Y, _cstioReport=Y, _cstmetricsReport=Y, _cstgeneralResultsReport=Y, _cstcheckIdResultsReport=Y);
Internal Framework Utility	%cstutil_createsubdir(_cstSubDir=);
Internal Framework Utility	%cstutil_createsublists();
Internal Framework	%cstutil_createtempmessages(_cstCreationFlag=);
Internal Framework Utility	%cstutil_createunixsubdir(_cstSubDir=);
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=, _cstsource=, _cstsublistnum=);

Exposure	Macro
Internal Framework Utility	%cstutil_parsescopesegment(_cstPart=, _cstVarName=, _cstMessageID=CST0004);
Internal Framework Utility	%cstutil_parsetablescope(_cstscopestr=, _cstopsource=, _cstsublistnum=);
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";
Internal Framework Utility	%cstutil_readxmltags(_cstxmlfilename=inxml, _cstxmlreporting=Results, cstxmlelementds=work.cstodmelements, _cstxmlattrds=work.cstodmattributes);
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);

Exposure	Macro
Automatically generated by the CST-Framework post-installation configuration component	<code>%cstutil_setcstgroot;</code>
Internal Framework Utility	<code>%cstutil_setmodel /des="Set Which Model Definition to Use";</code>
Internal CDISC CRT-DDS	<code>%cstutil_writecubexml(_cstXMLOut=, _cstMDPFile=, _cstDebug=);</code>
Internal Framework Utility	<code>%cstutil_writemetric(_cstMetricParameter=, _cstResultID=, _cstResultSeqParm=, _cstMetricCnt=, _cstSrcDataParm=, _cstMetricsDSParm=&_cstMetricsDS);</code>
Internal CDISC-CRT-DDS	<code>%cstutil_writeodmcubexml(_cstXMLOut=);</code>
Internal Framework Utility	<code>%cstutil_writeresult(_cstResultID=, _cstValCheckID=, _cstResultParm1=, _cstResultParm2=, _cstResultSeqParm=1, _cstSeqNoParm=1, _cstSrcDataParm=, _cstResultFlagParm=0, _cstRCParm=0, _cstActualParm=, _cstKeyValuesParm=, _cstResultDetails=, _cstResultsDSParm=&_cstResultsDS);</code>

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:

- (Required) `_cstStandard`: The name of a registered standard.
- (Optional) `_cstStandardVersion`: 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.
- (Required) `_cstType`: The type of data set to be created. This value comes from the TYPE column in the SASReferences file for the standard-version combination.
- (Optional) `_cstSubType`: 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.
- (Required) `_cstOutputDS`: The name of the data set to be created.
- (Optional) `_cstResultsOverrideDS`: 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.

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:

- (Required) _cstStandard: The name of a registered standard.
- (Optional) _cstStandardVersion: 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.
- (Required) _cstOutputLibrary: The LIBNAME in which the table shells should be created.

- (Optional) `_cstResultsOverrideDS`: 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:

- (Required) `_cstPropertiesLocation`: The location of the property file. The format depends on the value of `_cstLocationType`.
- (Required) `_cstLocationType`: 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).
- (Optional) `_cstResultsOverrideDS`: 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_getstandardssubtypes`

```
%cst_getstandardssubtypes(_cstStandard=CDISC-TERMINOLOGY, _cstOutputDS=,
_cstResultsDS=);
```

[Exposure: external] [Macro Type: framework]

Generates a data set containing the installed Clinical Terminology subtypes (for example, SDTM, CDASH, ADAM, any user-customizations).

Parameters:

- (Required) `_cstStandard`: The name of a registered standard.
- `_cstOutputDS`: The libname.memname of the data set to create.
- `_cstResultsDS`: The results of the creation process.

File: `cst_getstandardssubtypes.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.

Parameters:

- (Required) **_cstSASReferences**: The (LIBNAME.)member that refers to a valid SASReferences file.
- (Optional) **_cstResultsOverrideDS**: 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:

- (Required) **_cstStandard**: The name of a registered standard.
- (Optional) **_cstStandardVersion**: 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.
- (Required) **_cstOutputDS**: The (LIBNAME.)member name of the output data set to be created.
- (Optional) **_cstResultsOverrideDS**: 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:

- (Optional) `_cstSASReferences`: 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.
- (Required) `_cstOutputDS`: The output data set to create that contains the completed information.
- `_cstAddRequiredCSTRefs`
- (Optional) `_cstResultsOverrideDS`: 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`

`%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:

- (Required) `_cstPropertiesLocation`: The location of the property file. The format depends on the value of `_cstLocationType`.
- (Required) `_cstLocationType`: 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).
- (Optional) `_cstResultsOverrideDS`: 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 you. For example, `CST_FRAMEWORK` provides a property subType of 'required' that points to a property file that has default settings for required properties. You can call this method using this code to set these properties:

```
%cst_setstandardproperties(
_cstStandard=CST_FRAMEWORK,
_cstStandardVersion=1.2,
_cstSubType=required);
```

Parameters:

- (Required) `_cstStandard`: The name of a registered standard.
- (Optional) `_cstStandardVersion`: Specified that the standard has a default set. Otherwise, it is mandatory. This specifies the version of the standard.
- (Required) `_cstSubType`: The name of the properties subtype that is to be read and from where properties are set.
- (Optional) `_cstResultsOverrideDS`: 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:

- (Required) `_cstPropertiesLocation`: The location of the property file. The format depends on the value of `_cstLocationType`.
- (Required) `_cstLocationType`: 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).
- (Optional) `_cstResultsOverrideDS`: 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_columnexists

%cstcheck_columnexists (_cstControl=) /des='CST: Does column exist?';

[Exposure: external] [Macro Type: Validation Check]

cstcheck_columnexists

Determines whether one or more of the columns defined in columnScope exist in each of the tables defined in tableScope.

Note: By default, this check does not require the use of codeLogic. If the check metadata includes a non-null value of codeLogic, it will be used. If codeLogic is used, it must populate the macro variable _cstDataRecords with a count that represents the number of columns found in the specific table defined by _cstDSName_cstDomainOnly. A _cstDataRecords count of 0 will be reported as an error.

Note: Care must be exercised when columnScope contains either multiple columns (for example, TRTP+TRTPn) or a column when using a wildcard (for example, TRT**P). In both cases, the default code will report an error ONLY if NONE of the columns are found.

Note: This is a metadata-only check against column and table metadata files. No source data sets are referenced.

Example validation checks that use this macro:

- ADAM0090: (for BDS data sets) Does the column TRTP exist?

Parameters:

- `_cstControl`: The single observation data set containing check-specific metadata.

File: `cstcheck_columnexists.sas`

%cstcheck_columnvarlist

`%cstcheck_columnvarlist (_cstControl=) /des='CST: Column varlist processing';`

[Exposure: external] [Macro Type: Validation Check]

`cstcheck_columnvarlist`

Supports comparison of multiple columns within the same data set or across multiple data sets.

Note: As a general rule, this macro expects a check metadata `columnScope` syntax of `{_cstList:var1+var2+var3...varn}` for within-data set assessments and `{_cstList:var1...varn} {_cstList:var1...varn}` for multi-data set assessments.

Note: Macro requires use of `_cstCodeLogic` at a DATA step level (for example, a full DATA step or PROC SQL invocation). `_cstCodeLogic` creates a work file (`_cstproblems`) containing records in error. `_cstCodeLogic` must handle any data set joins when multiple data sets are involved in the column comparisons.

Example validation checks that use this macro:

- ADAM0152 - (for BDS data sets) BASE is populated and BASE is not equal to AVAL where ABLFL is equal to "Y" for a given value of PARAM and BASETYPE.

Parameters:

- `_cstControl`: The single observation data set containing check-specific metadata.

%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_crossstdcomparedomains

%cstcheck_crossstdcomparedomains(_cstControl=) /des='CST: Cross-std Matching column values not found';

[Exposure: external] [Macro Type: Validation Check]

This macro generally compares values for 1+ columns in one table against either those same columns in another domain in another standard, or compares values against metadata from the comparison standard.

Note: This macro requires the use of _cstCodeLogic as a full DATA step or PROC SQL invocation. This DATA step or PROC SQL invocation 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.

Example validation checks that use this macro include:

- ADaM subject not found in SDTM DM domain.
- ADaM SDTM domain reference (for traceability) is found, but SDTM domain is unknown.

Required global macro variables: (none)

Parameters:

- _cstControl: The single-observation data set that contains check-specific metadata.

File: cstcheck_crossstdcomparedomains.sas

%cstcheck_crossstdmetamismatch

%cstcheck_crossstdmetamismatch(_cstControl=) /des='CST: Cross-std metadata inconsistencies';

[Exposure: external] [Macro Type: Validation Check]

This macro identifies inconsistencies between metadata across registered standards.

Note: This macro requires the use of _cstCodeLogic as a full DATA step or PROC SQL invocation. This DATA step or PROC SQL invocation 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 are available to assess compliance.
- _cstProblems includes two columns (or more if needed):
 - &_cstMnemonic._value (for example, ADaM_value containing the value of the column of interest from the primary standard)
 - &_cstCRMnemonic._value (for example, SDTM_value containing the value of the column of interest from the comparison standard)

The mnemonics are from the global standards library Standards data set.

Required macro variables:

- _cstcrossstd: The name of the comparison standard.
- _cstcrossstdver: The version of the comparison standard.
- _cstrunstd: The name of the primary standard.
- _cstrunstdver: The version of the primary standard.

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_crossstdmetamismatch.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_java`

`%cstcheck_java()/des='CST: Were there Java issues in the DATA step?';`

[Exposure: internal] [Macro Type: System Check]

This macro determines whether any Java or related to Java issues exist in the previous DATA step. This macro must be called immediately after the DATA step that declares the Java object.

Note: This macro will be deprecated in future releases of the SAS Clinical Standards Toolkit. It will be replaced in the SAS Clinical Standards Toolkit 1.5 with the `csutil_checkjava` macro.

The following Java issues and issues related to Java are caught:

No Java installed:

- ERROR: The Java proxy is not responding.
- ERROR: The Java proxys JNI call to start the VM failed.
- ERROR: Could not create Java VM.
- SYSERR 0
- SYSERRORTEXT Could not create Java VM.

No picklist:

- ERROR: The Java picklist file was not found.
- ERROR: Could not initialize classpath from picklist file.
- SYSERR 0
- SYSERRORTEXT Could not initialize classpath from picklist file.

Edit picklist has the wrong content:

- ERROR: File(s) are missing from the Java repository.
- ERROR: Could not initialize classpath from picklist file.
- SYSERR 0
- SYSERRORTEXT Could not initialize classpath from picklist file.

Missing JAR file:

- ERROR: Could not find class com/sas/ptc/transform/xml/StandardXMLTransformerParams at line 3 column 22. Please ensure that the CLASSPATH is correct.
- ERROR: DATA STEP Component Object failure. Aborted during the EXECUTION phase.
- SYSERR 1012
- SYSERRORTEXT DATA STEP Component Object failure. Aborted during the EXECUTION phase

Required global macro variables: (none)

Required file inputs: (none)

File: cstcheck_java.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 invocation 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 you do 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 your responsibility code correct joins and lookup logic in code logic.

When LOOKUPTYPE=CODELIST, functionality is deferred for the 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 you do 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 this style:

```
[LBTEST+VSTEST][LBTESTCD+VSTESTCD]
```

These 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.

Here 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_recnofound`

`%cstcheck_recnofound(_cstControl=);`

[Exposure: external] [Macro type: Validation Check]

`cstcheck_recnofound`

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 these 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: The location of the SASReferences input file.
- _cstSASRefsName: The name of the SASReferences input file.
- _cstSASRefs: The Work library version of SASReferences.
- _cstFMTLibraries: Specifies whether to include Work and Library in fmtsearch.
- _cstMessageOrder: Specifies whether to (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.
- (Optional) _cstStdRef: Reference in standard supporting checks.
- (Optional) _cstOrderBy: 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`: The name of the data set with column metadata returned when `_cstFormatType=DATASET`.
- `_cstDomDSName`: The name of the 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: The name of the 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_buildddomlist.sas

%cstutil_buildformatsfromxml

```
%cstutil_buildformatsfromxml(_cstFmtLib=, _cstReplaceFmtCat=Y,
_cstFmtCatPrefix=, _cstFmtCatLang=, _cstFmtCatLangOption=English) /des='CST:
Build formats from xml codelists';
```

[Exposure: external] [Macro Type: Utility]

This utility macro is designed for use with CDISC XML-based standards such as CRTDDS and ODM. Those standards capture acceptable values in codelists. This module reads that codelist information to create one or more SAS format catalogs, based on the xml:lang language tags.

This macro is called by the odm_read and crtdds_read macros.

Parameters:

- (Optional) `_cstFmtLib`: The location where catalogs are written. If this parameter is not specified, the default value is first derived from SASReferences, then WORK.
- (Optional) `_cstReplaceFmtCat`: Specifies whether an existing format catalog by the same name in `_cstFmtLib` is replaced. Values: N | Y. Default behavior: Y (overwrite existing catalog).
- (Optional) `_cstFmtCatPrefix`: The prefix to use for catalog names. If this parameter is not specified, the default value is <standard mnemonic>FmtCat (such as ODMFmtCat). This default produces an English format catalog name of ODMFmtCat_en.
- (Optional) `_cstFmtCatLang`: If this parameter is specified, it creates a format catalog only for the specified language. Example: `_cstFmtCatLang=en`. If no records exist for the specified language, an empty catalog is created.
- (Optional) `_cstFmtCatLangOption`: Specifies the action to take when no language tag is provided in the XML. Values: Ignore | English | Use_cstFmtCatLang.
 - If the value is Ignore, records are ignored (but they are reported in the SAS log).
 - If the value is English, records are added to the English catalog (default).
 - If the value is Use_cstFmtCatLang, records are added to the language catalog specified in the `_cstFmtCatLang` parameter.

Assumptions:

- If EnumeratedItems are encountered, these items are added to each language-specific format catalog that is created.

File: cstutil_buildformatsfromxml.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:

- (Required) `_cstdsname`: The two-level name of the data set to validate.
- (Required) `_csttype`: The type of data set to be created. This value comes from the TYPE column in the SASReferences file for the standard-version combination.
- (Optional) `_cstsubtype`: 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. Otherwise, it must be provided.
- (Optional) `_cststandard`: The name of the data standard to validate against. By default, all standards are included.
- (Optional) `_cststandardversion`: The version of the data standard to validate against. By default, all values of standardversion are included.

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`: Title that defines the title2 statement for all reports.
- (Required) `_cstValidationDS`: 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 you.
- (Optional) `_cstValidationDSWhClause`: WHERE clause applied to `_cstValidationDS`.
- (Required) `_cstMessagesDS`: 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, _cstreporterroronly=N, _cstreportobs=,
 _cstreportbytable=N, _csttablechecksds=, _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:

- `_cstsasreferencesdsset`: The SASReferences data set used by a SAS Clinical Standards Toolkit process. Either this data set or the `_cstresultsdsset` must exist.
- `_cstresultsdsset`: The Results data set created by a SAS Clinical Standards Toolkit process. Either this data set or the `_cstsasreferencesdsset` must exist.
- (Optional) `_cstmetricsdsset`: The Metrics data set created by a SAS Clinical Standards Toolkit process.
- `_cstreporterroronly`: (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.
- `_csttablechecksdsset`: 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.
- `_csttablecheckcode`: The code module (macro) to build `_csttablechecksdsset` if it does not exist or is not passed as a parameter. Required only if `_cstreportbytable=Y` and `_csttablechecksdsset` is not provided.
- `_cstkeepablechecklist`: 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.
- (Required) `_cstreportoutput`: 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_creareport.sas`

%cstutil_createsubdir

`%cstutil_createsubdir(_cstSubDir=)/des='CST: Create subdirectory';`

[Exposure: internal] [Macro Type: Framework utility]

This macro creates subdirectories on a computer that is not Microsoft Windows. The SAS Clinical Standards Toolkit sample drivers create output files that need to have Read and Write access to the subdirectories. This macro creates the subdirectories in the specified workspace. If a value is missing, the StudyOutputPath points to the Work directory, and any subdirectories are created under it. StudyOutputPath is referenced in SASReferences.

Required global macro variables: (none)

File: cstutil_createsubdir.sas

%cstutil_createsublists

%cstutil_createsublists() /des ="CST: Create sublists from columnscope";

[Exposure: internal] [Macro Type: Framework utility]

This macro creates the work._cstsublists data set that has interpreted validation check metadata as specified in the columnScope column in the expected form of [var1][var2]. This macro is called directly only as a validation check metadata code logic value.

Note: This macro is not always called for the derivation of work._cstsublists.

Required global macro variables (beyond reporting and debugging variables):
_cstColumnScope

Required file inputs: work._cstcolumnmetadata

File: cstutil_createsublists.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:

- (Optional) _cstCreationFlag: 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_createunixsubdir

Note: This macro will be deprecated in future releases of the SAS Clinical Standards Toolkit. It is replaced in the SAS Clinical Standards Toolkit 1.4 with the cstutil_createsubdir(_cstSubDir=) utility macro.

%cstutil_createunixsubdir(_cstSubDir=)/des='CST: Create subdirectories on UNIX';

[Exposure: internal] [Macro Type: Framework utility]

This macro creates subdirectories on a UNIX computer. The SAS Clinical Standards Toolkit sample drivers create output data sets that need to have Read and Write access to

the subdirectories. On UNIX computers, these sample study programs are in read-only areas. This macro creates the subdirectories in a temporary workspace. StudyOutputPath points to the Work directory, and any subdirectories are created under it. StudyOutputPath is referenced in SASReferences.

Required global macro variables: workpath, typically pathname(work)

File: cstutil_createunixsubdir.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:

- (Required) _cstDataSetName: 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.

- (Required) `_cstSASRefType`: File or data type from `sasreferences.type`. Representative values: `autocall`, `control`, `fintsearch`, `messages`, `properties`, `referencecontrol`, `referencemetadata`, `results`, `sourcedata`, and `sourcemetadata`.
- (Optional) `_cstSASRefSubtype`: 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=, _cststopsource=, _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.
- `_cststopsource`: 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=, _cstsource=, _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.
- `_cstsource`: 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_readxmltags`

`%cstutil_readxmltags(_cstxmlfilename=inxml, _cstxmlreporting=Results, cstxmlelementds=work.cstodmelements, _cstxmlattrds=work.cstodmattributes) / des='CST: Parse xml for elements and attributes';`

[Exposure: internal] [Macro Type: Framework tool]

This code provides a proof-of-concept implementation of a tool to read the element tags and attributes of an XML file to identify those element tags and attributes that the SAS Clinical Standards Toolkit does not currently handle using the CDISC ODM `odm_read` macro.

This macro relies on a defined set of XSLT modules, metadata that specifies a SAS representation of ODM, and a SAS XML map file that reads a derived cubexml file. Each of these makes assumptions about the XML content to be read.

Assumptions:

- The XML file has previously been defined with a SAS fileref.
- ODM reference metadata is available as defined in `SASReferences`.

Limitations:

- The code does not work on a continuous-stream (no line returns) XML file.
- The code might not work well on multi-element rows like
<Study><MetaDataVersion OID><...>.
- The code might not handle PCDATA.

Parameters:

- (Required) `_cstxmlfilename`: The fileref for the input XML file.
- (Required) `_cstxmlreporting`: Specifies how results are reported. Valid values: Dataset or Results. If the value is Dataset, these two parameters are referenced. If the value is Results, the differences that are detected are reported in the process results data set (as defined by the `&_cstResultsDS` global macro variable).
- (Optional) `_cstxmlelementds`: The libref.dataset for file elements.
Default=work.cstodmelements.
- (Optional) `_cstxmlattrds`: The libref.dataset for file attributes.
Default=work.cstodmattributes.

File: `cstutil_readxmltags.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 the 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 the SAS Clinical Standards Toolkit validation check metadata) and results (report on the 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 CDISC 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 bulddomlist and buldcollist 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_writeodmcubexml`

`%cstutil_writeodmcubexml(_cstXMLOut=);`

[Exposure: internal] [Macro Type: CDISC-CRT-DDS]

`cstutil_writeodmcubexml`

Build XML file for `define.xml`

Utility macro to create an XML file to be used by the `define.xml` process. There is one input to this macro: the MDP SAS data set that contains the member names and library references needed for the `define` process. There is one output: the XML file.

Example:

`%cstutil_odmwritecubexml(_cstXMLOut=c:\crtdds\test2.xml)`

Required global macro variables: (none)

Parameters:

- (Required) `_cstXMLOut`: The destination and filename for XML output.
- (Required) `_cstMDPFile`: The libref (for example, `srcdata.mdp`). If the libref is omitted, the macro fails.
- (Optional) `_cstDebug`: If set, information useful for validating the macro is included in the output.

File: `cstutil_writeodmcubexml.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 A3.4 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; <i>Note:</i> Not used and will be deprecated in the next release.
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';
	%sdtmutil_getchecks(_cstControl=, _cstMeta=, _cstMsg=, _cstDomain="*", _cstOutDS=, _cstIncludeDraft=true); <i>Note:</i> Deprecated.
Internal SDTM Utility	%sdtmutil_iso8601(_cstString=); %sdtmutil_listsettings(group=_ALL_); <i>Note:</i> Deprecated.

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: `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

Note: Not used and will be deprecated in the next release.

`%sdtmutil_buildsasreferences;`

[Exposure: Not specified] [Macro type: Not specified]

File: `sdtmutil_buildsasreferences.sas`

%sdtmutil_createformatsfromcrtdds

`%sdtmutil_createformatsfromcrtdds;`

[Exposure: external] [Macro Type: SDTM Tool]

This sample utility macro attempts to derive codelists from a CRT-DDS data library derived from the `define.xml` file for a CDISC SDTM study.

These source metadata files are used by the SAS Clinical Standards Toolkit to create codelists as provided in the CDISC CRT-DDS (`define.xml`) files:

- `codelists`
- `codelistitems`
- `clitemdecodetranslatedtext`

This is the general strategy:

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]

This sample utility macro attempts to derive source metadata files from a CRT-DDS data library derived from the `define.xml` file for a CDISC SDTM study.

The `itemgroupleaf` data set is used by this macro to generate a list of XPT files.

This is the general strategy:

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 specify 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]

This sample utility macro attempts to derive source metadata files from a CRT-DDS data library derived from the define.xml file for a CDISC SDTM study.

These 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

This is the general strategy:

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]

SAS library for a CDISC SDTM study.

These 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

This is the general strategy:

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). If it is not specified, data set keys are estimated based on columns that 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 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 values are marked as `<--- HARDCODE`.

Limitations:

Here are two scenarios that have not been addressed:

- Split domains, such as QS**
- SDTM 3.1.2 FA multiple domains (for example, FACM)

File: sdtmutil_createsrcmetafromsaslib.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.

These 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

This is the general strategy:

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:

Here 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]

Note: Deprecated.

Parameters:

- `_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 specifies 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 that specifies a message that describes the validity of the input string.

`_cstISOinfo`: String that specifies 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]

Note: Deprecated.

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 A3.5 Module SDTM V3.1.2 (Run Time) Macro Summary

Exposure	Macro
External SDTM Validation Process	<code>%sdtm_validate /des='CST: Validate CDISC SDTM model files';</code>
Internal SDTM Validation Check Utility	<code>%sdtmcheckutil_recordlookup(_cstSourceDS=&_cstDSName, _cstSourceLib=&_cstRefOnly);</code>

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 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: `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.

These 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`

This is the general strategy:

1. Combine CRT-DDS data to create the `entlin` 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.

This is the general strategy:

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.

These 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

This is the general strategy:

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.

These 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

This is the general strategy:

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:

Here 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 that specifies a message that describes the validity of the input string.
- _cstISOinfo:** String that specifies 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 A3.6 Module ODM V1.3.0 (Run Time) Macro Summary

Exposure	Macro
External Framework	%odm_getstatic(_cstName=, _cstVar=);
External CDISC ODM	%odm_read;

Exposure	Macro
External ODM Validation Process	%odm_validate;
External CDISC ODM	%odm_write(_cstCreateDisplayStyleSheet=0, _cstOutputEncoding=UTF-8, _cstHeaderComment=, _cstResultsOverrideDS=, _cstLogLevel=info);
External CDISC ODM	%odm_xmlvalidate(_cstLogLevel=info,_cstResultsOverrideDS=);

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: The location of an existing Results data set. If not specified, work._cstResults is used.

File: odm_read.sas

%odm_validate

%odm_validate /des='CST: Validate CDISC ODM model files';

[Exposure: external] [Macro Type: ODM Validation Process]

Validate CDISC ODM model files.

The basic function of this code module is to cycle through the validation checks to be run, and to write validation results to the process results and (optionally) metrics data

sets. These results are persisted to any permanent location based on type=results records in SASReferences. 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: odm_validate

%odm_write

```
%odm_write(_cstCreateDisplayStyleSheet=0, _cstOutputEncoding=UTF-8,
_cstHeaderComment=, _cstResultsOverrideDS=, _cstLogLevel=info ) /des='CST: Write
ODM 1.3.0 XML file';
```

[Exposure: external] [Macro Type: CDISC-ODM]

Writes an ODM 1.3.0 XML file.

This macro uses the SAS representation of an ODM XML file as source data and converts it to the required XML structure. The inputs and outputs are specified through SASReferences.

Required global macro variables:

- The framework initialization properties.
- The ODM 1.3.0 initialization properties.
- _cstresultsds should point to an existing results data set, or it should override this value using the _cstResultsOverrideDS parameter.

Parameters:

- (Optional) _cstCreateDisplayStyleSheet: Specifies whether the macro should create a style sheet in the same directory as the output XML file. If this value is set to 1, the macro looks in SASReferences for a record with a type or subtype of referencexml/stylesheet and uses that record. If this value is set to 0, the macro does not create the XSL, even if a record is specified in SASReferences. Default=0.
- (Optional) _cstOutputEncoding: The XML encoding to use for the ODM file that is created. Default=UTF-8.
- (Optional) _cstHeaderComment: A short comment that is added to the top of the ODM XML file that is produced. If no comment is provided, a default is used.
- (Optional) _cstResultsOverrideDS: The (libname.)member that refers to a results data set to be created. If this parameter is not specified, the results data set specified by &_cstResultsDS is the data set written to.
- (Optional) _cstLogLevel: The level of error reporting. Valid values are Info, Warning, Error, and Fatal Error. Default=Info.

File: odm_write

%odm_xmlvalidate

```
%odm_xmlvalidate(_cstLogLevel=info, _cstResultsOverrideDS=) /des='CST: Validate
ODM V1.3.0 XML file';
```

[Exposure: external] [Macro Type: CDISC-ODM]

Perform XML-level (not SAS) validation on an ODM V1.3.0 XML file.

General use of this macro will be in combination with another macro (such as odm_write or odm_read). Conditional code is included that optionally writes metadata to the results

data set and optionally checks the validity of the SASReferences data set if this macro is run independently.

Parameters:

- (Optional) `_cstResultsOverrideDS`: The (libname.)member that refers to a results data set to be created. If this parameter is not specified, the results data set specified by `&_cstResultsDS` is the data set written to. This parameter is optional.
- (Optional) `_cstLogLevel`: The level of error reporting. Valid values are Info, Warning, Error, and Fatal Error.

File: `odm_xmlvalidate`

Appendix 4

CDISC SDTM Validation Checks

Validation Checks (with SDTM V3.1.1) 351

Validation Checks (with SDTM V3.1.2) 379

Validation Checks (with SDTM V3.1.1)

This table provides a complete list of CDISC SDTM validation checks. The version of SDTM is V3.1.1.

Table A4.1 Validation Checks (with SDTM V3.1.1)

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0001	Janus	IR4000	Identifies domain table that has zero rows and, therefore, contains no data.	_ALL_	
SDTM0001	WebSDM	IR4000	Identifies domain table that has zero rows and, therefore, contains no data.	_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.	DM+DS+EX	
SDTM0003	WebSDM	SAS0018	WebSDM and the SDTM model require only the DM domain be present.	DM	
SDTM0004	SAS	SAS0033	Source metadata includes domain data set not found in reference metadata.	_ALL_	
SDTM0005	SAS	SAS0034	Custom domain data set does not conform to specification naming guidelines.	_ALL_	

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0006	SAS	SAS0035	Source data library contains domain data not found in study metadata.	_ALL_	
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_	
SDTM0011	WebSDM	IR4250	Identifies a column that was described in the domain description, but not included in the SAS data set for that domain.	_ALL_	
SDTM0012	Janus	IR4252	Identifies a column listed in the domain description as required (Req), but not included in the SAS data set for that domain.	_ALL_	
SDTM0012	WebSDM	IR4252	Identifies a column listed in the domain description as required (Req), but not included in the SAS data set for that domain.	_ALL_	
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_	
SDTM0013	WebSDM	IR4253	Identifies a column listed in the domain description as expected (Exp), but not included in the SAS data set for that domain.	_ALL_	
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_	
SDTM0015	Janus	IR4254	Identifies a column that appears in the SAS data set, but is not listed in the domain description.	_ALL_	
SDTM0015	WebSDM	IR4254	Identifies a column that appears in the SAS data set, but is not listed in the domain description.	_ALL_	
SDTM0017	Janus	IR4258	Identifies a domain that appears to contain supplemental qualifier data, but does not contain the unique subject identifier (USUBJID).	SUPP**	

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0017	WebSDM	IR4258	Identifies a domain that appears to contain supplemental qualifier data, but does not contain the unique subject identifier (USUBJID).	SUPP**	
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_	
SDTM0019	WebSDM	IR4259	Identifies a variable where data type in (study-specific) description is not consistent with data type implicit in SAS data set.	_ALL_	
SDTM0020	SAS	SAS0006	Column order does not match standard.	_ALL_	
SDTM0022	SAS	SAS0001	Column length < length defined in standard.	_ALL_	
SDTM0023	SAS	SAS0002	Column length > length defined in standard.	_ALL_	
SDTM0030	WebSDM	IR4264	Column label inconsistent with label defined in standard.	_ALL_	
SDTM0031	SAS	SAS0004	Column format found, but column not subject to controlled terminology.	_ALL_	
SDTM0032	SAS	SAS0005	Column format found, but format name mismatch with standard controlled terminology name.	_ALL_	
SDTM0033	WebSDM	IR4266	Identifies a variable that has been deprecated according to the CDISC SDTM standard.	_ALL_	
SDTM0101	JanusFR	IR4002	Identifies values that do not conform to the ISO 8601 standard for datetimes.	_ALL_	**DTC +**STDTC +**ENDTC +BRTHDTC +RFSTDTC +RFENDTC

checkid	check source	sourceid	source description	tablescape	columnscope
SDTM0101	WebSDM	IR4002	Identifies values that do not conform to the ISO 8601 standard for datetimes.	_ALL_	**DTC +**STDTC +**ENDTC +BRTHDTC +RFSTDTC +RFENDTC
SDTM0102	JanusFR	IR4002	Identifies values that do not conform to the ISO 8601 standard for durations.	_ALL_	**DUR
SDTM0102	WebSDM	IR4002	Identifies values that do not conform to the ISO 8601 standard for durations.	_ALL_	**DUR
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
SDTM0124	WebSDM	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
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
SDTM0125	WebSDM	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
SDTM0126	SAS	SAS0017	Qualifier variable label (QLABEL) length > 40.	SUPP**	QLABEL
SDTM0127	SAS	SAS0018	Qualifier variable name (QNAM) length > 8, starts with a number, or contains special characters.	SUPP**	QNAM
SDTM0128	Janus	IR4115	Identifies records that violate the condition [LENGTH(Trial Summary Parameter (**PARM)) less than or equal to 40 characters].	TS	TSPARM

checkid	check source	sourceid	source description	tablescape	columnscope
SDTM0128	WebSDM	IR4115	Identifies records that violate the condition [LENGTH(Trial Summary Parameter (**PARM)) less than or equal to 40 characters].	TS	TSPARM
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	TSPARMCD
SDTM0129	WebSDM	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	TSPARMCD
SDTM0201	Janus	IR4001	Identifies a null (empty) value found in a column where (Standard) Core attribute is Req.	_ALL_	
SDTM0201	WebSDM	IR4001	Identifies a null (empty) value found in a column where (Standard) Core attribute is Req.	_ALL_	
SDTM0202	SAS	SAS0015	Identifies a null (empty) value found in a column where (Standard) Core attribute is Exp.	_ALL_	
SDTM0203	SAS	SAS0010	Character column value is not correctly uppercased per specification.	_ALL_	
SDTM0204	SAS	SAS0011	Character column value contains the numeric missing '.' or any special missing value like '.N'.	_ALL_	
SDTM0205	SAS	SAS0012	Column value is not left-justified.	_ALL_	
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

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0206	WebSDM	IR4003	Identifies records where the value in the Domain Abbreviation column (DOMAIN) does not match the name of Domain.	_ALL_ -SUPP*- RELREC	DOMAIN
SDTM0207	Janus	IR4010	Identifies records where the value for Visit Number (VISITNUM) is formatted to more than three decimal places.	_ALL_	VISITNUM
SDTM0207	WebSDM	IR4010	Identifies records where the value for Visit Number (VISITNUM) is formatted to more than 3 decimal places.	_ALL_	VISITNUM
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]
SDTM0209	WebSDM	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]
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]
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]

checkid	check source	sourceid	source description	tablescope	columnscope
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]
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]
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]
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]
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]
SDTM0212	WebSDM	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]

checkid	check source	sourceid	source description	tablescope	columnscope
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]
SDTM0213	WebSDM	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]
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]
SDTM0214	WebSDM	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]
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]
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]
SDTM0218	Janus	IR4107	Identifies records that violate the condition [Status (**STAT) equals NOT DONE], limited to records where **STAT is not null.	_ALL_	**STAT

checkid	check source	sourceid	source description	tablescape	columnscope
SDTM0218	WebSDM	IR4107	Identifies records that violate the condition [Status (**STAT) equals NOT DONE], limited to records where **STAT is not null.	_ALL_	**STAT
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	[**REASND] [**STAT]
SDTM0219	WebSDM	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	[**REASND] [**STAT]
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
SDTM0220	WebSDM	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
SDTM0221	Janus	IR4136	Identifies records where values are not found in the study-specific codelist attached to a variable.	_ALL_	
SDTM0221	WebSDM	IR4136	Identifies records where values are not found in the study-specific codelist attached to a variable.	_ALL_	
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
SDTM0222	WebSDM	IR4137	Identifies records that violate the condition [Study Day of Visit/Collection/Exam (**DY) does not equal 0].	_ALL_	**DY+**STDY +**ENDY +VISITDY
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]

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0225	WebSDM	IR4162	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]
SDTM0226	WebSDM	IR4163	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]
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
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
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
SDTM0271	SAS	SAS0036	Value for column defined as a data set key is null.	_ALL_	
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
SDTM0301	WebSDM	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
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

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0302	WebSDM	IR4106	Identifies records that violate the condition [Occurrence (**OCCUR) in (Y , N)], limited to records where **OCCUR is not null.	CLASS: EVENTS +CLASS: INTER- VENTIONS	**OCCUR
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: INTER- VENTIONS	**STRF
SDTM0303	WebSDM	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: INTER- VENTIONS	**STRF
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: INTER- VENTIONS	[**DOSE] [**DOSU]
SDTM0351	WebSDM	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: INTER- VENTIONS	[**DOSE] [**DOSU]
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: INTER- VENTIONS	**DOSE
SDTM0352	WebSDM	IR4109	Identifies records that violate the condition [Dose (**DOSE) greater than or equal to 0], limited to records where **DOSE is not null.	CLASS: INTER- VENTIONS	**DOSE

checkid	check source	sourceid	source description	tablescope	columnscope
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: INTER- VENTIONS	[**DOSE] [**DOSU]
SDTM0353	WebSDM	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: INTER- VENTIONS	[**DOSE] [**DOSU]
SDTM0354	WebSDM	IR4139	Identifies records that violate the condition [Related Domain (RDOMAIN) is not null].	SUPP** +RELREC	RDOMAIN
SDTM0355	SAS	SAS0040	Value for Related Domain (RDOMAIN) is inconsistent with data set name.	SUPP**- SUPPQUAL	RDOMAIN
SDTM0401	Janus	IR4102	Identifies records that violate the condition [Baseline Flag (**BLFL) either 'Y' or null].	CLASS: FINDINGS	**BLFL
SDTM0401	WebSDM	IR4102	Identifies records that violate the condition [Baseline Flag (**BLFL) either 'Y' or null].	CLASS: FINDINGS	**BLFL
SDTM0402	JanusFR	IR4103	Identifies records that violate the condition [Derived Flag (**DRVFL) either 'Y' or null].	CLASS: FINDINGS	**DRVFL
SDTM0402	WebSDM	IR4103	Identifies records that violate the condition [Derived Flag (**DRVFL) either 'Y' or null].	CLASS: FINDINGS	**DRVFL
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
SDTM0403	WebSDM	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

checkid	check source	sourceid	source description	tablescope	columnscope
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]
SDTM0405	WebSDM	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]
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]
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]
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]
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]
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]

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0408	WebSDM	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]
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]
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]
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]
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]
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: FINDINGS	[**STNRHI] [**STNRLO]

checkid	check source	sourceid	source description	tablescope	columnscope
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: FINDINGS-IE	[**STRESC] [**STRESU]
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: FINDINGS-IE	[**STRESC] [**STRESU]
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]
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]
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]
SDTM0414	WebSDM	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]
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
SDTM0451	JanusFR	IR4007	Identifies records where the value for the preferred term could not be found in the MedDRA dictionary.	AE	AEDECOD

checkid	check source	sourceid	source description	tablescape	columnscope
SDTM0451	WebSDM	IR4007	Identifies records where the value for the preferred term could not be found in the MedDRA dictionary.	AE	AEDECOD
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
SDTM0452	WebSDM	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
SDTM0453	JanusFR	R4019	Identifies records where value for [Serious Event (AESER)] is not found in codelist [YESNO].	AE	AESER
SDTM0453	WebSDM	R4019	Identifies records where value for [Serious Event (AESER)] is not found in Codelist [YESNO].	AE	AESER
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

checkid	check source	sourceid	source description	tablescape	columnscope
SDTM0454	WebSDM	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
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
SDTM0455	WebSDM	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
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
SDTM0456	WebSDM	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
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
SDTM0457	WebSDM	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
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

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0458	WebSDM	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
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
SDTM0459	WebSDM	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
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
SDTM0460	WebSDM	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
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
SDTM0461	WebSDM	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
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]

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0462	WebSDM	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]
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]
SDTM0463	WebSDM	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]
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	AECONTRT
SDTM0464	WebSDM	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	AECONTRT
SDTM0500	WebSDM	IR4172	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]
SDTM0501	Janus	IR4011	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa].	DM	[ARM] [ARMCD]
SDTM0501	WebSDM	IR4011	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa].	DM+TA	[ARM] [ARMCD]

checkid	check source	sourceid	source description	tablescope	columnscope
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]
SDTM0502	WebSDM	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]
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]
SDTM0503	WebSDM	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]
SDTM0504	JanusFR	R4007	Identifies records where value for [SEX] is not found in codelist [SEX].	DM	SEX
SDTM0504	WebSDM	R4007	Identifies records where value for [SEX] is not found in codelist [SEX].	DM	SEX
SDTM0505	Janus	R4008	Identifies records where value for [COUNTRY] is not found in codelist [COUNTRY].	DM	COUNTRY
SDTM0505	WebSDM	R4008	Identifies records where value for [COUNTRY] is not found in codelist [COUNTRY].	DM	COUNTRY
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

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0506	WebSDM	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
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]
SDTM0507	WebSDM	R4106	Identifies records that violate the condition [age units (AGEU) is not null], limited to records where AGE is not null.	DM	[AGE][AGEU]
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
SDTM0508	WebSDM	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
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
SDTM0531	WebSDM	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
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 IEORES is not null.	IE	IEORRES

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0532	WebSDM	R4071	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES)] is not found in codelist[YESNO], limited to records where IEORES is not null.	IE	IEORRES
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
SDTM0533	WebSDM	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
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]
SDTM0534	WebSDM	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]
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]
SDTM0541	OpenCDISC	SD0092	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]
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]

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0561	WebSDM	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]
SDTM0601	SAS	SAS0013	Domain not sorted by keys as defined in standard.	_ALL_	
SDTM0602	SAS	SAS0007	Records are not unique by the expected keys	_ALL_	
SDTM0603	JanusFR	IR4004	Identifies records where non-unique values for Sequence Number variable (**SEQ) exist within a subject.	_ALL_-TS	**SEQ
SDTM0603	WebSDM	IR4004	Identifies records where non-unique values for Sequence Number variable (**SEQ) exist within a subject.	_ALL_-TS	**SEQ
SDTM0604	SAS	SAS0009	Sequence Number (**SEQ) values are not consecutively incremented beginning at 1 for each USUBJID.	TS	TSSEQ
SDTM0604	SAS	SAS0009	Sequence Number (**SEQ) values are not consecutively incremented beginning at 1 for each USUBJID.	_ALL_-TS	**SEQ
SDTM0605	SAS	SAS0014	Report any variable for the domain that contains all missing or null values.	_ALL_	_ALL_
SDTM0606	JanusFR	SAS0022	Identify any column defined as numeric in the standard that contains non-numeric values.	_ALL_	
SDTM0607	JanusFR	SAS0038	Site Study Identifier (SITEID) is null for all records.	DM	SITEID
SDTM0621	WebSDM	IR4005	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
SDTM0622	WebSDM	IR4142	Inconsistency between test (**TEST) and test code (**TESTCD).	CLASS: FINDINGS	[**TEST] [**TESTCD]

checkid	check source	sourceid	source description	tablescape	columnscope
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]
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]
SDTM0631	WebSDM	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]
SDTM0641	JanusFR	R4005	Identifies records where values for Unique Subject ID (USUBJID) are not unique, limited to records where USUBJID is not null.	DM	
SDTM0641	WebSDM	R4005	Identifies records where values for Unique Subject ID (USUBJID) are not unique, limited to records where USUBJID is not null.	DM	
SDTM0642	SAS	SAS0028	Inconsistency between Description of Arm (ARM) and Arm Code (ARMCD) values across all records.	DM	[ARM] [ARMCD]
SDTM0643	SAS	SAS0016	AGE precision inconsistent across records.	DM	AGE
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
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

checkid	check source	sourceid	source description	tablescope	columnscope
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**	
SDTM0661	WebSDM	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**	
SDTM0662	WebSDM	IR4161	Identifies qualifier variable name (QNAM) where variable label value (Qualifier Variable Label QLABEL) is not consistent across all records.	SUPP**	[QNAM] [QLABEL]
SDTM0671	SAS	SAS0032	Inconsistency between Trial Summary Parameter (TSPARM) and Trial Summary Parameter Short Name (TSPARMCD).	TS	[TSPARM] [TSPARMCD]
SDTM0801	JanusFR	IR4500	Identifies non-demographics domain subjects (USUBJID) not found in the demographics domain.	[_ALL_-DM][DM]	STUDYID +USUBJID
SDTM0801	WebSDM	IR4500	Identifies non-demographics domain subjects (USUBJID) not found in the demographics domain.	[_ALL_-DM][DM]	STUDYID +USUBJID
SDTM0802	Janus	IR4505	Identifies demographics subjects where no record for the subject is found in the disposition domain.	[DM][DS]	STUDYID +USUBJID
SDTM0802	WebSDM	IR4505	Identifies demographics subjects where no record for the subject is found in the disposition domain.	[DM][DS]	STUDYID +USUBJID

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0803	Janus	IR4506	Identifies demographics subjects where no record for the subject is found in the exposure domain.	[DM][EX]	STUDYID +USUBJID
SDTM0803	WebSDM	IR4506	Identifies demographics subjects where no record for the subject is found in the exposure domain.	[DM][EX]	STUDYID +USUBJID
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
SDTM0804	WebSDM	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
SDTM0805	Janus	IR4502	Identifies records where the value for ARM code (ARMCD) is not found in the TA domain, excluding 'SCRNFAIL'.	[DM][TA]	ARMCD
SDTM0805	WebSDM	IR4502	Identifies records where the value for ARM code (ARMCD) is not found in the TA domain, excluding 'SCRNFAIL'.	[DM][TA]	ARMCD
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
SDTM0806	WebSDM	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
SDTM0807	JanusFR	SAS0039	TA domain is not provided and Planned Arm Code (ARMCD) is null for all rows in the demographics domain.	DM	ARMCD

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0808	WebSDM	IR4170	Identifies records that violate the condition [Visit Name (VISIT) must be the same for a given value of Visit Number (VISITNUM)].	SV	[VISIT] [VISITNUM]
SDTM0809	WebSDM	IR4171	Identifies records that violate the condition [Visit Number (VISITNUM) must be the same for a given value of Visit Name (VISIT)].	SV	[VISITNUM] [VISIT]
SDTM0811	Janus	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[TA][TE]	ETCD
SDTM0811	Janus	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[SE][TE]	ETCD
SDTM0811	WebSDM	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[TA][TE]	ETCD
SDTM0811	WebSDM	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[SE][TE]	ETCD
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
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
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
SDTM0831	JanusFR	SAS0020	The Study Identifier (STUDYID) in the TA domain does not match STUDYID in the DM domain.	[TA][DM]	STUDYID

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0836	JanusFR	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
SDTM0851	JanusFR	IR4508	Identifies comments (CO) domain reference to an unknown related domain.	CO	RDOMAIN
SDTM0851	WebSDM	IR4508	Identifies comments (CO) domain reference to an unknown related domain.	CO	RDOMAIN
SDTM0861	Janus	IR4509	Identifies Related Records (RELREC) domain reference to an unknown related domain.	RELREC	RDOMAIN
SDTM0861	WebSDM	IR4509	Identifies Related Records (RELREC) domain reference to an unknown related domain.	RELREC	RDOMAIN
SDTM0862	JanusFR	IR4510	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to an unknown related domain.	SUPP**	RDOMAIN
SDTM0862	WebSDM	IR4510	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to an unknown related domain.	SUPP**	RDOMAIN
SDTM0863	Janus	IR4511	Identifies Related Records (RELREC) domain reference to a key variable that is not defined in the target domain.	RELREC	IDVAR
SDTM0863	WebSDM	IR4511	Identifies Related Records (RELREC) domain reference to a key variable that is not defined in the target domain.	RELREC	IDVAR
SDTM0864	JanusFR	IR4512	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a key variable that is not defined in the target domain.	SUPP**	IDVAR

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0864	WebSDM	IR4512	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a key variable that is not defined in the target domain.	SUPP**	IDVAR
SDTM0865	Janus	IR4513	Identifies Related Records (RELREC) domain reference to a record that does not exist in the target domain.	RELREC	IDVAR
SDTM0865	WebSDM	IR4513	Identifies Related Records (RELREC) domain reference to a record that does not exist in the target domain.	RELREC	IDVAR
SDTM0866	JanusFR	IR4514	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a record that does not exist in the target domain.	SUPP**	IDVAR
SDTM0866	WebSDM	IR4514	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a record that does not exist in the target domain.	SUPP**	IDVAR
SDTM0871	Janus	SAS0024	Identifies comments (CO) domain reference to a record that does not exist in the target domain.	CO	IDVAR

Validation Checks (with SDTM V3.1.2)

This table provides a complete list of CDISC SDTM validation checks. The version of SDTM is V3.1.2. All WebSDM IR numbers begin with 5000.

Table A4.2 Validation Checks (with SDTM V3.1.2)

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0001	WebSDM	IR5000	Identifies domain table that has zero rows and, therefore, contains no data.	_ALL_	
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	

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0003	SAS	SAS0018	WebSDM and the SDTM model require only the DM domain be present.	DM	
SDTM0004	SAS	SAS0033	Source metadata includes domain data set not found in reference metadata.	_ALL_	
SDTM0005	SAS	SAS0034	Custom domain data set does not conform to specification naming guidelines.	_ALL_	
SDTM0006	SAS	SAS0035	Source data library contains domain data not found in study metadata.	_ALL_	
SDTM0011	WebSDM	IR5250	Identifies a column that was described in the domain description, but not included in the SAS data set for that domain.	_ALL_	
SDTM0012	WebSDM	IR5252	Identifies a column listed in the domain description as required (Req), but not included in the SAS data set for that domain.	_ALL_	
SDTM0013	WebSDM	IR5253	Identifies a column listed in the domain description as expected (Exp), but not included in the SAS data set for that domain.	_ALL_	
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_	
SDTM0015	WebSDM	IR5254	Identifies a column that appears in the SAS data set, but is not listed in the domain description.	_ALL_	
SDTM0017	WebSDM	IR5258	Identifies a domain that appears to contain supplemental qualifier data, but does not contain the unique subject identifier (USUBJID).	SUPP**	
SDTM0019	WebSDM	IR5259	Identifies a variable where data type in (study-specific) description is not consistent with data type implicit in SAS data set.	_ALL_	

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0020	SAS	SAS0006	Column order does not match standard.	_ALL_	
SDTM0022	SAS	SAS0001	Column length < length defined in standard.	_ALL_	
SDTM0023	SAS	SAS0002	Column length > length defined in standard.	_ALL_	
SDTM0030	WebSDM	IR5264	Column label inconsistent with label defined in standard.	_ALL_	
SDTM0031	SAS	SAS0004	Column format found, but column not subject to controlled terminology.	_ALL_	
SDTM0032	SAS	SAS0005	Column format found, but format name mismatch with standard controlled terminology name.	_ALL_	
SDTM0033	WebSDM	IR5266	Identifies a variable that has been deprecated according to the CDISC SDTM standard.	_ALL_	
SDTM035	WebSDM	IR5261	Identifies a domain that is referenced in a description file, but for which there is no source data.	_ALL_	
SDTM036	WebSDM	IR5262	Identifies a domain whose source data failed to load.	_ALL_	
SDTM037	WebSDM	IR5263	Identifies a variable that uses an unsupported event dictionary.	_ALL_	
SDTM038	WebSDM	IR5265	Identifies a variable whose referenced codelist is not properly defined in the associated define.xml file.	_ALL_	
SDTM039	WebSDM	IR5267	Identifies a domain for which metadata has not been provided.	_ALL_	
SDTM0101	WebSDM	IR5002	Identifies values that do not conform to the ISO 8601 standard for datetimes.	_ALL_	**DTC +**STDTC +**ENDTC +BRTHDTC +RFSTDTC +RFENDTC
SDTM0102	WebSDM	IR5002	Identifies values that do not conform to the ISO 8601 standard for durations.	_ALL_	**DUR

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0124	WebSDM	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
SDTM0125	WebSDM	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
SDTM0126	SAS	SAS0017	Qualifier variable label (QLABEL) length > 40.	SUPP**	QLABEL
SDTM0127	SAS	SAS0018	Qualifier variable name (QNAM) length > 8, starts with a number, or contains special characters.	SUPP**	QNAM
SDTM0128	WebSDM	IR5115	Identifies records that violate the condition [LENGTH(Trial Summary Parameter (**PARM)) less than or equal to 40 characters].	TS	TSPARM
SDTM0129	WebSDM	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	TSPARMCD
SDTM0130	OpenCDISC	SD1004	The value of planned arm code (ARMCD) must not be more than 20 characters in length.		
SDTM0131	OpenCDISC	SD1009	The value of the element code (ETCD) must not be no more than 8 characters in length.		
SDTM0201	WebSDM	IR5001	Identifies a null (empty) value found in a column where (Standard) Core attribute is Req.	_ALL_	
SDTM0202	SAS	SAS0015	Identifies a null (empty) value found in a column where (Standard) Core attribute is Exp.	_ALL_	

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0203	SAS	SAS0010	Character column value is not correctly uppercased per specification.	_ALL_	
SDTM0204	SAS	SAS0011	Character column value contains the numeric missing '.' or any special missing value like '.N'.	_ALL_	
SDTM0205	SAS	SAS0012	Column value is not left-justified.	_ALL_	
SDTM0206	WebSDM	IR5003	Identifies records where the value in the Domain Abbreviation column (DOMAIN) does not match the name of Domain.	_ALL_ -SUPP*- RELREC	DOMAIN
SDTM0207	WebSDM	IR5010	Identifies records where the value for Visit Number (VISITNUM) is formatted to more than 3 decimal places.	_ALL_	VISITNUM
SDTM0209	WebSDM	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]
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]
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	[**STRF] [**ENRF]

checkid	check source	sourceid	source description	tablescope	columnscope
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]
SDTM0212	WebSDM	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]
SDTM0213	WebSDM	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]
SDTM0214	WebSDM	IR5133	Identifies records that violate the condition [Time Point Reference (**TPTRF) is not null], limited to records where [Elapsed Time from Reference Point (**ELTM) is not null].	_ALL_-PP	[**TPTRF] [**ELTM]
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]
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]
SDTM0218	WebSDM	IR5107	Identifies records that violate the condition [Status (**STAT) equals NOT DONE], limited to records where **STAT is not null.	_ALL_	**STAT

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0219	WebSDM	IR5122	Identifies records that violate the condition [reason not done (**REASND) is null], limited to records where [status (**STST) is null].	_ALL_	[**REASND] [**STAT]
SDTM0220	WebSDM	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
SDTM0221	WebSDM	IR5136	Identifies records where values are not found in the study-specific codelist attached to a variable.	_ALL_	
SDTM0222	WebSDM	IR5137	Identifies records that violate the condition [Study Day of Visit/Collection/Exam (**DY) does not equal 0].	_ALL_	**DY+**STDY +**ENDY +VISITDY
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]
SDTM0223	SAS	SAS0030	Identifies records with the condition [subcategory (**SCAT) is not null when category of related records (**CAT) is null].	TI	[IESCAT] [IECAT]
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]
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]
SDTM0231	OpenCDISC	SD1003	When age units (AGEU) are not null, then age (AGE) should be provided.	DM	[AGE] [AGEU]

checkid	check source	sourceid	source description	tablescope	columnscope
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]
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]
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
SDTM0271	SAS	SAS0036	Value for column defined as a data set key is null.	_ALL_	
SDTM0301	WebSDM	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: INTER- VENTIONS	**ENRF
SDTM0302	WebSDM	IR5106	Identifies records that violate the condition [Occurrence (**OCCUR) in (Y , N)], limited to records where **OCCUR is not null.	CLASS: EVENTS +CLASS: INTER- VENTIONS	**OCCUR
SDTM0303	WebSDM	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: INTER- VENTIONS	**STRF
SDTM0351	WebSDM	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: INTER- VENTIONS	[**DOSE] [**DOSU]

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0352	WebSDM	IR5109	Identifies records that violate the condition [Dose (**DOSE) greater than or equal to 0], limited to records where **DOSE is not null.	CLASS: INTER- VENTIONS	**DOSE
SDTM0353	WebSDM	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: INTER- VENTIONS	[**DOSE] [**DOSU]
SDTM0354	WebSDM	IR5139	Identifies records that violate the condition [Related Domain (RDOMAIN) is not null].	SUPP** +RELREC	RDOMAIN
SDTM0355	SAS	SAS0040	Value for Related Domain (RDOMAIN) is inconsistent with data set name.	SUPP**_ SUPPQUAL	RDOMAIN
SDTM0401	WebSDM	IR5102	Identifies records that violate the condition [Baseline Flag (**BLFL) either 'Y' or null].	CLASS: FINDINGS	**BLFL
SDTM0402	WebSDM	IR5103	Identifies records that violate the condition [Derived Flag (**DRVFL) either 'Y' or null].	CLASS: FINDINGS	**DRVFL
SDTM0403	WebSDM	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
SDTM0405	WebSDM	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]
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]

checkid	check source	sourceid	source description	tablescope	columnscope
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]
SDTM0408	WebSDM	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]
SDTM0409	WebSDM	IR5126	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]
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] [LBSTNRLO]
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: FINDINGS	[**STNRHI] [**STNRLO]
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: FINDINGS- IE-PE	[**STRESC] [**STRESU]
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]

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0414	WebSDM	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]
SDTM0415	WebSDM	IR5143	Identifies records that violate the condition [if non-null occurrence (**OCCUR) is provided, then pre-specified (**PRES P) must equal 'Y'].	CE+CM+SU +MH	[**OCCUR] [**PRES P]
SDTM0416	WebSDM	IR5144	Identifies records that violate the condition [if non-null occurrence (**OCCUR) is provided and pre-specified (**PRES P) equal 'Y', then completion status (**STAT) must equal 'NOT DONE'].	CLASS: EVENTS +CLASS: INTER- VENTIONS	[**OCCUR] [**STAT]
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]
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]
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]
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]
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]

checkid	check source	sourceid	source description	tablescope	columnscope
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
SDTM0451	WebSDM	IR5007	Identifies records where the value for the preferred term could not be found in the MedDRA dictionary.	AE	AEDECOD
SDTM0452	WebSDM	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
SDTM0453	WebSDM	IR5019	Identifies records where value for [Serious Event (AESER)] is not found in Codelist [YESNO].	AE	AESER
SDTM0454	WebSDM	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
SDTM0455	WebSDM	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
SDTM0456	WebSDM	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

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0457	WebSDM	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
SDTM0458	WebSDM	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
SDTM0459	WebSDM	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
SDTM0460	WebSDM	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
SDTM0461	WebSDM	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
SDTM0462	WebSDM	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]
SDTM0463	WebSDM	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]
SDTM0464	WebSDM	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	AECONTRT

checkid	check source	sourceid	source description	tablescope	columnscope
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
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
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
SDTM0470	OpenCDISC	CT0003	Variable values should be populated with terms found in AGESPAN (C66780) CDISC controlled terminology codelist.	TS	TSVAL
SDTM0471	OpenCDISC	CT0005	Variable values should be populated with terms found in AGEU (C66781) CDISC terminology codelist.	TS	TSVAL
SDTM0472	OpenCDISC	CT0007	Variable values should be populated with terms found in drug accountability test name (C66731) CDISC controlled terminology codelist.	DA	DATEST
SDTM0473	OpenCDISC	CT0008	Variable values should be populated with terms found in drug accountability test code (C66732) CDISC controlled terminology codelist.	DA	DATESTCD
SDTM0475	OpenCDISC	CT0016	Variable values should be populated with terms found in evaluator (C78735) CDISC controlled terminology codelist.	CLASS: FINDINGS	**EVAL
SDTM0476	OpenCDISC	CT0017	Variable values should be populated with terms found in evaluator (C78735) CDISC controlled terminology codelist.	SUPP**	QEVAL

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0477	OpenCDISC	CT0024	Variable values should be populated with terms found in MARISTAT (C76348) CDISC controlled terminology codelist.	SC	SCSTRESC
SDTM0478	OpenCDISC	CT0026	Variable values should be populated with terms found in 'Reference Range Indicator' (C78736) CDISC controlled terminology codelist.	CLASS: FINDINGS	**NRIND
SDTM0479	OpenCDISC	CT0032	Variable values should be populated with terms found in ROUTE (C66729) CDISC controlled terminology codelist.	TS	TSVAL
SDTM0480	OpenCDISC	CT0035	Variable values should be populated with terms found in SEXPOP (C66732) CDISC controlled terminology codelist.	TS	TSVAL
SDTM0481	OpenCDISC	CT0037	Variable values should be populated with terms found in AGESPAN (C66780) CDISC controlled terminology codelist.	CLASS: FINDINGS +CLASS: EVENTS	**BODSYS
SDTM0482	OpenCDISC	CT0040	Variable values should be populated with terms found in TBLIND (C66735) CDISC controlled terminology codelist.	TS	TSVAL
SDTM0483	OpenCDISC	CT0041	Variable values should be populated with terms found in TCNTRL (C66785) CDISC controlled terminology codelist.	TS	TSVAL
SDTM0484	OpenCDISC	CT0042	Variable values should be populated with terms found in TDIGRP (C66787) CDISC controlled terminology codelist.	TS	TSVAL
SDTM0485	OpenCDISC	CT0043	Variable values should be populated with terms found in TINDTP (C66736) CDISC controlled terminology codelist.	TS	TSVAL
SDTM0486	OpenCDISC	CT0045	Variable values should be populated with terms found in TPHASE (C66737) CDISC controlled terminology codelist.	TS	TSVAL

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0487	OpenCDISC	CT0046	Variable values should be populated with terms found in Trial Summary Parameter Test Name (C67152) CDISC controlled terminology codelist.	TS	TSPARM
SDTM0488	OpenCDISC	CT0047	Variable values should be populated with terms found in Trial Summary Parameter Test Code (C66738) CDISC controlled terminology codelist.	TS	TSPARMCD
SDTM0489	OpenCDISC	CT0035	Variable values should be populated with terms found in TTYPE (C66739) CDISC controlled terminology codelist.	TS	TSVAL
SDTM0490	WebSDM	IR5150	Identifies records that violate the condition [Pre-specified (**PRESF) is either 'Y' or null].	_ALL_	**PRESF
SDTM0491	WebSDM	IR5159	Identifies records that violate the condition [route of administration (**ROUTE) is in codelist ROUTE or is null].	CLASS: INTER- VENTIONS	**ROUTE
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
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
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
SDTM0495	WebSDM	IR5173	Identifies records that violate the condition [dose units (**DOSU) is in codelist UNIT or is null].	_ALL_	**DOSU
SDTM0496	WebSDM	IR5174	Identifies records that violate the condition [Original Units (**ORRESU) is in codelist UNIT or is null].	_ALL_-VS	**ORRESU

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0497	WebSDM	IR5175	Identifies records that violate the condition [Standard Units (**STRESU) is in codelist UNIT or is null].	_ALL_-VS	**STRESU
SDTM0498	WebSDM	IR5176	Identifies records that violate the condition [location used for the measurement (**LOC) is in codelist LOC or is null].	_ALL_	**LOC
SDTM0499	WebSDM	IR5177	Identifies records that violate the condition [dosing frequency per interval (**DOSFRQ) is in codelist FREQ or is null].	CLASS: INTER- VENTIONS	**DOSFRQ
SDTM0500	WebSDM	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]
SDTM0501	WebSDM	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]
SDTM0502	WebSDM	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]
SDTM0503	WebSDM	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]
SDTM0504	WebSDM	R5007	Identifies records where value for [SEX] is not found in codelist [SEX].	DM	SEX
SDTM0505	WebSDM	R5008	Identifies records where value for [COUNTRY] is not found in codelist [COUNTRY].	DM	COUNTRY

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0506	WebSDM	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
SDTM0507	WebSDM	R5106	Identifies records that violate the condition [age units (AGEU) is not null], limited to records where AGE is not null.	DM	[AGE][AGEU]
SDTM0508	WebSDM	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
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
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
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]
SDTM0512	WebSDM	R5122	Identifies records where value for [category for disposition event (DSCAT)] is not found in codelist [DSCAT].	DS	DSCAT
SDTM0513	WebSDM	R5131	Identifies records where value for [subject characteristic short name (SCTESTCD)] is not found in codelist [SCCD].	SC	SCTESTCD
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
SDTM0515	WebSDM	R5123	Identifies records where value for [ECG Test or Examination Name (EGTEST)] is not found in Codelist [EGTEST].	EG	EGTEST

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0516	WebSDM	R5124	Identifies records where value for [ECG test or examination short name (EGTESTCD)] is not found in codelist [EGTESTCD].	EG	EGTESTCD
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
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
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
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
SDTM0531	WebSDM	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
SDTM0532	WebSDM	R5071	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES)] is not found in codelist[YESNO], limited to records where IEORES is not null.	IE	IEORRES
SDTM0533	WebSDM	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

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0534	WebSDM	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]
SDTM0541	WebSDM	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]
SDTM0561	WebSDM	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]
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
SDTM0570	WebSDM	R5114	Identifies records where value for [lab test or examination name (LBTEST)] is not found in codelist [LBTEST].	LB	LBTEST
SDTM0571	WebSDM	R5115	Identifies records where value for [lab test or examination code (LBTESTCD)] is not found in codelist [LBTESTCD].	LB	LBTESTCD
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
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

checkid	check source	sourceid	source description	tablescope	columnscope
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
SDTM0575	WebSDM	R5119	Identifies records where value for [vital signs test name (VSTEST)] is not found in codelist [VSTEST].	VS	VSTEST
SDTM0576	WebSDM	R5120	Identifies records where value for [vital signs test short name (VSTESTCD)] is not found in codelist [VSTESTCD].	VS	VSTESTCD
SDTM0580	WebSDM	R5112	Variable values should be populated with terms found in completion/reason for non-completion (C66727) CDISC controlled terminology codelist.	DS	DSDECOD
SDTM0601	SAS	SAS0013	Domain not sorted by keys as defined in standard.	_ALL_	
SDTM0602	SAS	SAS0007	Records are not unique by the expected keys	_ALL_	
SDTM0603	WebSDM	IR5004	Identifies records where non-unique values for Sequence Number variable (**SEQ) exist within a subject.	_ALL_-TS	**SEQ
SDTM0604	SAS	SAS0009	Sequence Number (**SEQ) values are not consecutively incremented beginning at 1 for each USUBJID.	TS	TSSEQ
SDTM0604	SAS	SAS0009	Sequence Number (**SEQ) values are not consecutively incremented beginning at 1 for each USUBJID.	_ALL_-TS	**SEQ
SDTM0605	SAS	SAS0014	Report any variable for the domain that contains all missing or null values.	_ALL_	_ALL_
SDTM0606	SAS	SAS0022	Identify any columns defined as numeric in the standard that contains non-numeric values.	_ALL_	
SDTM0607	SAS	SAS0038	Site study identifier (SITEID) is null for all records.	DM	SITEID

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0621	WebSDM	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
SDTM0622	WebSDM	IR5142	Inconsistency between test (**TEST) and test code (**TESTCD).	CLASS: FINDINGS	[**TEST] [**TESTCD]
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]
SDTM0631	WebSDM	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]
SDTM0641	WebSDM	R5005	Identifies records where values for Unique Subject ID (USUBJID) are not unique, limited to records where USUBJID is not null.	DM	
SDTM0642	SAS	SAS0028	Inconsistency between Description of Arm (ARM) and Arm Code (ARMCD) values across all records.	DM	[ARM] [ARMCD]
SDTM0643	SAS	SAS0016	AGE precision inconsistent across records.	DM	AGE
SDTM0644	SAS	SAS0019	STUDYID should have the same value for all records within a study.	DM	STUDYID
SDTM0645	OpenCDISC	SD1005	Study identifier (STUDYID) values must match the STUDYID in demographics (DM) domain.	[_ALL_ -DM][DM]	STUDYID
SDTM0661	WebSDM	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**	

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0662	WebSDM	IR5161	Identifies qualifier variable name (QNAM) where variable label value (Qualifier Variable Label QLABEL) is not consistent across all records.	SUPP**	[QNAM] [QLABEL]
SDTM0671	SAS	SAS0032	Inconsistency between Trial Summary Parameter (TSPARM) and Trial Summary Parameter Short Name (TSPARMCD).	TS	[TSPARM] [TSPARMCD]
SDTM0801	WebSDM	IR5500	Identifies non-demographics domain subjects (USUBJID) not found in the demographics domain.	[_ALL_ -DM][DM]	STUDYID +USUBJID
SDTM0802	WebSDM	IR5505	Identifies demographics subjects where no record for the subject is found in the disposition domain.	[DM][DS]	STUDYID +USUBJID
SDTM0803	WebSDM	IR5506	Identifies demographics subjects where no record for the subject is found in the exposure domain.	[DM][EX]	STUDYID +USUBJID
SDTM0804	WebSDM	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
SDTM0805	WebSDM	IR5502	Identifies records where the value for ARM code (ARMCD) is not found in the TA domain, excluding 'SCRNFAIL'.	[DM][TA]	ARMCD
SDTM0806	WebSDM	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
SDTM0807	SAS	SAS0039	TA domain is not provided and Planned Arm Code (ARMCD) is null for all rows in the demographics domain.	DM	ARMCD
SDTM0808	WebSDM	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]

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0809	WebSDM	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]
SDTM0811	WebSDM	IR5503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[TA][TE]	ETCD
SDTM0811	WebSDM	IR5503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain.	[SE][TE]	ETCD
SDTM0812	WebSDM	IR5516	Identifies records in exposure that should not be present since the subject has Arm Code (ARMCD)='NOTASSGN'.	[EX][DM]	USUBJID
SDTM0821	WebSDM	IR5504	Identifies records where the value for Inclusion/Exclusion Criterion Short Name (IETESTCD) is not found in the TI domain.	[IE][TI]	IETESTCD
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
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
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

checkid	check source	sourceid	source description	tablescope	columnscope
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]
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
SDTM0845	OpenCDISC	SD1015	Epoch (EPOCH) values must match the entries in the Trial Arms (TA) data set.	[_ALL_ -TA][TA]	EPOCH
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
SDTM0851	WebSDM	IR5508	Identifies comments (CO) domain reference to an unknown related domain.	CO	RDOMAIN
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
SDTM0861	WebSDM	IR5509	Identifies Related Records (RELREC) domain reference to an unknown related domain.	RELREC	RDOMAIN
SDTM0862	WebSDM	IR5510	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to an unknown related domain.	SUPP**	RDOMAIN
SDTM0863	WebSDM	IR5511	Identifies Related Records (RELREC) domain reference to a key variable that is not defined in the target domain.	RELREC	IDVAR

checkid	check source	sourceid	source description	tablescope	columnscope
SDTM0864	WebSDM	IR5512	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a key variable that is not defined in the target domain.	SUPP**	IDVAR
SDTM0865	WebSDM	IR5513	Identifies Related Records (RELREC) domain reference to a record that does not exist in the target domain.	RELREC	IDVAR
SDTM0866	WebSDM	IR5514	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a record that does not exist in the target domain.	SUPP**	IDVAR
SDTM0871	OpenCDISC	SD1007	Identifies comments (CO) domain reference to a record that does not exist in the target domain.	CO	IDVAR
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

Appendix 5

CDISC CRT-DDS 1.0 Validation Checks

This 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	_ALL_		No two values for the source column can be equivalent within the same source data set.
CRT0101	Content	_ALL_		Data is required for this field.
CRT0105	Structural	FormDefItemGroupRefs	OrderNumber	Duplicate (nonmissing) OrderNumber element found.
CRT0105	Structural	ItemGroupDefItemRefs	OrderNumber	Duplicate (nonmissing) OrderNumber element found.
CRT0105	Structural	ProtocolEventRefs	OrderNumber	Duplicate (nonmissing) OrderNumber element found.
CRT0105	Structural	StudyEventFormRefs	OrderNumber	Duplicate (nonmissing) OrderNumber element found.
CRT0105	Structural	ValueListItemRefs	OrderNumber	Duplicate (nonmissing) OrderNumber element found.
CRT0106	Content	CLItemDecodeTranslatedText	lang	The data in the _cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]

checkid	checktype	tablescope	columnscope	messagetext
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	MUtranslatedText	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	RCErrorsTranslatedText	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_].
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.

checkid	checktype	tablescope	columnscope	messagetext
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]	[CLItemDecodeTranslated 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.
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]	[FormDefItemGroupRefs. 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	[FormDefItemGroupRefs] [ItemGroupDefs]	[FormDefItemGroupRefs.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.ComputationMethodOID] [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.
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.

checkid	checktype	tablescope	columnscope	messagetext
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.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	[ItemMURRefs] [ItemDefs]	[ItemMURRefs.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemMURRefs] [MeasurementUnits]	[ItemMURRefs.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.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[ItemQuestionTranslatedText] [ItemDefs]	[ItemQuestionTranslatedText] [ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ItemRangeCheckValues] [ItemRangeChecks]	[ItemRangeCheckValues] [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.MU RefOID] [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.Val ueListOID] [ValueLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
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] [MDVLeafTitles]	[MDVLeaf.ID] [MDVLeafTitles.FK_ MDVLeaf]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.

checkid	checktype	tablescope	columnscope	messagetext
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	[RCErrTranslatedText] [ItemRangeChecks]	[RCErrTranslatedText.F ItemRangeChecks] [ItemRangeChecks.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
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.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[StudyEventFormRefs] [FormDefs]	[StudyEventFormRefs.F ormOID] [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.FileO ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[SupplementalDocs] [MDVLeaf]	[SupplementalDocs.leaf ID] [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.Rol eCodeListOID] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ValueListItemRefs] [ImputationMethods]	[ValueListItemRefs.Imp utationMethodOID] [ImputationMethods.OI D]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
CRT0110	Content	[ValueListItemRefs] [ItemDefs]	[ValueListItemRefs.Ite mOID] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2.
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.

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[ValueLists] [MetaDataVersion]	[ValueLists.FK_ MetaDataVersion] [MetaDataVersion.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.
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	[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	ItemGroupDefItemRefs	[ItemOID] [FK_ItemGroupDefs]	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.

checkid	checktype	tablescope	columnscope	messagetext
CRT0113	Content	StudyEventFormRefs	[FormOID] [FK_StudyEventDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique.
CRT0114	Content	_ALL_		Coded value is either incorrect, missing, or wrong case.

Appendix 6

CDISC ODM 1.3.0 Validation Checks

This table provides a complete list of all CDISC ODM 1.3.0 validation checks.

Table A6.1 Validation Checks

checkid	checktype	tablescope	columnscope	messagetext
ODM0100	Structural	_ALL_		No two values for the source column can be equivalent within the same source data set
ODM0101	Content	_ALL_	Any Required Columns	Data is required for this field
ODM0106	Content	CLITEMDECODETRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	CONDITIONDEFTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	FORMDEFTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	ITEMDEFTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	ITEMGROUPDEFTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]

checkid	checktype	tablescope	columnscope	messagetext
ODM0106	Content	ITEMQUESTIONTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	METHODDEFTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	MUTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	PRESENTATION	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	PROTOCOLTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	RCERRORTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0106	Content	STUDYEVENTDEFTRANSLATEDTEXT	LANG	The data in the &_cstparm1 field is improperly constructed. Must be in the form [A-Za-z-0-9]
ODM0107	Content	FORMDEFARCHLAYOUTS	PDFFILENAME	The data in the &_cstparm1 field is an improperly constructed filename. Must be in the form [A-Za-z0-9_.]
ODM0108	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_]

checkid	checktype	tablescope	columnscope	messagetext
ODM0108	Content	ITEMGROUPDEFS	SASDATASETNAME	The data in the &_cstparm1 field is an improperly constructed SAS name. Must be in the form [A-Za-z0-9_]
ODM0109	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_]
ODM0110	Content	[ADMINDATA] [ODM]	[ADMINDATA.FK_ODM] [ODM.FILEOID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ANNOTATIONFLAG] [ANNOTATION]	[ANNOTATIONFLAG] [ANNOTATION.GENERATEDID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ANNOTATIONFLAG] [CODELISTS]	[ANNOTATIONFLAG.FLAGTYPECODELISTOID] [CODELISTS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ANNOTATIONFLAG] [CODELISTS]	[ANNOTATIONFLAG.FLAGVALUECODELISTOID] [CODELISTS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ASSOCIATION] [METADATAVERSION]	[ASSOCIATION.METADATAVERSIONOID] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ASSOCIATION] [ODM]	[ASSOCIATION.FK_ODM] [ODM.FILEOID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ASSOCIATION] [STUDY]	[ASSOCIATION.STUDYOID] [STUDY.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[AUDITRECORD] [LOCATION]	[AUDITRECORD.LOCATIONOID] [LOCATION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[AUDITRECORD] [USER]	[AUDITRECORD.USEROID] [USER.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[CLINICALDATA] [METADATAVERSION]	[CLINICALDATA.METADATAVERSIONOID] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[CLINICALDATA] [ODM]	[CLINICALDATA.FK_ODM] [ODM.FILEOID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[CLINICALDATA] [STUDY]	[CLINICALDATA.STUDYOID] [STUDY.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[CLITEMDECODETRANSLATEDTEXT] [CODELISTITEMS]	[CLITEMDECODETRANSLATEDTEXT] [CODELISTITEMS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[CODELISTITEMS] [CODELISTS]	[CODELISTITEMS.FK_CODELISTS] [CODELISTS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[CODELISTS] [METADATAVERSION]	[CODELISTS.FK_METADATAVERSION] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[CONDITIONDEFFORMAL EXPRESSION] [CONDITIONDEFS]	[CONDITIONDEFFORMAL EXPRESSION] [CONDITIONDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[CONDITIONDEFS] [METADATAVERSION]	[CONDITIONDEFS.FK_ METADATAVERSION] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[CONDITIONDEFTRANSLATEDTEXT] [CONDITIONDEFS]	[CONDITIONDEFTRANSLATEDTEXT] [CONDITIONDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ENUMERATEDITEMS] [CODELISTS]	[ENUMERATEDITEMS] [CODELISTS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[EXTERNALCODELISTS] [CODELISTS]	[EXTERNALCODELISTS] [CODELISTS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDATA] [FORMDEFARCHLAYOUTS]	[FORMDATA.ARCHIVE] [FORMDEFARCHLAYOUTS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDATA] [FORMDEFS]	[FORMDATA.FORM] [FORMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDATA] [STUDYEVENTDATA]	[FORMDATA.FK_ STUDYEVENTDATA] [STUDYEVENTDATA.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDEFARCHLAYOUTS] [FORMDEFS]	[FORMDEFARCHLAYOUTS] [FORMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDEFARCHLAYOUTS] [PRESENTATION]	[FORMDEFARCHLAYOUTS.PRESENTATION] [PRESENTATION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[FORMDEFITEMGROUPPREFS] [CONDITIONDEFS]	[FORMDEFITEMGROUPPREFS.COLLECTIONEXCEPTIONCONDITIONOID] [CONDITIONDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDEFITEMGROUPPREFS] [FORMDEFS]	[FORMDEFITEMGROUPFORMDEFS] [FORMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDEFITEMGROUPPREFS] [ITEMGROUPDEFS]	[FORMDEFITEMGROUPPREFS.ITEMGROUPOID] [ITEMGROUPDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDEFS] [METADATAVERSION]	[FORMDEFS.FK_METADATAVERSION] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[FORMDEFTRANSLATEDTEXT] [FORMDEFS]	[FORMDEFTRANSLATEDFORMDEFS] [FORMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[IMPUTATIONMETHODS] [METADATAVERSION]	[IMPUTATIONMETHODSMETADATAVERSION] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMALIASES] [ITEMDEFS]	[ITEMALIASES.FK_ITEMDEFS] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMDATA] [ANNOTATION]	[ITEMDATA.ANNOTATIONID] [ANNOTATION.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMDATA] [AUDITRECORD]	[ITEMDATA.AUDITRECORDID] [AUDITRECORD.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[ITEMDATA] [ITEMDEFS]	[ITEMDATA.ITEMOID] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMDATA] [ITEMGROUPDATA]	[ITEMDATA.FK_ITEMGROUPDATA] [ITEMGROUPDATA.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMDATA] [MEASUREMENTUNITS]	[ITEMDATA.MEASUREMENTUNITOID] [MEASUREMENTUNITS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMDATA] [SIGNATURE]	[ITEMDATA.SIGNATUREID] [SIGNATURE.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMDEFS] [CODELISTS]	[ITEMDEFS.CODELISTSTREF] [CODELISTS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMDEFS] [METADATAVERSION]	[ITEMDEFS.FK_METADATAVERSION] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMDEFTRANSLATEDTEXT] [ITEMDEFS]	[ITEMDEFTRANSLATEDITEMDEFS] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPALIASES] [ITEMGROUPDEFS]	[ITEMGROUPALIASESITEMGROUPDEFS] [ITEMGROUPDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDATA] [FORMDATA]	[ITEMGROUPDATA.FORMDATA] [FORMDATA.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[ITEMGROUPDATA] [ITEMGROUPDEFS]	[ITEMGROUPDATA. ITEMGROUPOID] [ITEMGROUPDEFS. OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDATA] [REFERENCEDATA]	[ITEMGROUPDATA.FI REFERENCEDATA] [REFERENCEDATA. GENERATEDID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDEFITEMRE FS] [CODELISTS]	[ITEMGROUPDEFIT EMREFS.ROLECOD ELISTOID] [CODELISTS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDEFITEMRE FS] [CONDITIONDEFS]	[ITEMGROUPDEFIT EMREFS.COLLECTI ONEXCEPTIONCON DITIONOID] [CONDITIONDEFS.O ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDEFITEMRE FS] [IMPUTATIONMETHODS]	[ITEMGROUPDEFIT EMREFS.IMPUTATI ONMETHODOID] [IMPUTATIONMET HODS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDEFITEMRE FS] [ITEMDEFS]	[ITEMGROUPDEFIT EMREFS.ITEMOID] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDEFITEMRE FS] [ITEMGROUPDEFS]	[ITEMGROUPDEFIT ITEMGROUPDEFS] [ITEMGROUPDEFS. OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDEFITEMRE FS] [METHODDEFS]	[ITEMGROUPDEFIT EMREFS.METHOD ID] [METHODDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMGROUPDEFS] [METADATAVERSION]	[ITEMGROUPDEFS.FK METADATAVERSION] [METADATAVERSI ON.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[ITEMGROUPDEFTRANSLATEDTEXT] [ITEMGROUPDEFS]	[ITEMGROUPDEFTRANSLATEDTEXT] [ITEMGROUPDEFS] [ITEMGROUPDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMMUREFS] [ITEMDEFS]	[ITEMMUREFS.FK_ITEMDEFS] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMMUREFS] [MEASUREMENTUNITS]	[ITEMMUREFS.MEASUREMENTUNITOID] [MEASUREMENTUNITS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMQUESTIONEXTERNAL] [ITEMDEFS]	[ITEMQUESTIONEXTERNAL] [ITEMDEFS] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMQUESTIONTRANSLATEDTEXT] [ITEMDEFS]	[ITEMQUESTIONTRANSLATEDTEXT] [ITEMDEFS] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMRANGECHECKS] [ITEMDEFS]	[ITEMRANGECHECKS] [ITEMDEFS] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMRANGECHECKS] [MEASUREMENTUNITS]	[ITEMRANGECHECKS.MUREFOID] [MEASUREMENTUNITS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMRANGECHECKVALUES] [ITEMRANGECHECKS]	[ITEMRANGECHECKVALUES] [ITEMRANGECHECKS] [ITEMRANGECHECKS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[ITEMRCFORMALEXPRESSION] [ITEMRANGECHECKS]	[ITEMRCFORMALEXPRESSION] [ITEMRANGECHECKS] [ITEMRANGECHECKS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[ITEMROLE] [ITEMDEFS]	[ITEMROLE.FK_ITEMDEFS] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[KEYSET] [ASSOCIATION]	[KEYSET.FK_ASSOCIATION] [ASSOCIATION.GENERATEDID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[KEYSET] [FORMDEFS]	[KEYSET.FORMOID] [FORMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[KEYSET] [ITEMDEFS]	[KEYSET.ITEMOID] [ITEMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[KEYSET] [ITEMGROUPDEFS]	[KEYSET.ITEMGROUPOID] [ITEMGROUPDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[KEYSET] [STUDYEVENTDEFS]	[KEYSET.STUDYEVENTOID] [STUDYEVENTDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[KEYSET] [STUDY]	[KEYSET.STUDYOID] [STUDY.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[LOCATIONVERSION] [LOCATION]	[LOCATIONVERSION.LOCATION] [LOCATION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[LOCATIONVERSION] [METADATAVERSION]	[LOCATIONVERSION.METADATAVERSIONOID] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[LOCATIONVERSION] [STUDY]	[LOCATIONVERSION] N.STUDYOID] [STUDY.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[LOCATION] [ADMINDATA]	[LOCATION.FK_ ADMINDATA] [ADMINDATA.GEN ERATEDID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[MEASUREMENTUNITS] [STUDY]	[MEASUREMENTUNI STUDY] [STUDY.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[METADATAVERSION] [STUDY]	[METADATAVERSION] STUDY] [STUDY.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[METHODDEFFORMALEX PRESSION] [METHODDEFS]	[METHODDEFFORMA METHODDEFS] [METHODDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[METHODDEFS] [METADATAVERSION]	[METHODDEFS.FK_ METADATAVERSION] [METADATAVERSI ON.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[METHODDEFTRANSLAT EDTEXT] [METHODDEFS]	[METHODDEFTRANS] METHODDEFS] [METHODDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[MUTRANSLATEDTEXT] [MEASUREMENTUNITS]	[MUTRANSLATEDTE] MEASUREMENTUNIT [MEASUREMENTU NITS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[PRESENTATION] [METADATAVERSION]	[PRESENTATION.FK_ METADATAVERSION] [METADATAVERSI ON.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[PROTOCOLEVENTREFS] [CONDITIONDEFS]	[PROTOCOLEVENT REFS.COLLECTION EXCEPTIONCONDIT IONOID] [CONDITIONDEFS.O ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[PROTOCOLEVENTREFS] [METADATAVERSION]	[PROTOCOLEVENTRE METADATAVERSION] [METADATAVERSI ON.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[PROTOCOLEVENTREFS] [STUDYEVENTDEFS]	[PROTOCOLEVENT REFS.STUDYEVENT OID] [STUDYEVENTDEF S.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[PROTOCOLTRANSLATED TEXT] [METADATAVERSION]	[PROTOCOLTRANSLA METADATAVERSION] [METADATAVERSI ON.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[RCERRORTRANSLATEDT EXT] [ITEMRANGECHECKS]	[RCERRORTRANSLAT ITEMRANGECHECKS] [ITEMRANGECHEC KS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[REFERENCEDATA] [METADATAVERSION]	[REFERENCEDATA. METADATAVERSIO N.OID] [METADATAVERSI ON.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[REFERENCEDATA] [ODM]	[REFERENCEDATA.FK ODM] [ODM.FILEOID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[REFERENCEDATA] [STUDY]	[REFERENCEDATA. STUDYOID] [STUDY.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[SIGNATUREDEF] [ADMINDATA]	[SIGNATUREDEF.FK_ ADMINDATA] [ADMINDATA.GEN ERATEDID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[SIGNATURE] [LOCATION]	[SIGNATURE.LOCATIONOID] [LOCATION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[SIGNATURE] [SIGNATUREDEF]	[SIGNATURE.SIGNATUREDEFID] [SIGNATUREDEF.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[SIGNATURE] [USER]	[SIGNATURE.USEROID] [USER.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[STUDYEVENTDATA] [STUDYEVENTDEFS]	[STUDYEVENTDATA.STUDYEVENTOID] [STUDYEVENTDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[STUDYEVENTDATA] [SUBJECTDATA]	[STUDYEVENTDATA.SUBJECTDATA] [SUBJECTDATA.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[STUDYEVENTDEFS] [METADATAVERSION]	[STUDYEVENTDEFS.METADATAVERSION] [METADATAVERSION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[STUDYEVENTDEFTRANSLATEDTEXT] [STUDYEVENTDEFS]	[STUDYEVENTDEFTRANSLATEDTEXT] [STUDYEVENTDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[STUDYEVENTFORMREFS] [CONDITIONDEFS]	[STUDYEVENTFORMREFS.COLLECTIONEXCEPTIONCONDITIONOID] [CONDITIONDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[STUDYEVENTFORMREFS] [FORMDEFS]	[STUDYEVENTFORMREFS.FORMOID] [FORMDEFS.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[STUDYEVENTFORMREFS] [STUDYEVENTDEFS]	[STUDYEVENTFORM] STUDYEVENTDEFS] [STUDYEVENTDEF S.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[STUDY] [ODM]	[STUDY.FK_ODM] [ODM.FILEOID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[SUBJECTDATA] [CLINICALDATA]	[SUBJECTDATA.FK_ CLINICALDATA] [CLINICALDATA.OI D]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[SUBJECTDATA] [LOCATION]	[SUBJECTDATA.SIT EREFOID] [LOCATION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[SUBJECTDATA] [USER]	[SUBJECTDATA.IN VESTIGATORREFOI D] [USER.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[USERADDRESSSTREETN AME] [USERADDRESS]	[USERADDRESSSTRE USERADDRESS] [USERADDRESS.GE NERATEDID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[USERADDRESS] [USER]	[USERADDRESS.FK_ USER] [USER.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[USEREMAIL] [USER]	[USEREMAIL.FK_ USER] [USER.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[USERFAX] [USER]	[USERFAX.FK_ USER] [USER.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
ODM0110	Content	[USERLOCATIONREF] [LOCATION]	[USERLOCATIONREF.F.LOCATIONOID] [LOCATION.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[USERLOCATIONREF] [USER]	[USERLOCATIONREF.USER] [USER.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[USERPHONE] [USER]	[USERPHONE.FK_USER] [USER.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0110	Content	[USER] [ADMINDATA]	[USER.FK_ADMINDATA] [ADMINDATA.GENERATEDID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
ODM0111	Content	[ITEMGROUPDEFS] [ITEMGROUPDEFITEMREFS]	[ITEMGROUPDEFS.OID] [ITEMGROUPDEFITEMREFS]	Each distinct value of &_cstparm1 must have a corresponding value in the target data set &_cstparm2
ODM0111	Content	[ITEMRANGECHECKS] [ITEMRANGECHECKVALUES]	[ITEMRANGECHECKS.OID] [ITEMRANGECHECKVALUES]	Each distinct value of &_cstparm1 must have a corresponding value in the target data set &_cstparm2
ODM0112	Content	[CODELISTITEMS] [ENUMERATEDITEMS]	[CODELISTITEMS.FK_CODELISTS] [ENUMERATEDITEMS.CODELISTS]	No value in &_cstparm1 can be equal to any value in &_cstparm2
ODM0112	Content	[CODELISTITEMS] [EXTERNALCODELISTS]	[CODELISTITEMS.FK_CODELISTS] [EXTERNALCODELISTS.CODELISTS]	No value in &_cstparm1 can be equal to any value in &_cstparm2
ODM0112	Content	[EXTERNALCODELISTS] [ENUMERATEDITEMS]	[EXTERNALCODELISTS.CODELISTS] [ENUMERATEDITEMS.CODELISTS]	No value in &_cstparm1 can be equal to any value in &_cstparm2
ODM0114	Content	_ALL_		Coded value is either incorrect, missing, or wrong case.

checkid	checktype	tablescope	columnscope	messagetext
ODM0200	Content	ODM	ODMVERSION	Document based on ODM 1.3 does not have ODMVersion=1.3
ODM0201	Content	ODM	PRIORFILEOID	External file reference found. PriorFileOID is not missing. Will be ignored by CST.
ODM0202	Content	METADATAVERSION	INCLUDEDROID	MetadataVersion IncludedOID is non-null. Will be ignored by CST.
ODM0203	Content	METADATAVERSION	INCLUDEDSTUDYOID	MetadataVersion IncludedStudyOID is non-null. Will be ignored by CST.
ODM0211	Structural	FORMDEFITEMGROUPREFS	ORDERNUMBER	Duplicate (nonmissing) OrderNumber element found.
ODM0211	Structural	ITEMGROUPDEFITEMREFS	ORDERNUMBER	Duplicate (nonmissing) OrderNumber element found.
ODM0211	Structural	PROTOCOLEVENTREFS	ORDERNUMBER	Duplicate (nonmissing) OrderNumber element found.
ODM0211	Structural	STUDYEVENTFORMREFS	ORDERNUMBER	Duplicate (nonmissing) OrderNumber element found.
ODM0216	Column	ITEMDEFS	[DATATYPE] [LENGTH]	ItemDef length attribute is missing when data type is text, string, integer, or float.
ODM0217	Column	CODELISTS	[DATATYPE] [SASFORMATNAME]	A CodeList contains a SASFormatName value that does not have the correct format: CodeList datatype is text or string and the FormatName does not start with \$.
ODM0218	Metadata	ODM		ODM Data set either does not exist or has zero observations.
ODM0219	Content	ITEMDATA	[VALUE] [ITEMDATATYPE]	Invalid integer value.

checkid	checktype	tablescope	columnscope	messagetext
ODM0220	Column	ITEMDEFS	[DATATYPE] [SIGNIFICANTDIGITS]	Date type is float and SignificantDigits is missing.
ODM0221	Column	ITEMDATA	[VALUE] [ITEMDATATYPE]	Invalid float value.
ODM0222	Column	ITEMDATA	[VALUE] [ITEMDATATYPE]	Invalid date value.
ODM0223	Column	ITEMDATA	[VALUE] [ITEMDATATYPE]	Invalid time value.
ODM0224	Column	ITEMDATA	[VALUE] [ITEMDATATYPE]	Invalid datetime value.
ODM0226	Column	ITEMDATA	ITEMDATATYPE	Invalid data type found in ITEMDATA.
ODM0227	Column	CLITEMDECODETRANSLATEDTEXT	[FK_CODELISTITEMS][LANG]	Duplicate Lang tags exist.
ODM0227	Column	CONDITIONDEFTRANSLATEDTEXT	[FK_CONDITIONDEFS][LANG]	Duplicate Lang tags exist.
ODM0227	Column	FORMDEFTRANSLATEDTEXT	[FK_FORMDEFS][LANG]	Duplicate Lang tags exist.
ODM0227	Column	ITEMDEFTRANSLATEDTEXT	[FK_ITEMDEFS][LANG]	Duplicate Lang tags exist.
ODM0227	Column	ITEMGROUPDEFTRANSLATEDTEXT	[FK_ITEMGROUPDEFS][LANG]	Duplicate Lang tags exist.
ODM0227	Column	ITEMQUESTIONTRANSLATEDTEXT	[FK_ITEMDEFS][LANG]	Duplicate Lang tags exist.
ODM0227	Column	METHODDEFTRANSLATEDTEXT	[FK_METHODDEFS][LANG]	Duplicate Lang tags exist.
ODM0227	Column	MUTRANSLATEDTEXT	[FK_MEASUREMENTUNIT]	Duplicate Lang tags exist.
ODM0227	Column	PRESENTATION	[FK_METADATAVERSION]	Duplicate Lang tags exist.
ODM0227	Column	PROTOCOLTRANSLATEDTEXT	[FK_METADATAVERSION]	Duplicate Lang tags exist.
ODM0227	Column	RCERRORTRANSLATEDTEXT	[FK_ITEMRANGECHECKS]	Duplicate Lang tags exist.
ODM0227	Column	STUDYEVENTDEFTRANSLATEDTEXT	[FK_STUDYEVENTDEFS][LANG]	Duplicate Lang tags exist.

checkid	checktype	tablescope	columnscope	messagetext
ODM0228	Column	ITEMDATA	[ITEMDATATYPE] [MEASUREMENTU NITOID]	Non-numeric data contains measurement unit.

Appendix 7

CDISC ADaM 2.1 Validation Checks

This table provides a complete list of all CDISC ADaM 2.1 validation checks.

Table A7.1 Validation Checks

checkid	source description	table scope	column scope
ADAM0001	There is only one ADSL per study	ADSL	
ADAM0002	Any ADaM variable whose name is the same as an SDTM variable must be a copy of the SDTM variable, and its label and values must not be modified	_ALL_	
ADAM0003	Any ADaM variable whose name is the same as an SDTM variable must be a copy of the SDTM variable, and its label and values must not be modified	_ALL_	
ADAM0004	Any ADaM variable whose name is the same as an SDTM variable must be a copy of the SDTM variable, and its label and values must not be modified	_ALL_	
ADAM0005	The names of all other character flag (or indicator) variables end in FL	_ALL_	**FL
ADAM0006	The names of the corresponding numeric flag (or indicator) variables end in FN	_ALL_	**FN
ADAM0007	If the numeric flag is used, the character version (*FL) is required	_ALL_	*FN+*FL
ADAM0010	*FN and *FL must be a one-to-one mapping	_ALL_	{**FL}{**FN}
ADAM0011	*FN and *FL must be a one-to-one mapping	_ALL_	{**FL}{**FN}
ADAM0012	*FN and *FL must be a one-to-one mapping	_ALL_	{**FL}{**FN}

checkid	source description	table scope	column scope
ADAM0013	ADaM variable names must be no more than 8 characters in length	_ALL_	_ALL_
ADAM0014	ADaM variable names must start with a letter (not underscore), and be comprised only of letters (A-Z), underscore (_), and numerals (0-9)	_ALL_	_ALL_
ADAM0015	ADaM variable names must start with a letter (not underscore), and be comprised only of letters (A-Z), underscore (_), and numerals (0-9)	_ALL_	_ALL_
ADAM0016	All ADaM variable labels must be no more than 40 characters in length	_ALL_	_ALL_
ADAM0017	All ADaM character variables must be no more than 200 characters in length	_ALL_	_ALL_
ADAM0018	In general, the variable labels specified in the tables in Section 3 are required. There are only two exceptions to this rule (1) descriptive text is allowed at the end of the labels of variables whose names contain indexes "y", "xx", or "zz"; and (2) asterisks (*) and ellipses (...) in specified variable labels should be replaced by the sponsor with appropriate text	_ALL_	_ALL_
ADAM0019	For subject-level character population flag variables: N = no (not included in the population), Y = yes (included). Null values are not allowed	ADSL	COMPLFL
ADAM0020	For subject-level character population flag variables: N = no (not included in the population), Y = yes (included). Null values are not allowed	ADSL	FASFL
ADAM0021	For subject-level character population flag variables: N = no (not included in the population), Y = yes (included). Null values are not allowed	ADSL	ITTFL
ADAM0022	For subject-level character population flag variables: N = no (not included in the population), Y = yes (included). Null values are not allowed	ADSL	PPROTFL

checkid	source description	table scope	column scope
ADAM0023	For subject-level character population flag variables: N = no (not included in the population), Y = yes (included). Null values are not allowed.	ADSL	SAFFL
ADAM0024	For subject-level character population flag variables: N = no (not included in the population), Y = yes (included). Null values are not allowed.	ADSL	RANDFL
ADAM0025	For subject-level character population flag variables: N = no (not included in the population), Y = yes (included). Null values are not allowed.	ADSL	ENRLFL
ADAM0026	For subject-level numeric population flag variables: 0 = no (not included), 1 = yes (included). Null values are not allowed.	ADSL	COMPLFN
ADAM0027	For subject-level numeric population flag variables: 0 = no (not included), 1 = yes (included). Null values are not allowed.	ADSL	FASFN
ADAM0028	For subject-level numeric population flag variables: 0 = no (not included), 1 = yes (included). Null values are not allowed.	ADSL	ITTFN
ADAM0029	For subject-level numeric population flag variables: 0 = no (not included), 1 = yes (included). Null values are not allowed.	ADSL	PPROTFN
ADAM0030	For subject-level numeric population flag variables: 0 = no (not included), 1 = yes (included). Null values are not allowed.	ADSL	SAFFN
ADAM0031	For subject-level numeric population flag variables: 0 = no (not included), 1 = yes (included). Null values are not allowed.	ADSL	RANDFN
ADAM0032	For subject-level numeric population flag variables: 0 = no (not included), 1 = yes (included). Null values are not allowed.	ADSL	ENRLFN

checkid	source description	table scope	column scope
ADAM0033	For record-level character population flag variables: Y = yes (included). Null values are allowed	_ALL_-ADSL	**RFL
ADAM0034	For parameter-level character population flag variables: Y = yes (included). Null values are allowed	_ALL_-ADSL	**PFL
ADAM0035	For record-level numeric population flag variables: 1 = yes (included). Null values are allowed	_ALL_-ADSL	**RFN
ADAM0036	For parameter-level numeric population flag variables: 1 = yes (included). Null values are allowed	_ALL_-ADSL	**PFN
ADAM0037	The *GRy and associated *GRyN variable must have a one-to-one relationship	ADSL	{**GR#}{**GR#N}
ADAM0038	The *GRy and associated *GRyN variable must have a one-to-one relationship	ADSL	{**GR#}{**GR#N}
ADAM0039	The names of date imputation flag variables end in DTF	_ALL_	**DTF
ADAM0040	The names of time imputation flag variables end in TMF	_ALL_	**TMF
ADAM0041	Numeric dates, times and datetimes should be formatted so as to be human readable with no loss of precision	_ALL_	**DT
ADAM0042	Numeric dates, times and datetimes should be formatted so as to be human readable with no loss of precision	_ALL_	**TM-**DTM
ADAM0043	Numeric dates, times and datetimes should be formatted so as to be human readable with no loss of precision	_ALL_	**DTM
ADAM0044	If a *DTM and associated *TM variable exist, then the *TM variable must match the time part of the *DTM variable	_ALL_	[**DTM][**TM-**DTM]
ADAM0045	If a *DTM and associated *DT variable exist, then the *DT variable must match the date part of the *DTM variable	_ALL_	[**DTM][**DT]

checkid	source description	table scope	column scope
ADAM0046	*DY cannot = 0	_ALL_-ADSL	**DY
ADAM0047	ADSL must have the variables SUBJID, SITEID, AGE, AGEU, SEX, RACE, ARM	ADSL	SITEID
ADAM0048	ADSL must have at least one variable that ends in FL because you need at least one population flag	ADSL	**FL
ADAM0049	ADSL must have the variables SUBJID, SITEID, AGE, AGEU, SEX, RACE, ARM	ADSL	AGE
ADAM0050	ADSL must have the variables SUBJID, SITEID, AGE, AGEU, SEX, RACE, ARM	ADSL	AGEU
ADAM0051	ADSL must have the variables SUBJID, SITEID, AGE, AGEU, SEX, RACE, ARM	ADSL	SEX
ADAM0052	ADSL must have the variables SUBJID, SITEID, AGE, AGEU, SEX, RACE, ARM	ADSL	RACE
ADAM0053	Invalid STUDYID/USUBJID combination not found in the SDTM Demographics domain	_ALL_	STUDYID+USUBJID
ADAM0054	ADSL is one record per USUBJID	ADSL	
ADAM0055	ADSL must have the variables SUBJID, SITEID, AGE, AGEU, SEX, RACE, ARM	ADSL	SUBJID
ADAM0058	All *DT variables must be numeric	_ALL_	**DT
ADAM0059	All *TM variables must be numeric	_ALL_	**TM
ADAM0061	TRTSDT or TRTSDTM variables are required if there is an investigational product	ADSL	TRTSDT+TRTSDTM
ADAM0062	Any ADSL variable beginning with TRT and ending in A or beginning in TRT and ending in AN must include xx where xx is a two-digit number	ADSL	TRT**A-TRTSEQA
ADAM0063	Any ADSL variable beginning with TRT and ending in A or beginning in TRT and ending in AN must include xx where xx is a two-digit number	ADSL	TRT**AN-TRTSEQAN

checkid	source description	table scope	column scope
ADAM0064	Any variable beginning with TRT and ending in AN must have a corresponding variable beginning with TRT, having the same increment and end in A	ADSL	[TRT##AN][TRT##A]
ADAM0065	Any ADSL variable beginning with TR and containing PG must have a padded numeric of 01 and increment	ADSL	TR**PG
ADAM0066	Any variable beginning with TR, containing PG and ending in N must have a corresponding variable beginning with TR, containing PG, and having the same increment	ADSL	TR**
ADAM0068	Any ADSL variable beginning with TR and containing AG must have numeric where xx is a two-digit number	ADSL	TR**AG
ADAM0069	Any variable beginning with TR, containing AG and ending in N must have a corresponding variable beginning with TR, containing AG, and having the same increment	ADSL	TR**
ADAM0070	Any variable beginning with TR, containing AG and ending in N must have a corresponding variable beginning with TR, containing AG, and having the same increment	ADSL	TR**
ADAM0071	ADSL must have the variables SUBJID, SITEID, AGE, AGEU, SEX, RACE, ARM	ADSL	ARM
ADAM0072	ADSL must have at least one TRTxxP variable	ADSL	TRT**P
ADAM0073	Any ADSL variable beginning with TRT and ending in P must have a suffix that is a two-digit integer [01-99]	ADSL	TRT**P-TRTSEQP
ADAM0074	Any ADSL variable beginning with TRT and ending in PN must have a suffix that is a two-digit integer [01-99]	ADSL	TRT**PN-TRTSEQPN
ADAM0075	Any variable beginning with TRT and ending in PN must have a corresponding variable beginning with TRT, having the same increment and end in P	ADSL	[TRT##PN][TRT##P]

checkid	source description	table scope	column scope
ADAM0076	Any variable beginning with TRT and ending in PN must have a corresponding variable beginning with TRT, having the same increment and end in P	ADSL	[TRT##PN][TRT##P]
ADAM0078	If there is more than one treatment period, then TRxxSDT and TRxxEDT should exist in ADSL	ADSL	[TRT##P][TR##SDT]
ADAM0079	If there is more than one treatment period, then TRxxSDT and TRxxEDT should exist in ADSL	ADSL	[TRT##P][TR##EDT]
ADAM0080	If TRTxxA exists, then TRTxxP should exist	ADSL	[TRT##A][TRT##P]
ADAM0081	If TRT(xx+1)P exists, then TRTxxP should exist for xx+1 > 01	ADSL	TRT**P
ADAM0083	If TR01SDT exists, then TRTSDT=TR01SDT	ADSL	[TR01SDT][TRTSDT]
ADAM0084	TRTEDT = maximum(TRxxEDT)	ADSL	{TRTEDT} {TR##EDT}
ADAM0085	BDS variables STUDYID, USUBJID, SUBJID, SITEID, and so on, must match ADSL variable in metadata	_ALL_	_ALL_
ADAM0086	BDS variables STUDYID, USUBJID, SUBJID, SITEID, and so on, must match ADSL variable in metadata	_ALL_	_ALL_
ADAM0087	BDS variables STUDYID, USUBJID, SUBJID, SITEID, and so on, must match ADSL variable in metadata	_ALL_	_ALL_
ADAM0088	All data sets must have the variables STUDYID and USUBJID	_ALL_	STUDYID
ADAM0089	All data sets must have the variables STUDYID and USUBJID	_ALL_	USUBJID
ADAM0090	BDS must have TRTP variable	_ALL_-ADSL	TRTP
ADAM0091	TRTP must match at least one value in TRT01P-TRTxxP	[_ALL_-ADSL] [ADSL]	{_cstList:USUBJID+TRT##P} {_cstList:USUBJID+TRTP}
ADAM0092	TRTPN must be a one-to-one match to TRTP	_ALL_-ADSL	[TRTP][TRTPN]

checkid	source description	table scope	column scope
ADAM0094	Any variable beginning with TRTPG must have a suffix of 1 and increment up to 9	_ALL_	TRTPG#
ADAM0095	TRTAN must be a one-to-one match to TRTA	_ALL_-ADSL	[TRTA][TRTAN]
ADAM0097	Any variable beginning with TRTPG and ending in N must have a corresponding variable beginning with TRTPG, and having the same increment	_ALL_-ADSL	[TRTPG#N][TRTPG#]
ADAM0098	*SDY is less than or equal to *EDY if both are nonmissing	_ALL_-ADSL	[**SDY][**EDY]
ADAM0099	*STDY is less than or equal to *ENDY if both are nonmissing	_ALL_-ADSL	[**STDY][**ENDY]
ADAM0100	APEREDT must have corresponding APxxEDT value	[_ALL_-ADSL] [ADSL]	{_cstlist:APEREDT+APERIOD} {_cstlist:AP##EDT}
ADAM0101	APEREDTM must have corresponding APxxEDTM value	[_ALL_-ADSL] [ADSL]	{_cstlist:APEREDTM+APERIOD} {_cstlist:AP##EDTM}
ADAM0102	APERIOD value must have corresponding TRTxxP/TRxxSDT/TRxxEDT variables	[_ALL_-ADSL] [ADSL]	{_cstlist:APERIOD} {_cstlist:TRT##P}
ADAM0103	APERIOD value must have corresponding TRTxxP/TRxxSDT/TRxxEDT variables	[_ALL_-ADSL] [ADSL]	{_cstlist:APERIOD} {_cstlist:TR##SDT}
ADAM0104	APERIOD value must have corresponding TRTxxP/TRxxSDT/TRxxEDT variables	[_ALL_-ADSL] [ADSL]	{_cstlist:APERIOD} {_cstlist:TR##EDT}
ADAM0105	APERIODC must have one-to-one mapping with APERIOD	_ALL_-ADSL	[APERIOD][APERIODC]
ADAM0107	APERSDT must have corresponding APxxSDT value	[_ALL_-ADSL] [ADSL]	{_cstlist:APERSDT+APERIOD} {_cstlist:AP##SDT}
ADAM0108	APERSDTM must have corresponding APxxSDTM value	[_ALL_-ADSL] [ADSL]	{_cstlist:APERSDTM+APERIOD} {_cstlist:AP##SDTM}
ADAM0109	AVISITN is a one-to-one mapping with AVISIT	_ALL_-ADSL	[AVISITN][AVISIT]

checkid	source description	table scope	column scope
ADAM0111	When ARELTM is present, the anchor time variable and ARELTMU must also be included in the data set, and the anchor time variable must be identified in the metadata for ARELTM	_ALL_-ADSL	[ARELTM][ARELTMU]
ADAM0112	When ARELTM is present, the anchor time variable and ARELTMU must also be included in the data set, and the anchor time variable must be identified in the metadata for ARELTM	_ALL_-ADSL	[ARELTM][ARELTMU]
ADAM0113	When ARELTM is present, the anchor time variable and ARELTMU must also be included in the data set, and the anchor time variable must be identified in the metadata for ARELTM	_ALL_-ADSL	[ARELTMU][ARELTM]
ADAM0114	When ARELTM is present, the anchor time variable and ARELTMU must also be included in the data set, and the anchor time variable must be identified in the metadata for ARELTM	_ALL_-ADSL	[ARELTMU][ARELTM]
ADAM0115	If ATPTREF is populated, then ATPPT must be populated	_ALL_-ADSL	[ATPTREF][ATPT]
ADAM0116	If ATPPTREF is populated, then ATPPT must be populated	_ALL_-ADSL	[ATPT][ATPTREF]
ADAM0117	Within the same parameter there must be a one-to-one mapping between ATPPT and ATPPTN if both variables are present	_ALL_-ADSL	[ATPT][ATPTN]
ADAM0121	If *SDT and *EDT are nonmissing, then *SDT <= *EDT	_ALL_	[**SDT][**EDT]
ADAM0122	If *SDTM and *EDTM are nonmissing, then *SDTM <= *EDTM	_ALL_	[**SDTM][**EDTM]
ADAM0123	PARAMTYP has the same value for all records within a parameter	_ALL_-ADSL	[PARAMTYP][PARAMCD]
ADAM0124	PARCATy has the same value for all records within a parameter	_ALL_-ADSL	[PARCAT#][PARAMCD]
ADAM0125	PARCATy and PARCATyN have a one-to-one mapping	_ALL_-ADSL	[PARCAT#][PARCAT#N]

checkid	source description	table scope	column scope
ADAM0127	If BASE is populated, then there must be a corresponding AVAL value with ABLFL=Y	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+ BASE+ABLFL}
ADAM0128	If BASEC is populated, then there must be a corresponding AVALC value with ABLFL=Y	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+ BASEC+ABLFL}
ADAM0129	If both BASE and BASEC are populated, then there must be a one-to-one mapping	_ALL_-ADSL	[BASE][BASEC]
ADAM0131	If BASETYPE is populated for at least one record within a parameter, then it must be populated for all records within that parameter	_ALL_-ADSL	[PARAMCD][BASETYPE]
ADAM0132	R2BASE must equal AVAL/BASE	_ALL_-ADSL	{_cstList:R2BASE+AVAL+BASE}
ADAM0133	R2AyLO must equal AVAL/AyLO	_ALL_-ADSL	{_cstList:R2A#LO} {_cstList:A#LO}
ADAM0134	R2AyHI must equal AVAL/AyHI	_ALL_-ADSL	{_cstList:R2A#HI} {_cstList:A#HI}
ADAM0135	SHIFTyN must be a one-to-one mapping with SHIFTy	_ALL_-ADSL	[SHIFT#N][SHIFT#]
ADAM0137	If CRITyFL is populated, then CRITy must be populated	_ALL_-ADSL	[CRIT#FL][CRIT#]
ADAM0138	If CRITyFL is populated, then CRITy must be populated	_ALL_-ADSL	[CRIT#][CRIT#FL]
ADAM0141	PARAM and PARAMCD are present and have a one-to-one mapping	_ALL_-ADSL	[PARAM][PARAMCD]
ADAM0143	PARAMCD values should follow SAS V5 variable naming conventions	_ALL_-ADSL	PARAMCD
ADAM0145	PARAMCD values should follow SAS V5 variable naming conventions	_ALL_-ADSL	PARAMCD
ADAM0146	PARAMN is a one-to-one mapping with PARAM if present	_ALL_-ADSL	[PARAM][PARAMN]
ADAM0148	PARAMN must be an integer	_ALL_-ADSL	PARAMN
ADAM0149	If both AVAL and AVALC are populated, then there must be a one-to-one mapping	_ALL_-ADSL	[AVAL][AVALC]

checkid	source description	table scope	column scope
ADAM0151	The values of CRITy within a parameter must be constant on all rows on which it is populated	_ALL_-ADSL	{_cstList:PARAMCD+CRIT#}
ADAM0152	If BASE is populated, then it must be with a value flagged for some record via ABLFL for that parameter	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+BASETYPE+BASE+AVAL+ABLFL}
ADAM0153	If there are multiple baseline records flagged for a given parameter within a subject, then BASETYPE should be populated and contain different values for the baseline records within a subject	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+BASETYPE+ABLFL}
ADAM0154	If there are multiple baseline records flagged for a given parameter within a subject, then BASETYPE should be populated and contain different values for the baseline records within a subject	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+BASETYPE+ABLFL}
ADAM0155	Whenever there is more than one definition of baseline, the BASETYPE column is required. BASETYPE identifies the definition of baseline that corresponds to the value of BASE in each row. There is only one BASE column, and only one column for each qualifying function of AVAL and BASE	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+ ABLFL}
ADAM0156	Variable CRITyFL must be present on the data set if variable CRITy is present, and vice versa.	_ALL_-ADSL	[CRIT#FL][CRIT#]
ADAM0157	Variable CRITyFL must be present on the data set if variable CRITy is present, and vice versa	_ALL_-ADSL	[CRIT#][CRIT#FL]
ADAM0158	If AWTDIFF is present, then AWTARGET must be present	_ALL_-ADSL	[AWTDIFF][AWTARGET]
ADAM0159	If AWTDIFF is populated, then AWTARGET must be populated	_ALL_-ADSL	[AWTDIFF][AWTARGET]
ADAM0160	If AWU is present, then AWLO and AWHI must be present	_ALL_-ADSL	{_cstList:AWU+AWLO+AWHI}
ADAM0161	If AWU is populated, then AWLO and AWHI must be populated	_ALL_-ADSL	{_cstList:AWU+AWLO+AWHI}
ADAM0162	*LO must be less than or equal to *HI	_ALL_-ADSL	[**LO][**HI]

checkid	source description	table scope	column scope
ADAM0163	If BTOXGR is present, then ATOXGR and ABLFL must be present	_ALL_-ADSL	[BTOXGR][ATOXGR]
ADAM0164	If BTOXGR is present, then ATOXGR and ABLFL must be present	_ALL_-ADSL	[BTOXGR][ABLFL]
ADAM0165	If BTOXGR is populated, then there must be a corresponding ATOXGR value with ABLFL=Y	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+BASETYPE+BTOXGR+ATOXGR+ABLFL}
ADAM0166	If BNRIND is present, then ANRIND and ABLFL must be present	_ALL_-ADSL	[BNRIND][ANRIND]
ADAM0167	If BNRIND is present, then ANRIND and ABLFL must be present	_ALL_-ADSL	[BNRIND][ABLFL]
ADAM0168	If BNRIND is populated, then there must be a corresponding ANRIND value with ABLFL=Y	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+BASETYPE+BNRIND+ANRIND+ABLFL}
ADAM0169	CNSR should be an integer	_ALL_-ADSL	CNSR
ADAM0170	If STARTDT is present, then CNSR must be present	_ALL_-ADSL	[STARTDT][CNSR]
ADAM0171	If ONTRTFL ^=Y then TRTSDT <= ADT <= TRTEDT should not be true	[_ALL_-ADSL] [ADSL]	{_cstList:USUBJID+ONTRTFL+ADT} {_cstList:USUBJID+TRTSDT}
ADAM0172	If ONTRTFL ^=Y then TRTSDT <= ADT <= TRTEDT should not be true	[_ALL_-ADSL] [ADSL]	{_cstList:USUBJID+ONTRTFL+ADT} {_cstList:USUBJID+TRTEDT}
ADAM0173	If ONTRTFL ^=Y then TRTSDT <= ADT <= TRTEDT should not be true	[_ALL_-ADSL] [ADSL]	{_cstList:USUBJID+ONTRTFL+ADT} {_cstList:USUBJID+TRTSDT+TRTEDT}
ADAM0174	If LVOTFL=Y then TRTSDT <= ADT <= TRTEDT should be true	[_ALL_-ADSL] [ADSL]	{_cstList:USUBJID+LVOTFL+ADT} {_cstList:USUBJID+TRTSDT}
ADAM0175	A maximum of one record within a parameter timepoint can have LVOTFL=Y	_ALL_-ADSL	[PARAMCD][LVOTFL]
ADAM0176	ABLFL must have a value of Y or null	_ALL_-ADSL	ABLFL
ADAM0177	A maximum of one record within a parameter can have ABLFL=Y	_ALL_-ADSL	[BASETYPE][ABLFL]

checkid	source description	table scope	column scope
ADAM0178	ANLzzFL must have a value of Y or null	_ALL_-ADSL	ANL##FL
ADAM0179	If LVOTFL=Y then TRTSDT <= ADT <= TRTEDT should be true	[_ALL_-ADSL] [ADSL]	{_cstList:USUBJID+LVOTFL+ADT} {_cstList:USUBJID+TRTEDT}
ADAM0180	The value of SRCDOM should reference a valid SDTM domain.	_ALL_-ADSL	SRCDOM
ADAM0181	If BASE is populated, then it must be with a value flagged for some record via ABLFL for that parameter	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+ BASE +AVAL+ABLFL}
ADAM0182	If BTOXGR is populated, then there must be a corresponding ATOXGR value with ABLFL=Y	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+ BTOXGR +ATOXGR+ABLFL}
ADAM0183	If BNRIND is populated, then there must be a corresponding ANRIND value with ABLFL=Y	_ALL_-ADSL	{_cstList:USUBJID+PARAMCD+ BNRIND +ANRIND+ABLFL}

Appendix 8

ODM 1.3.0 SAS Data Sets

Conventions Used in the Tables	449
ODM SAS Data Set	449
Study SAS Data Set	450
MeasurementUnits SAS Data Set	450
MUTranslatedText SAS Data Set	450
MetaDataVersion SAS Data Set	451
ProtocolEventRefs SAS Data Set	451
ProtocolTranslatedText SAS Data Set	452
StudyEventDefs SAS Data Set	452
StudyEventDefTranslatedText SAS Data Set	452
StudyEventFormRefs SAS Data Set	453
FormDefs SAS Data Set	453
FormDefTranslatedText SAS Data Set	454
FormDefItemGroupRefs SAS Data Set	454
FormDefArchLayouts SAS Data Set	454
ItemGroupDefs SAS Data Set	455
ItemGroupDefTranslatedText SAS Data Set	455
ItemGroupDefItemRefs SAS Data Set	456
ItemGroupAliases SAS Data Set	456
ItemDefs SAS Data Set	457
ItemDefTranslatedText SAS Data Set	457
ItemQuestionTranslatedText SAS Data Set	458
ItemQuestionExternal SAS Data Set	458
ItemMUREfs SAS Data Set	459
ItemRangeChecks SAS Data Set	459
ItemRangeCheckValues SAS Data Set	459
RCErrorsTranslatedText SAS Data Set	460
ItemRCFormalExpression SAS Data Set	460

ItemRole SAS Data Set	460
ItemAliasesSAS Data Set	461
CodeLists SAS Data Set	461
ExternalCodeLists SAS Data Set	461
EnumeratedItems SAS Data Set	462
CodeListItems SAS Data Set	462
CLItemDecodeTranslatedText SAS Data Set	463
ImputationMethods SAS Data Set	463
Presentation SAS Data Set	463
MethodDefs SAS Data Set	464
MethodDefTranslatedText SAS Data Set	464
MethodDefFormalExpression SAS Data Set	464
ConditionDefs SAS Data Set	465
ConditionDefTranslatedText SAS Data Set	465
ConditionDefFormalExpression SAS Data Set	465
ClinicalData SAS Data Set	466
SubjectDataSAS Data Set	466
StudyEventData SAS Data Set	466
FormData SAS Data Set	467
ItemGroupData SAS Data Set	467
ItemData SAS Data Set	468
ReferenceData SAS Data Set	469
AdminData SAS Data Set	469
User SAS Data Set	469
UserAddress SAS Data Set	470
UserAddressStreetName SAS Data Set	470
UserEmail SAS Data Set	471
UserFax SAS Data Set	471
UserPhone SAS Data Set	471
UserLocationRef SAS Data Set	472
Location SAS Data Set	472
LocationVersion SAS Data Set	472
SignatureDef SAS Data Set	473
AuditRecord SAS Data Set	473
Signature SAS Data Set	474
Annotation SAS Data Set	474
AnnotationFlag SAS Data Set	475

Association SAS Data Set	475
KeySet SAS Data Set	475

Conventions Used in the Tables

The tables that describe the SAS data sets use these conventions:

- The names of key variables are appended with two asterisks (**). Some data sets do not have a key.
- Foreign key variable names are prepended with two caret 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.
- All data set names and column names are case sensitive. They must be specified exactly as shown.

ODM SAS Data Set

Table A8.1 ODM SAS Data Set

Variable Name	SAS Data Type	Length (if char)
**FileOID	character	64 (OID)
Archival	character	3
AsOfDateTime	character	32
CreationDateTime	character	32
Description	character	2000 (text)
FileType	character	13
Granularity	character	15
Id	character	64 (OID)
ODMVersion	character	2000 (text)
Originator	character	2000 (text)
PriorFileOID	character	64 (OID)
SourceSystem	character	2000 (text)

Variable Name	SAS Data Type	Length (if char)
SourceSystemVersion	character	2000 (text)

Study SAS Data Set

Table A8.2 Study SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
StudyName	character	128 (name)
StudyDescription	character	2000 (text)
ProtocolName	character	128 (name)
^^FK_ODM [ODM]	character	64 (OID)

MeasurementUnits SAS Data Set

Table A8.3 MeasurementUnits SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)

MUTranslatedText SAS Data Set

Table A8.4 MUTranslatedText SAS Data Set

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)
lang	character	128 (name)
^^FK_MeasurementUnits [MeasurementUnits]	character	64 (OID)

Variable Name	SAS Data Type	Length (if character)
^^FK_Study [Study]	character	64 (OID)
Name	character	128 (name)

MetaDataVersion SAS Data Set

Table A8.5 MetaDataVersion SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Name	character	128 (name)
Description	character	2000 (text)
IncludedOID	character	64 (OID)
IncludedStudyOID	character	64 (OID)
^^FK_Study [Study]	character	64 (OID)

ProtocolEventRefs SAS Data Set

Table A8.6 ProtocolEventRefs SAS Data Set

Variable Name	SAS Data Type	Length (if character)
Mandatory	character	3
OrderNumber	numeric	8
^^StudyEventOID [StudyEventDefs]	character	64 (OID)
^^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

ProtocolTranslatedText SAS Data Set

Table A8.7 *ProtocolTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)
lang	character	17
^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)
^CollectionExceptionCon ditionOID [ConditionDefs]	character	64 (OID)

StudyEventDefs SAS Data Set

Table A8.8 *StudyEventDefs SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Category	character	2000 (text)
Name	character	128 (name)
Repeating	character	3
Type	character	11
^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

StudyEventDefTranslatedText SAS Data Set

Table A8.9 *StudyEventDefTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)

Variable Name	SAS Data Type	Length (if character)
lang	character	17
^^FK_StudyEventDefs [StudyEventDefs]	character	64 (OID)

StudyEventFormRefs SAS Data Set

Table A8.10 StudyEventFormRefs SAS Data Set

Variable Name	SAS Data Type	Length (if character)
^^FormOID [FormDefs]	character	64 (OID)
Mandatory	character	3
OrderNumber	numeric	8
^^FK_StudyEventDefs [StudyEventDefs]	character	64 (OID)
^^CollectionExceptionCon ditionOID [ConditionDefs]	character	64 (OID)

FormDefs SAS Data Set

Table A8.11 FormDefs SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Name	character	128 (name)
Repeating	character	3
^^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

FormDefTranslatedText SAS Data Set

Table A8.12 *FormDefTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)
lang	character	17
^FK_FormDefs [FormDefs]	character	64 (OID)

FormDefItemGroupRefs SAS Data Set

Table A8.13 *FormDefItemGroupRefs SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
^ItemGroupOID [ItemGroupDefs]	character	64 (OID)
Mandatory	character	3
OrderNumber	numeric	8
^FK_FormDefs [FormDefs]	character	64 (OID)
^^CollectionExceptionCon ditionOID [ConditionDefs]	character	64 (OID)

FormDefArchLayouts SAS Data Set

Table A8.14 *FormDefArchLayouts SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
PdfFileName	character	512 (path)

Variable Name	SAS Data Type	Length (if character)
^^PresentationOID [Presentation]	character	64 (OID)
^^FK_FormDefs [FormDefs]	character	64 (OID)

ItemGroupDefs SAS Data Set

Table A8.15 ItemGroupDefs SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Name	character	128 (name)
Repeating	character	3
IsReferenceData	character	3
SASDatasetName	character	8
Domain	character	2000 (text)
Origin	character	2000 (text)
Role	character	128 (name)
Purpose	character	2000 (text)
Comment	character	2000 (text)
^^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

ItemGroupDefTranslatedText SAS Data Set

Table A8.16 ItemGroupDefTranslatedText SAS Data Set

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)

Variable Name	SAS Data Type	Length (if character)
lang	character	17
^^FK_ItemGroupDefs [ItemGroupDefs]	character	64 (OID)

ItemGroupDefItemRefs SAS Data Set

Table A8.17 *ItemGroupDefItemRefs SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
^^ItemOID [ItemDefs]	character	64 (OID)
Mandatory	character	3
OrderNumber	numeric	8
KeySequence	numeric	8
^^ImputationMethodOID [ImputationMethods]	character	64 (OID)
^^MethodOID [MethodDefs]	character	64 (OID)
Role	character	128 (name)
^^RoleCodeListOID [CodeLists]	character	64 (OID)
^^FK_ItemGroupDefs [ItemGroupDefs]	character	64 (OID)

ItemGroupAliases SAS Data Set

Table A8.18 *ItemGroupAliases SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
Context	character	2000 (text)
Name	character	2000 (text)

Variable Name	SAS Data Type	Length (if character)
^^FK_ItemGroupDefs [ItemGroupDefs]	character	64 (OID)
^^CollectionExceptionCon ditionOID [ConditionDefs]	character	64 (OID)

ItemDefs SAS Data Set

Table A8.19 ItemDefs SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Name	character	128 (name)
DataType	character	18
Length	numeric	8
SignificantDigits	numeric	8
SASFieldName	character	8
SDSVarName	character	8
Origin	character	2000 (text)
Comment	character	2000 (text)
^^CodeListRef [CodeLists]	character	64 (OID)
^^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

ItemDefTranslatedText SAS Data Set

Table A8.20 ItemDefTranslatedText SAS Data Set

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)

Variable Name	SAS Data Type	Length (if character)
lang	character	17
^^FK_ItemDefs [ItemDefs]	character	64 (OID)

ItemQuestionTranslatedText SAS Data Set

Table A8.21 *ItemQuestionTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)
lang	character	17
^^FK_ItemDefs [ItemDefs]	character	64 (OID)

ItemQuestionExternal SAS Data Set

Table A8.22 *ItemQuestionExternal SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
Dictionary	character	2000 (text)
Version	character	2000 (text)
Code	character	2000 (text)
^^FK_ItemDefs [ItemDefs]	character	64 (OID)

ItemMURefs SAS Data Set

Table A8.23 ItemMURefs SAS Data Set

Variable Name	SAS Data Type	Length (if character)
^^MeasurementUnitOID [MeasurementUnits]	character	64 (OID)
^^FK_ItemDefs [ItemDefs]	character	64 (OID)

ItemRangeChecks SAS Data Set

Table A8.24 ItemRangeChecks SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Comparator	character	5
SoftHard	character	4
^^MURefOID [MeasurementUnits]	character	64 (OID)
^^FK_ItemDefs [ItemDefs]	character	64 (OID)

ItemRangeCheckValues SAS Data Set

Table A8.25 ItemRangeCheckValues SAS Data Set

Variable Name	SAS Data Type	Length (if character)
CheckValue	character	512 (value)
^^FK_ItemRangeChecks [ItemRangeChecks]	character	64 (OID)

RCErrTranslatedText SAS Data Set

Table A8.26 *RCErrTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)
lang	character	17
^FK_ItemRangeChecks [ItemRangeChecks]	character	64 (OID)

ItemRCFormalExpression SAS Data Set

Table A8.27 *ItemRCFormalExpression SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
Context	character	2000 (text)
Expression	character	2000 (text)
^FK_ItemRangeChecks [ItemRangeChecks]	character	64 (OID)

ItemRole SAS Data Set

Table A8.28 *ItemRole SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
Name	character	2000 (text)
^FK_ItemDefs [ItemDefs]	character	64 (OID)

ItemAliasesSAS Data Set

Table A8.29 *ItemAliasesSAS Data Set*

Variable Name	SAS Data Type	Length (if character)
Context	character	2000 (text)
Name	character	2000 (text)
^FK_ItemDefs [ItemDefs]	character	64 (OID)

CodeLists SAS Data Set

Table A8.30 *CodeLists SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Name	character	128 (name)
DataType	character	7
SASFormatName	character	8
^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

ExternalCodeLists SAS Data Set

Table A8.31 *ExternalCodeLists SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
Dictionary	character	2000 (text)
Version	character	2000 (text)
ref	character	2000 (text)

Variable Name	SAS Data Type	Length (if character)
href	character	2000 (text)
^^FK_CodeLists [CodeLists]	character	64 (OID)

EnumeratedItems SAS Data Set

Table A8.32 *EnumeratedItems SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
CodedValue	character	512 (value)
Rank	numeric	8
^^FK_CodeLists [CodeLists]	character	64 (OID)

CodeListItems SAS Data Set

Table A8.33 *CodeListItems SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
##OID	character	64 (OID)
CodedValue	character	512 (value)
^^FK_CodeLists [CodeLists]	character	64 (OID)
Rank	numeric	8

CLItemDecodeTranslatedText SAS Data Set

Table A8.34 *CLItemDecodeTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)
lang	character	17
^FK_CodeListItems [CodeListItems]	character	64 (OID)

ImputationMethods SAS Data Set

Table A8.35 *ImputationMethods SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
method	character	2000 (text)
^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

Presentation SAS Data Set

Table A8.36 *Presentation SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
presentation	character	2000 (text)
lang	character	17
^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

MethodDefs SAS Data Set

Table A8.37 *MethodDefs SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Name	character	128 (name)
Type	character	11
^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

MethodDefTranslatedText SAS Data Set

Table A8.38 *MethodDefTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)
lang	character	17
^FK_MethodDefs [MethodDefs]	character	64 (OID)

MethodDefFormalExpression SAS Data Set

Table A8.39 *MethodDefFormalExpression SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
Context	character	2000 (text)
Expression	character	2000 (text)
^FK_MethodDefs [MethodDefs]	character	64 (OID)

ConditionDefs SAS Data Set

Table A8.40 ConditionDefs SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Name	character	128 (name)

ConditionDefTranslatedText SAS Data Set

Table A8.41 ConditionDefTranslatedText SAS Data Set

Variable Name	SAS Data Type	Length (if character)
TranslatedText	character	2000 (text)
lang	character	17
^FK_ConditionDefs [ConditionDefs]	character	64 (OID)
^FK_MetaDataVersion [MetaDataVersion]	character	64 (OID)

ConditionDefFormalExpression SAS Data Set

Table A8.42 ConditionDefFormalExpression SAS Data Set

Variable Name	SAS Data Type	Length (if character)
Context	character	2000 (text)
Expression	character	2000 (text)
^FK_ConditionDefs [ConditionDefs]	character	64 (OID)

ClinicalData SAS Data Set

Table A8.43 *ClinicalData SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
##OID	character	64 (OID)
^^StudyOID [Study]	character	64 (OID)
^^MetaDataVersionOID [MetaDataVersion]	character	64 (OID)
^^FK_ODM [ODM]	character	64 (OID)

SubjectDataSAS Data Set

Table A8.44 *SubjectDataSAS Data Set*

Variable Name	SAS Data Type	Length (if character)
##OID	character	64 (OID)
SubjectKey	character	2000 (text)
TransactionType	character	7
^^InvestigatorRefOID [User]	character	64 (OID)
^^SiteRefOID [Location]	character	64 (OID)
^^FK_ClinicalData [ClinicalData]	character	64 (OID)

StudyEventData SAS Data Set

Table A8.45 *StudyEventData SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
##OID	character	64 (OID)

Variable Name	SAS Data Type	Length (if character)
^^StudyEventOID [StudyEventDefs]	character	64 (OID)
StudyEventRepeatKey	character	2000 (text)
TransactionType	character	7
^^FK_SubjectData [SubjectData]	character	64 (OID)

FormData SAS Data Set

Table A8.46 FormData SAS Data Set

Variable Name	SAS Data Type	Length (if character)
##OID	character	64 (OID)
^^FormOID [FormDefs]	character	64 (OID)
FormRepeatKey	character	2000 (text)
TransactionType	character	7
^^ArchiveLayoutRefOID [FormDefArchLayouts]	character	64 (OID)
^^FK_StudyEventData [StudyEventData]	character	64 (OID)

ItemGroupData SAS Data Set

Table A8.47 ItemGroupData SAS Data Set

Variable Name	SAS Data Type	Length (if character)
##OID	character	64 (OID)
^^ItemGroupOID [ItemGroupDefs]	character	64 (OID)
ItemGroupRepeatKey	character	2000 (text)

Variable Name	SAS Data Type	Length (if character)
TransactionType	character	7
^^FK_FormData [FormData]	character	64 (OID)

ItemData SAS Data Set

Table A8.48 ItemData SAS Data Set

Variable Name	SAS Data Type	Length (if character)
##OID	character	64 (OID)
^^ItemOID [ItemDefs]	character	64 (OID)
TransactionType	character	7
TransactionOrder	numeric	8
Value	character	2000 (text)
IsNull	character	3
ItemDataType	character	18
^^FK_ItemGroupData [ItemGroupData]	character	64 (OID)
AuditRecordID [AuditRecords]	character	64 (OID)
SignatureID [Signatures]	character	64 (OID)
AnnotationID [Annotations]	character	64 (OID)
MeasurementUnitOID [MeasurementUnits]	character	64 (OID)
^^FK_ReferenceData [ReferenceData]	character	64 (OID)

ReferenceData SAS Data Set

Table A8.49 *ReferenceData SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
##GeneratedID	character	64 (OID)
^^StudyOID [Study]	character	64 (OID)
^^MetaDataVersionOID [MetaDataVersion]	character	64 (OID)
^^FK_ODM [ODM]	character	64 (OID)

AdminData SAS Data Set

Table A8.50 *AdminData SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
##GeneratedID	character	64 (OID)
StudyOID	character	64 (OID)
^^FK_ODM [ODM]	character	64 (OID)

User SAS Data Set

Table A8.51 *User SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
UserType	character	12 (enum)
LoginName	character	2000 (text)
DisplayName	character	2000 (text)
FullName	character	2000 (text)

Variable Name	SAS Data Type	Length (if character)
FirstName	character	2000 (text)
LastName	character	2000 (text)
Organization	character	2000 (text)
PictureImageType	character	2000 (text)
PictureFileName	character	2000 (text)
Pager	character	2000 (text)
^^FK_AdminData [AdminData]	character	64 (OID)

UserAddress SAS Data Set

Table A8.52 UserAddress SAS Data Set

Variable Name	SAS Data Type	Length (if character)
##GeneratedID	character	64 (OID)
City	character	2000 (text)
StateProv	character	2000 (text)
Country	character	2 (text)
PostalCode	character	2000 (text)
OtherText	character	2000 (text)
^^FK_User [User]	character	64 (OID)

UserAddressStreetName SAS Data Set

Table A8.53 UserAddressStreetName SAS Data Set

Variable Name	SAS Data Type	Length (if character)
StreetName	character	2000 (text)

Variable Name	SAS Data Type	Length (if character)
^^FK_UserAddress [UserAddress]	character	64 (OID)

UserEmail SAS Data Set

Table A8.54 UserEmail SAS Data Set

Variable Name	SAS Data Type	Length (if character)
Email	character	2000 (text)
^^FK_User [User]	character	64 (OID)

UserFax SAS Data Set

Table A8.55 UserFax SAS Data Set

Variable Name	SAS Data Type	Length (if character)
Fax	character	2000 (text)
^^FK_User [User]	character	64 (OID)

UserPhone SAS Data Set

Table A8.56 UserPhone SAS Data Set

Variable Name	SAS Data Type	Length (if character)
Phone	character	2000 (text)
^^FK_User [User]	character	64 (OID)

UserLocationRef SAS Data Set

Table A8.57 *UserLocationRef SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
^^LocationOID [Location]	character	64 (OID)
^^FK_User [User]	character	64 (OID)

Location SAS Data Set

Table A8.58 *Location SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Name	character	2000 (text)
LocationType	character	7 (enum)
^^FK_AdminData [AdminData]	character	64 (OID)

LocationVersion SAS Data Set

Table A8.59 *LocationVersion SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
^^StudyOID [Study]	character	64 (OID)
^^MetaDataVersionOID [MetaDataVersion]	character	64 (OID)
EffectiveDate	character	10 (date)
^^FK_Location [Location]	character	64 (OID)

SignatureDef SAS Data Set

Table A8.60 *SignatureDef SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)
Methodology	character	10 (enum)
Meaning	character	2000 (text)
LegalReason	character	2000 (text)
^^FK_AdminData [AdminData]	character	64 (OID)

AuditRecord SAS Data Set

Table A8.61 *AuditRecord SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
ID	character	2000 (text)
EditPoint	character	14 (enum)
UsedImputationMethod	character	3 (Yes/No)
^^UserOID [User]	character	64 (OID)
^^LocationOID [Location]	character	64 (OID)
DateTimeStamp	character	25 (datetime)
ReasonForChange	character	2000 (text)
SourceID	character	2000 (text)
ParentType	character	14 (datanode)
ParentKey	character	64 (OID)

Signature SAS Data Set

Table A8.62 *Signature SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
ID	character	2000 (text)
^^UserID [User]	character	64 (OID)
^^LocationOID [Location]	character	64 (OID)
^^SignatureDefOID [SignatureDef]	character	64 (OID)
DateTimeStamp	character	25 (datetime)
ParentType	character	14 (datanode)
ParentKey	character	64 (OID)

Annotation SAS Data Set

Table A8.63 *Annotation SAS Data Set*

Variable Name	SAS Data Type	Length (if character)
##GeneratedID	character	64 (OID)
ID	character	2000 (text)
SeqNum	numeric	8 (integer)
TransactionType	character	7 (enum)
CommentSponsorOrSite	character	7 (enum)
Comment	character	2000 (text)
ParentType	character	14 (datanode)
ParentKey	character	64 (OID)

AnnotationFlag SAS Data Set

Table A8.64 AnnotationFlag SAS Data Set

Variable Name	SAS Data Type	Length (if character)
FlagValue	character	2000 (text)
^^FlagValueCodeListOID [CodeLists]	character	64 (OID)
FlagType	character	128 (name)
^^FlagTypeCodeListOID [CodeLists]	character	64 (OID)
^^FK_Annotation [Annotation]	character	64 (OID)

Association SAS Data Set

Table A8.65 Association SAS Data Set

Variable Name	SAS Data Type	Length (if character)
##GeneratedID	character	64 (OID)
^^StudyOID [Study]	character	64 (OID)
^^MetaDataVersionOID [MetaDataVersion]	character	64 (OID)
^^FK_ODM [ODM]	character	64 (OID)

KeySet SAS Data Set

Table A8.66 KeySet SAS Data Set

Variable Name	SAS Data Type	Length (if character)
**OID	character	64 (OID)

Variable Name	SAS Data Type	Length (if character)
^^StudyOID [Study]	character	64 (OID)
SubjectKey	character	2000 (text)
^^StudyEventOID [StudyEventDefs]	character	64 (OID)
StudyEventRepeatKey	character	2000 (text)
^^FormOID [FormDefs]	character	64 (OID)
FormRepeatKey	character	2000 (text)
^^ItemGroupOID [ItemGroupDefs]	character	64 (OID)
ItemGroupRepeatKey	character	2000 (text)
^^ItemOID [ItemDefs]	character	64 (OID)
^^FK_Association [Association]	character	64 (OID)

Appendix 9

CRT-DDS 1.0 SAS Data Sets

Conventions Used in the Tables	478
DefineDocument SAS Data Set	478
Study SAS Data Set	479
MeasurementUnits SAS Data Set	480
MUTranslatedText SAS Data Set	480
MetaDataVersion SAS Data Set	480
AnnotatedCRFs SAS Data Set	481
SupplementalDocs SAS Data Set	481
MDVLeaf SAS Data Set	482
MDVLeafTitles SAS Data Set	482
ComputationMethods SAS Data Set	482
ValueLists SAS Data Set	483
ValueListItemRefs SAS Data Set	483
ProtocolEventRefs SAS Data Set	483
StudyEventDefs SAS Data Set	484
StudyEventFormRefs SAS Data Set	484
FormDefs SAS Data Set	485
FormDefItemGroupRefs SAS Data Set	485
FormDefArchLayouts SAS Data Set	485
ItemGroupDefs SAS Data Set	486
ItemGroupDefItemRefs SAS Data Set	487
ItemGroupAliases SAS Data Set	487
ItemGroupLeaf SAS Data Set	488
ItemGroupLeafTitles SAS Data Set	488
ItemDefs SAS Data Set	488
ItemQuestionTranslatedText SAS Data Set	489
ItemQuestionExternal SAS Data Set	489
ItemMUREfs SAS Data Set	490

ItemRangeChecks SAS Data Set	490
ItemRangeCheckValues SAS Data Set	490
RCErrorsTranslatedText SAS Data Set	491
ItemRole SAS Data Set	491
ItemAliases SAS Data Set	491
ItemValueListRefs SAS Data Set	492
CodeLists SAS Data Set	492
ExternalCodeLists SAS Data Set	492
CodeListItems SAS Data Set	493
CLItemDecodeTranslatedText SAS Data Set	493
ImputationMethods SAS Data Set	493
Presentation SAS Data Set	494

Conventions Used in the Tables

The tables that describe the SAS data sets use these conventions:

- The names of key variables are appended with two asterisks (**). Some data sets do not have a key.
- Foreign key variable names are prepended with two caret 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.
- All data set names and column names are case sensitive. They must be specified exactly as shown.
- 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.

DefineDocument SAS Data Set

Table A9.1 DefineDocument SAS Data Set

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

Variable Name	SAS Data Type	Length (if char)
AsOfDateTime	character	24
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 SAS Data Set

Table A9.2 Study SAS Data Set

Variable Name	SAS Data Type	Length (if char)
**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 SAS Data Set

Table A9.3 *MeasurementUnits SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
**OID [X]	character	128 (oid)
Name [X]	character	128 (name)
^FK_Study [Study] [X]	character	128 (oid)

MUTranslatedText SAS Data Set

Table A9.4 *MUTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
TranslatedText	character	2000 (text)
lang	character	128 (name)
^FK_MeasurementUnits [MeasurementUnits][X]	character	128 (oid)

MetaDataVersion SAS Data Set

Table A9.5 *MetaDataVersion SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
**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)

Variable Name	SAS Data Type	Length (if char)
StandardName [X]	character	2000 (text)
StandardVersion [X]	character	2000 (text)
^FK_Study [Study] [X]	character	128 (oid)

AnnotatedCRFs SAS Data Set

Table A9.6 *AnnotatedCRFs SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
DocumentRef	character	2000 (text)
^leafID [MDVLeaf] [X]	character	128 (oid)
^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

SupplementalDocs SAS Data Set

Table A9.7 *SupplementalDocs SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
DocumentRef	character	2000 (text)
^leafID [MDVLeaf] [X]	character	128 (oid)
^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

MDVLeaf SAS Data Set

Table A9.8 MDVLeaf SAS Data Set

Variable Name	SAS Data Type	Length (if char)
**ID [X]	character	128 (oid)
href	character	512 (path)
^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

MDVLeafTitles SAS Data Set

Table A9.9 MDVLeafTitles SAS Data Set

Variable Name	SAS Data Type	Length (if char)
title	character	2000 (text)
^FK_MDVLeaf [MDVLeaf] [X]	character	128 (oid)

ComputationMethods SAS Data Set

Table A9.10 ComputationMethods SAS Data Set

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

ValueLists SAS Data Set

Table A9.11 ValueLists SAS Data Set

Variable Name	SAS Data Type	Length (if char)
**OID [X]	character	128 (oid)
^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

ValueListItemRefs SAS Data Set

Table A9.12 ValueListItemRefs SAS Data Set

Variable Name	SAS Data Type	Length (if char)
^ItemOID [ItemDefs] [X]	character	128 (oid)
OrderNumber	numeric	8
Mandatory [X]	character	3
KeySequence	numeric	8
^ImputationMethodOID [ImputationMethods]	character	128 (oid)
Role	character	128 (name)
^RoleCodeListOID [CodeLists]	character	128 (oid)
^FK_ValueLists [ValueLists] [X]	character	128 (oid)

ProtocolEventRefs SAS Data Set

Table A9.13 ProtocolEventRefs SAS Data Set

Variable Name	SAS Data Type	Length (if char)
Mandatory [X]	character	3

Variable Name	SAS Data Type	Length (if char)
OrderNumber	numeric	8
^^StudyEventOID [StudyEventDefs] [X]	character	128 (oid)
^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

StudyEventDefs SAS Data Set

Table A9.14 StudyEventDefs SAS Data Set

Variable Name	SAS Data Type	Length (if char)
**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 SAS Data Set

Table A9.15 StudyEventFormRefs SAS Data Set

Variable Name	SAS Data Type	Length (if char)
^^FormOID [FormDefs] [X]	character	128 (oid)
Mandatory [X]	character	3
OrderNumber	numeric	8
^^FK_StudyEventDefs [StudyEventDefs] [X]	character	128 (oid)

FormDefs SAS Data Set

Table A9.16 FormDefs SAS Data Set

Variable Name	SAS Data Type	Length (if char)
**OID [X]	character	128 (oid)
Name [X]	character	128 (name)
Repeating [X]	character	3
^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

FormDefItemGroupRefs SAS Data Set

Table A9.17 FormDefItemGroupRefs SAS Data Set

Variable Name	SAS Data Type	Length (if char)
^ItemGroupOID [ItemGroupDefs] [X]	character	128 (oid)
Mandatory [X]	character	3
OrderNumber	numeric	8
^FK_FormDefs [FormDefs] [X]	character	128 (oid)

FormDefArchLayouts SAS Data Set

Table A9.18 FormDefArchLayouts SAS Data Set

Variable Name	SAS Data Type	Length (if char)
**OID [X]	character	128 (oid)
PdfFileName [X]	character	512 (path)
^PresentationOID [Presentation]	character	128 (oid)

Variable Name	SAS Data Type	Length (if char)
^^FK_FormDefs [FormDefs] [X]	character	128 (oid)

ItemGroupDefs SAS Data Set

Table A9.19 ItemGroupDefs SAS Data Set

Variable Name	SAS Data Type	Length (if char)
**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)
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 SAS Data Set

Table A9.20 *ItemGroupDefItemRefs SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
^^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 SAS Data Set

Table A9.21 *ItemGroupAliases SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
Context [X]	character	2000 (text)
Name [X]	character	2000 (text)
^^FK_ItemGroupDefs [ItemGroupDefs] [X]	character	128 (oid)

ItemGroupLeaf SAS Data Set

Table A9.22 *ItemGroupLeaf SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
**ID [X]	character	128 (oid)
href	character	512 (path)
^FK_ItemGroupDefs [ItemGroupDefs] [X]	character	128 (oid)

ItemGroupLeafTitles SAS Data Set

Table A9.23 *ItemGroupLeafTitles SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
title	character	2000 (text)
^FK_ItemGroupLeaf [ItemGroupLeaf] [X]	character	128 (oid)

ItemDefs SAS Data Set

Table A9.24 *ItemDefs SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
**OID [X]	character	128 (oid)
Name [X]	character	128 (name)
DataType [X]	character	8
Length	numeric	8
SignificantDigits	numeric	8
SASFieldName	character	8

Variable Name	SAS Data Type	Length (if char)
SDSVarName	character	8
Origin	character	2000 (text)
Comment	character	2000 (text)
^^CodeListRef [CodeLists]	character	128 (oid)
Label	character	2000 (text)
DisplayFormat	character	2000 (text)
^^ComputationMethodOID[ComputationMethods]	character	128 (oid)
^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

ItemQuestionTranslatedText SAS Data Set

Table A9.25 ItemQuestionTranslatedText SAS Data Set

Variable Name	SAS Data Type	Length (if char)
TranslatedText	character	2000 (text)
lang	character	17
^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)

ItemQuestionExternal SAS Data Set

Table A9.26 ItemQuestionExternal SAS Data Set

Variable Name	SAS Data Type	Length (if char)
Dictionary	character	2000 (text)
Version	character	2000 (text)
Code	character	2000 (text)
^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)

ItemMURefs SAS Data Set

Table A9.27 *ItemMURefs SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
^^MeasurementUnitOID [MeasurementUnits] [X]	character	128 (oid)
^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)

ItemRangeChecks SAS Data Set

Table A9.28 *ItemRangeChecks SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
**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)

ItemRangeCheckValues SAS Data Set

Table A9.29 *ItemRangeCheckValues SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
CheckValue	character	512 (value)
^^FK_ItemRangeChecks [ItemRangeChecks] [X]	character	128 (oid)

RCErrTranslatedText SAS Data Set

Table A9.30 *RCErrTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
TranslatedText	character	2000 (text)
lang	character	17
^FK_ItemRangeChecks [ItemRangeChecks] [X]	character	128 (oid)

ItemRole SAS Data Set

Table A9.31 *ItemRole SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
Name	character	2000 (text)
^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)

ItemAliases SAS Data Set

Table A9.32 *ItemAliases SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
Context [X]	character	2000 (text)
Name [X]	character	2000 (text)
^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)

ItemValueListRefs SAS Data Set

Table A9.33 *ItemValueListRefs SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
^^ValueListOID [ValueLists] [X]	character	128 (oid)
^^FK_ItemDefs [ItemDefs] [X]	character	128 (oid)

CodeLists SAS Data Set

Table A9.34 *CodeLists SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
**OID [X]	character	128 (oid)
Name [X]	character	128 (name)
DataType [X]	character	7
SASFormatName	character	8
^^FK_MetaDataVersion [MetaDataVersion] [X]	character	128 (oid)

ExternalCodeLists SAS Data Set

Table A9.35 *ExternalCodeLists SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
Dictionary	character	2000 (text)
Version	character	2000 (text)
^^FK_CodeLists [CodeLists] [X]	character	128 (oid)

CodeListItems SAS Data Set

Table A9.36 *CodeListItems SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
**OID [X]	character	128 (oid)
CodedValue	character	512 (value)
^FK_CodeLists [CodeLists] [X]	character	128 (oid)
Rank	numeric	8

CLItemDecodeTranslatedText SAS Data Set

Table A9.37 *CLItemDecodeTranslatedText SAS Data Set*

Variable Name	SAS Data Type	Length (if char)
TranslatedText	character	2000 (text)
lang	character	17
^FK_CodeListItems [CodeListItems] [X]	character	128 (oid)

ImputationMethods SAS Data Set

Table A9.38 *ImputationMethods SAS Data Set*

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

Presentation SAS Data Set

Table A9.39 *Presentation SAS Data Set*

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)

Index

C

CDISC [1](#)

CDISC ADaM

- Analysis data set metadata [224](#)
- analysis results metadata [228](#)
- analysis variable metadata [225](#)
- cross-standard validation [233](#)
- data set templates [230](#)
- key clinical reporting components [237](#)
- overview [223](#)
- sample data [234](#)
- sample reporting [236](#)
- SAS representation [224](#)
- TLF metadata [238](#)
- unique validation properties [232](#)
- validation check macros [233](#)
- validation of analysis data sets [231](#)

CDISC ADaM 2.1 [46](#)

- purpose [46](#)
- reference standard [47](#)
- regulatory basis [47](#)
- release date [47](#)

CDISC CRT-DDS 1.0 [51](#)

- purpose [51](#)
- reference standard [51](#)
- regulatory basis [51](#)
- release data [51](#)

CDISC CRT-DDS standard

- sample XML style sheet [36](#)

CDISC ODM 1.3.0 [56](#)

- purpose [56](#)
- reference standard [56](#)
- release date [56](#)

CDISC SDTM 3.1.1 [40](#)

- description [42](#)
- purpose [40](#)
- reference standard [41](#)
- release dates [40](#)

CDISC SDTM 3.1.2

- description [44](#)
- reference standard [43](#)

CDISC Terminology standard [62](#)

- purpose [62](#)
- reference standard [63](#)

Clinical Data Interchange Standards Consortium

See [CDISC](#)

clinical research activities [1](#)

columns

- in data tables [35](#)

common framework metadata [9](#)

controlled terminology [84](#)

- alternatives [170](#)

D

data set templates

- for CDISC ADaM [230](#)

data sets

- creating data sets used by framework [13](#)
- list of data sets associated with registered standard [12](#)

data standards

- creating table shells based on [13](#)
- getting a copy of the reference metadata for [14](#)

data tables [35](#)

- columns in [35](#)

default version for a standard

- setting [18](#)

default version of standards

- referencing [11](#)

F

files

- list of files associated with registered standard [12](#)

folder hierarchy

- global standards library [38](#)

framework

- creating data sets used by 13
 - creating table shells based on a data standard 13
 - determining which revision of a standard version is installed 12
 - getting a copy of the reference metadata for a data standard 14
 - getting a list of files and data sets associated with a registered standard 12
 - getting a list of installed standards 11
 - initializing global macro variables 10
 - inserting information from registered standards into SASReferences files 15
 - referencing default version of standards 11
 - usage scenarios 10
 - framework metadata 9
 - Framework module 6
- G**
- global macro variables
 - initializing 10
 - global standards library 6
 - directories in 6
 - directory structure 7
 - folder hierarchy 38
- I**
- initializing global macro variables 10
 - installed standards
 - getting a list of 11
- L**
- list of files and data sets associated with registered standard 12
 - list of installed standards 11
- M**
- macro variables
 - initializing framework's global macro variables 10
 - macros
 - utility macros for metadata files 68
 - maintenance usage scenarios 17
 - Messages data set 10
 - file content and structure 31
 - metadata
 - getting a copy of reference metadata 14
 - metadata directory 6
 - metadata files
 - additional files 35
 - common framework metadata 9
 - descriptions of 23
 - SASReferences files 67
 - metadata repository
 - See [global standards library](#)
- P**
- process controls 83
 - properties 10, 83
 - properties files
 - structure of 29
- R**
- Reference_Columns data set 35
 - Reference_Tables data set 35
 - reference metadata 83
 - getting a copy of 14
 - reference standards 38
 - references 2
 - referencing default version of standards 11
 - registered standards
 - inserting information from SASReferences files into 15
 - list of files and data sets associated with 12
 - registering
 - new standards 17
 - new version of a standard 17
 - unregistering a standard version 18
 - unregistering an old version of a standard, then registering a new version of a standard 19
 - releases
 - determining which release is installed 12
 - results 84
 - Results data set 10
 - file content and structure 33
 - revisions
 - determining which revision is installed 12
- S**
- SAS Clinical Standards Toolkit 1
 - SAS sessions
 - translating content of SASReferences file for 78
 - SASReferences data set 9
 - file content and structure 27
 - SASReferences file

- assessing structural integrity and content 76
- communicating filename and location to SAS Clinical Standards Toolkit 75
- how it's used 75
- translating content for SAS sessions 78
- SASReferences files 67
 - building 67
 - inserting information from registered standards into 15
 - sample files 67
 - templates 68
 - utility macros 68
- scenarios
 - maintenance usage scenarios 17
- scenarios for framework usage 10
- schema-repository directory 8
- set of checks to run 84
- Source_Columns data set 35
- Source_Tables data set 35
- source data 83
- source metadata 83
- standard versions
 - unregistering 18
- Standardlookup data set 9, 68
 - file content and structure 26
 - type and subtype values 69
- standards 1
 - CDISC ADaM 46
 - CDISC CRT-DDS 1.0 51
 - CDISC ODM 1.3.0 56
 - CDISC SDTM 40
 - CDISC SDTM 3.1.2 43
 - CDISC Terminology 62
 - creating table shells based on a data standard 13
 - defined 8
 - determining which revision is installed 12
 - getting a copy of the reference metadata for a data standard 14
 - getting a list of installed standards 11
 - inserting information from registered standards into SASReferences files 15
 - list of files and data sets associated with registered standard 12
 - reference standards 38
 - referencing default version of 11
 - registering a new standard 17
 - registering a new version 17
 - SAS representation of 37
 - setting the default version for a standard 18
 - supported 37

- unregistering an old version of a standard, then registering a new version of a standard 19
- Standards data set 9
 - file content and structure 23
- standards directory 7
- StandardSASReferences data set 9
 - file content and structure 25
- style sheet 36
- Summary data set 35
- supported standards 37

T

- table shells
 - creating, based on a data standard 13
- TLF metadata
 - CDISC ADaM 238
- translating content of SASReferences file 78

U

- unregistering
 - a standard version 18
 - an old version of a standard, and then registering a new version of a standard 19
- usage scenarios
 - maintenance scenarios 17
- usage scenarios 10
- utility macros 68

V

- validation checks 35
- Validation Control data set 35
- validation framework 82
 - building a validation process 104
 - components of 83
 - cross-standard validation 102
 - debugging validation processes 159
 - how SAS Clinical Standards Toolkit interprets validation check metadata 150
 - messages 99
 - metadata requirements 84
 - performance considerations 174
 - reference metadata 85
 - running a validation process 111
 - sample CDISC CRT-DDS 1.0 driver
 - program: validate_crtds_data.sas 120
 - sample CDISC SDTM 3.1.1 driver
 - program: validate_data.sas 111
 - SAS implementation of ISO 8601 154

- SASReferences customization 104
- setting properties for the validation process 110
- source metadata 88
- supplemental validation check metadata: domains by check 97
- supplemental validation check metadata: validation standard references 95
- validation check macros 144
- validation check metadata: Validation Master data set 88
- validation checks by standard 120
- validation control: specification of run-time checks 105
- validation customization 164
- validation metrics 100
- validation properties 97
- validation results and metrics 116
- Validation Master data set 35
- validation metrics 35
- variables
 - initializing framework's global macro variables 10
- versions
 - determining which revision is installed 12
 - referencing default version of a standard 11
 - registering a new version 17
 - setting the default version for a standard 18
 - unregistering a standard version 18
 - unregistering an old version of a standard, then registering a new version of a standard 19
- X**
 - XML style sheet 36
 - xsl-repository directory 8