



THE
POWER
TO KNOW.

SAS[®] Clinical Standards Toolkit 1.5

User's Guide

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS® Clinical Standards Toolkit 1.5: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Clinical Standards Toolkit 1.5: User's Guide

Copyright © 2013, SAS Institute Inc., Cary, NC, USA

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

For a hard-copy 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.

May 2013

SAS 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 **support.sas.com/bookstore** 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>vii</i>
Chapter 1 • Introduction to the SAS Clinical Standards Toolkit	1
What Is the SAS Clinical Standards Toolkit?	1
References	2
Chapter 2 • Framework	7
Overview	8
Global Standards Library	8
What Is a Standard?	13
Common Framework Metadata	13
Common Usage Scenarios for the Framework	16
Maintenance Usage Scenarios	25
Chapter 3 • Metadata File Descriptions	33
Overview	34
Standards	34
StandardSASReferences	37
Standardlookup	39
SASReferences	42
Properties	45
Messages	47
Results	50
Additional Metadata Files	54
Chapter 4 • Supported Standards	57
SAS Representation of Standards	58
CDISC SDTM	62
CDISC ADaM 2.1	69
CDISC CRT-DDS 1.0	74
CDISC ODM	79
CDISC SEND 3.0	87

CDISC Terminology	89
Chapter 5 • SASReferences File	93
Overview	93
Building a SASReferences File	94
How Is a SASReferences File Used?	106
Chapter 6 • Compliance Assessment Against a Reference Standard	115
Validation Framework Overview	117
Metadata Requirements	120
Cross-Standard Validation	150
Building a Validation Process	153
Running a Validation Process	160
Validation Checks by Standard	172
Special Topic: Validation Check Macros	185
Special Topic: How the SAS Clinical Standards Toolkit Interprets Validation Check Metadata	191
Special Topic: SAS Implementation of ISO 8601	196
Special Topic: Debugging a Validation Process	204
Special Topic: Validation Customization	212
Special Topic: Using Alternative Controlled Terminologies	222
Special Topic: Performance Considerations	227
Chapter 7 • Internal Validation	229
Overview	229
Supporting Macros	231
Validating a SASReferences Data Set	233
Sample Driver Modules	235
Validation Checks	243
Chapter 8 • XML-Based Standards	249
SAS Support of XML-Based Standards	250
Reading XML Files	251
Writing XML Files	288
Validation of XML-Based Standards	309

Special Topic: A Round Trip Exercise Involving the CDISC SDTM and CDISC CRT-DDS Standards	322
Special Topic: A Round Trip Exercise Involving the CDISC CRT-DDS Standard: Importing and Exporting the define.xml File	327
Special Topic: Identifying Unsupported Elements and Attributes in a CDISC ODM File	334
Chapter 9 • Working with CDISC ADaM Data	341
Overview	341
SAS Representation of CDISC ADaM Metadata	342
ADaM Data Set Templates	353
Validation of ADaM Data Sets	354
Sample Reporting Methodology	361
Chapter 10 • Reporting	373
Sample Reports	373
Process Results Reporting	374
Validation Check Metadata Reporting	386
Appendix 1 • Global Macro Variables	393
Overview	393
Global Macro Variables and Their Associated Metadata	394
Index	397

What's New

What's New in the SAS Clinical Standards Toolkit

Overview

Here are some of the new capabilities in the SAS Clinical Standards Toolkit 1.5:

- Deployment of the SAS Clinical Standards Toolkit 1.5 now includes the sample library.

Previous versions of the SAS Clinical Standards Toolkit deployed all sample files into the !sasroot folder hierarchy. During the installation and configuration of the SAS Clinical Standards Toolkit 1.5, the installer is prompted for the location in which to install the sample study library. The configuration process creates a series of directories in this location.

- The introduction of a set of tools to validate the SAS Clinical Standards Toolkit metadata itself.

This functionality is referred to as internal validation. A subset of validation checks serves to support installation and operational qualification of the SAS Clinical Standards Toolkit. For a description of the implementation, see [Chapter 7, “Internal Validation,”](#) on page 229.

- The CDISC SDTM 3.1.3 standard, including all metadata and validation checks, has been fully implemented.

This includes definitions of the 36 domains itemized in the *Study Data Tabulation Model Implementation Guide: Human Clinical Trials (Version 3.1.3)*. For a description of the implementation, see [Chapter 4, “Supported Standards,” on page 57](#).

- The SDTM check utility macro `sdmcheckutil_recordlookup` was completely rewritten to optimize performance.

This macro is used to identify records in a source data set that cannot be found in the referenced lookup data set. Examples of the source data set include SUPPxx, CO, and RELREC.

- The CDISC ODM 1.3.1 standard, including all metadata and validation checks, has been fully implemented.

Support for the extraction of ODM Clinical data and ODM Reference data into SAS data sets has been added to both ODM 1.3.0 and ODM 1.3.1. For a description and limitations of the implementation, see [Chapter 4, “Supported Standards,” on page 57](#) and [Chapter 8, “XML-Based Standards,” on page 249](#).

- The CRT-DDS 1.0.0 standard (`define.xml`) was updated to include support for creating a `define.xml` for the CDISC ADaM standard.

Support was added for the definitions of study source metadata for value level metadata (SDTM), parameter value level metadata (ADaM), and document metadata for annotated CRFs and supplemental documents (SDTM).

The ability to create a `define.pdf` for SDTM and ADaM was added to the CRT-DDS standard.

- An initial implementation of the CDISC SEND 3.0 standard, including the definitions of all domains and columns as specified in the *Standard for Exchange of Nonclinical Data Implementation Guide: Nonclinical Studies, Version 3.0*, has been provided. For a brief description of the implementation, see [Chapter 4, “Supported Standards,” on page 57](#).

Note: No support of CDISC SEND validation checks is provided in the SAS Clinical Standards Toolkit 1.5.

- Support for the CDISC ADaM Data Structure for Adverse Event Analysis (ADAE) Version 1.0 has been added to the SAS Clinical Standards Toolkit ADaM reference metadata.

An example implementation of the CDISC ADaM Basic Data Structure for Time-to-Event (ADTTE) Analysis Version 1.0 has been added to the ADaM sample study.

New validation checks in support of ADAE and ADTTE have been added to the ADaM Validation Master data set.

- The implementation of CT 1.0.0, a tool to support the import of the National Cancer Institute (NCI) CDISC Controlled Terminology in the ODM XML format into SAS data sets and SAS format catalogs, has been added.

Samples are provided to import the latest versions of controlled terminology for ADaM, CDASH, SDTM, SEND, Questionnaires, and the Clinical Data Element Glossary. For a description of the implementation, see [Chapter 8, “XML-Based Standards,” on page 249](#).

- Additional columns have been added to all SASReferences and StandardSASReferences data sets provided by SAS.

The iotype, filetype, allowoverwrite, and relpathprefix columns primarily support validation of the SASReferences data set and a broader capability to define relative study paths. For a full description of these columns, see [Chapter 3, “Metadata File Descriptions,” on page 33](#).

- Full i18n multiple-byte character support for data has been added.

All SAS Clinical Standards Toolkit validation macros have been updated to handle multiple-byte data characters encoded in UTF-8 or Shift JIS.

Note: This initial i18n implementation is limited to clinical data only. The SAS Clinical Standards Toolkit 1.5 metadata does not support i18n with the exception of directory pathnames. The SAS macros and code logic field of the validation_master data set were modified by replacing SAS functions with i18n-compatible K functions.

- The SAS Clinical Standards Toolkit metadata and code base have been updated.
- A number of new framework macros are available, including three new validation check macros and six new validation check utility macros.

These macros are described briefly below. For more information about the check macros, see [Table 6.20 on page 193](#). Information about all macros is in the online macro API reference documentation.

Changes to Metadata and Code Base

Framework Changes

These autocall macros are new:

- `cst_createdsfromtemplate.sas` creates a zero-observation data set that is based on a template. The template is returned from the `Standardlookup` data set based on a specified standard, `standardversion`, `type`, and `subtype`.
- `cstcheckcompareallcolumns.sas` compares all columns in one domain with the same columns in other domains.
- `cstcheckentitynotfound.sas` reports that an entity, typically a file, folder, or column, cannot be found.
- `cstcheckforeignkeynotfound.sas` compares the consistency of one or more columns across two tables, where a column in the first table is a foreign key that points to a primary key in the second table.
- `cstcheckutilcheckfile.sas` determines whether a file exists as defined by columns in a source data set. This macro is used primarily by internal validation to confirm that files exist based on the content of the *global standards library directory/metadata/standards.sas7bdat* data set.
- `cstcheckutilcheckfolder.sas` determines whether a folder exists as defined by columns in a source data set. This macro is used primarily by internal validation to confirm that folders exist based on the content of the *global standards library directory/metadata/standards.sas7bdat* data set.

- `cstcheckutilcheckstructure.sas` compares the structure of data sets referenced within `StandardSASReferences` or `SASReferences` data sets against a template.
- `cstcheckutilfindsasrefsfile.sas` determines whether designated files in the referenced `SASReferences` data set exist.
- `cstcheckutillookupvalues.sas` determines whether metadata column values for discrete columns can be found in the `Standardlookup` data set.
- `cstupdatestandardsasrefs.sas` expands all relative paths to full paths in a `SASReferences` data set.
- `cstutil_getcstversion.sas` returns the SAS Clinical Standards Toolkit product version.
- `cstutil_setcstsroot.sas` sets the value of the global macro variable `_cstSRoot`, which provides the location of the sample library. If the default installation location is used, this is `C:\cstSampleLibrary` (Microsoft Windows) or `/usr/local/cstSampleLibrary` (UNIX).
- `cstutilbuildmetadatafromsasrefs.sas` builds the framework `reference_tables` and `reference_columns` data sets from available `SASReferences` data sets as a part of internal validation.
- `cstutilbuildstdvalidationcode.sas` generates the validation-specific macro `_cstreadStds` to build the internal validation workflow for one or more standards.
- `cstutilcheckforproblem.sas` handles any error condition that sets the global macro variable `_cst_rc` to 1. This macro variable can be set after a call to any SAS Clinical Standards Toolkit macro.
- `cstutilcheckjava.sas` determines whether issues related to Java exist in the previous DATA step.

Note: In the SAS Clinical Standards Toolkit 1.4, `cstutilcheckjava.sas` was known as `cstcheck_java`. It has been renamed in version 1.5 to follow naming conventions.

- `cstutilcheckwriteaccess.sas` checks for Write access for an entity that has been defined as an output object in a `SASReferences` data set.
- `cstutilcomparestructure.sas` compares the metadata structure of two data sets using a return code to provide information about the result of the comparison.

- `cstutilcreateattribfromds.sas` creates a DATA step ATTRIB statement for all columns in a specified data set.
- `cstutildropmissingvars.sas` drops variables from a data set that have only missing values.
- `cstutilfindvalidfile.sas` checks whether a folder, file, data set, catalog, or catalog member exists. It is used most often in the validation of a SASReferences data set.
- `cstutilnobs.sas` returns the number of observations in a data set or an error.
- `cstutilprocessfailed.sas` returns a Boolean value to report whether a SAS Clinical Standards Toolkit process failed.
- `cstutiltrimcharvars.sas` trims character variables to their minimum length.
- `cstutilvalidatesasreferences.sas` validates the structure and content of a SASReferences data set. Eight different conditions are evaluated.
- `cstutilvalidationsummary.sas` summarizes the contents of the validation process Results data set, reporting the number of validation warnings or errors that were generated and whether some validation checks were not run.
- `cstvalidate.sas` validates the SAS Clinical Standards Toolkit framework metadata.
- `csutilwriteresultsintro.sas` adds process metadata records to the Results data set.

These macros are located in the `!sasroot/cstframework/sasmacro` directory (Microsoft Windows) or in the `!sasroot/sasautos` directory (UNIX). A description of each new macro is provided in the online macro API reference documentation.

Sample Library

The SAS Clinical Standards Toolkit 1.5 installs all sample study files into a folder hierarchy outside the `!sasroot` location that was used in prior versions. For example, in version 1.4, the sample library for CDISC-SDTM 3.1.2 was located for SAS 9.3 here:

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

This !sasroot deployment location caused access problems for customers whose Write access permission to !sasroot was restricted.

During the installation and configuration of the SAS Clinical Standards Toolkit 1.5, the installer is prompted for the location in which to install the sample study library. These are default locations:

- `C:\cstSampleLibrary` (Microsoft Windows)
- `/usr/local/cstSampleLibrary` (UNIX)

The global macro variable `&_cstSRoot` is now set to the location of the sample study library (paralleling the `&_cstGRoot` global macro variable that points to the global standards library).

In many sample driver programs, the global macro variable `studyRootPath` was set in previous versions to the !sasroot location for the standard, using code such as this:

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

Now, the same initialization of `studyRootPath` is this:

```
call
symput('studyRootPath', cats("&_
cstSRoot", "/cdisc-sdtm-3.1.2-1.5/sascstdemodata"));
```

The sample library is used to illustrate use of the SAS Clinical Standards Toolkit. Most standards that are provided by SAS use a sample set of data, metadata, and code to provide an instance of each specific standard. In reality, the SAS Clinical Standards Toolkit sample library is simply a proxy for your clinical study data, metadata, and code.

As a part of each standard definition, the *global standards library directory/standards/<standard and version>/control/standards.sas7bdat* data set contains a column named `studylibraryrootpath`. This column, by default, has been set to the rootpath of the sample study for that standard. You can choose to use this column to point to the rootpath of some study hierarchy within your organization. There are a number of alternative ways to reference your study data and metadata as well.

Internal Validation

The SAS Clinical Standards Toolkit 1.5 provides a new set of functionality, called internal validation, to help verify that metadata files are consistent and correct. This feature is especially useful as you customize the SAS Clinical Standards Toolkit. This new set of tools uses the SAS Clinical Standards Toolkit validation framework and methodology that assess standard-specific files against a defined reference standard.

Given the central role that the SASReferences data set plays in submission of SAS Clinical Standards Toolkit processes, more rigorous validation of this data set is provided with the `cstutilvalidatesasreferences` macro.

For more information, see [Chapter 7, “Internal Validation,”](#) on page 229.

CDISC CRT-DDS Changes

These changes were made to CDISC CRT-DDS:

- These macros were added to support the creation of a `define.xml` file for the CDISC ADaM standard:
 - `crtdds_adamtodef`
 - `crtdds_itemgroupdefitemrefs_adam`
 - `crtdds_itemgroupdefs_adam`
- The `crtdds_writepdf` macro was added to support the creation of a `define.pdf` for the CDISC SDTM and CDISC ADaM standards.
- The `crtdds_sourcevalues` and `crtdds_sourcedocuments` macros were added to import metadata that describes value level metadata (SDTM), parameter value level metadata (ADaM), and document metadata for annotated CRFs and supplemental documents (SDTM).
- New conventions were implemented for the creation of the various OID attributes in the `define.xml` file. Here are some examples:
 - `MetaDataVersion OID="MDV.1"`

- ❑ `def:ComputationMethod` `OID="CM.EG.EGTESTCD.QTCB"`
- ❑ `def:ValueListDef` `OID="VL.EG.EGTESTCD"`
- ❑ `ItemGroupDef` `OID="IG.AE"`
- ❑ `ItemDef` `OID="IT.AE.AESTDTC"`
- ❑ `CodeList` `OID="CL.AESEV"`

CDISC Controlled Terminology

The SAS Clinical Standards Toolkit support for controlled terminology has been updated to the most recent version of the NCI CDISC controlled terminology as of April 1, 2013.

This table lists the implemented controlled terminology versions. Every controlled terminology standard (ADaM, CDASH, SDTM, and SEND) also contains a **current** folder, which is a copy of the most recent controlled terminology version for that standard.

Implemented Controlled Terminology

Standard	201101	201104	201107	201201	201212
ADaM	x		x		
CDASH		x			x
SDTM		x			x
SEND				x	x

Introduction to the SAS Clinical Standards Toolkit

<i>What Is the SAS Clinical Standards Toolkit?</i>	1
<i>References</i>	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 the <i>CDISC SDTM Implementation Guide V3.1.1 Final</i> and the <i>CDISC Study Data Tabulation Model Version 1.1 Final</i> .
CDISC SDTM 3.1.2	http://www.cdisc.org/sdtm	Provides access to the <i>Study Data Tabulation Model, Version 1.2</i> and the <i>Study Data Tabulation Model Implementation Guide: Human Clinical Trials (Version 3.1.2)</i> .

Reference	Web Address **	Description
CDISC SDTM 3.1.3	http://www.cdisc.org/sdtm	Provides access to the <i>Study Data Tabulation Model (Version 1.3)</i> and the <i>Study Data Tabulation Model Implementation Guide: Human Clinical Trials (Version 3.1.3)</i> .
CDISC SEND 3.0	http://www.cdisc.org/send	Provides access to the <i>Standard for Exchange of Nonclinical Data Implementation Guide: Nonclinical Studies, Version 3.0</i> .
CDISC CRT-DDS 1.0	http://www.cdisc.org/define-xml	Provides access to the <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/odm	Provides access to ODM Version 1.3.0 files and documentation.
CDISC ODM 1.3.1	http://www.cdisc.org/odm	Provides access to ODM Version 1.3.1 files and documentation.
NCI CDISC Controlled Terminology	http://www.cancer.gov/cancertopics/terminologyresources/page6	Provides access to a directory of supported CDISC terminology. Note: http://evs.nci.nih.gov/ftp1/CDISC/SDTM/ offers a current and cumulative set of terminology that supports CDISC SDTM.

Reference	Web Address **	Description
CDISC ADaM 2.1	http://www.cdisc.org/adam	Provides access to the <i>Analysis Data Model, Version 2.1</i> and the <i>ADaM Implementation Guide, Version 1.0</i> . Note: Registration might be required.
CDISC ADaM 2.1 Validation Checks Version 1.1	http://www.cdisc.org/adam-validation	Provides access to the CDISC ADaM Validation Checks Version 1.1. Note: Access to the CDISC members-only site might be required.
CDISC ADaM 2.1 Validation Checks Version 1.2	http://www.cdisc.org/adam-validation	Provides access to the CDISC ADaM Validation Checks Version 1.2. Note: Access to the CDISC members-only site might be required.
Data Structure for Adverse Event Analysis Version 1.0	http://www.cdisc.org/adam	Provides access to the Analysis Data Model (ADaM) Data Structure for Adverse Event Analysis Version 1.0.
Data Structure for Time-to-Event Analyses Version 1.0	http://www.cdisc.org/adam	Provides access to the ADaM Basic Data Structure for Time-to-Event Analyses Version 1.0.
OpenCDISC Validation Rules	http://www.opencdisc.org/projects/validator/cdisc-validation-rules-repository	Provides access to the 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.

Reference	Web Address **	Description
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	Provides information about the ISO 8601 standard.
SAS Technical Support	Online form: http://support.sas.com/ctx/supportform/createForm	Provides access to a form on which any problems experienced with the product and technical questions should be documented. Or, you can call (in 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	Provides current information, documentation, technical papers, and presentations about the SAS Clinical Standards Toolkit.
<i>SAS Clinical Standards Toolkit 1.5: User's Guide</i>	http://support.sas.com/documentation/onlinedoc/clinical/index.html	Provides a link to this document and other documents.
SAS Clinical Standard Toolkit: Papers	http://support.sas.com/rnd/base/cdisc/cst/index.html	Provides links to papers written about the SAS Clinical Standards Toolkit.
SAS Clinical Standards Toolkit Samples and SAS Notes	http://support.sas.com/notes/index.html	Provides a way to search SAS installation problems, usage problems, samples, and SAS Notes that are associated with the SAS Clinical Standards Toolkit. (Type <i>Clinical Standards Toolkit</i> in the search field.)

Reference	Web Address **	Description
SAS in Health Care Related Fields and Clinical Trials Forum	http://communities.sas.com/ community/ sas_and_clinical_trials	Provides access to a primary public discussion forum for the SAS Clinical Standards Toolkit.
SAS Training	http://support.sas.com/ training/	Currently, SAS is pursuing the development of SAS Clinical Standards Toolkit training classes. Some information about the SAS Clinical Standards Toolkit is provided in the SAS Clinical Data Integration: Essentials training course.
External Vendor Tutorials		Offers product tutorials from vendors, often as a part of an industry-related user conference.

** Accessed on March 11, 2013.

2

Framework

Overview	8
Global Standards Library	8
What Is a Standard?	13
Common Framework Metadata	13
Overview	13
Standards Data Set	14
StandardSASReferences	14
Standardlookup	14
SASReferences Data Set	15
Properties Files	15
Messages Data Set	15
Results Data Set	15
Common Usage Scenarios for the Framework	16
Overview	16
Initializing the Framework's Global Macro Variables	16
Referencing the Default Version of a Standard	17
Getting a List of the Standards That Are Installed	17
Determining Which Revision (Release) of a Standard Version Is Installed	18
Getting a List of the Files and Data Sets That Are Associated with a Registered Standard	18
Creating Data Sets Used by the Framework	20
Creating Table Shells Based on a Data Standard	20

Getting a Copy of the Reference Metadata for a Data Standard	21
Inserting Information from Registered Standards into a SASReferences File	22
<i>Maintenance Usage Scenarios</i>	25
Overview	25
Registering a New Version of a Standard	25
Setting the Default Version for a Standard	27
Unregistering a Standard Version	27
Unregistering an Old Version of a Standard, and Then Registering a New Version of a Standard	28

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

Global Standards Library

The global standards library is the metadata repository for the SAS Clinical Standards Toolkit. By default, the global standards library contains the metadata for the

Framework module and the metadata for each data standard that is provided by SAS (such as the CDISC SDTM 3.1.2 standard).

During the installation and configuration of the SAS Clinical Standards Toolkit, 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 13](#).
- **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 three data sets and one XML file: Standards, Standardlookup, and StandardSASReferences, and availabletransforms.xml. 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	groupname	groupversion	comment	rootpath	studytraintopath
1	CDISC-ADAM	ADAM	2.1	ADAM	2.1	CDISC ADAM V2.1	_cstRoot/standards/cdisc-adam-2.1.1.5	_cstRoot/cdisc-adam-2.1.1.5/sascdmetadata
2	CDISC-CRTDQS	CRT	1.0	CRTDQS	1.0	CDISC CRT DQS V1.0	_cstRoot/standards/cdisc-crtdds-1.0.1.5	_cstRoot/cdisc-crtdds-1.0.1.5
3	CDISC-CT	CTX	1.0.0	CTX	1.0.0	CDISC CT XML V1.0.0	_cstRoot/standards/cdisc-ct-1.0.0.1.5	_cstRoot/cdisc-ct-1.0.0.1.5
4	CDISC-ODM	ODM	1.3.0	ODM	1.3.0	CDISC ODM V1.3.0	_cstRoot/standards/cdisc-odm-1.3.0.1.5	_cstRoot/cdisc-odm-1.3.0.1.5
5	CDISC-ODM	ODM	1.3.1	ODM	1.3.1	CDISC ODM V1.3.1	_cstRoot/standards/cdisc-odm-1.3.1.1.5	_cstRoot/cdisc-odm-1.3.1.1.5
6	CDISC-SDTM	SDTM	3.1.1	SDTM	3.1.1	CDISC SDTM V3.1.1	_cstRoot/standards/cdisc-odm-3.1.1.1.5	_cstRoot/cdisc-odm-3.1.1.1.5/sascdmetadata
7	CDISC-SDTM	SDTM	3.1.2	SDTM	3.1.2	CDISC SDTM V3.1.2	_cstRoot/standards/cdisc-odm-3.1.2.1.5	_cstRoot/cdisc-odm-3.1.2.1.5/sascdmetadata
8	CDISC-SDTM	SDTM	3.1.3	SDTM	3.1.3	CDISC SDTM V3.1.3	_cstRoot/standards/cdisc-odm-3.1.3.1.5	_cstRoot/cdisc-odm-3.1.3.1.5/sascdmetadata
9	CDISC-SEND	SEND	3.0	SEND	3.0	CDISC SEND V3.0	_cstRoot/standards/cdisc-send-3.0.1.5	_cstRoot/cdisc-send-3.0.1.5/sascdmetadata
10	CDISC-TERMINOLOGY	CT	NO_THESAURUS	TERMINOLOGY	NO_THESAURUS	CDISC Terminology	_cstRoot/standards/cdisc-terminology-1.5	
11	CST FRAMEWORK	CST	1.2	FRAMEWORK	1.2	Clinical Standards Toolkit Framework	_cstRoot/standards/cst-framework-1.5	_cstRoot/cst-framework-1.5

	standard	mnemonic	standardversion	controlsubid	templatemultid	standarddefaut	isctframework	isdatastandard	isreportvaliddefaut	ismlstandard	importref	importref	schema	productversion
1	CDISC-ADAM	ADAM	2.1	control	templates	Y	N	Y	Y	N				1.5
2	CDISC-CRTDQS	CRT	1.0	control	templates	Y	N	Y	Y	Y	CRT DQS/1.0/report/Root.xml	CRT DQS/1.0/report/Root.xml	cdisc-crtdds-1.0.0/define1.0.0.xsd	1.5
3	CDISC-CT	CTX	1.0.0	control	templates	Y	N	Y	Y	Y	CT/1.0/report/Root.xml	CT/1.0/report/Root.xml	cdisc-ct-1.0.0/control/terminology1.0.0.xsd	1.5
4	CDISC-ODM	ODM	1.3.0	control	templates	N	N	Y	Y	Y	ODM/1.3.0/report/Root.xml	ODM/1.3.0/report/Root.xml	cdisc-odm-1.3.0/ODM1-3.0.xsd	1.5
5	CDISC-ODM	ODM	1.3.1	control	templates	Y	N	Y	Y	Y	ODM/1.3.1/report/Root.xml	ODM/1.3.1/report/Root.xml	cdisc-odm-1.3.1/ODM1-3.1.xsd	1.5
6	CDISC-SDTM	SDTM	3.1.1	control	templates	N	N	Y	Y	N				1.5
7	CDISC-SDTM	SDTM	3.1.2	control	templates	N	N	Y	Y	N				1.5
8	CDISC-SDTM	SDTM	3.1.3	control	templates	Y	N	Y	Y	N				1.5
9	CDISC-SEND	SEND	3.0	control	templates	Y	N	Y	N	N				1.5
10	CDISC-TERMINOLOGY	CT	NO_THESAURUS	control		Y	N	N	N	N				1.5
11	CST FRAMEWORK	CST	1.2	control	templates	Y	Y	N	Y	N				1.5

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	isotype	filetype	allowoverwrite	relpathprefix	path	order	memname
1	CDISC-ADAM	2.1	autocall		autocall	fleref	input	folder	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/macros	1	
2	CDISC-ADAM	2.1	classmetadata	column	refmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/metadata		class_columns.sas7bdat
3	CDISC-ADAM	2.1	classmetadata	table	refmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/metadata		class_tables.sas7bdat
4	CDISC-ADAM	2.1	csimetadata	lookup	stdmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/control		standardlookup.sas7bdat
5	CDISC-ADAM	2.1	csimetadata	macrovariabledetails	stdmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/control		standardmacrovariabledetails.sas7bdat
6	CDISC-ADAM	2.1	csimetadata	macrovariables	stdmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/control		standardmacrovariables.sas7bdat
7	CDISC-ADAM	2.1	csimetadata	sasreferences	stdmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/control		standardsasreferences.sas7bdat
8	CDISC-ADAM	2.1	csimetadata	standard	stdmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/control		standards.sas7bdat
9	CDISC-ADAM	2.1	lookup		lookup	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/control		standardlookup.sas7bdat
10	CDISC-ADAM	2.1	messages		messages	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/messages	1	messages.sas7bdat
11	CDISC-ADAM	2.1	properties	initialize	initprop	fleref	input	file	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/programs		initialize.properties
12	CDISC-ADAM	2.1	properties	report	rlprop	fleref	input	file	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/programs		report.properties
13	CDISC-ADAM	2.1	properties	validation	valprop	fleref	input	file	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/programs		validation.properties
14	CDISC-ADAM	2.1	referencecontrol	checktable	refcntrl	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/validation/control		validation_classbycheck.sas7bdat
15	CDISC-ADAM	2.1	referencecontrol	validation	refcntrl	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/validation/control		validation_master.sas7bdat
16	CDISC-ADAM	2.1	referencemetadata	column	refmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/metadata		reference_columns.sas7bdat
17	CDISC-ADAM	2.1	referencemetadata	table	refmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/metadata		reference_tables.sas7bdat
18	CDISC-ADAM	2.1	template		tmplit	lbref	input	folder	N		t_cstGRoot/standards/cdisc-adam-2.1.1.5/templates		
19	CDISC-CRTDDS	1.0	autocall		autocall	fleref	input	folder	N		t_cstGRoot/standards/cdisc-crtdds-1.0-1.5/macros	1	
20	CDISC-CRTDDS	1.0	csimetadata	lookup	stdmeta	lbref	input	dataset	N		t_cstGRoot/standards/cdisc-crtdds-1.0-1.5/control		standardlookup.sas7bdat

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.

The Standardlookup data set contains discrete lookup values specific to a SAS Clinical Standards Toolkit registered standard. It provides specific information for column values and data set template names. In addition, this data set is used to perform internal validation of the SAS Clinical Standards Toolkit.

This display shows the entire column list.

Display 2.3 Global Standards Library: Metadata Standardlookup Data Set

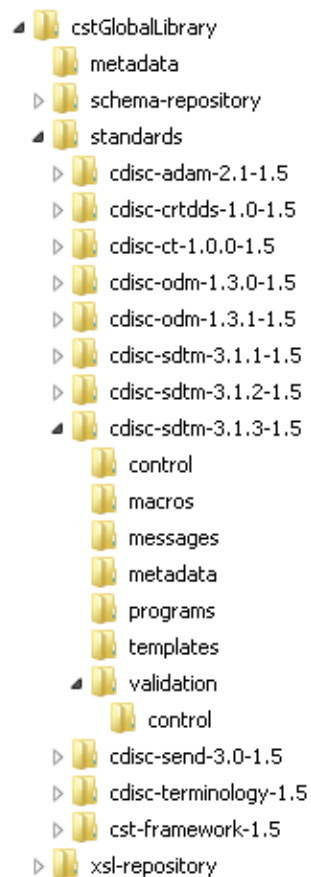
	standard	standardversion	SASref	table	column	refcolumn	refvalue	value	default	nonnull
108	CDISC-ADAM	2.1	stdmeta	standardsasreferences	filetype			DATASET	Y	Y
109	CDISC-ADAM	2.1	stdmeta	standardsasreferences	filetype			FILE	N	Y
110	CDISC-ADAM	2.1	stdmeta	standardsasreferences	filetype			FOLDER	N	Y
111	CDISC-ADAM	2.1	stdmeta	standardsasreferences	ioltype			BOTH	N	Y
112	CDISC-ADAM	2.1	stdmeta	standardsasreferences	ioltype			INPUT	Y	Y
113	CDISC-ADAM	2.1	stdmeta	standardsasreferences	ioltype			OUTPUT	N	Y
114	CDISC-ADAM	2.1	stdmeta	standardsasreferences	reftype			FILEREF	N	Y
115	CDISC-ADAM	2.1	stdmeta	standardsasreferences	reftype			LIBREF	Y	Y
116	CDISC-ADAM	2.1	stdmeta	standardsasreferences	subtype	type	CLASSMET	COLUMN	N	N
117	CDISC-ADAM	2.1	stdmeta	standardsasreferences	subtype	type	CLASSMET	TABLE	Y	N
118	CDISC-ADAM	2.1	stdmeta	standardsasreferences	subtype	type	CONTROL	REFERENCE	Y	N
119	CDISC-ADAM	2.1	stdmeta	standardsasreferences	subtype	type	CONTROL	VALIDATION	N	N
120	CDISC-ADAM	2.1	stdmeta	standardsasreferences	subtype	type	CSTMETA	LOOKUP	N	N
121	CDISC-ADAM	2.1	stdmeta	standardsasreferences	subtype	type	CSTMETA	MACROVARIABLEDETAIL	N	N

	standard	standardversion	SASref	table	order	templatetype	template	comment
108	CDISC-ADAM	2.1	stdmeta	standardsasreferences	1			
109	CDISC-ADAM	2.1	stdmeta	standardsasreferences	3			
110	CDISC-ADAM	2.1	stdmeta	standardsasreferences	4			
111	CDISC-ADAM	2.1	stdmeta	standardsasreferences	3			
112	CDISC-ADAM	2.1	stdmeta	standardsasreferences	1			
113	CDISC-ADAM	2.1	stdmeta	standardsasreferences	2			
114	CDISC-ADAM	2.1	stdmeta	standardsasreferences	2			
115	CDISC-ADAM	2.1	stdmeta	standardsasreferences	1			
116	CDISC-ADAM	2.1	stdmeta	standardsasreferences	2	dataset	tmplt.columnmetadata	
117	CDISC-ADAM	2.1	stdmeta	standardsasreferences	1	dataset	tmplt.class_tables	
118	CDISC-ADAM	2.1	stdmeta	standardsasreferences	1	dataset	csttmplt.sasreferences	
119	CDISC-ADAM	2.1	stdmeta	standardsasreferences	2	dataset	csttmplt.validation_master	
120	CDISC-ADAM	2.1	stdmeta	standardsasreferences	3	dataset	csttmplt.standardlookup	
121	CDISC-ADAM	2.1	stdmeta	standardsasreferences	5	dataset	csttmplt.standardmacrovariabledetails	

The availabletransforms.xml file is for XML-based standards. It defines the location of the XML schema, the location of the XSL transformation style sheets, and the import and export locations of XML documents.

This display shows the directory structure for a Microsoft Windows global standards library with **cdisc-sdtm-3.1.3-1.5** expanded.

Display 2.4 *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 9](#), 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 9](#), 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 25](#).

For more information about what a SAS Clinical Standards Toolkit standard is, see [Chapter 4, “Supported Standards,” on page 57](#).

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 33](#). 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 33](#). 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 in the *global standards library directory* and 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 the online macro API reference documentation.

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-1.5/
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 3.1.3 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 3.1.3) 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 9](#).

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.

Here is another, simpler method to determine the current SAS Clinical Standards Toolkit release:

```
%put CST Version: %cstutil_getcstversion;
```

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 2.5 StandardSASReferences Returned in work.sasrefs Data Set (Column Subset)

	standard	standardversion	type	subtype	SASref	reftype	iotype	filetype	allowoverwrite	relpathprefix	path
1	CST-FRAMEWORK	1.2	control	reference	csttmp	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
2	CST-FRAMEWORK	1.2	cstmetadata	lookup	control	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
3	CST-FRAMEWORK	1.2	cstmetadata	lookup	cstmeta	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
4	CST-FRAMEWORK	1.2	cstmetadata	macrovariabledetails	control	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
5	CST-FRAMEWORK	1.2	cstmetadata	macrovariabledetails	cstmeta	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
6	CST-FRAMEWORK	1.2	cstmetadata	macrovariables	control	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
7	CST-FRAMEWORK	1.2	cstmetadata	macrovariables	cstmeta	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
8	CST-FRAMEWORK	1.2	cstmetadata	sasreferences	control	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
9	CST-FRAMEWORK	1.2	cstmetadata	sasreferences	cstmeta	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
10	CST-FRAMEWORK	1.2	cstmetadata	standard	control	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
11	CST-FRAMEWORK	1.2	cstmetadata	standard	cstmeta	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
12	CST-FRAMEWORK	1.2	lookup		lookup	libref	input	dataset	N		%_cstGRoot/metadata
13	CST-FRAMEWORK	1.2	messages		cstmsg	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
14	CST-FRAMEWORK	1.2	properties	initialize	cstprop	fileref	input	file	N		%_cstGRoot/standards/cst-framework-1.
15	CST-FRAMEWORK	1.2	properties	validation	valprop	fileref	input	file	N		%_cstGRoot/standards/cst-framework-1.
16	CST-FRAMEWORK	1.2	referencecontrol	validation	cstcntrl	libref	input	dataset	N		%_cstGRoot/standards/cst-framework-1.
17	CST-FRAMEWORK	1.2	template		csttmpl	libref	input	folder	N		%_cstGRoot/standards/cst-framework-1.

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_GETSTANDARDSASREFERENCES will stop executing.
```

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

Creating Data Sets Used by the Framework

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

```
/*
Create the empty SASReferences data set used in the next
step
*/
%cst_createdsfromtemplate(
    _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 14 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_createdsfromtemplate(
    _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';
    iotype='input';
    filetype='dataset';
    allowoverwrite='N';
    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_createdsfromtemplate(
    _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='cstmsg';
    reftype='libref';
    order=1;
    iotype='input';
    filetype='dataset';
    allowoverwrite='N';
    output;
    standard='CST-FRAMEWORK';
    standardversion='1.2';
    type='lookup';
    subtype='';
    sasref='cstlkup';
    reftype='libref';
    order=1;
    iotype='input';
    filetype='dataset';
    allowoverwrite='N';

```

```

output;
standard='CST-FRAMEWORK';
standardversion='1.2';
type='results';
subtype='validationresults';
sasref='cstrslt';
reftype='libref';
order=1;
iotype='output';
filetype='dataset';
allowoverwrite='Y';
output;
run;

```

Here is what the data set looks like:

Display 2.6 Example SASReferences Data Set

	standard	standardversion	type	subtype	SASref	iotype	filetype	path	order	memname
1	CST-FRAMEWORK	1.2	messages		cstmsg	input	dataset		1	
2	CST-FRAMEWORK	1.2	lookup		cstlkup	input	dataset		1	
3	CST-FRAMEWORK	1.2	results	results	cstrslt	output	dataset		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 2.7 work.outSASRefs Data Set with Added Content

	standard	standardversion	type	subtype	SASref	reftype	iotype	filetype	path	order	memname
1	CST-FRAMEWORK	1.2	lookup		cstlkup	libref	input	dataset	&_cstGRoot./metadata	1	standardlookup.sas7bdat
2	CST-FRAMEWORK	1.2	messages		cstmsg	libref	input	dataset	&_cstGRoot./standards/cst-framework-1.5/messages	1	messages.sas7bdat
3	CST-FRAMEWORK	1.2	results	results	cstrslt	libref	output	dataset		1	

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 the online macro API reference documentation.

TIP Best Practice Recommendation: Do not modify global standards library files provided by SAS. Instead, modify copies of these files. Leaving the SAS 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),
    _cstStdLookupDSName=standardlookup;

```

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.

Note: This subfolder must exist before registering the standard.

- 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 name of the Standardlookup data set. This data set has the same structure as the data set in the *global standards library directory/metadata* directory. This data set is optional.

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 13](#) 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 33](#). 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

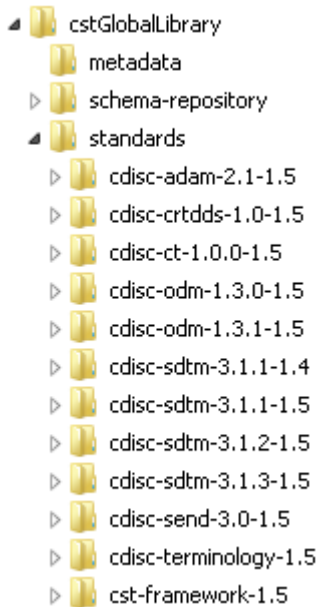
Suppose that the SAS Clinical Standards Toolkit 1.4 is currently installed and used. The SAS Clinical Standards Toolkit 1.5 is released. You want the product updates for a standard version. In the following steps, the CDISC SDTM standard is used as an example. However, the steps apply to all other standard versions. You want to set version 3.1.3 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 1.4 code base or metadata that you want to preserve. Or, you might want to test the SAS Clinical Standards Toolkit 1.5 CDISC SDTM 3.1.3 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 2.8 *Multiple Versions per Standard in the Global Standards Library*



The `cdisc-sdtm-3.1.1-1.4` directory contains files installed with the SAS Clinical Standards Toolkit 1.4. The `cdisc-sdtm-3.1.1-1.5` directory contains files installed with the SAS Clinical Standards Toolkit 1.5.

- 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.-1.4 indicates that it is the original version that was shipped with the SAS Clinical Standards Toolkit 1.4. It is defined as the default version for the CDISC SDTM standard.

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

standard	mnemonic	standardversion	rootpath	isstandarddefault	productrevision
CDISC-ADAM	ADAM	2.1	&_cstGRoot./standards/cdisc-adam-2.1-1.5	Y	1.5
CDISC-CRTDDS	CRT	1.0	&_cstGRoot./standards/cdisc-crtdds-1.0-1.5	Y	1.5
CDISC-CT	CTX	1.0.0	&_cstGRoot./standards/cdisc-ct-1.0.0-1.5	Y	1.5
CDISC-ODM	ODM	1.3.0	&_cstGRoot./standards/cdisc-odm-1.3.0-1.5	N	1.5
CDISC-ODM	ODM	1.3.1	&_cstGRoot./standards/cdisc-odm-1.3.1-1.5	Y	1.5
CDISC-SDTM	SDTM	3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.4	Y	1.4
CDISC-SDTM	SDTM	3.1.2	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5	N	1.5

Step 2: Register the updated CDISC SDTM 3.1.1 metadata in the global standards library to use the SAS Clinical Standards Toolkit 1.5.

- 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-3.1.1-1.5/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 1.5, and CDISC SDTM 3.1.2 is registered as the default standard.

Display 2.10 Global Standards Library Metadata Standards Data Set after Updates

standard	mnemonic	standardversion	rootpath	isstandarddefault	productrevision
CDISC-ADAM	ADAM	2.1	&_cstGRoot./standards/cdisc-adam-2.1-1.5	Y	1.5
CDISC-CRTDDS	CRT	1.0	&_cstGRoot./standards/cdisc-crtdds-1.0-1.5	Y	1.5
CDISC-CT	CTX	1.0.0	&_cstGRoot./standards/cdisc-ct-1.0.0-1.5	Y	1.5
CDISC-ODM	ODM	1.3.0	&_cstGRoot./standards/cdisc-odm-1.3.0-1.5	N	1.5
CDISC-ODM	ODM	1.3.1	&_cstGRoot./standards/cdisc-odm-1.3.1-1.5	Y	1.5
CDISC-SDTM	SDTM	3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.5	N	1.5
CDISC-SDTM	SDTM	3.1.2	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5	Y	1.5

3

Metadata File Descriptions

Overview	34
Standards	34
StandardSASReferences	37
Standardlookup	39
SASReferences	42
Properties	45
Messages	47
Results	50
Additional Metadata Files	54
Overview	54
Validation Master (Validation Control)	54
Reference Tables (Source Tables)	54
Reference Columns (Source Columns)	55
Validation Metrics	55
CDISC CRT-DDS Style Sheet	55

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.
groupname	(\$20)	The standard group across versions, such as STDM or TERMINOLOGY.
groupversion	(\$20)	The version of the groupname, often the same as standardversion.
comment	(\$200)	A description of the registered standard version.

Column Name	Column Length	Description
rootpath	(\$200)	The root path for the standard version's directory in the global standards library.
studylibraryrootpath	(\$200)	The root path to the study repository. This can be used to initialize the studyRootPath and studyOutputPath global macro variables and to use relative paths to study library subfolders. By default, this is set to the sample library that is associated with each standard provided by SAS.
controlsubfolder	(\$200)	The control folder path (relative to rootpath). This value provides the location of data sets that are required for standard registration (such as Standards and StandardSASReferences).
templatesubfolder	(\$200)	The template folder path (relative to rootpath). This value provides the location of data sets that are specific to the standard that serve as templates for standard-specific processes.
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.

Column Name	Column Length	Description
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 1.5 (the columns are continued in the subsequent two images):

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

	standard	mnemonic	standardversion	groupname	groupversion	comment	rootpath
1	CDISC-ADAM	ADAM	2.1	ADAM	2.1	CDISC ADAM V2.1	&_cstGRoot./standards/cdisc-adam-2.1-1.5
2	CDISC-CRTDDS	CRT	1.0	CRTDDS	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1.0-1.5
3	CDISC-CT	CTX	1.0.0	CTX	1.0.0	CDISC CT XML V1.0.0	&_cstGRoot./standards/cdisc-ct-1.0.0-1.5
4	CDISC-ODM	ODM	1.3.0	ODM	1.3.0	CDISC ODM V1.3.0	&_cstGRoot./standards/cdisc-odm-1.3.0-1.5
5	CDISC-ODM	ODM	1.3.1	ODM	1.3.1	CDISC ODM V1.3.1	&_cstGRoot./standards/cdisc-odm-1.3.1-1.5
6	CDISC-SDTM	SDTM	3.1.1	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.5
7	CDISC-SDTM	SDTM	3.1.2	SDTM	3.1.2	CDISC SDTM V3.1.2	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5
8	CDISC-SDTM	SDTM	3.1.3	SDTM	3.1.3	CDISC SDTM V3.1.3	&_cstGRoot./standards/cdisc-sdtm-3.1.3-1.5
9	CDISC-SEND	SEND	3.0	SEND	3.0	CDISC SEND V3.0	&_cstGRoot./standards/cdisc-send-3.0-1.5
10	CDISC-TERMINOLOGY	CT	NCI_THESAURUS	TERMINOLOGY	NCI_THESAURUS	CDISC Terminology	&_cstGRoot./standards/cdisc-terminology-1.5
11	CST-FRAMEWORK	CST	1.2	FRAMEWORK	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework-1.5

	standard	standardversion	studylibraryrootpath	controlsubfolder	templatesubfolder	isstandarddefault	iscstframework	isdatastandard	supportvalidation
1	CDISC-ADAM	2.1	&_cstSRoot./cdisc-adam-2.1-1.5/sascstdemodata	control	templates	Y	N	Y	Y
2	CDISC-CRTDDS	1.0	&_cstSRoot./cdisc-crtdds-1.0-1.5	control	templates	Y	N	Y	Y
3	CDISC-CT	1.0.0	&_cstSRoot./cdisc-ct-1.0.0-1.5	control	templates	Y	N	Y	Y
4	CDISC-ODM	1.3.0	&_cstSRoot./cdisc-odm-1.3.0-1.5	control	templates	N	N	Y	Y
5	CDISC-ODM	1.3.1	&_cstSRoot./cdisc-odm-1.3.1-1.5	control	templates	Y	N	Y	Y
6	CDISC-SDTM	3.1.1	&_cstSRoot./cdisc-sdtm-3.1.1-1.5/sascstdemodata	control	templates	N	N	Y	Y
7	CDISC-SDTM	3.1.2	&_cstSRoot./cdisc-sdtm-3.1.2-1.5/sascstdemodata	control	templates	N	N	Y	Y
8	CDISC-SDTM	3.1.3	&_cstSRoot./cdisc-sdtm-3.1.3-1.5/sascstdemodata	control	templates	Y	N	Y	Y
9	CDISC-SEND	3.0	&_cstSRoot./cdisc-send-3.0-1.5/sascstdemodata	control	templates	Y	N	Y	N
10	CDISC-TERMINOLOGY	NCI_THESAURUS		control	templates	Y	N	N	N
11	CST-FRAMEWORK	1.2	&_cstSRoot./cst-framework-1.5	control	templates	Y	Y	N	Y

	standard	standardversion	isxmlstandard	importxsl	exportxsl	schema	productrevision
1	CDISC-ADAM	2.1	N				1.5
2	CDISC-CRTDDS	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	1.5
3	CDISC-CT	1.0.0	Y	CT/1.0/import/Root.xsl	CT/1.0/export/Root.xsl	cdisc-ct-1.0.0/controlledterminology1-0-0	1.5
4	CDISC-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	1.5
5	CDISC-ODM	1.3.1	Y	ODM/1.3.1/import/Root.xsl	ODM/1.3.1/export/Root.xsl	cdisc-odm-1.3.1/ODM1-3-1.xsd	1.5
6	CDISC-SDTM	3.1.1	N				1.5
7	CDISC-SDTM	3.1.2	N				1.5
8	CDISC-SDTM	3.1.3	N				1.5
9	CDISC-SEND	3.0	N				1.5
10	CDISC-TERMINOLOGY	NCI_THESAURUS	N				1.5
11	CST-FRAMEWORK	1.2	N				1.5

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

`&_cstSRoot` in the **studylibraryrootpath** column maps to the *sample study library directory* that is set by calling the `cstutil_setcstsroot` 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-3.1.2-1.5/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-3.1.2-1.5/control/standardsasreferences.sas7bdat
```

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

standard	standardversion	type	subtype	SASref	reftype	iotype	filetype	allowoverwrite	relpathprefix	path	order	memname
CDISC-SDTM	3.1.2	autocall		autocall	fileref	input	folder	N	rootpath	macros	1	
CDISC-SDTM	3.1.2	classmetadata	column	refmeta	libref	input	dataset	N	rootpath	metadata		. class_columns.sas7bdat
CDISC-SDTM	3.1.2	classmetadata	table	refmeta	libref	input	dataset	N	rootpath	metadata		. class_tables.sas7bdat
CDISC-SDTM	3.1.2	cstmadata	lookup	stdmeta	libref	input	dataset	N	rootpath	control		. standardlookup.sas7bdat
CDISC-SDTM	3.1.2	cstmadata	macrovariabledetails	stdmeta	libref	input	dataset	N	rootpath	control		. standardmacrovariabledetails.sas7bdat
CDISC-SDTM	3.1.2	cstmadata	macrovariables	stdmeta	libref	input	dataset	N	rootpath	control		. standardmacrovariables.sas7bdat
CDISC-SDTM	3.1.2	cstmadata	sasreferences	stdmeta	libref	input	dataset	N	rootpath	control		. standardsasreferences.sas7bdat
CDISC-SDTM	3.1.2	cstmadata	standard	stdmeta	libref	input	dataset	N	rootpath	control		. standards.sas7bdat
CDISC-SDTM	3.1.2	lookup		lookup	libref	input	dataset	N	rootpath	control		. standardlookup.sas7bdat
CDISC-SDTM	3.1.2	messages		messages	libref	input	dataset	N	rootpath	messages	1	messages.sas7bdat
CDISC-SDTM	3.1.2	properties	initialize	initprop	fileref	input	file	N	rootpath	programs	1	initialize.properties
CDISC-SDTM	3.1.2	properties	validation	valprop	fileref	input	file	N	rootpath	programs	2	validation.properties
CDISC-SDTM	3.1.2	referencecontrol	checktable	refcntl	libref	input	dataset	N	rootpath	validation/control		. validation_domainsbycheck.sas7bdat
CDISC-SDTM	3.1.2	referencecontrol	standardref	refcntl	libref	input	dataset	N	rootpath	validation/control		. validation_stdref.sas7bdat
CDISC-SDTM	3.1.2	referencecontrol	validation	refcntl	libref	input	dataset	N	rootpath	validation/control		. validation_master.sas7bdat
CDISC-SDTM	3.1.2	referencemetadata	column	refmeta	libref	input	dataset	N	rootpath	metadata		. reference_columns.sas7bdat
CDISC-SDTM	3.1.2	referencemetadata	table	refmeta	libref	input	dataset	N	rootpath	metadata		. reference_tables.sas7bdat
CDISC-SDTM	3.1.2	template		tmplt	libref	input	folder	N	rootpath	templates		.

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 contains a relative path. The **relpathprefix** value **rootpath** instructs the code to use the rootpath location that is specified in the standard-specific Standards data set. The resolved path is shown in [Display 3.3 on page 39](#).

The cross-standard global standards library StandardSASReferences data set that is provided with the SAS Clinical Standards Toolkit is located 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 following enhancements are the only enhancements to the data set during concatenation:

- the **path** column is resolved to the full global standards library path for each record, based on the **relpathprefix** value
- the **relpathprefix** column is reset to null

This display shows the content for the CDISC SDTM StandardSASReferences data set that is described in [Display 3.2 on page 38](#). 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	SASref	reftype	iotype	filetype	allowoverwrite	relpathprefix	path
103	CDISC-SDTM	3.1.2	autocall		autocall	fileref	input	folder	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/macros
104	CDISC-SDTM	3.1.2	classmetadata	column	refmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/metadata
105	CDISC-SDTM	3.1.2	classmetadata	table	refmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/metadata
106	CDISC-SDTM	3.1.2	cstmetadata	lookup	stdmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/control
107	CDISC-SDTM	3.1.2	cstmetadata	macrovariabledetails	stdmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/control
108	CDISC-SDTM	3.1.2	cstmetadata	macrovariables	stdmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/control
109	CDISC-SDTM	3.1.2	cstmetadata	sasreferences	stdmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/control
110	CDISC-SDTM	3.1.2	cstmetadata	standard	stdmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/control
111	CDISC-SDTM	3.1.2	lookup		lookup	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/control
112	CDISC-SDTM	3.1.2	messages		messages	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/messages
113	CDISC-SDTM	3.1.2	properties	initialize	initprop	fileref	input	file	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/programs
114	CDISC-SDTM	3.1.2	properties	validation	valprop	fileref	input	file	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/programs
115	CDISC-SDTM	3.1.2	referencecontrol	checktable	refcntl	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/validation/control
116	CDISC-SDTM	3.1.2	referencecontrol	standardref	refcntl	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/validation/control
117	CDISC-SDTM	3.1.2	referencecontrol	validation	refcntl	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/validation/control
118	CDISC-SDTM	3.1.2	referencemetadata	column	refmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/metadata
119	CDISC-SDTM	3.1.2	referencemetadata	table	refmeta	libref	input	dataset	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/metadata
120	CDISC-SDTM	3.1.2	template		tmpit	libref	input	folder	N		&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5/templates

The structure of all StandardSASReferences data sets is the same for all standards provided by SAS in the SAS Clinical Standards Toolkit. This structure is described in “SASReferences” on page 42.

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
standard	(\$20)	The name of the registered standard.

Column Name	Column Length	Description
standardversion	(\$20)	The version number of the registered standard. This must be unique within the standard.
SASref	(\$8)	SAS libref
table	(\$32)	A SAS Clinical Standards Toolkit table name
column	(\$32)	A SAS Clinical Standards Toolkit column name
refcolumn	(\$32)	Associated SAS Clinical Standards Toolkit column name
refvalue	(\$200)	Associated SAS Clinical Standards Toolkit column value
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
templatetype	(\$8)	For the given record, a non-null value (for example, data set) indicates that a template is available. For example, the macro call <code>%cst_createdsfromtemplate(_cstStandard=CST-FRAMEWORK, _cstType=control,_cstSubType=reference, _cstOutputDS=work.sasreferences)</code> finds that a template is available as <code>csttmplt.sasreferences</code> .
template	(\$40)	The SAS reference (libref.dset or fileref) to the templatetype. For example, <code>csttmplt.sasreferences</code> points to <i>global standards library directory/standards/cst-framework-1.5/templates/sasreferences.sas7bdat</i> .
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 in the definition and registration of custom standards in the SAS Clinical Standards Toolkit.

The cross-standard global standards library Standardlookup data set that is provided with the SAS Clinical Standards Toolkit is located here:

*global standards library directory/metadata/
standardlookup.sas7bdat*

This data set contains the concatenation of each Standardlookup data set that is provided for each supported standard in the SAS Clinical Standards Toolkit.

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

Display 3.4 Standardlookup Data Set Content in the Global Standards Library

standard	standardversion	SASref	table	column	refcolumn	refvalue	value	default	nonnull	order	templatetype	template
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	allowoverwrite			N	Y	Y	1		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	allowoverwrite			Y	N	Y	2		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	filetype			CATALOG	N	Y	2		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	filetype			DATASET	Y	Y	1		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	filetype			FILE	N	Y	3		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	filetype			FOLDER	N	Y	4		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	ioctype			BOTH	N	Y	3		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	ioctype			INPUT	Y	Y	1		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	ioctype			OUTPUT	N	Y	2		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	reftype			FILEREF	N	Y	2		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	reftype			LIBREF	Y	Y	1		
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	subtype	type	CLASSMETADATA	COLUMN	N	N	2	dataset	tmplt.columnmetadata
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	subtype	type	CLASSMETADATA	TABLE	Y	N	1	dataset	tmplt.tablemetadata
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	subtype	type	CONTROL	REFERENCE	Y	N	1	dataset	csttmplt.sasreferences
CDISC-SDTM	3.1.2	stdmeta	standardsasreferences	subtype	type	CONTROL	VALIDATION	N	N	2	dataset	cattmplt.validation_master

These records show the valid values for discrete columns in any SDTM 3.1.2 SASReferences (including StandardSASReferences) data set. For example, **filetype** can have values of CATALOG, DATASET, FILE, or FOLDER. These records also show that a SASReferences data set allows two **subtype** values (REFERENCE and VALIDATION) when **type** is CONTROL. When **type** is CONTROL, the **subtype** value must always be non-null.

Templates are available for both the SASReferences data set and the validation_master data sets. For more information about the columns and values in SASReferences data sets, 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 93, 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 <i>global standards library directory/metadata</i> 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 <i>global standards library directory/metadata</i> 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 <i>global standards library directory/standards/cst-framework-1.5/control/standardlookup</i> data set. These values are also itemized in Table 5.1 on page 96 . This column is required.

Column Name	Column Length	Description
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 <i>global standards library directory/standards/cst-framework-1.5/control/standardlookup</i> data set. These values are also itemized in Table 5.1 on page 96 . 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)	The reference type. This column is required. Valid values are libref and fileref.
iotype	(\$8)	The input/output type (input, output, or both) of the entity. Entities defined as “input” or “both” must exist and be accessible. If not, calls to the cstutilvalidatesasreferences macro report an error condition and halt the process.
filetype	(\$8)	The file type (folder, dataset, catalog, or file).
allowoverwrite	(\$1)	Allow the file to be overwritten (Y/N), for files with an iotype value of “output” or “both”.
relpathprefix	(\$41)	The relative path prefix (for example, rootpath, studylibraryrootpath, or &mypath). If non-null, the value of the path is assumed to be relative to the resolved relpathprefix. The reserved values rootpath and studylibraryrootpath have special significance: they instruct the SAS Clinical Standards Toolkit to use the standard-specific values for these columns in the <i>global standards library directory/metadata/standards.sas7bdat</i> data set.

Column Name	Column Length	Description
path	(\$200)	<p>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).</p>
order	(8.)	<p>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:</p> <ol style="list-style-type: none"> 1 Multiple records exist within these types: autocall, fmtsearch, cmplib, messages. 2 Library concatenation is wanted (multiple librefs are within the same value of SASref for a type). 3 There is a need to establish precedence within a type (for example, look first in this library and then look in another library).
memname	(\$48)	<p>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.</p>
comment	(\$200)	<p>Explanatory comments. This column is optional.</p>

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	iotype	filetype	allowoverwrite	path	order	memname
CDISC-SDTM	3.1.2	autocall		autocall	fileref	input	folder	N		1	
CDISC-SDTM	3.1.2	control	reference	srcntli	libref	input	dataset	N	&studyRootPath/control	.	sasreferences.sas7bdat
CDISC-SDTM	3.1.2	control	validation	srcntli	libref	input	dataset	N	&studyRootPath/control	.	validation_control.sas7bdat
CDISC-SDTM	3.1.2	fmtsearch		fmnts	libref	input	catalog	N	&studyRootPath/terminology/formats	1	formats.sas7bcat
CDISC-SDTM	3.1.2	lookup		lookup	libref	input	dataset	N		.	
CDISC-SDTM	3.1.2	messages		messages	libref	input	dataset	N		.	
CDISC-SDTM	3.1.2	properties	initialize	initprop	fileref	input	file	N	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.5/messages	2	messages.sas7bdat
CDISC-SDTM	3.1.2	properties	validation	valprop	fileref	input	file	N	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.5/programs	1	initialize.properties
CDISC-SDTM	3.1.2	referencecontrol	checktable	refcntli	libref	input	dataset	N		.	
CDISC-SDTM	3.1.2	referencecontrol	standardref	refcntli	libref	input	dataset	N		.	
CDISC-SDTM	3.1.2	referencecontrol	validation	refcntli	libref	input	dataset	N		.	
CDISC-SDTM	3.1.2	referencectem		ctref	libref	input	dataset	N	&studyRootPath/terminology/coding-dictionaries	1	meddra.sas7bdat
CDISC-SDTM	3.1.2	referencemetadata	column	refmeta	libref	input	dataset	N		.	
CDISC-SDTM	3.1.2	referencemetadata	table	refmeta	libref	input	dataset	N		.	
CDISC-SDTM	3.1.2	results	validationmetrics	results	libref	output	dataset	Y	&studyOutputPath/results	.	validation_metrics.sas7bdat
CDISC-SDTM	3.1.2	results	validationresults	results	libref	output	dataset	Y	&studyOutputPath/results	.	validation_results.sas7bdat

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, minimum SASReferences file structure. All columns defined in [Table 3.1 on page 34](#) are expected; additional columns are allowed. No changes to column attributes are allowed (for example, changing column lengths).

Note: SASReferences data sets from the SAS Clinical Standards Toolkit releases prior to version 1.5 can be used in version 1.5 if they do not include any of the columns added in version 1.5 (iotype, filetype, allowoverwrite, and relpathprefix).

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. For more information about the SAS Clinical Standards Toolkit global macro variables, see [Appendix 1, “Global Macro Variables,” on page 393](#). 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*.

Table 3.4 *Properties File Structure*

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

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

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 <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. For CDISC standards supporting validation, the resultid should match the checkid from the Validation Master data set for standard records that support validation.	Required
standardversion	(\$20)	A specific version of a standard. This value must match one of the standard versions that is associated with a registered standard. This value must also match the standardversion field in the SASReferences data set. The only exception to this rule is that *** can be used to signify that the check applies to all supported versions of the standard (for example, 3.1.1, 1.0, ***). If a subsequent version of the standard is released, then *** would be applicable if the check is valid for the new version.	Required

Column Name	Column Length	Description	Optional or Required
checksource	(\$40)	A string that identifies the source of the message. This string is used to provide source-specific messages generated within the SAS Clinical Standards Toolkit. CDISC examples include Janus, OpenCDISC, SAS, and WebSDM. This field can contain any user-defined value.	Required
sourceid	(\$8)	A reference identifier for this message from the checksource .	Optional
checkseverity	(\$40)	The severity as assigned by checksource . This value is mapped to 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

Column Name	Column Length	Description	Optional or Required
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-1.5/
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

Certain 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>

Column Name	Column Length	Description
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>
srcdata	(\$200)	<p>Source data. This string generally specifies:</p> <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 <p>Value should be non-null.</p>
message	(\$500)	<p>Resolved message text from Messages data set. The message value includes up to two run-time parameter values in message text.</p> <p>Value should be non-null.</p>

Column Name	Column Length	Description
resultseverity	(\$40)	<p>Result severity (for example, warning or error).</p> <p>Info Informational note</p> <p>Note Problem detected, low severity</p> <p>Warning Problem detected, medium severity</p> <p>Warning: Check not run No assessment able to be made</p> <p>Warning: Check not completed Full compliance assessment could not be made</p> <p>Error Problem detected, high severity</p> <p>Value should be non-null.</p>
resultflag	(8.)	<p>A value that determines whether a problem has been detected. The values are 0=no, otherwise, yes.</p> <p>-1 Validation check not run</p> <p>0 No problem detected (value always 0 when resultseverity=Info)</p> <p>1 Validation check run, error detected</p> <p>Value should be non-null.</p>
_cst_rc	(8.)	<p>Process status. Values are nonzero and aborted. A nonzero value typically indicates that the process ended abnormally.</p> <p>Value should be non-null.</p>
actual	(\$240)	<p>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.</p>
keyvalues	(\$2000)	<p>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.</p>
resultdetails	(\$200)	<p>Basis or explanation for result. This column is optional.</p>

For an example of a SAS Clinical Standards Toolkit Results data set, see [Display 6.9 on page 169](#) and [Display 6.10 on page 169](#).

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.supportsvalidation` field, see [Table 3.1 on page 34](#).) This data set should have the columns as defined in [Table 6.3 on page 128](#), 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. Use of Validation Control SAS views is supported.

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 121](#) 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 123](#) 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 147](#) describes the metrics metadata for the CDISC SDTM standard, and [Display 6.2 on page 149](#) provides sample content for the CDISC SDTM standard.

CDISC CRT-DDS Style Sheet

A sample XML style sheet (`define1-0-0.xml`) 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.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. Alternative style sheets can be used to provide metadata support for CDISC CRT-DDS.

4

Supported Standards

<i>SAS Representation of Standards</i>	58
Overview	58
<i>CDISC SDTM</i>	62
Purpose	62
Release Dates	63
Description	63
CDISC SDTM 3.1.1 Reference Standard	66
CDISC SDTM 3.1.2 Reference Standard	67
CDISC SDTM 3.1.3 Reference Standard	68
<i>CDISC ADaM 2.1</i>	69
Purpose	69
Release Date	70
Regulatory Basis	70
CDISC ADaM 2.1 Reference Standard	71
<i>CDISC CRT-DDS 1.0</i>	74
Purpose	74
Release Date	74
Regulatory Basis	74
CDISC CRT-DDS 1.0 Reference Standard	75
<i>CDISC ODM</i>	79
Purpose	79
Release Dates	79
CDISC ODM 1.3.0 Reference Standard	80

CDISC ODM 1.3.1 Reference Standard	86
CDISC SEND 3.0	87
Purpose	87
Release Date	87
Overview of the CDISC SEND 3.0 Domains	88
CDISC Terminology	89
Purpose	89
CDISC Terminology Reference Standard	89

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 definition of any type of standard.

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 25](#).

SAS provides new standards and updates based on customer requirements, changes to source guidelines, and changes to 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.

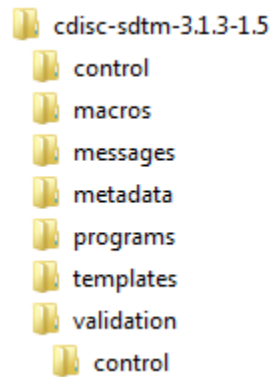
For example, consider two of the CDISC standards supported in the SAS Clinical Standards Toolkit. Each CDISC Terminology standard can be represented in its simplest form as either a SAS data set or SAS format catalog of acceptable values. Each CDISC SDTM standard can be represented as a set of domains (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:

<code>class_tables</code>	identifies a limited set of column collections specific to one or more SDTM domains.
<code>class_columns</code>	identifies the full set of column definitions used in the SDTM domains.
<code>reference_tables</code>	provides metadata for the specific data sets (domains) that are supported for CDISC SDTM. This information is different for each version of the CDISC SDTM standard.
<code>reference_columns</code>	provides metadata for the specific columns in the domains that are supported for CDISC SDTM. This information is different for each version of the CDISC SDTM standard.

- 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 7](#).

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 3.1.3

- CDISC SDTM Model, Final Version 1.3, July 16, 2012
- *CDISC SDTM Implementation Guide*, Final Version 3.1.3, July 16, 2012

Description

CDISC standards, including SDTM, 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, which 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 each CDISC SDTM standard are derived from the global standards library in these formats:

- as empty data sets (using the utility macro `cst_createTablesForDataStandard`)
- as table metadata (See [Table 4.1 on page 64.](#))
- as column metadata for each domain (See [Table 4.2 on page 64.](#))

Table 4.1 Sample reference_tables Record (CDISC SDTM 3.1.1)

Column Name	Column Value
SASref	REFMETA
Table	AE
Label	Adverse Events
Class	Events
XmlPath	.../transport/ae.xpt
XmlTitle	Adverse Events SAS transport file
Structure	One record per event per subject
Purpose	Tabulation
Keys	STUDYID USUBJID AETERM AESTDTC
State	Final
Date	2005-08-26
Standard	CDISC-SDTM
StandardVersion	3.1.1
Standardref	SDTM2.2.4
Comment	

Table 4.2 Sample reference_columns Record (CDISC SDTM 3.1.1)

Column Name	Column Value
sasref	REFMETA
table	AE

Column Name	Column Value
column	AESEV
label	Severity/Intensity
order	16
type	C
length	20
displayformat	
xmldatatype	text
xmlcodelist	
core	Perm
origin	CRF
role	RecordQualifier
term	*
algorithm	
qualifiers	UPPERCASE
standard	CDISC-SDTM
standardversion	3.1.1
standardref	
comment	The severity or intensity of the event. Examples: MILD, MODERATE, SEVERE.

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 superset of checks validating the domain structure and content for each specific SDTM version.
- 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 of the CDISC SDTM reference standards.

CDISC SDTM 3.1.1 Reference Standard

Overview of the CDISC SDTM 3.1.1 Domains

The SAS Clinical Standards Toolkit representation of the CDISC SDTM 3.1.1 standard is comprised of 25 domains (in the `reference_tables` metadata data set) and 495 columns (in the `reference_columns` metadata data set).

The 25 supported domains are shown in this table.

Table 4.3 *Supported Domains (CDISC SDTM 3.1.1)*

Adverse Events - AE	Relate Records - RELREC
Concomitant Medications - CM	Subject Characteristics - SC
Comments - CO	Subject Elements - SE
Demographics - DM	Supplemental Qualifiers - SUPPAE
Disposition - DS	Substance Use - SU
Protocol Deviations - DV	Subject Visits - SV

ECG Tests - EG	Trial Arms - TA
Exposure - EX	Trial Elements - TE
Inclusion/Exclusion Exceptions - IE	Trial Inclusion/Exclusion Criteria - TI
Laboratory Tests - LB	Trial Summary - TS
Medical History - MH	Trial Visits - TV
Physical Examinations - PE	Vital Signs - VS
Questionnaires - QS	

CDISC SDTM 3.1.2 Reference Standard

Overview of the CDISC SDTM 3.1.2 Domains

The SAS Clinical Standards Toolkit representation of the CDISC SDTM 3.1.2 standard is comprised of 32 domains (in the reference_tables metadata data set) and 723 columns (in the reference_columns metadata data set).

The 32 supported domains are shown in this table.

Table 4.4 Supported Domains (CDISC SDTM 3.1.2)

Adverse Events - AE	PK Concentrations - PC
Clinical Events - CE	Physical Examination - PE
Concomitant Medications - CM	PK Parameters - PP
Comments - CO	Questionnaires - QS
Drug Accountability - DA	Related Records - RELREC
Demographics - DM	Subject Characteristics - SC
Disposition - DS	Subject Elements - SE

Protocol Deviations - DV	Substance Use - SU
ECG Test Results - EG	Supplemental Qualifiers - AE - SUPPAE
Exposure - EX	Subject Visits - SV
Findings About - FA	Trial Arms - TA
Inclusion/Exclusion Criterion Not Met - IE	Trial Elements - TE
Laboratory Test Results - LB	Trial Inclusion/Exclusion Criteria - TI
Microbiology Specimen - MB	Trial Summary - TS
Medical History - MH	Trial Visits - TV
Microbiology Susceptibility Test - MS	Vital Signs - VS

CDISC SDTM 3.1.3 Reference Standard

Overview of the CDISC SDTM 3.1.3 Domains

The SAS Clinical Standards Toolkit representation of the CDISC SDTM 3.1.3 standard is comprised of 36 domains (in the reference_tables metadata data set) and 821 columns (in the reference_columns metadata data set).

The 36 supported domains are shown in this table.

Table 4.5 Supported Domains (CDISC SDTM 3.1.3)

Adverse Events - AE	Clinical Events - CE
Concomitant Medications - CM	Comments - CO
Drug Accountability - DA	Demographics - DM
Disposition - DS	Protocol Deviations - DV
ECG Test Results - EG	Exposure - EX

Findings About - FA	Inclusion/Exclusion Criterion Not Met - IE
Laboratory Test Results - LB	Microbiology Specimen - MB
Medical History - MH	Microbiology Susceptibility - MS
PK Concentrations - PC	Physical Examination - PE
Pool Definition - POOLDEF	PK Parameters - PP
Questionnaire - QS	Related Records - RELREC
Disease Response - RS	Subject Characteristics - SC
Subject Elements - SE	Substance Use - SU
Supplemental Qualifiers - AE - SUPPAE	Subject Visits - SV
Trial Arms - TA	Trial Elements - TE
Trial Inclusion/Exclusion Criteria - TI	Tumor Results - TR
Trial Summary - TS	Tumor Identification - TU
Trial Visits - TV	Vital Signs - VS

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

The ADaM Basic Data Structure for Time-to-Event Analyses, Version 1.0, May 8, 2012

Analysis Data Model (ADaM) Data Structure for Adverse Event Analysis, Version 1.0, May 10, 2012

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 1.1/December 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/UCM312964.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.

The CDISC ADaM 2.1 reference standard currently supports these 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 Time-to-Event analysis data set is an example implementation of the BDS structure.
- The Adverse Event analysis data set (ADAE) structure is built on the nomenclature of the CDISC SDTM Implementation Guides for collected data. The ADAE data set adds attributes, variables, and data structures that are required for statistical analyses. The primary SDTM source domain for the ADAE data set is AE, with the corresponding SUPPAE. Additional variables can be added from the ADaM ADSL data set. The ADAE data set is required when SDTM AE is not sufficient to support all adverse event analyses. The ADAE structure for the standard adverse event safety data set has at least one record per each AE recorded in the SDTM AE domain.

Metadata for the ADSL, BDS, and ADAE data sets is defined in the SAS Clinical Standards Toolkit `reference_tables` data set in the standard metadata folder.

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

Table 4.6 *ADaM Metadata Types and SAS Clinical Standards Toolkit Locations*

ADaM Metadata Type	SAS Clinical Standards Toolkit Location
Analysis data set metadata	global standards library reference_tables.sas7bdat
Analysis variable metadata	global standards library reference_columns.sas7bdat
Analysis parameter-value-level metadata	global standards library valuemetadata.sas7bdat template sample library metadata source_values.sas7bdat example
Analysis results metadata	global standards library analysis_results.sas7bdat template sample library metadata analysis_results.sas7bdat example

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 ADaM analysis data sets. This set of ADSL and BDS columns has been supplemented with Version 1.0 of the Analysis Data Model (ADaM) Data Structure for Adverse Event Analysis. Metadata for the 290 columns in the SAS representation of ADSL, BDS, and ADAE is defined in the SAS Clinical Standards Toolkit reference_columns data set in the standard metadata folder. Empty ADSL, BDS, and ADAE data sets containing these columns can be derived from the SAS Clinical Standards Toolkit global standards library using the utility macro `cst_createTablesForDataStandard`.

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 versions 1.1 and 1.2 of the CDISC ADaM Validation Checks as prepared by the CDISC ADaM team, as well as selected checks that are unique to the SAS Clinical Standards Toolkit.
- 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 tabulation and analysis 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 4.7 on page 75](#)])
- as column metadata for 176 columns in the 39 data sets (`reference_columns` in the standard metadata folder)

Table 4.7 *reference_tables* (CDISC CRT-DDS 1.0)

AnnotatedCRFs	ItemGroupAliases	MDVLeafTitles
CLItemDecodeTranslatedText	ItemGroupDefItemRefs	MUTranslatedText
CodeListLitems	ItemGroupDefs	MeasurementUnits
CodeLists	ItemGroupLeaf	MetaDataVersion
ComputationMethods	ItemGroupLeafTitles	Presentation
DefineDocument	ItemMUREfs	ProtocolEventRefs
ExternalCodeLists	ItemQuestionExternal	RCErrorsTranslatedText
FormDefArchLayouts	ItemQuestionTranslatedText	Study

FormDefItemGroupRefs	ItemRangeCheckValues	StudyEventDefs
FormDefs	ItemRangeChecks	StudyEventFormRefs
ImputationMethods	ItemRole	SupplementalDocs
ItemAliases	ItemValueListRefs	ValueListItemRefs
ItemDefs	MDVLeaf	ValueLists

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	IG.TE
Name	TE

Column	Value
Repeating	No
IsReferenceData	Yes
SASDatasetName	TE
Domain	TE
Origin	
Role	
Purpose	Tabulation
Comment	Elements are the building blocks of Arms. Arms consisting of Elements are the paths subjects will follow.
Label	Trial Elements
Class	Trial Design
Structure	One record per planned element
DomainKeys	STUDYID, ETCD
ArchiveLocationID	Location.TE
FK_MetaDataVersion	MDV.1

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

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 4.9 CDISC CRT-DDS Default Lengths by Data Type

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

Note: CRT-DDS and ODM use slightly different lengths.

CDISC CRT-DDS SAS Data Set Construction

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, the structural integrity of the define.xml file is validated, and a define.pdf file can be generated. To support this functionality, supplemental files include these global standards library files:

- A SAS format catalog (crtdsct.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 superset 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.xsl` and `define-v1-updated-html.xsl` XSL style sheets.

The `define1-0-0.xsl` style sheet was the original style sheet published by CDISC in 2005. It can be found at <http://www.cdisc.org/define-xml>.

The `define-v1-updated-html.xsl` style sheet was used in the 2013 update to the first CDISC SDTM/ADaM Pilot Project (<http://www.cdisc.org/sdtm-adam-pilot-project>).

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

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 Dates

- CDISC ODM, Version 1.3.0, December 15, 2006
- CDISC ODM, Version 1.3.1, February 11, 2010

CDISC ODM 1.3.0 Reference Standard

The SAS Clinical Standards Toolkit 1.5 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
- extracting one or more data sets from the ClinicalData or ReferenceData sections of the ODM XML file

The SAS Clinical Standards Toolkit 1.5 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
- 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 1.5 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 [Table 4.11 on page 82](#)])
- as column metadata for 315 columns in the 66 data sets (reference_columns in the standard metadata folder)

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 4.10 Sample Mapping of odm.xml File to SAS Representation

XML Element or Attribute	SAS Data Set	SAS Column	SAS Column Value
<ClinicalData StudyOID="P2006-101" MetadataVersionOID="101.01" >	ClinicalData	StudyOID MetadataVersionOID	"P2006-101" "101.01"
<SubjectData SubjectKey="1000" TransactionType="Insert">	SubjectData	SubjectKey TransactionType	"1000" "Insert"
<StudyEventData StudyEventOID="101.Screen" >	StudyEventData	StudyEventOID	"101.Screen"

XML Element or Attribute	SAS Data Set	SAS Column	SAS Column Value
<FormData FormOID="101.DEMOG">	FormData	FormOID	"101.DEMOG"
<ItemGroupData ItemGroupOID="101.DM">	ItemGroupData	ItemGroupOID	"101.DM"
<ItemDataString ItemOID="101.USUBJID">101 -01-01</ItemDataString>	ItemData	ItemOID ItemDataType Value	"101.USUBJID" "ItemDataString" "101-01-01"
<ItemDataString ItemOID="101.SEX">F</ ItemDataString>	ItemData	ItemOID ItemDataType Value	"101.SEX" "ItemDataString" "F"

This table lists the complete set of 66 tables that form the SAS Clinical Standards Toolkit 1.5 SAS representation of the CDISC ODM 1.3.0 standard.

Table 4.11 *reference_tables (CDISC ODM 1.3.0)*

admindata	itemrangecheckvalues
annotation	itemrcformalexpression
annotationflag	itemrole
association	keyset
auditrecord	location
clinicaldata	locationversion
clitemdecodetranslatedtext	measurementunits
codelistitems	metadataversion
codelists	methoddefformalexpression

conditiondeformalexpression	methoddefs
conditiondefs	methoddeftranslatedtext
conditiondeftranslatedtext	mutranslatedtext
enumerateditems	odm
externalcodelists	presentation
formdata	protocoleventrefs
formdefarchlayouts	protocoltranslatedtext
formdefitemgrouprefs	rcerrortranslatedtext
formdefs	referencedata
formdeftranslatedtext	signature
imputationmethods	signaturedef
itemaliases	study
itemdata	studyeventdata
itemdefs	studyeventdefs
itemdeftranslatedtext	studyeventdeftranslatedtext
itemgroupaliases	studyeventformrefs
itemgroupdata	subjectdata
itemgroupdefitemrefs	user
itemgroupdefs	useraddress
itemgroupdeftranslatedtext	useraddressstreetname
itemmurefs	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 4.12 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-1.3.0-1.5/metadata
```

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

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.

In the SAS Clinical Standards Toolkit 1.5, the CDISC ODM standard supports reading and representing in SAS a complete `odm.xml` file, and building an `odm.xml` file. The SAS Clinical Standards Toolkit validates both the structure and content of the SAS

representation of each odm.xml file and the structural integrity of that file. The SAS Clinical Standards Toolkit also supports the extraction of subject or reference data for a data set (such as an SDTM AE domain) from the odm.xml file.

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

- A SAS format catalog (odmct.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 1.3.0 reference standard.

CDISC ODM 1.3.1 Reference Standard

The CDISC ODM 1.3.1 reference standard has the same functionality as CDISC ODM 1.3.0, with the following differences:

- The SAS representation of CDISC ODM 1.3.1 includes 10 data sets in addition to those shown in [Table 4.11 on page 82](#). The 10 additional data sets are listed in this table:

Table 4.13 Additional CDISC ODM 1.3.1 Tables Not Included with CDISC ODM 1.3.0

codelistaliases	formaliases
codelistitemaliases	methodaliases
codelisttranslatedtext	mualias
conditionalaliases	protocolaliases

enumerateditemaliases

studyeventaliases

- The table metadata for these 76 data sets can be found in the reference_tables data set in the standard metadata folder. Column metadata for the 352 columns in these 76 data sets can be found in the reference_columns data set in the standard metadata folder.

This set of files, in whole or in part, defines the CDISC ODM 1.3.1 reference standard.

CDISC SEND 3.0

Purpose

The CDISC SEND standard defines a standard structure for data tabulations that are designed to support single-dose general toxicology studies, repeat-dose general toxicology studies, and carcinogenicity non-clinical studies. CDISC SEND is based on CDISC SDTM. These data tabulations are submitted as part of a product application to a regulatory authority such as the FDA.

The data sets and columns required for a product application are not prescribed by the standard. Instead, requirements are based on the trial protocol and discussions with the regulatory authority in charge of reviewing the application. Therefore, any SAS Clinical Standards Toolkit standard, including the CDISC SEND standard, is only a representative sample or template.

Release Date

CDISC Standard for Exchange of Nonclinical Data (SEND), Final Version 3.0, May 19, 2011

Overview of the CDISC SEND 3.0 Domains

The SAS Clinical Standards Toolkit representation of the CDISC SEND 3.0 standard is comprised of 28 domains (in the reference_tables metadata data set) and 563 columns (in the reference_columns metadata data set).

The 28 domains are shown in this table:

Table 4.14 Supported Domains (CDISC SEND 3.0)

Body Weight Gains - BG	Pharmacokinetics Concentrations - PC
Body Weights - BW	Palpable Masses - PM
Clinical Observations - CL	Pool Definition - POOLDEF
Comments - CO	Pharmacokinetics Parameters - PP
Death Diagnosis - DD	Related Records - RELREC
Demographics - DM	Subject Characteristics - SC
Disposition - DS	Subject Elements - SE
ECG Test Results - EG	Supplemental Qualifiers - SUPPQUAL
Exposure - EX	Trial Arms - TA
Food and Water Consumption - FW	Trial Elements - TE
Laboratory Test Results - LB	Tumor Findings - TF
Macroscopic Findings - MA	Trial Summary - TS
Microscopic Findings - MI	Trial Sets - TX
Organ Measurements - OM	Vital Signs - VS

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.

The SAS Clinical Standards Toolkit offers a tool to import controlled terminology from the ODM XML files that can be downloaded from the NCI CDISC Controlled Terminology FTP site (<http://evs.nci.nih.gov/ftp1/CDISC/SDTM/>).

For SDTM, these snapshots are supplied:

- The 201104 snapshot was taken from NCI EVS Controlled Terminology for SDTM, released April 2011, in support of the SAS Clinical Standards Toolkit 1.5. This snapshot supports the *Study Data Tabulation Model Implementation Guide*.

- The 201212 snapshot was taken from NCI EVS Controlled Terminology for SDTM, released December 2012, in support of the SAS Clinical Standards Toolkit 1.5. This snapshot supports the *Study Data Tabulation Model Implementation Guide*.

For SEND, these snapshots are supplied:

- The 201201 snapshot was taken from NCI EVS Controlled Terminology for SEND, released January 2012, in support of the SAS Clinical Standards Toolkit 1.5. This snapshot supports the *Standard for Exchange of Nonclinical Data Implementation Guide: Nonclinical Studies* Version 3.0 (SENDIG V3.0).
- The 201212 snapshot was taken from NCI EVS Controlled Terminology for SEND, released December 2012, in support of the SAS Clinical Standards Toolkit 1.5. This snapshot supports the *Standard for Exchange of Nonclinical Data Implementation Guide: Nonclinical Studies* Version 3.0 (SENDIG V3.0).

For ADaM, these snapshots are supplied:

- The 201101 snapshot was taken from NCI EVS Controlled Terminology for ADaM, released January 2011, in support of the SAS Clinical Standards Toolkit 1.5. This snapshot supports the *Analysis Data Model (ADaM) Implementation Guide* Version 1.0 (ADaMIG v1.0).
- The 201107 snapshot was taken from NCI EVS Controlled Terminology for ADaM, released July 2011, in support of the SAS Clinical Standards Toolkit 1.5. This snapshot supports the *Analysis Data Model (ADaM) Implementation Guide* Version 1.0 (ADaMIG v1.0).

For CDASH, these snapshots are supplied:

- The 201104 snapshot was taken from NCI EVS Controlled Terminology for CDASH, released April 2011, in support of the SAS Clinical Standards Toolkit 1.5. This snapshot supports CDISC CDASH.
- The 201212 snapshot was taken from NCI EVS Controlled Terminology for CDASH, released December 2012, in support of the SAS Clinical Standards Toolkit 1.5. This snapshot supports CDISC CDASH.

Note: Although SAS does not provide the SAS Clinical Standards Toolkit with the CDASH standard, the terminology is provided as a convenience. The terminology snapshots from 2011 were also available in the SAS Clinical Standards Toolkit 1.4.

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 *YYYYMM* is the specific snapshot (201104, 201212, and so on):

```
global standards library directory/standards/  
cdisc-terminology1.5/cdisc-xxxx/<current OR YYYYMM>/formats
```


5

SASReferences File

<i>Overview</i>	93
<i>Building a SASReferences File</i>	94
<i>How Is a SASReferences File Used?</i>	106
Overview	106
Communicating the Filename and Location to the SAS Clinical Standards Toolkit	106
Assessing Structural Integrity and Content	108
Translating Content for a SAS Session	113

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 33](#). 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 42](#) and example content is provided in [Display 3.5 on page 45](#).

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:

```
sample study library directory\cdisc-sdtm-3.1.2-1.5\  
sascstdemodata\control
```

An excerpt of this sample SASReferences file is provided in [Display 3.5 on page 45](#).

- 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-1.5/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-3.1.2-1.5/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 33](#) 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-1.5/
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.
cstmetadata	lookup, macrovariabledetails, macrovariables, sasreferences, standard, or standardsubtypes	Identifies the SAS data set templates that are used for Clinical Standards Toolkit Standards Library internal validation
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.

Type	Subtype	Comments
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.
fmtsearch		Provides a way to build the format search path for a validation process. The SAS Clinical Standards Toolkit sets the SAS fmtsearch type based on each record, specifying a SAS catalog that uses the order=n sequence. This type is not provided by default in StandardSASReferences, so you must specify a value. The type=fmtsearch value is optional unless one or more checks are to be run that assess value compliance against a SAS format.
globalmetadata	sasreferences or standard	Identifies the SAS data set templates that are used for the internal validation of the SAS Clinical Standards Toolkit global standards library.
lookup	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.

Type	Subtype	Comments
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 superset 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>

Type	Subtype	Comments
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.
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>

Type	Subtype	Comments
report	library or outputfile	Specifies the storage location of the SAS Clinical Standards Toolkit process reports. If a single, specific report is referenced, then it can be specified with a subtype of outputfile, a valid path, and valid memname values. If the process produces multiple reports, then a subtype of library is used with a valid path to the directory or folder. In the latter case, default report names as defined in the code are used.
results	analysis or 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. Note: Analysis has been added for the SAS Clinical Standards Toolkit 1.5, but it is not used.
resultspackage	xml or log	This type is not used in the SAS Clinical Standards Toolkit 1.5. 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.

Type	Subtype	Comments
sourcemetadata	analyses, column, document, value, table, or study	Identifies the SAS data sets (sasref.memname) that contain the column, document, analyses, value (for value level metadata), 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.
standardmetadata	attribute or element	Identifies the SAS data set templates for valid_attributes and valid_elements when validating ODM files.
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.
targetmetadata	analyses, document, value, column, table, or study	Identifies the SAS data sets (sasref.memname) that contain the analyses, document, value (for value level metadata), 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.

Type	Subtype	Comments
transport		This type is not used in the SAS Clinical Standards Toolkit 1.5. 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	reftype	path	memname
CST-FRAMEWORK	1.2	control	reference	csttmp	libref	&_cstGRoot./standards/cst-framework-1.5/templates	sasreferences.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	lookup	control	libref	&_cstGRoot./standards/cst-framework-1.5/control	standardlookup.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	lookup	cstmeta	libref	&_cstGRoot./standards/cst-framework-1.5/control	standardlookup.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	macrovariabledetails	control	libref	&_cstGRoot./standards/cst-framework-1.5/control	standardmacrovariabledetails.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	macrovariabledetails	cstmeta	libref	&_cstGRoot./standards/cst-framework-1.5/control	standardmacrovariabledetails.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	macrovariables	control	libref	&_cstGRoot./standards/cst-framework-1.5/control	standardmacrovariables.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	macrovariables	cstmeta	libref	&_cstGRoot./standards/cst-framework-1.5/control	standardmacrovariables.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	macrovariables	control	libref	&_cstGRoot./standards/cst-framework-1.5/control	standardsasreferences.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	macrovariables	cstmeta	libref	&_cstGRoot./standards/cst-framework-1.5/control	standardsasreferences.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	standard	control	libref	&_cstGRoot./standards/cst-framework-1.5/control	standards.sas7bdat
CST-FRAMEWORK	1.2	cstmetadata	standard	cstmeta	libref	&_cstGRoot./standards/cst-framework-1.5/control	standards.sas7bdat
CST-FRAMEWORK	1.2	lookup		lookup	libref	&_cstGRoot./metadata	standardlookup.sas7bdat
CST-FRAMEWORK	1.2	messages		cstmsg	libref	&_cstGRoot./standards/cst-framework-1.5/messages	messages.sas7bdat
CST-FRAMEWORK	1.2	properties	initialize	cstprop	fileref	&_cstGRoot./standards/cst-framework-1.5/programs	initialize.properties
CST-FRAMEWORK	1.2	properties	validation	valprop	fileref	&_cstGRoot./standards/cst-framework-1.5/programs	validation.properties
CST-FRAMEWORK	1.2	referencecontrol	validation	cstcntrl	libref	&_cstGRoot./standards/cst-framework-1.5/validation/control	validation_master.sas7bdat
CST-FRAMEWORK	1.2	template		csttmplt	libref	&_cstGRoot./standards/cst-framework-1.5/templates	

The standard SASReferences data set has been expanded in the SAS Clinical Standards Toolkit 1.5. The **SASref** field, with values of **cstmeta** and **control**, points to the same path field value. The SAS Clinical Standards Toolkit 1.5 was enhanced with new features, and the **control** SASref was retained to ensure backward compatibility with past releases.

In addition, there are new memnames, such as **standardmacrovariabledetails.sas7bdat** and **standardmacrovariables.sas7bdat**. These are new data sets that contain metadata about the SAS Clinical Standards Toolkit 1.5 global macro variables. They are used for internal validation.

Display 5.2 on page 103 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	reftype	path	order	memname
1	CDISC-SDTM	3.1.3	autocall		autocall	fileref	%_cstGRoot/standards/cdisc-sdtm-3.1.3	1	
2	CDISC-SDTM	3.1.3	classmetadata	column	refmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		class_columns.sas7bdat
3	CDISC-SDTM	3.1.3	classmetadata	table	refmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		class_tables.sas7bdat
4	CDISC-SDTM	3.1.3	cstmetadata	lookup	stdmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		standardlookup.sas7bdat
5	CDISC-SDTM	3.1.3	cstmetadata	macrovariabledetails	stdmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		standardmacrovariabledetails.sas7bdat
6	CDISC-SDTM	3.1.3	cstmetadata	macrovariables	stdmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		standardmacrovariables.sas7bdat
7	CDISC-SDTM	3.1.3	cstmetadata	sasreferences	stdmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		standardsasreferences.sas7bdat
8	CDISC-SDTM	3.1.3	cstmetadata	standard	stdmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		standards.sas7bdat
9	CDISC-SDTM	3.1.3	lookup		lookup	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		standardlookup.sas7bdat
10	CDISC-SDTM	3.1.3	messages		messages	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3	1	messages.sas7bdat
11	CDISC-SDTM	3.1.3	properties	initialize	initprop	fileref	%_cstGRoot/standards/cdisc-sdtm-3.1.3	1	initialize.properties
12	CDISC-SDTM	3.1.3	properties	validation	valprop	fileref	%_cstGRoot/standards/cdisc-sdtm-3.1.3	2	validation.properties
13	CDISC-SDTM	3.1.3	referencecontrol	checktable	refcntl	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		validation_domainsbycheck.sas7bdat
14	CDISC-SDTM	3.1.3	referencecontrol	standardref	refcntl	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		validation_stdref.sas7bdat
15	CDISC-SDTM	3.1.3	referencecontrol	validation	refcntl	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		validation_master.sas7bdat
16	CDISC-SDTM	3.1.3	referencemetadata	column	refmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		reference_columns.sas7bdat
17	CDISC-SDTM	3.1.3	referencemetadata	table	refmeta	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		reference_tables.sas7bdat
18	CDISC-SDTM	3.1.3	template		tmplit	libref	%_cstGRoot/standards/cdisc-sdtm-3.1.3		

A comparison of Display 5.1 on page 102 and Display 5.2 on page 103 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:

```
sample study library directory\cdisc-sdtm-3.1.2-1.5\
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	memname
1	CDISC-SDTM	3.1.2	autocall		autocall	libref		
2	CDISC-SDTM	3.1.2	control	reference	srcctrl	libref	&studyRootPath/control	sasreferences.sas7bdat
3	CDISC-SDTM	3.1.2	control	validation	srcctrl	libref	&studyRootPath/control	validation_control.sas7bdat
4	CDISC-SDTM	3.1.2	fmtsearch		fmts	libref	&studyRootPath/terminology/formats	formats.sas7bcat
5	CDISC-SDTM	3.1.2	lookup		lookup	libref		
6	CDISC-SDTM	3.1.2	messages		messages	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.5/messages	messages.sas7bdat
7	CDISC-SDTM	3.1.2	properties	initialize	initprop	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.2-1.5/programs	initialize.properties
8	CDISC-SDTM	3.1.2	properties	validation	valprop	libref	&studyRootPath/programs	validation.properties
9	CDISC-SDTM	3.1.2	referencecontrol	checktable	refctrl	libref		
10	CDISC-SDTM	3.1.2	referencecontrol	standardref	refctrl	libref		
11	CDISC-SDTM	3.1.2	referencecontrol	validation	refctrl	libref		
12	CDISC-SDTM	3.1.2	referenceceterm		ctref	libref	&studyRootPath/terminology/coding-dictionaries	meddra.sas7bdat
13	CDISC-SDTM	3.1.2	referencecemetadata	column	refmeta	libref		
14	CDISC-SDTM	3.1.2	referencecemetadata	table	refmeta	libref		
15	CDISC-SDTM	3.1.2	results	validationmetrics	results	libref	&studyOutputPath/results	validation_metrics.sas7bdat
16	CDISC-SDTM	3.1.2	results	validationresults	results	libref	&studyOutputPath/results	validation_results.sas7bdat
17	CDISC-SDTM	3.1.2	sourcedata		srcdata	libref	&studyRootPath/data	
18	CDISC-SDTM	3.1.2	sourcecemetadata	column	srcmeta	libref	&studyRootPath/metadata	source_columns.sas7bdat
19	CDISC-SDTM	3.1.2	sourcecemetadata	table	srcmeta	libref	&studyRootPath/metadata	source_tables.sas7bdat
20	CDISC-SDTM	3.1.2	template		tmpplt	libref		
21	CDISC-TERMINOLOGY	NCL_THESAURUS	fmtsearch		ctfmt	libref	&_cstGRoot/standards/cdisc-terminology-1.5/cdisc-sdtm/201104/formats	ctrms.sas7bcat
22	CST-FRAMEWORK	1.2	messages		cstmsg	libref	&_cstGRoot/standards/cst-framework-1.5/messages	messages.sas7bdat
23	CST-FRAMEWORK	1.2	template		csttmpplt	libref		

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, 21	Two standards are referenced to create a format search path. Line 4 references the SDTM study-specific formats catalog. Line 21 references the more general CDISC Terminology cterms catalog. The precedence is set by the order column.
6, 22	These records are identical to the CST-FRAMEWORK and CDISC-SDTM StandardSASReferences records.
7	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.

Lines	Comment
8	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.
9–11 13–14, 20, 23	Points to the reference standard for CDISC SDTM 3.1.2, but unlike the template defaults in Display 5.2 on page 103 , 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 103 .
12	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.
15–16	Specifies that process results are to be stored in a location in the study hierarchy.
17	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.
18–19	These source metadata references are new. These values follow the style used in line 17 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_createdsfromtemplate utility macro.

```
%cst_createdsfromtemplate(_cstStandard=CST-FRAMEWORK,_cstType=control,
_cstSubType=reference,_cstOutputDS=work.sasreferences);
proc sql;
insert into work.sasreferences
values(CST-FRAMEWORK 1.2 messages messages libref 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 programs. 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>	<p>Specifies the initial source that setup should be based on.</p> <p>Valid values are <code>SASReferences</code> (default) or <code>Results</code>.</p> <p>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>.</p>
<code>_cstSASReferencesLocation</code>	<p>Specifies the path (folder location) of the <code>SASReferences</code> data set. The default is the path to the Work library. This is the value of the global macro variable.</p>
<code>_cstSASReferencesName</code>	<p>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.</p>

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.

```
%cstutil_setcstroot;
data _null_;
call symput('studyRootPath',cats("&_cstSRoot",
                                "/cdisc-sdtm-3.1.2-&_cstVersion/sascstdemodata"));
run;
%* Look for the data set named sasreferences in the specified folder ;
%cstutil_processsetup(_cstSASReferencesLocation=&studyrootpath/control);
```

The call to the `%cstutil_setcstroot` macro sets the SAS Clinical Standards Toolkit global macro variable `&_cstSRoot` to the sample library.

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 104](#). This macro attempts to update only records for a supported standard (and standardversion) that has 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.

In the SAS Clinical Standards Toolkit 1.5, the `cstutilvalidatesasreferences` macro replaces the `cstutil_checkds` macro. This new macro checks the structure and content of the SASReferences data set against a defined gold standard.

If you have used previous versions of the SAS Clinical Standards Toolkit, you might see failures when you use the `cstutilvalidatesasreferences` macro against SASReferences data sets that were created in a version before the SAS Clinical Standards Toolkit 1.5. These failures are caused by the stricter adherence to the SASReferences metadata model that the `cstutilvalidatesasreferences` macro enforces.

Here is the syntax of this macro:

```
%macro cstutilvalidatesasreferences
(_cstDSName=,_cstStandard=,_cstStandardversion=, _cstSASRefsGoldStd=,
_cstallowoverride=, _cstResultsType=, _cstPreAllocated, _cstVerbose= );
```

`_cstDSName` specifies the two-level name of the data set to be validated. This value is required. The default value is `&_cstSASRefs` derived from the process setup macro.

`_cstStandard` specifies the name of a registered data standard. This value is required. The default value is `CST-FRAMEWORK`.

`_cstStandardversion` specifies the version of a registered data standard. This value is required. The default value is `1.2`.

`_cstSASRefsGoldStd` specifies the two-level name of a comparative gold standard against which this SASReferences data set is compared. This value is required. By default, the global standards library metadata `StandardSASReferences` is assumed.

`_cstallowoverride` specifies whether to ignore one or more of the values defined above. Specify the check code in a blank-delimited string (for example, `CHK01 CHK07`). If null, all conditions are tested.

`_cstResultsType` specifies where to store report findings: in the SAS log or in the Results data set. This value is required. It must be either `LOG` or `RESULTS`. The default value is `LOG`.

`_cstPreAllocated` specifies whether to allocate librefs and filerefs when this macro is called. If they are not allocated, the validation of data sets and catalogs is performed based on paths and memnames, not on libref.memnames. This value is required. It must be either N or Y. The default value is N.

`_cstVerbose` specifies whether to report specific problems or the absence of problems in `_cst_rc`. Otherwise, only success or failure is reported. This value is required. It must be either N or Y. The default value is N.

This macro is typically used as a part of the normal process setup. It is called either before or as a part of `cstutil_allocatesasreferences()` or as a stand-alone call outside the context of use in the normal process setup. The macro sets the `_cst_rc` and `_cst_rcmsg` global macro variables to indicate that the SASReferences data set is valid (`_cst_rc=0`) or not valid (`_cst_rc ne 0`).

There are eight checks associated with this macro when validating a SASReferences data set.

- CHK01: The data set is structurally correct.
- CHK02: An unknown standard or standardversion exists.
- CHK03: The referenced input and output files and folders can be accessed.
- CHK04: All required look-throughs to the global standards library defaults work.
- CHK05: All discrete character field values are found in the Standardlookup data set.
- CHK06: For the given context, path and memname macro variables are resolved.
- CHK07: Multiple fmtsearch records exist, but valid ordering is not provided.
- CHK08: Multiple autocall records exist, but valid ordering is not provided.

In the SAS Clinical Standards Toolkit 1.5, additional columns are included in the SASReferences data set to facilitate internal validation. Two of these columns are `iotype` and `filetype`. To remain backward compatible, if the SASReferences data set is missing these two columns, CHK03 is ignored because the `cstutil_validatesasreferences` macro assumes that the SASReferences data set was created in a version before the SAS Clinical Standards Toolkit 1.5.

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_validatesasreferences` macro. Solutions are suggested. All errors appear in the Results data set.

- CHK01 - A problem with the structure of the data set exists.

The macro has detected a structural difference in the data set that needs to be addressed.

Fix the issues as described in the Results data set.

- CHK02 - An unknown standard or standardversion value exists.

The macro has detected a standard or standardversion value that does not exist in the SAS Clinical Standards Toolkit. This can be caused by a typographical error for the value or by a standard that has not yet been registered with the SAS Clinical Standards Toolkit.

Correct the erroneous value or register the unknown standard.

- CHK03 - The referenced input and output files cannot be accessed.

This check uses a new metadata variable in SASReferences called `iotype`. This variable is not available in previous versions of the SAS Clinical Standards Toolkit. To maintain backward compatibility, a special Boolean macro variable exists. It is named `&_cstCurrentStyle` and has a value of 1 (version 1.5 SASReferences) or 0 (previous version of SASReferences). When set to 0, the SAS Clinical Standards Toolkit ignores this check.

Based on the value of `iotype`, the macro has detected a specified input file, data set, or catalog that does not exist in the path provided by SASReferences. For `iotype` equal to 'output' or 'both,' the specified path is Read-Only and does not allow the SAS Clinical Standards Toolkit to create an output file.

Correct this issue by ensuring that pathnames, filenames, data set names, and catalog names are entered correctly. For output file references, ensure that the user

account has Write access permission to the folders that are specified in SASReferences.

- CHK04 - Required look-throughs to the global standards library defaults do not work.

For this check to be meaningful, ensure that a call to `cst_insertStandardSASRefs` has been performed before running this check. Otherwise, empty pathnames might exist that are populated with a call to `cst_insertStandardSASRefs`.

This check is not applicable to stand-alone use. This check detects pathnames that are missing or null.

Correct this issue by verifying that the call to `cst_insertStandardSASRefs` was made before running this check. Otherwise, provide a valid pathname for each missing value.

- CHK05 - Not all discrete character fields were found in the Standardlookup data set.

This check detects missing or incorrect names for the following columns in SASReferences: `reftype`, `type+subtype` combinations, `iotype`, `filetype`, and `allowoverwrite`.

Note: Because `iotype`, `filetype`, and `allowoverwrite` are new to the SAS Clinical Standards Toolkit 1.5, these columns are ignored when `&_cstCurrentStyle=0`. (See check CHK03.)

Correct this issue by providing valid values for these columns in SASReferences. If needed, update the Standardlookup data set.

Note: Updating the Standardlookup data set is an advanced use of the SAS Clinical Standards Toolkit and should be performed by an administrator.

- CHK06 - For the given context, all macro variables have not been resolved.

This check detects unresolved macro variables used in the `memname` and `path` columns.

Correct this issue by making sure all macro references used in SASReferences have been resolved.

- CHK07

To ensure proper FMTSEARCH functionality in SAS, the order in which the `fmtsearch` string is built is very important for the proper functioning of the SAS Clinical Standards Toolkit. This check detects multiple `fmtsearch` records with invalid order values. Invalid order values could be missing or duplicate values.

Correct this issue by assigning valid order values for multiple `fmtsearch` records.

■ CHK08

To ensure proper AUTOCALL macro functionality in SAS, the order in which the `autocall` macro string is built is very important for the proper functioning of the SAS Clinical Standards Toolkit. This check detects multiple `autocall` records with invalid order values. Invalid order values could be missing or duplicate values.

Correct this issue by assigning valid order values for multiple `autocall` records.

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_validatesasreferences` 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 111](#).

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.

6

Compliance Assessment Against a Reference Standard

<i>Validation Framework Overview</i>	117
<i>Metadata Requirements</i>	120
Overview	120
Reference Metadata	121
Source Metadata	126
Validation Check Metadata: Validation Master	127
Supplemental Validation Check Metadata:	
Validation Standard References	138
Supplemental Validation Check Metadata:	
CDISC SDTM Domains by Check	141
Supplemental Validation Check Metadata:	
CDISC ADaM Class by Check	143
Validation.Properties	143
Messages	145
Validation Metrics	146
<i>Cross-Standard Validation</i>	150
Overview	150
The cstcheck_crossstdcomparedomains Macro	150
The cstcheck_crossstdmetamismatch Macro	151
<i>Building a Validation Process</i>	153
Overview	153
SASReferences Customizations	154

Validation Control: Specification of Run-Time Checks	156
Setting Properties for the Validation Process	159
Running a Validation Process	160
Sample CDISC SDTM 3.1.3 Driver Program:	
validate_data.sas	160
Validation Results and Metrics	168
Validation Checks by Standard	172
Overview	172
ADaM 2.1	173
CDISC CRT-DDS 1.0	174
CDISC ODM 1.3.0 and 1.3.1	177
CDISC SDTM 3.1.1, 3.1.2, and 3.1.3	183
CDISC CT 1.0.0	185
The SAS Clinical Standards Toolkit Framework	185
Special Topic: Validation Check Macros	185
Special Topic: How the SAS Clinical Standards	
Toolkit Interprets Validation Check Metadata	191
Overview	191
Case Study 1: CDISC SDTM Check SDTM0604	192
Case Study 2: CDISC SDTM 3.1.1 Check SDTM0623	194
Special Topic: SAS Implementation of ISO 8601	196
Overview	196
Example ISO 8601 Values	197
SAS ISO 8601 References	203
Special Topic: Debugging a Validation Process	204
Overview	204
Errors in Setting Up the SAS Clinical Standards	
Toolkit Environment	205
Errors in Performing Some Primary SAS Clinical	
Standards Toolkit Action	208
Other Debugging Tips	211
Special Topic: Validation Customization	212

Overview	212
Case Study 1: Modifying an Existing Standard or Defining a New Reference Standard	213
Case Study 2: Using Any Set of Source Data and Metadata ...	214
Case Study 3: Modifying the SAS Validation Checks for Supported Standards	214
Case Study 4: Adding New Validation Checks for Supported Standards	215
Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros	217
Case Study 6: Modifying the SAS Clinical Standards Toolkit Messaging, Including Internationalization ...	218
Case Study 7: Validation of Multiple Studies	220
<i>Special Topic: Using Alternative Controlled Terminologies ...</i>	222
<i>Special Topic: Performance Considerations</i>	227

Validation Framework Overview

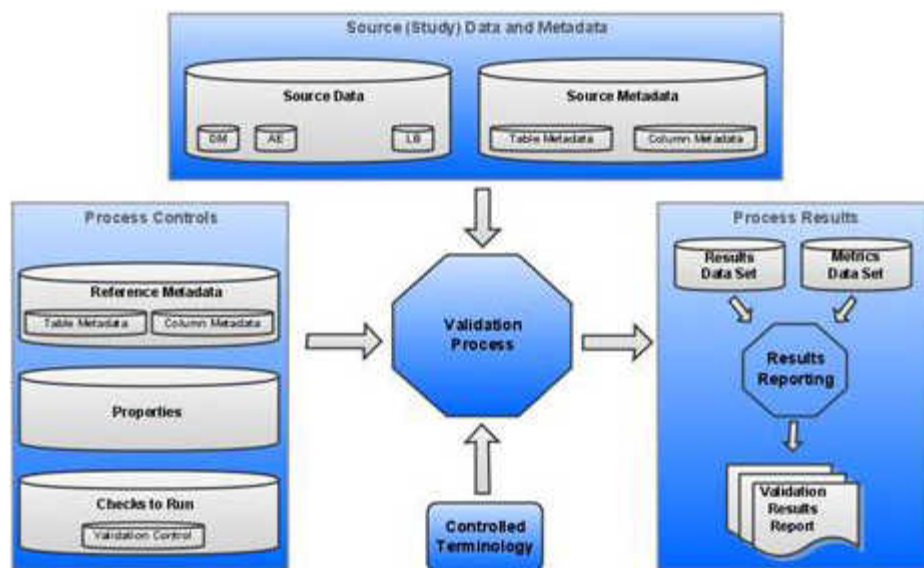
The SAS Clinical Standards Toolkit validation assesses the compliance of data, and the metadata describing the data, with an accepted reference standard. It assesses the consistency of values in a specific column, between columns, across records in a specific data set, and across data sets. The primary 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 64](#) and [Table 4.2 on page 64](#).
 - *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 127](#). For information about the SAS Clinical Standards Toolkit global macro variables, see [Appendix 1, “Global Macro Variables,” on page 393](#).
 - *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 57](#), a standard consists of properties, messages, and metadata files that collectively represent the standard in the SAS Clinical Standards Toolkit. Each SAS Clinical Standards Toolkit registered standard can support validation if the `standards.supportsvalidation` flag is set to Y. This setting indicates that the required set of validation files defining the standard exist. By default, the set of validation files that supports the standards that are supplied by SAS is in the `cstGlobalLibrary` folder hierarchy.

For example, validation files that define the CDISC SDTM 3.1.3 standard are in this folder hierarchy:

`global standards library directory/standards/cdisc-sdtm-3.1.3-1.5`

The following sections describe each metadata type used by typical validation processes. For information about metadata files that are common to all SAS Clinical

Standards Toolkit processes, see [Chapter 3, “Metadata File Descriptions,” on page 33](#). Metadata characteristics specific to compliance assessments are described in the sections in this chapter.

Reference Metadata

For CDISC standards, reference metadata about data sets 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 6.1 on page 121](#) and an example of a `reference_columns` record is provided in [Table 6.2 on page 123](#).

Note: The structure and content of the reference metadata data sets can vary for other standards.

As noted in [Chapter 4, “Supported Standards,” on page 57](#), 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=referencemetadatas</code> and <code>subtype=table</code> . This column is required.
<code>table</code>	\$32	The name of the tabulation domain or analysis data set 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.

Column Name	Column Length	Description
class	\$40	The observation class in the standard. Example CDISC SDTM values are Events, Findings, Interventions, Relates, Special Purpose, and Trial Design. This column is optional and not relevant for all standards.
xmlpath	\$200	The path to the SAS transport file. This path can be specified as a relative path. The value can be used when creating define.xml to populate the value for the def:leaf xlink:href link to the domain file. The value should be the pathname and filename of the SAS transport file relative to the location of define.xml file. This column is optional and not relevant for all standards.
xmltitle	\$200	The title of the SAS transport file. The value can be used when creating a define.xml file to populate the value for the def:leaf def:title value. It can provide a meaningful description, label, or location of the domain leaf (for example, crt/datasets/Protocol 1234/AE.xpt). This column is optional and not relevant for all standards.
structure	\$200	The description of the general structure of the table. An example value is one record per event per subject. This column is optional and not relevant for all standards.
purpose	\$20	The description of the general purpose of the table. Examples are Tabulation (required for CDISC SDTM) and Analysis (required for CDISC ADaM). This column is optional and not relevant for all standards.
keys	\$200	A space-delimited string of keys that captures the table columns that uniquely define records in the table. This set of keys can also define the sort order of records in the table. Example is STUDYID USUBJID. This column is required.
state	\$20	A description of the table state, such as Draft or Final. This column is optional.
date	\$20	A meaningful, distinguishing date that describes the table, such as the release date, the creation date, or the modified date. This column is optional.

Column Name	Column Length	Description
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 8 . 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.

Note: The column length can vary to match submission requirements or corporate conventions.

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 tabulation domain or analysis data set being defined in the standard. The value must conform to SAS naming conventions. This column is required.

Column Name	Column Length	Description
column	\$32	The name of the column in the table. The value must conform to SAS naming conventions. This column is required.
label	\$200	The label of the column. The value must conform to SAS naming conventions. This column is optional.
order	8.	The order of the columns in each table. Values must be integers >0 and unique in each table. This column is required.
type	\$1	The SAS type, N for numeric, C for character. This column is required.
length	8.	The length of the column. Numeric columns have a length of 8. This column is required.
displayformat	\$32	The display format for numeric variables. For example, 8.2 indicates that floating-point variable values should be displayed to the second decimal place. This value is optional and not relevant for all standards.
xmldatatype	\$8	The data type of the column as it is defined in the define.xml file. Values are integer float date datetime time text. This column is optional and not relevant for all standards.
xmlcodelist	\$32	A SAS format name that is used to assess conformance to controlled terminology. This value does not have a \$ prefix for character formats and does not have the trailing period. This value is also the codelist name in the define.xml file. The SAS format name must be in the format search path for successful column-value validation. This record is optional and not relevant for all standards.
core	\$10	The value indicates whether the column is required. Sample CDISC SDTM values are Req (required), Exp (expected), Perm (permissible), and Dep (deprecated). This column is optional and not relevant for all standards.

Column Name	Column Length	Description
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 8 . 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.

Column Name	Column Length	Description
standardref	\$200	Any reference to an associated standard definition, implementation guide, schema, and so on, that provides additional information about the column or describes the column in greater detail. This column is optional.
comment	\$1000	Any character string that provides comments relevant to the column. This column is optional.

Note: The column length can vary to match submission requirements or corporate conventions.

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 3.1.3 standard, the location is:

```
global standards library directory/standards/  
cdisc-sdtm-3.1.3-1.5/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 8 . 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.

Column Name	Column Length	Description
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.
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>

Column Name	Column Length	Description
codesource	\$32	The name of the check macro. The name must conform to SAS naming conventions. The value must be in the SAS autocall path. An example is cstcheck_notunique. This column is required.
usesourcemetadata	\$1	The value indicates whether to use source metadata rather than reference metadata. The metadata controls the derivation of domains and column lists to be validated, program flow, and looping. Values are Y and N (default). This column is optional.
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_-DM-DS</u>: 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_-DM</u>][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>_ALL_: All columns (equivalent to ** or a null value).</p> <p>_NA_: Not applicable (that is, domain-level check).</p> <p>AGE: Any single column. This value can be specified as libref.domain.column or domain.column.</p> <p>ARM+ARMCD: Multiple columns delimited with a +.</p> <p>**BLFL-LBBLFL: Multiple columns that exclude specific columns delimited with a -.</p> <p>**DTC: Wildcard to include multiple columns with ** representing the domain name.</p> <p>xxx**: (For example, AE**, where ** is a column wildcard).</p> <p>[**STDTC][**ENDTC]: Bracket syntax to define sublists for comparative purposes. In this example, all start dates are compared with all end dates. The number of columns in each sublist must be equivalent.</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 _ALL_.)</p>

Column Name	Column Length	Description
codeologic	\$2000	<p>Check-specific code segment that is inserted into the check macro defined in codesource and consistent with codetype. The codeologic value enables check-level customization and allows the reuse of more general check macros. The field length of \$2000 limits the code to short code segments, although referencing another macro or using %include expands this capability. The codeologic 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 */</pre> <p>The previous code is limited to filerefs set outside of the SAS Clinical Standards Toolkit or in the SASReferences control data set.</p> <pre>%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 codelogic and what type of codelogic can be used in the validation code. Values include:</p> <p>0: No codelogic used.</p> <p>1: DATA step statement level. (For example, if &_cstColumn <0 then _cstError=1.)</p> <p>2: Full DATA step, PROC SQL step, or multiple steps.</p> <p>3: Calls a SAS macro or %include that can contain only DATA step statement level code. (For example, codetype=1.)</p> <p>4: Calls a SAS macro or %include that can contain only full DATA step or PROC SQL step code. (For example, codetype=2.)</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 xmlcodelist 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, medDRA). 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>

Column Name	Column Length	Description
lookupsource	\$32	<p>The specific SAS format or file associated with lookuptype. For example:</p> <p>If lookuptype is metadata, then lookupsource should be blank. The code gets the value from the source_columns.xmlcodelist field.</p> <p>If lookuptype is format, then lookupsource should be the SAS format and must be in the format search path if it is specified. This value should generally match any value in source_columns.xmlcodelist for the columns specified in columnscope. This field allows a run-time validation check against another format.</p> <p>If lookuptype is dataset, then lookupsource should be the name of a SAS data set. This value is specified as the data set name (for example, meddra) or libref.dataset. If a value is provided without a libref, then the SAS Clinical Standards Toolkit looks for any SASReferences type=referenceceterm records for the sasref 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 columnscope for code-processing purposes and to help resolve errors. If this value is specified, then it should be a space-delimited list of columns in the domains specified in the tablescope field. The values of these columns can be reported in the Results data set. This column is optional.</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) >0: (active) -1: (deprecated, archived) -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) N: (no)</p> <p>This column is required although not all check macro modules support abbreviated (N) reporting.</p>

Column Name	Column Length	Description
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 SDTM000100CST120SDTM3112009-05-12T12:00:00C DI.</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.

Column Name	Column Value	Comment
standard	CDISC-SDTM	The registered standard.
standardversion	***	The standard version. A value of *** indicates that the check is applicable to all versions of the standard.
checksource	WebSDM	This check originated as a WebSDM check.
sourceid	IR5010	WebSDM check IR5010.
checkseverity	Warning	
checktype	ColumnValue	
codesource	cstcheck_column	This check uses the cstcheck_column check macro in the SAS Clinical Standards Toolkit autocall library.
usesourcemetadata	Y	This check is run on source data domains.
tablescope	_ALL_	This check is run on all domains.
columnscope	VISITNUM	This check evaluates VISITNUM values from each domain.
codelogic	<pre>_vnum=kstrip(put(&_ cstColumn,best.));_ dot=kindexc(_vnum,".");if _dot then if length(ksubstr(_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		

Column Name	Column Value	Comment
reportingcolumns		
checkstatus	1	
reportall	Y	This check reports all errors that are identified.
uniqueid	SDTM020701CST150SDT M3122012-06-08T10:49:21 CST	
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 191](#).

**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 128.)
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.

Column Name	Column Value	Comment
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: CDISC SDTM 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 191](#). The `Validation_DomainsByCheck` data set is supplied in `global standards library directory/standards/cdisc-sdtm-3.1.x/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 10, “Reporting,” on page 373](#). It is also

available to other programs and applications that might need to subset checks that are applicable to specific domains.

The SDTM 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 128.)
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.

Supplemental Validation Check Metadata: CDISC ADaM Class by Check

For CDISC ADaM, the supplemental data set is called `Validation_ClassByCheck`. It is located at: `global standards library directory/standards/cdisc-adam-2.1-1.5/validation/control`.

This data set is patterned after the data set that is described in [Table 6.8 on page 142](#). However, the column class (\$40, Observation Class within Standard) has been added. This addition accommodates the different way that the ADaM reference standard is defined. For example, the `reference_tables` data set, located in `/standards/cdisc-adam-2.1-1.5/metadata`, includes a BDS record that serves as a class template for all specific implementations of BDS that are required for a study. The SAS Clinical Standards Toolkit does not know each of the specific analysis data sets, so the `Validation_ClassByCheck` data set includes records by class, not by domain, for each check in the ADaM Validation Master data set.

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 _DATA_ is used, then the data set order is assumed. Or, _cstCheckSortOrder can be set to sort the Validation Control data set at run time by any fields in that data set (for example, CHECKSOURCE CHECKID).
_cstMetrics	This property determines whether to calculate and report metrics. An example value is 1=Yes.
_cstMetricsDS	This property sets the SAS data set name to use to accumulate metrics during the process. The default value is work._cstmtrics.
_cstMetricsNumSubj _cstMetricsCntNumSubj	This property determines whether to calculate and report subject-level counts. An example value is 1=Yes, initialize _cstMetricsCntNumSubj to 0. The calculation of subject-level counts might not be appropriate for all check macros.
_cstMetricsNumRecs _cstMetricsCntNumRecs	This property determines whether to calculate and report record-level counts. An example value is 1=Yes, initialize cstMetricsCntNumRecs to 0.
_cstMetricsNumChecks _cstMetricsCntNumChecks	This property determines whether to summarize and report the number of checks run. An example value is 1=Yes, initialize cstMetricsCntNumChecks to 0.
_cstMetricsNumBadChecks _cstMetricsCntNumBadChecks	This property determines whether to summarize and report the number of check invocations that failed. An example is 1=Yes, initialize cstMetricsCntNumBadChecks to 0.
_cstMetricsNumErrors _cstMetricsCntNumErrors	This property determines whether to summarize and report the total number of errors (resultseverity=Error) found. An example is 1=Yes, initialize cstMetricsCntNumErrors to 0.

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

By default, for all standards that support validation, Validation.Properties 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 located at: *sample study library directory/cdisc-sdtm-3.1.3-1.5*.

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 33.

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`.

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 144](#).

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 147](#).

Cross-Standard Validation

Overview

The implementation of the ADaM 2.1 standard in the SAS Clinical Standards Toolkit 1.5 requires the use of a number of cross-standard validation checks. These cross-standard validation checks compare data and metadata between two different standards, such as ADaM 2.1 and SDTM 3.1.2.

The SAS Clinical Standards Toolkit 1.5 provides two macros that enable cross-standard comparisons: `cstcheck_crossstdcomparedomains.sas` and `cstcheck_crossstdmetamismatch.sas`. These macros are located at: `!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.

```
%let _cstCRDomName=DM;
proc sql noprint;
  create table work._cstproblems as
  select &_cstSQLColList
  from &_cstDSName
  except select &_cstSQLColList from &_cstCrossDataLib..&_cstCRDomName;
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 &_cstStMne..sasref, &_cstStMne..table, &_cstStMne..column,
         &_cstStMne..&_cstAttr as &_cstStMne._value,
         &_cstCrMne..&_cstAttr as &_cstCrMne._value
  from work._cstcolumnmetadata &_cstStMne
       left join
       work._cstcrosscolumnmetadata &_cstCrMne
       on upcase(&_cstStMne..column)=upcase(&_cstCrMne..column)
       where &_cstCrMne..column ne "" and
          (&_cstStMne..&_cstAttr ne &_cstCrMne..&_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.

This table shows the standard-specific validation macros for all SAS Clinical Standards Toolkit standards that support validation.

Table 6.11 *Standard-Specific Validation Macros for Standards Supporting Validation*

Standard and Version	Validation Macro
CDISC-ADAM 2.1	%adam_validate
CDISC-CRTDDS 1.0	%crtdds_validate
CDISC-CT 1.0.0	%ct_validate
CDISC-ODM 1.3.0	%odm_validate
CDISC-ODM 1.3.1	%odm_validate
CDISC-SDTM 3.1.1	%sdtm_validate
CDISC-SDTM 3.1.2	%sdtm_validate
CDISC-SDTM 3.1.3	%sdtm_validate
CST-FRAMEWORK 1.2	%cstvalidate

The remainder of this section uses SDTM 3.1.3 as an example.

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	3.1.3	referencecontrol	validation	refcntrl	libref		1	
CDISC-SDTM	3.1.3	referencemetadata	column	refmeta	libref		2	
CDISC-SDTM	3.1.3	referencemetadata	table	refmeta	libref		3	

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	3.1.3	properties	validation	valprop	fileref	&studyRootPath/programs	2	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 143](#). For information about the derived global macro variables, see [Appendix 1, “Global Macro Variables,” on page 393](#). 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	3.1.3	results	validation	results	libref	&studyOutputPath/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 146](#) and [“Validation Results and Metrics” on page 168](#). For information about the global macro variables that govern metrics output, see [Appendix 1, “Global Macro Variables,” on page 393](#). 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	reftype	path	order	memname
CDISC-SDTM	3.1.3	fmtsearch		fmts	libref	&studyRootPath/terminology/formats	1	formats.sas7bcat
CDISC-SDTM	3.1.3	referenceceterm		ctref	libref	&studyRootPath/terminology/coding-dictionaries	1	meddra.sas7bdat
CDISC-TERMINOLOGY	NCI_THESAURUS	fmtsearch		ctfmt	libref	&_cstGRoot/standards/cdisc-terminology-1.5/cdisc-sdtm/201104/formats	2	ctterms.sas7bcat

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=referenceceterm 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	reftype	path	order	memname
CDISC-SDTM	3.1.3	control	validation	srocntrl	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 127.) 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 3.1.3 Validation Control data set is deployed to this directory:

```
sample study library directory/cdisc-sdtm-3.1.3-1.5/  
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 155](#).)

The &studyRootPath value is assumed to have been set to *sample study library directory/cdisc-sdtm-3.1.3/sascstdemodata*.

The Validation Master data set (illustrated in [Display 6.3 on page 154](#) 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

standard	standardversion	type	subtype	SASref	reftype	path	order	memname
CDISC-SDTM	3.1.3	referencecontrol	validation	refcntl	libref	\$_cstGRoot/standards/cdisc-sdtm-3.1.3-1.5/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 6.12 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>

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>
Checks new to CDISC SDTM 3.1.3.	<pre>data control.validation_control; set refcntl.validation_master (where=(standardVersion = "3.1.3")); 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>

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 more information about the full set of validation properties, see [Appendix 1, “Global Macro Variables,” on page 393](#).) 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 re-sort 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 3.1.3 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, the CDISC SDTM 3.1.3 validation driver module is in: *sample study library directory/cdisc-sdtm-3.1.3-1.5*.

Step 1: Define macro variables required by the validation process.

```
%let _cstStandard=CDISC-SDTM;
%let _cstStandardVersion=3.1.3;
%let _cstVersion=;
%let _cstCTPath=;
%let _cstCTMemname=;
%let _cstCTDescription=;
```

These macro variables are used as substitution parameters later in the driver module to reduce the number of code changes required.

```
%cst_setStandardProperties(_cstStandard=CST-FRAMEWORK,_cstSubType=initialize);
```

Initialize the minimum set of global macro variables used to run any SAS Clinical Standards Toolkit process. This includes the names of work data sets, default locations of files, and metadata used to populate the process Results data set.

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 the properties are processed when the %cstutil_allocatesasreferences macro is called. This latter approach is followed in the SDTM validate_data.sas driver module.

```
%cst_getRegisteredStandards(_cstOutputDS=work._cstStandards);
data _null_;
  set work._cstStandards (where=(standard="CST-FRAMEWORK"));
  call symputx('_cstVersion',strip(productrevision));
run;
```

Get the list of registered standards to determine the version of the SAS Clinical Standards Toolkit.

```
* 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'));
  * User can override CT version of interest by specifying a different where clause:          *;
  * Example: (where=(standardversion="&_cstStandard" and standardsubtypeversion='201104')) *;
  call symputx('_cstCTPath',path);
  call symputx('_cstCTMemname',memname);
  call symputx('_cstCTDescription',description);
run;

proc datasets lib=work nolist;
  delete _cstStandards _cstStdSubTypes;
quit;
```

Choose the default controlled terminology that is associated with the _cstStandard and _cstStandardVersion. Cleanup work files.

```
*****;
* The following data step sets (at a minimum) the studyrootpath and studyoutputpath. These *;
* are used to make the driver programs portable across platforms and allow the code to be *;
* run with minimal modification. These macro variables by default point to locations within *;
```

```

* the cstSampleLibrary, set during install but modifiable thereafter. The cstSampleLibrary *;
* is assumed to allow write operations by this driver module. *;
*****;

%cstutil_setcstroot;
data _null_;
  call symput('studyRootPath',cats("&_cstSRoot",
    "/cdisc-sdtm-3.1.3-&_cstVersion/sascstdemodata"));
  call symput('studyOutputPath',cats("&_cstSRoot",
    "/cdisc-sdtm-3.1.3-&_cstVersion/sascstdemodata"));
run;

```

Note: `&_cstSRoot` is set by the call to `%cstutil_setcstroot` to the location of the `cstSampleLibrary` that was defined during the product installation.

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

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

Step 2: Build and populate the SASReferences data set

```

%let _cstSetupSrc=SASREFERENCES;

*****;
* One strategy to defining the required library and file metadata for a CST process *;
* is to optionally build SASReferences in the WORK library. An example of how to do *;
* this follows. *;
* *;
* The call to cstutil_processsetup below tells CST how SASReferences will be provided *;
* and referenced. If SASReferences is built in work, the call to cstutil_processsetup *;
* may, assuming all defaults, be as simple as %cstutil_processsetup() *;
*****;

*****;
* Build the SASReferences data set *;
* column order: standard, standardversion, type, subtype, sasref, reftype, iotype, *;
* filetype, allowoverwrite, relpathprefix, path, order, memname, comment *;
* note that _cstGRoot points to the Global Library root directory *;
* path and memname are not required for Global Library references - defaults will be used*;
*****;
%cst_createdsfromtemplate(_cstStandard=CST-FRAMEWORK, _cstType=control, _cstSubType=reference,
  _cstOutputDS=work.sasreferences);

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

```

```

        "N"   ""   ""   1   ""   "")
values ("&_cstStandard" "&_cstStandardVersion" "control" "validation" "cntl_v" "libref"
       "input" "dataset" "N"   ""   "&studyRootPath/control" . "validation_control.sas7bdat" "")
[etc.]
;
quit;

```

The `cst_createdsfromtemplate` macro initializes the SASReferences data set that is required for SDTM validation. The SASReferences data set defines the location and name of each input metadata source, input data source, and output file that is created by the validation process, including the Validation Control data set. The Validation Control data set contains the set of checks to include in the validation process. The sample `validate_data.sas` driver program sets the path of the Validation Control data set to `&studyRootPath/control` and sets the name to `validation_control.sas7bdat`. Based on the code executed in step 1, this is the path:

```

sample study library directory/cdisc-sdtm-3.1.3/
sascstdemodata/control/validation_control.sas7bdat.

```

For an explanation of the purpose and content of each SASReferences file, see [“SASReferences File” on page 93](#). For a fully initialized SASReferences data set for SDTM validation, see [Display 5.3 on page 104](#).

Step 3: 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 22.](#)
- 2 Multiple calls to the %cstutilvalidatesasreferences macro are made to perform internal validation on the SASReferences data set.

The validation performed by the %cstutilvalidatesasreferences macro is described in the [“Assessing Structural Integrity and Content” on page 108.](#)

- 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 204](#) and [“Assessing Structural Integrity and Content” on page 108](#).

Step 4: 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 re-sorts the Validation Control data set based on the `_cstCheckSortOrder` property or global macro variable value. This step is optional.

- 3 Metadata about the validation process, such as the standard/version, key files referenced, and process datetimes, is added to the process Results data set.
- 4 For each check in the Validation Control data set with a checkstatus > 0, this macro calls the check macro specified in the Validation Control codesource field. It passes all of the check metadata to the check macro.
- 5 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 204](#).

Step 5: 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,
  _cstResetCmpLib=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 6.13 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 cstlnitSASAutos. 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 _cstlnitSASAutos does not exist.
_cstClearLibRefs	Clear all filerefs and librefs included in SASReferences, except any autocall filerefs.
_cstResetCmpLib	Reset the compiled library path based on the cmplib 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.
_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.

Macro Parameter	Action Attempted
<code>_cstDeleteGlobalMacroVars</code>	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 169](#) summarizes a sample validation process. Here are a few facts about the sample validation process:

- 1 The validation process was run on CDISC SDTM 3.1.3 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-framework-1.5/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\sasses\AppData\Local\Temp\SAS Temporary Files_TD4156_D72672_sasreferences	Info	0	0
4	CST0200		1	1	CST_INSERTSTANDARDSASREFS	SASReferences data set was successfully validated	Info	0	0
5	CST0200		1	2	CSTUTIL_ALLOCATESASREFERENCES	SASReferences data set was successfully validated	Info	0	0
6	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.3-1.5/programs/initialize.properties	Info	0	0
7	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstSampleLibrary/cdisc-sdtm-3.1.3-1.5/sascstdemodata/programs/validation.properties	Info	0	0
8	CST0200		1	1	SDTM_VALIDATE	PROCESS STANDARD: CDISC-SDTM	Info	0	0
9	CST0200		1	2	SDTM_VALIDATE	PROCESS STANDARDVERSION: 3.1.3	Info	0	0
10	CST0200		1	3	SDTM_VALIDATE	PROCESS DRIVER: SDTM_VALIDATE	Info	0	0
11	CST0200		1	4	SDTM_VALIDATE	PROCESS DATE: 2012-10-01T09:57:14	Info	0	0
12	CST0200		1	5	SDTM_VALIDATE	PROCESS TYPE: VALIDATION	Info	0	0
13	CST0200		1	6	SDTM_VALIDATE	PROCESS SASREFERENCES: C:\Users\sasses\AppData\Local\Temp\SAS Temporary Files_TD4156_D72672_sasreferences.sas7bdat	Info	0	0
14	CST0200		1	7	SDTM_VALIDATE	PROCESS STUDYROOTPATH: c:/cstSampleLibrary/cdisc-sdtm-3.1.3-1.5/sascstdemodata	Info	0	0
15	CST0200		1	8	SDTM_VALIDATE	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info	0	0
16	CST0200		1	9	SDTM_VALIDATE	PROCESS CSTVERSION: 1.5	Info	0	0
17	CST0200		1	10	SDTM_VALIDATE	PROCESS CONTROLLED TERMINOLOGY SOURCE: c:/cstGlobalLibrary/standards/cdisc-terminology-1.5/cdisc-sdtm/201104/formats/ctterms (Controlled Terminology released by NCI on 2011-04-08)	Info	0	0
18	CST0100	SDTM0101	1	1	SRCDATA.DM	No errors detected in SRCDATA.DM	Info	0	0
19	CST0100	SDTM0130	1	1	SRCDATA.DM	No errors detected in SRCDATA.DM	Info	0	0
20	SDTM0435	SDTM0435	1	1	SRCDATA.RS	RSEVAL must be populated when RSEVALID is populated	Error	1	0
21	CST0032	SDTM0451	1	1	CSTCHECK_NOTINCODELIST	Reference terminology data set meddra was set to ctfref.meddra	Info	0	0
22	SDTM0451	SDTM0451	1	2	SRCDATA.AE.AELLT	Invalid value for Coded Term	Warning	1	0

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	CST0200		1	1			
5	CST0200		1	2			
6	CST0108		1	1			
7	CST0108		1	1			
8	CST0200		1	1			
9	CST0200		1	2			
10	CST0200		1	3			
11	CST0200		1	4			
12	CST0200		1	5			
13	CST0200		1	6			
14	CST0200		1	7			
15	CST0200		1	8			
16	CST0200		1	9			
17	CST0200		1	10			
18	CST0100	SDTM0101	1	1			
19	CST0100	SDTM0130	1	1			
20	SDTM0435	SDTM0435	1	1	RSEVAL=RSEVALID=INVESTIGATOR	STUDYID=SASCSTDDEMADATA.USUBJID=S001P011,RSSEQ=3	
21	CST0032	SDTM0451	1	1			The lookupSource value did not have an associated libref
22	SDTM0451	SDTM0451	1	2	AELLT=	STUDYID=SASCSTDDEMADATA.USUBJID=S001P008,AEDECOD=Rash prunhc.AESTDTC=2008-03-01T16:30	Has coding been done against the same version of MedDRA?

Table 6.14 Comments about the Validation Results Data Sets in Displays 6.9 and 6.10

Lines	Comment
1,6,7	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.
4-5	Informational notes that inform you that the process SASReferences data set passed internal validation using the cstutilvalidatesasreferences macro called from two different macros.
8-17	Informational summary that provides internal documentation about the process.
18-19	Checks SDTM0101 and SDTM0130 ran without error.
20	An error was detected in the SRCDATA.RS domain. The keyvalues column identifies the problem RS record, and the actual column reports the values that are in error.
21-22	Check SDTM0451 performs a terminology lookup for the AELLT column in SRCDATA.AE using the ctref.meddra data set. The ctref SAS libref was defined in the SASReferences type=referenceceterm record pointing to the SAS library containing the medDRA data set. The keyvalues column identifies the problem AE record, and the actual column reports that the problem AELLT value in error was blank.

Display 6.11 Example of a Validation Metrics Data Set

	metricparameter	reccount	resultid	srcdata	resultseq
1	# of subjects	70	SDTM0101	SRCDATA.DM	1
2	# of records tested	350	SDTM0101	SRCDATA.DM	1
3	Elapsed time to run check: 0:00:01	.	SDTM0101	CSTCHECK_COLUMN	1
4	# of subjects	70	SDTM0130	SRCDATA.DM	1
5	# of records tested	70	SDTM0130	SRCDATA.DM	1
6	Elapsed time to run check: 0:00:01	.	SDTM0130	CSTCHECK_COLUMN	1
7	# of subjects	1	SDTM0435	SRCDATA.RS	1
8	# of records tested	3	SDTM0435	SRCDATA.RS	1
9	Elapsed time to run check: 0:00:01	.	SDTM0435	CSTCHECK_COLUMNCOMPARE	1
10	# of records tested	106	SDTM0451	SRCDATA.AE	1
11	Elapsed time to run check: 0:00:01	.	SDTM0451	CSTCHECK_NOTINCODELIST	1
12	# of distinct check invocations	4	METRICS	SDTM_VALIDATE	1
13	# check invocations not run	0	METRICS	SDTM_VALIDATE	1
14	Errors (severity=High) reported	1	METRICS	SDTM_VALIDATE	1
15	Warnings (severity=Medium) reported	1	METRICS	SDTM_VALIDATE	1
16	Notes (severity=Low) reported	0	METRICS	SDTM_VALIDATE	1
17	Structural errors, warnings and notes	0	METRICS	SDTM_VALIDATE	1
18	Content errors, warnings and notes	2	METRICS	SDTM_VALIDATE	1

Table 6.15 Comments about the Validation Metrics Data Set

Lines	Comment
1-2	In check SDTM0101, 70 subjects and 5 date columns for each DM subject were evaluated.
3	Check SDTM0101 took one second to run using cstcheck_column.
10	Check SDTM0451 evaluated the AELLT column for each of the 106 SRCDATA.AE records.
12	A summary metric of unique check invocations.
13	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).
14-18	Summary metric counts of the number of records, by type of metric, in the Results data set.

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 10, “Reporting,”](#) on page 373.

Validation Checks by Standard

Overview

The SAS Clinical Standards Toolkit 1.5 provides a set of defined checks for each standard, where the *global standards library directory/metadata* standards data set supportsvalidation flag is set to “Y”. By default, each Validation Master data set is located in the *global standards library directory/standards/<standard>/validation/control* folder.

This table summarizes the content of each standard-specific validation_master data set that is provided by SAS:

Table 6.16 Summary of Checks in Each validation_master Data Set That Is Provided by SAS

CDISC Standard and Version	Total Number of Check Records	Number of Unique Checks	Number of Check Macros Used
ADaM 2.1	264	257	14
CRT-DDS 1.0	83	12	7
CT 1.0.0	34	14	7
ODM 1.3.0	179	39	10
ODM 1.3.1	190	38	10
SDTM 3.1.1	257	150	14
SDTM 3.1.2	247	243	15
SDTM 3.1.3	290	263	15
CST-FRAMEWORK	130	86	11

ADaM 2.1

The CDISC ADaM 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).

In addition, SAS has added 45 unique checks (52 total records) to the Validation Master data set. These checks can be identified where checksource="SAS".

ADaM data sets are typically derived from a tabulation study, such as SDTM or SEND. Some checks require the comparison of ADaM content with data and metadata from the

tabulation source. Of the 264 validation_master records, 22 involve a comparison with another CDISC standard such as SDTM 3.1.3.

CDISC CRT-DDS 1.0

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

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

Checks fall into these general categories:

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

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

- an enumeration
- a regular expression

The SAS Clinical Standards Toolkit 1.5 provides 83 CDISC CRT-DDS validation checks. These validation checks were developed by SAS and are based on CRT-DDS and ODM implementation experience and careful review of the associated implementation guides, with special emphasis on the occurrence of “should” within each implementation guide. [Table 6.17 on page 175](#) 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 6.17 CRT-DDS Validation Check Types

Check Type	Category	Description
Unique in data set	Structural	No two values for the source column can be the same in the same source data set.
Required character value	Data	The trimmed (white space removed) value of the character data must consist of one or more characters.
Required numeric value	Data	The numeric value of the column cannot be missing.
Enumeration(s0,s1,...)	Data	If character data exists, its value must match one of the enumerated character strings. All string comparisons are case sensitive.
Foreign key(targetColumn)	Structural	Each existing value in this column must have an equivalent value in the target column.
Foreign key required(targetColumn)	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.

Check Type	Category	Description
Character format: language	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. Invalid 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: <code>[a-zA-Z]{1,8}(-[a-zA-Z0-9]{1,8})*</code> .
Character format: fileName	Data	The character data must not contain any characters other than uppercase and lowercase letters of the alphabet, numeric digits, an underscore (<code>_</code>), or a period. Regular expression: <code>[A-Za-z0-9_.]+</code> .
Character format: sasFormat	Data	The first character must be either a lowercase or uppercase letter, an underscore (<code>_</code>), or the dollar sign (<code>\$</code>). Any subsequent character must be either an uppercase or lowercase letter, a numeric digit, an underscore (<code>_</code>), or a period. Regular expression: <code>[A-Za-z_\$][A-Za-z0-9_.]*</code> .
Character format: sasName	Data	The first character must be either a lowercase or uppercase letter or an underscore (<code>_</code>). Any subsequent character must be either an uppercase or lowercase letter, a numeric digit, or an underscore (<code>_</code>). Regular expression: <code>[A-Za-z_][A-Za-z0-9_]*</code> .
Unique across data sets(targetcolumn0,...)	Structural	No value in this column can be the same as any value in any of the data set columns.
Primary key	Data	Must be unique in data set check type and the required character value check type.
Must Have Corresponding Value(targetColumn)	Structural	For each distinct value in this column, there must be at least one equivalent value in the target column.

Check Type	Category	Description
No Duplicates Per Unique Value(targetColumn)	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).

The CDISC CRT-DDS validation also checks 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-1.0-1.5/formats* folder.

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

CDISC ODM 1.3.0 and 1.3.1

The SAS Clinical Standards Toolkit provides check macros that validate the data in the SAS data sets representing CDISC ODM 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 1.5 provides 179 ODM 1.3.0 and 190 ODM 1.3.1 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.

By default, the ODM 1.3.0 Validation Master data sets are located in the *global standards library directory/standards/cdisc-odm-1.3.0-1.5/validation/control* and the *global standards library directory/standards/cdisc-odm-1.3.1-1.5/validation/control* folders.

[Table 6.18 on page 179](#) 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 6.18 ODM Validation Check Types

Check Type	Category	Description
Unique in data set	Structural	No two values for the source column can be equivalent within the same source data set.
	Structural	Duplicate OrderNumber element. The OrderNumber attribute must be unique within the same source data set when not null.
Required character value	Data	The trimmed (white space removed) value of the character data must consist of one or more characters.
Required numeric value	Data	The numeric value of the column cannot be missing.
Enumeration(s0,s1,...)	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)	Structural	Each existing value in this column must have an equivalent value in the given target column.
Foreign key required(targetColumn)	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.

Check Type	Category	Description
Character format: language	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, <code>e</code> is a legal value, as are <code>en-us</code> and <code>english</code> and <code>english-d842</code> . Invalid values include <code>1en</code> , <code>mumblespeak</code> , and <code>en_us</code> . The hyphen character sequence can be repeated any number of times also making a value such as <code>english-mumbly-growly-47</code> a legal value. Regular expression: “[a-zA-Z]{1,8}(-[a-zA-Z0-9]{1,8})*”.
Character format: fileName	Data	The character data must not contain any characters other than upper- and lower-case letters of the alphabet, numeric digits, the underscore (<code>_</code>) character, or a period. Regular expression: <code>[A-Za-z0-9_]+</code> .
Character format: sasName	Data	The first character must be either a lower- or upper-case letter or an underscore (<code>_</code>). Any subsequent character must be either an upper- or lowercase letter, a numeric digit, or the underscore (<code>_</code>). Regular expression: <code>[A-Za-z_][A-Za-z0-9_]*</code> .
Character format: sasFormat	Data	The first character must be either a lower- or upper-case letter, an underscore (<code>_</code>), or the dollar sign (<code>\$</code>). Any subsequent character must be either an upper- or lowercase letter, a numeric digit, the underscore (<code>_</code>), or a period. Regular expression: <code>[A-Za-z_\$][A-Za-z0-9_]*</code> .
Must Have Corresponding Value(targetColumn)	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,...)	Structural	No value in this column can be equal to any value in any of the given data set columns.
Primary key	Data	Must satisfy the Unique in data set check type and the required character value check type.

Check Type	Category	Description
Invalid Value	Data	Documents based on ODM 1.3 should have ODM version set to 1.3.
	Data	An invalid SAS format name. In case the data type is character, the format name needs to start with the \$ character.
	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+.
	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+)?.
	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.
	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)?.
	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	Data	External file reference found because the prior file OID is not missing (for example, ODM.PriorFileOID ne "")

Check Type	Category	Description
Referenced OID Not Found	Data	If Metadata version IncludedOID is non-null, the referenced OID must be found in this XML file.
	Data	If Metadata version IncludedStudyOID is non-null, the referenced OID must be found in this XML file.
Attribute is Required	Column	The ItemDef length attribute is required when data type is text, string, integer, or float and can be ignored for the other types.
	Column	The required attribute SignificantDigits cannot be empty or missing when Data type is Float.
	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	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	Multirecord	Typed and Untyped data transmission should not be mixed within a single ODM file.
Multiple Records Exists	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).

For the CDISC ODM validation checks that compare 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). For ODM 1.3.0, these are in the *global standards library directory/standards/cdisc-odm-1.3.0-1.5/formats* folder. Case-sensitivity compliance is required by the XML schema validation.

CDISC SDTM 3.1.1, 3.1.2, and 3.1.3

The SAS Clinical Standards Toolkit 1.5 provides validation checks in support of CDISC SDTM 3.1.1, 3.1.2, and 3.1.3. These checks are derived from multiple sources that have evolved over time, including:

- The SAS interpretation of the CDISC SDTM WebSDM 2.6 and 3.0 documented checks.
- Checks supporting loads into the FDA Janus study data repository.
- The SAS interpretation of the OpenCDISC CDISC SDTM validation rules (<http://www.opencdisc.org>)
- SAS checks based on SAS data management and cleaning experiences building CDISC SDTM domains.

Future updates will be guided in part by the FDA/PhUSE Working Groups (<http://www.phusewiki.org>), such as the SDTM Validation Rules project.

Each version of the CDISC SDTM Validation Master data set (such as SDTM 3.1.3) contains a different number of checks based on the rules that are in effect at the time of each version and the number and type of supported tabulation domains. For more information about the distribution of checks by version, see [Table 6.16 on page 173](#).

By default, the Validation Master data set is located in the *global standards library directory/standards/<specific standard and version>/validation/control* folder. It is named `validation_master.sas7bdat`.

Each 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.

Multiple instances of specific checks are provided 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 in 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 it reports the same error message for the check.

Note: The validation check data set column `checkstatus` indicates the state of each check. It indicates that the check is ready to be run in its current defined state, or that the check can be run based on some external criteria. Current valid values are 1 (active), 0 (inactive), -1 (deprecated), and -2 (not yet implemented). Values are extensible to meet your requirements. You can elect to use other values such as 1 (draft), 2 (test), and 3 (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 if the `checkstatus` value is > 0 .

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, see [“Messages” on page 145](#). By default, the `Validation_StdRef` data set is located in the *global standards*

library directory/standards/<specific standard and version>/validation/control folder.

CDISC CT 1.0.0

The CDISC CT validation checks are patterned in part after the CDISC ODM checks. The checks ensure that SAS rules for format names and non-duplicate values are followed. A total of 34 records are defined in the Validation Master data set, which, by default, is located at: *global standards library directory/standards/cdisc-ct-1.0.0-1.5/validation/control*.

The SAS Clinical Standards Toolkit Framework

Validation of the SAS Clinical Standards Toolkit framework files is referred to as internal validation. For more information, see [Chapter 7, “Internal Validation,” on page 229](#).

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. Twenty-one check macros have been defined in the SAS autocall library to support compliance assessments across supported standards.
- 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. For example, the check macros `cstcheck_column`, `cstcheck_notincodelist`, and `cstcheck_notunique` are used by every standard provided by SAS that supports validation.
- 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 6.19 SAS Clinical Standards Toolkit Validation Check Macros

Check Macro	Code Logic Style	Description of Purpose
<code>cstcheck_column</code>	Statement	Identifies any invalid column values or attributes.
<code>cstcheck_columncompare</code>	Step	Supports comparison of column values.
<code>cstcheck_columnexists</code>	By default, this check does not require the use of <code>codeLogic</code> . If the check metadata includes a non-null value of <code>codeLogic</code> , then DATA step code logic is required.	Determines whether one or more of the columns defined in <code>columnScope</code> exist in each of the tables defined in <code>tableScope</code> .

Check Macro	Code Logic Style	Description of Purpose
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.
cstcheck_notincodelist	If lookuptype=DATASET, DATA step code logic required Else, DATA step code logic is optional	Identifies any column values inconsistent with controlled terminologies. Requires reference to the SAS format search path built based on type=FMTSEARCH records in the SASReferences control file. Example is a **STAT value is found other than 'NOT DONE.'

Check Macro	Code Logic Style	Description of Purpose
cstcheck_notsorted	(not used)	Identifies any domain that is not sorted by the keys defined in the metadata.
cstcheck_notunique	Not used for functions 1 through 3; DATA step for function 4	<p>A multi-function macro that assesses the uniqueness of data sets, columns, or value-pairs from two columns.</p> <p>Function 1: Is data set unique by a set of columns?</p> <p>Function 2: For any subject, are column values unique?</p> <p>Function 3: Does a combination of two columns have unique values?</p> <p>Function 4: Are the values in one column (Column2) consistent in each value of another column (Column1)?</p>
cstcheck_recmismatch	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.
stcheckcompareallcolumns	Step	Compares all columns in one domain with the same columns in other domains.
cstcheckentitynotfound	Step	Reports that an entity, typically a file, folder, or column, cannot be found.

Check Macro	Code Logic Style	Description of Purpose
cstcheckforeignkeynotfound	Step	Compares the consistency of one or more columns across two tables, where a column in the first table is a foreign key that points to a primary key in the second table.

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.

The use of each check macro, by standard and version, is summarized in this display.

Display 6.12 Use of Validation Check Macros by Standard

Check Macro	ADaM 2.1	CRTDDS 1.0	CT 1.0.0	ODM 1.3.0	ODM 1.3.1	SDTM 3.1.1	SDTM 3.1.2	SDTM 3.1.3	CST Framework
cstcheck_column	✓	✓	✓	✓	✓	✓	✓	✓	✓
cstcheck_columncompare	✓		✓	✓	✓	✓	✓	✓	✓
cstcheck_columnexists	✓								
cstcheck_columnvarlist	✓					✓	✓	✓	
cstcheck_comparedomains	✓			✓	✓	✓	✓	✓	✓
cstcheck_crossstdcomparedomains	✓								
cstcheck_crossstdmetamismatch	✓								
cstcheck_dsmismatch	✓					✓	✓	✓	✓
cstcheck_metamismatch	✓					✓	✓	✓	
cstcheck_notconsistent	✓	✓		✓	✓	✓	✓	✓	✓
cstcheck_notimplemented							✓	✓	
cstcheck_notincodelist	✓	✓	✓	✓	✓	✓	✓	✓	✓
cstcheck_notsorted				✓	✓	✓	✓	✓	
cstcheck_notunique	✓	✓	✓	✓	✓	✓	✓	✓	✓
cstcheck_recismatch		✓				✓	✓	✓	✓
cstcheck_recnofound			✓	✓	✓	✓	✓	✓	✓
cstcheck_violatesstd		✓	✓	✓	✓	✓	✓	✓	
cstcheck_zeroobs	✓		✓	✓	✓	✓	✓	✓	✓
cstcheckcompareallcolumns	✓								
cstcheckentitynotfound									✓
cstcheckforeignkeynotfound		✓							

More complete documentation is provided for each check macro in the online macro API reference documentation. This information is derived from the code headers. See [“Special Topic: Validation Customization” on page 212](#).

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: `usesourcemetadata`, `tablescope`, `columnscope`, and `codeologic`.

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

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

For those checks that use `codeologic`, 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=%kscan(%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 standard-specific properties file. For CDISC-SDTM, the value of `_cstSubjectKeys` is set to `STUDYID USUBJID` by default.
- `_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 6.20 Multiple Validation Check Invocations for a Specific CheckID

checkid	tablescope	columnscope	code logic
SDTM0604	_ALL_-TS	**SEQ	<pre>%let _cstLastKey=%kscan(%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 3.1.1 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 128](#).)

Here is the code logic for CDISC SDTM check SDTM0623:

```
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 how 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 **IETESTCD** with **LBSTRESU** is not the intention.

Tablescope and columnScope syntax supports wildcarding and addition and subtraction operators. However, this flexible functionality is not required. You can submit explicit table and column references. CDISC SDTM check SDTM0623 could be defined in the Validation Master data set as 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 6.21 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 6.22 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 6.23 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.

String	Interpretation	Comment
2009---25	The 25th day of an unknown month in the year 2009. The month is missing.	If a missing value is embedded in the string, then it must always be denoted by a hyphen (-).
--03-25	The 25th day of March in an unknown year.	Missing year.
--03--T-:15	The 15th minute of an unknown hour of an unknown day of the third month of an unknown year.	Missing year, day, and hour.
2009-03	Month of March 2009.	Trailing partial values can be omitted (truncated). If time is omitted, then T must also be omitted.
2009-03--T12	The 12th hour of an unknown day in March 2009.	Missing day of month.

Durations: Template

Table 6.24 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 6.25 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>

String	Interpretation	Comment
P0000-00-01	The span of zero years + zero months + one day.	Durations can be expressed in an alternative format. When expressed, the length of time is stored in the same format as date and time, but preceded by a P. Instead of expressing a specific point in time, it expresses a period of time.
P1Y2M3DT4H5M6S	The span of 1 year, 2 months, 3 days, 4 hours, 5 minutes, and 6 seconds.	The units must be in the correct order. The T is required for all time values, but it should not be specified if no time value is given.

Intervals: Template

Table 6.26 Example ISO 8601 Values for Intervals: Template

String	Interpretation	Comment
PnYnMnDTnHnMnS/YYYY-MM-DDTHH:MM:SS or YYYY-MM-DDTHH:MM:SS/ PnYnMnDTnHnMnS or YYYY-MM-DDTHH:MM:SS/ PnYnMnDTnHnMnS or YYYY-MM-DDTHH:MM:SS/YYYY-MM-DDTHH:MM:SS	Intervals	This is a duration that is anchored to a specific point in time.

Intervals: Examples

Table 6.27 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 6.28 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
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#pla0qt18rxdrkn1b0rtdfh2t8zs.htm
CALL IS8601_CONVERT Routine	http://support.sas.com/documentation/cdl/en/lefuctionsref/63354/HTML/default/viewer.htm#p0bhy7ndmdivmmn10b2okmbgiqmj.htm

Topic	Link
\$N8601Bw.d Informat	http://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#n1mqdr981wjxx3n11kqndfer2ei5.htm
\$N8601Ew.d Informat	http://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#p17xoiovjnnngtrnlp8yw1r0xyyep.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#n09mk4hlba9wp1n1tc3e7x0eow8q.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 &_amp;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 program (`validate_data.sas` for SDTM 3.1.3) documented in “[Running a Validation Process](#)” on page 160, 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(("&sysfunc(tranwrd(options %cmpres(&_cstDebugOptions),
      %str( ), %str( no)))");));
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. To increase the default log size permitted, use the option DMSLOGSIZE . 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 94](#).

This table lists some common setup errors and possible causes.

Table 6.29 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/metadata</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>

Error	Location Where Error Is Reported	Possible Cause and Corrective Action
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 6.30 *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 tablescope value could be found in the work._csttablemetadata 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.

Error	Resultid in Results Data Set	Possible Cause or Corrective Action
No columns evaluated-check Validation Control specification.	CST0004	<p>No columns interpreted from the columnscope value could be found in the work._cstcolumnmetadata data set.</p> <p>The SAS Clinical Standards Toolkit looks at the union of both tablescope and columnscope to build work._cstcolumnmetadata. The specified column might exist in a domain, but not in any column specified in a tablescope domain.</p>
Lookup to SASReferences control data set failed.	CST0006	<p>The SAS Clinical Standards Toolkit code has a call to the cstutil_getsasreference utility macro for a type or type and subtype combination that cannot be found in the SASReferences data set. This indicates that SASReferences has been incompletely defined for the SAS Clinical Standards Toolkit validation process.</p>
Validation Control parsing of tablescope/ column results in inconsistent sublist lengths.	CST0023	<p>This check involves a comparison of tables or columns, as indicated by multiple sets of brackets in tablescope or columnscope. Each set of brackets constitutes a sublist. However, the number of items in the specified sublist is inconsistent or unexpected by the check macro. Options typically include a more accurate specification of sublist items, either using explicit table or column names or more restrictive tablescope syntax (that is, removing the domain causing the inconsistency using minus sign (-) syntax, such as _ALL_-DM).</p>
One or more check metadata column values is invalid.	CST0026	<p>A value in the Validation Control data set for the check being run is invalid in the context of the specific check macro. Examples include conditions that are required by the check macro but are not found, such as no code logic found, an unexpected usesourcemetadata value, or no lookuptype or lookupsource for valid value assessments.</p>

Error	Resultid in Results Data Set	Possible Cause or Corrective Action
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`, `_csttablemetadata`, and `_cstcolumnmetadata`). These files might remain in the Work directory after a process by default. Toggling the `_cstDebug` global macro variable to 1 forces the Work files to remain after the process ends.
- When debugging, avoid setting the parameter flags in `cstutil_cleanupcstsession` to 1 (if that cleanup macro is called).

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

- Use `work._cstcolumnmetadata` and `work._csttablemetadata` to resolve missing domain and column issues. These data sets can also be used to resolve sublist length differences for checks using sublist syntax `[]` in `tablescope` and `columnscope`.
- Use the resultid code (for example, CST0003) in the Results data set to search the check macro code module used for a specific check for information about the error. The name of the macro code module is set in the Validation Control `codesource` field.

Special Topic: Validation Customization

Overview

One of the significant benefits of the SAS Clinical Standards Toolkit is that 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 213](#)
- [“Case Study 2: Using Any Set of Source Data and Metadata” on page 214](#)
- [“Case Study 3: Modifying the SAS Validation Checks for Supported Standards” on page 214](#)
- [“Case Study 4: Adding New Validation Checks for Supported Standards” on page 215](#)
- [“Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros” on page 217](#)
- [“Case Study 6: Modifying the SAS Clinical Standards Toolkit Messaging, Including Internationalization” on page 218](#)

- [“Case Study 7: Validation of Multiple Studies” on page 220](#)

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 25](#).

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 3.1.3 sample study (*sample study library directory/cdisc-sdtm-3.1.3-1.5/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 codelogic field to alter the logic used to identify error conditions. This might be a necessary change if a check needs to be generalized to accommodate new domains or columns. Or, customer conventions might differ from those in the SAS Clinical Standards Toolkit checks.
- If customer code changes are sufficiently significant, then it might be better to create a new validation check macro. (See [“Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros” on page 217.](#)) 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 128.](#)
- 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 7.](#))

- 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 7](#).
- 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 145](#).
- 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 cstcheck_columncompare, cstcheck_columnvarlist, or cstcheck_notunique.
 - Does the check compare tables?
Consider Validation Master records where the codesource column is cstcheck_comparedomains or cstcheck_recnofound.
 - Does the check look across multiple standards?
Consider Validation Master records where the codesource column is cstcheck_crossstdcomparedomains or cstcheck_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 specific codelist), 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 21 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 185](#) and the online macro API reference documentation.

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 185](#).
- 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	tablescope
SDTM0013	CDISC-SDTM	***	Janus	IR4253	Note	_ALL_
SDTM0013	CDISC-SDTM	***	WebSDM	IR4253	Warning	_ALL_

The SAS Clinical Standards Toolkit representation of the SDTM0013 check in the Messages data set is:

Display 6.15 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=sourcemetadata) 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.

Source data libraries are considered when tablescope and columnscope parsing occurs in the SAS Clinical Standards Toolkit. However, if tablescope does not

include the libref, unintended comparisons of multiple columns or multiple domains from different studies can occur. As a result, this methodology is not recommended unless you consistently use multiple librefs in the source metadata and validation check metadata.

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	cterms.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 1.5, these are provided (by CDISC model and snapshot date) in the following location:

***global standards library directory/standards/
cdisc-terminology-1.5/***

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 directory/standards/
cdisc-terminology-1.5/control*** folder.

For example, suppose you want to add a new CDISC ADaM controlled terminology snapshot released on 01July2013. A new 201307 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 201307 catalog.

Display 6.20 New Controlled Terminology Record

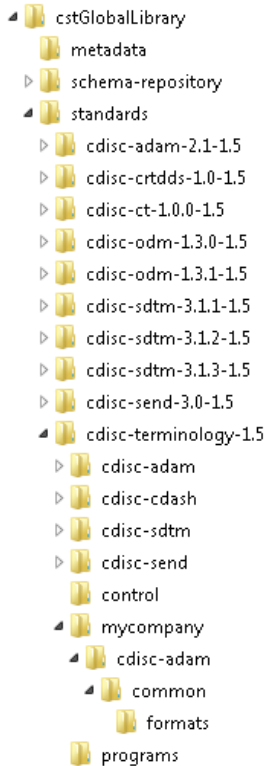
standard	standardversion	standardsubtype	standardsubtypeversion	path	isstandarddefault	description
CDISC-TERMINOLOGY	CDISC-ADAM	NCI_THESAURUS	201101	&_cstGRoot/standards/cdisc-terminology-1.5/cdisc-adam/201101/formats	N	CDISC ADaM Controlled Terminology, released by NCI on 2011-01-07
CDISC-TERMINOLOGY	CDISC-ADAM	NCI_THESAURUS	201107	&_cstGRoot/standards/cdisc-terminology-1.5/cdisc-adam/201107/formats	N	CDISC ADaM Controlled Terminology, released by NCI on 2011-07-22 (updated 2011-01 version)
CDISC-TERMINOLOGY	CDISC-ADAM	NCI_THESAURUS	201307	&_cstGRoot/standards/cdisc-terminology-1.5/cdisc-adam/201307/formats	Y	CDISC ADaM Controlled Terminology, released by NCI on 2013-07-01
CDISC-TERMINOLOGY	CDISC-ADAM	NCI_THESAURUS	current	&_cstGRoot/standards/cdisc-terminology-1.5/cdisc-adam/current/formats	N	Current CDISC ADaM Controlled Terminology, Copy of 2013-07-01

The SAS Clinical Standards Toolkit 1.5 provides sample programs that create the data sets that are needed to register controlled terminology. The programs also register these data sets. The programs are called `create_terminology_standarddatasets.sas` and `registerstandard.sas` and are located in the *global standards library directory/standards/cdisc-terminology-1.5/programs* folder.

Note: You must 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 25](#).

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-2.1-1.5	Y
CDISC-CRTDDS	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1.0-1.5	Y
CDISC-CT	CTX	1.0.0	CDISC CT XML V1.0.0	&_cstGRoot./standards/cdisc-ct-1.0.0-1.5	Y
CDISC-ODM	ODM	1.3.0	CDISC ODM V1.3.0	&_cstGRoot./standards/cdisc-odm-1.3.0-1.5	N
CDISC-ODM	ODM	1.3.1	CDISC ODM V1.3.1	&_cstGRoot./standards/cdisc-odm-1.3.1-1.5	Y
CDISC-SDTM	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1-1.5	N
CDISC-SDTM	SDTM	3.1.2	CDISC SDTM V3.1.2	&_cstGRoot./standards/cdisc-sdtm-3.1.2-1.5	N
CDISC-SDTM	SDTM	3.1.3	CDISC SDTM V3.1.3	&_cstGRoot./standards/cdisc-sdtm-3.1.3-1.5	Y
CDISC-SEND	SEND	3.0	CDISC SEND V3.0	&_cstGRoot./standards/cdisc-send-3.0-1.5	Y
CDISC-TERMINOLOGY	CT	COMPANY_STD	CDISC terminology used by our company	&_cstGRoot./standards/cdisc-terminology-1.5/mycompany	Y
CDISC-TERMINOLOGY	CT	NCI_THESAURUS	CDISC Terminology	&_cstGRoot./standards/cdisc-terminology-1.5	N
CST-FRAMEWORK	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework-1.5	Y

The standardsubtypes data set located in the *global standards library directory/standards/cdisc-terminology-1.5/control* folder now contains this CDISC ADaM record:

Display 6.23 CDISC ADaM Record Example

standard	standardversion	standardsubtype	standardsubtypeversion	path	isstandarddefault	description
CDISC-TERMINOLOGY	CDISC-ADAM	COMPANY_STD	common	&_cstGRoot/standards/cdisc-terminology-1.5/ mycompany/cdisc-adam/common	Y	Controlled Terminology (company standard as of 2013-01-01)

- 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	tablescope	columnscope	codelogic	lookuptype	lookupsource
SDTM0450	cstcheck_notincodelist	_ALL_	**DECOD	%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=referencecterm.

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.

7

Internal Validation

Overview	229
Supporting Macros	231
Validating a SASReferences Data Set	233
Sample Driver Modules	235
Overview	235
Internal Validation Driver Modules That Are Provided by SAS	236
Internal Validation Driver Module Workflow: validate_standard	237
Validation Checks	243
validation_master Data Set	243
validation_control SAS Views	245
Example Internal Validation Check: CSTV026	246

Overview

Each standard as defined in the SAS Clinical Standards Toolkit includes numerous SAS metadata files and SAS macros. For the SAS Clinical Standards Toolkit to function properly, each file must contain a core set of columns that have an expected variable type. Each macro is designed to use these core columns to perform certain functions.

The term *internal validation* refers to a set of tools that checks the consistency of the SAS metadata files. The tools use the SAS Clinical Standards Toolkit validation framework and methodology that assess standard-specific files against a defined

reference standard. The tools determine whether the metadata that the SAS Clinical Standards Toolkit expects is correctly defined.

The primary design goals of internal validation include:

- Verify that the SAS Clinical Standards Toolkit metadata files that are provided by SAS are consistent and correct.
- Use this functionality to facilitate definition, registration, and validation of new user-defined custom standards.
- Use the SAS Clinical Standards Toolkit validation framework whenever possible.
- Limit the amount of new metadata that is required to support internal validation.
- Enable the use of the functionality during product development as a part of the installation qualification process and operational qualification process and as users add new metadata or modify existing metadata.
- Significantly expand the internal validation of SASReferences data sets beyond the use of the cstutil_checkds autocall macro used in releases prior to SAS Clinical Standards Toolkit 1.5.
- Develop a suite of internal validation programs, tools, and validation processes that can be run independently or as part of any SAS Clinical Standards Toolkit process provided by SAS.

The SAS Clinical Standards Toolkit 1.5 provides a representative sample of programs, tools, and validation processes to support internal validation. Future releases are expected to more fully address all of the design goals and to better support and expand several of the following usage scenarios:

Table 7.1 *Status of Internal Validation Development*

Usage Scenario	SAS Clinical Standards Toolkit 1.5 Status
Support installation qualification and operational qualification assessment and reporting	Available; future additions planned
Support registration of a new standard and updates to an existing standard	Not yet available

Usage Scenario	SAS Clinical Standards Toolkit 1.5 Status
Assess metadata consistency across files	Available; future additions planned
Determine the structural validity of a metadata file	Available
Confirm valid content of a metadata file	Available; future additions planned
Validate a SASReferences data set	Available
Evaluate validation check metadata	Not yet available

Supporting Macros

The following macros have been added to the SAS Clinical Standards Toolkit 1.5. These macros directly support internal validation. Many of these macros are also used for other purposes.

These macros are located in the primary SAS Clinical Standards Toolkit autocall path:

- Microsoft Windows

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

- UNIX

```
!sasroot/sasautos
```

For complete macro documentation, see the online macro API reference documentation.

Table 7.2 Autocall Macros Added in Support of Internal Validation

Macro	Primary Purpose
cstcheckentitynotfound	Reports that a SAS Clinical Standards Toolkit entity (typically a file, folder, or column) cannot be found.

Macro	Primary Purpose
<code>cstcheckutilcheckfile</code>	Determines whether a file exists as defined by columns in a source data set.
<code>cstcheckutilcheckfolder</code>	Determines whether a folder exists as defined by columns in a source data set.
<code>cstcheckutilcheckstructure</code>	Compares the structure of data sets referenced within <code>StandardSASReferences</code> or <code>SASReferences</code> data sets against a template.
<code>cstcheckutilfindsasrefsfile</code>	Determines whether designated files in the referenced <code>SASReferences</code> data set exist.
<code>cstcheckutillookupvalues</code>	Determines whether metadata column values for discrete columns exist in the <code>Standardlookup</code> data set.
<code>cstutilbuildmetadatafromsasrefs</code>	Builds the framework <code>reference_tables</code> and <code>reference_columns</code> data sets.
<code>cstutilbuildstdvalidationcode</code>	Generates the validation-specific macro <code>_cstreadStds</code> to build the workflow.
<code>cstutilcheckforproblem</code>	Handles any error condition that sets error condition <code>_cst_rc</code> to 1.
<code>cstutilcheckwriteaccess</code>	Checks for Write access for an output object.
<code>cstutilcomparestructure</code>	Compares the metadata structure of two data sets.
<code>cstutilfindvalidfile</code>	Checks whether a folder, file, data set, catalog, or catalog member exists.
<code>cstutilprocessfailed</code>	Returns a Boolean value to report whether a process failed.
<code>cstutilvalidatesasreferences</code>	Validates the structure and content of a <code>SASReferences</code> data set.
<code>cstutilvalidationsummary</code>	Summarizes the contents of the validation process results data set.

Validating a SASReferences Data Set

A key internal validation design goal is to verify the content of each SASReferences data set. Each SAS Clinical Standards Toolkit process requires the use of a SASReferences data set. The SASReferences data set identifies all of the inputs that are required and the outputs that are created by the process. Each process might have its own unique SASReferences data set. For a description of the content and usage of SASReferences data sets, see [Chapter 5, “SASReferences File,” on page 93](#).

In most driver modules that are provided by SAS, a call to the `cstutil_processsetup` macro initiates a series of steps to establish the environment to perform a subsequent task, such as validating a study or building a `define.xml` file. SAS file and library references are allocated. Updates to the SAS autocall and format search paths are completed. These steps are completed based solely on the content of a SASReferences data set.

With the SAS Clinical Standards Toolkit 1.5, the SASReferences data set is automatically validated through a series of calls to the `cstutilvalidatesasreferences` macro. These calls to `cstutilvalidatesasreferences` are made within macros called in the `cstutil_processsetup` macro workflow. The following error conditions are reported by default:

Table 7.3 SASReferences Data Set Error Conditions Reported by the `cstutilvalidatesasreferences` Macro

Error Flag	Error Condition	Details
CHK01	The data set is structurally incorrect.	A structural comparison with the template that is provided by SAS is performed using <code>%cstutilcomparestructure</code> . Minor differences involving labels, informats, and formats are generally ignored.
CHK02	An unknown standard or standardversion exists.	The standard and standardversion must be registered in the <code><global standards library directory>/metadata/standards</code> data set.

Error Flag	Error Condition	Details
CHK03	A referenced input or output file or folder cannot be accessed.	If filetype="input" or "both", the file or folder must exist. If filetype="output", Write access to the output folder must be enabled.
CHK04	A required look-through to the global standards library defaults fails.	You might elect to leave the path or memname blank in your SASReferences data set, which indicates that you want to use the defaults as specified in the standard-specific StandardSASReferences data set. If the path or memname remains blank (unresolved) after the final call to cstutilvalidatesasreferences in cstutil_allocatesasreferences, this error is reported.
CHK05	One or more discrete character field values cannot be found in the Standardlookup data set.	Columns with discrete values (reftype, type +subtype combinations, iotype, filetype, allowoverwrite) must have values as defined in the standard-specific Standardlookup data set.
CHK06	For the given context, path or memname macro variables are not resolved.	If macro variables are used as part of the path or memname value, they must resolve to an accessible folder or file.
CHK07	Multiple fmtsearch records exist, but valid ordering is not provided.	To properly set the format search path, an unambiguous ordering of multiple type=fmtsearch records must be provided.
CHK08	Multiple autocall records exist, but valid ordering is not provided.	To properly set the autocall path, an unambiguous ordering of multiple type=autocall records must be provided.

The occurrence of any of these errors causes the process to terminate. The rationale is that if the process setup is incomplete, and the SAS Clinical Standards Toolkit cannot recognize a SASReferences column value or find a specified file, the process output might be unreliable. Correct problems reported in the process results data set (as typically defined by the `_cstResultsDS` global macro variable) and resubmit the process.

Sample Driver Modules

Overview

The SAS Clinical Standards Toolkit internal validation addresses two primary use cases:

1 Perform installation qualification and operational qualification.

This is implemented with and illustrated by the use of the `validate_iqoq` sample driver, located in the *sample study library directory/cst-framework-1.5/programs* folder. This is a two-step process:

- a** Select the CST-FRAMEWORK standard, and run the checks that are defined in the `validation_control_glmata` view of the internal validation `validation_master` data set.

This is a set of 64 checks (`checkid < CSTV100`) that look only at the global standards library metadata folder.

- b** Select 1 to n specific standards, and run the checks that are defined in the `validation_control_stdioq` view of the internal validation `validation_master` data set.

This is a set of 30 checks (`checkid > CSTV100` that are relevant to installation qualification and operational qualification issues) that look only at metadata libraries other than the global standards library metadata folder.

2 Perform validation on standard-specific metadata.

This is implemented with and illustrated by the use of the `validate_standard` sample driver. Select 1 to n specific standards, and run the checks that are defined in the `validation_control_std` view of the internal validation `validation_master` data set.

This is a set of 39 checks (`checkid > CSTV100`) that look only at metadata libraries other than the global standards library metadata folder.

The sample drivers that support internal validation are described in the following sections. The SASReferences data set is validated automatically as part of these sample driver programs during the call to the `cstutil_processsetup` macro.

Internal Validation Driver Modules That Are Provided by SAS

A summary of the driver modules that support internal validation, including these two specific use cases, is here:

- `validate_iqoq`

SASReferences: `stdvalidation_sasrefs` (modified in driver)

validation_control files used: `validation_control_glmata` view, `validation_control_stdqiqoq` view, checktype in ('GLMETA' 'STDIQOQ')

Purpose: First, runs checks only on CST-FRAMEWORK global standards library metadata (n=64 checks). Then, runs checks on one or more standards as specified in the driver. Thirty checks are run for each selected standard. These are the checks that support installation qualification and operational qualification for the SAS Clinical Standards Toolkit 1.5.

- `validate_standard`

SASReferences: `stdvalidation_sasrefs` (modified in driver)

validation_control files used: `validation_control_std` view, checktype in ('STD' 'STDIQOQ')

Purpose: Runs checks on one or more standards as specified in the driver. Thirty-nine checks are run for each selected standard.

- `validate_glmadata`

SASReferences: `stdvalidation_sasrefs` (modified in driver)

validation_control files used: `validation_control_glmata` view, checktype in ('GLMETA')

Purpose: Runs checks only on CST-FRAMEWORK global standards library metadata (n=64 checks).

- `validate_data`

SASReferences: `sasreferences`

validation_control files used: `validation_control` data set

Purpose: Runs checks only against CST-FRAMEWORK metadata. The `validation_control` data set is currently the same as the `validation_master` data set that is provided by SAS. Each of these data sets contains 103 checks.

The files are stored in these locations:

- Drivers: *sample study library directory/cst-framework-1.5/programs/<driver>.sas*
- SASReferences: *sample study library directory/cst-framework-1.5/control/<SASReferences>.sas7bdat*
- validation_control: *sample study library directory/cst-framework-1.5/control/<data set of view>*

The `validate_data` driver is similar in functionality to other standard-specific drivers (such as the CDISC-SDTM `validate_data` driver). It runs against a `validation_control` data set with no subsetting by standard or by check. For the simpler workflow, see the `validate_data` driver module in the online API documentation.

A complete discussion of the use of the `validate_iqoq` driver module is provided in *SAS Clinical Data Standards Toolkit: Installation Qualification*, which is available at: <http://support.sas.com/documentation/onlinedoc/clinical/index.html>.

Internal Validation Driver Module Workflow: `validate_standard`

Driver location: *sample study library directory/cst-framework-1.5/programs/validate_standard.sas*

This driver module performs all standard-specific validation checks. This excludes checks that target the *global standards library directory/metadata* folder files. Essentially, this is any check defined in `validation_master`, where checktype NE 'GLMETA'.

Here is the validate_standard driver workflow:

1 Select the standards of interest in work._cstStandardsforIV:

```
*****;
* User defines standard(s) of interest in the following data step      *;
*****;

data work._cstStandardsforIV;
  set work._cstAllStandards (where=(
    (upcase(standard) = 'CDISC-ADAM'          and standardversion='2.1')
    or (upcase(standard) = 'CDISC-CRTDDS'      and standardversion='1.0')
  /*
    or (upcase(standard) = 'CDISC-CT'          and standardversion='1.0.0')
    or (upcase(standard) = 'CDISC-ODM'        and standardversion='1.3.0')
    or (upcase(standard) = 'CDISC-ODM'        and standardversion='1.3.1')
    or (upcase(standard) = 'CDISC-SDTM'       and standardversion='3.1.1')
    or (upcase(standard) = 'CDISC-SDTM'       and standardversion='3.1.2')
    or (upcase(standard) = 'CDISC-SDTM'       and standardversion='3.1.3')
    or (upcase(standard) = 'CDISC-SEND'       and standardversion='3.0')
    or (upcase(standard) = 'CDISC-TERMINOLOGY' and standardversion='NCI_THESAURUS')
    or (upcase(standard) = 'CST-FRAMEWORK'    and standardversion='1.2')
  */
  ));
run;
```

In this example, validation is performed only for the CDISC ADaM and CDISC CRT-DDS standards.

2 Modify the standard validation SASReferences data set to point to the validation_control view of interest.

In the SAS Clinical Standards Toolkit 1.5, views have been provided to make defining the various check subsets more dynamic. Physical SAS data sets can be used, if preferred.

```
*****;
* Modify the sample SASReferences data set to point to the run-time      *;
* validation_control data set identifying the validation checks of interest.*;
*                                                                           *;
* The validation_control_std view of the validation_master data set includes *;
* just those checks specific to one or more standards and excludes those core*;*;
* framework checks that look only within the <cstGlobalLibrary>/metadata   *;
* folder.                                                                   *;
*****;
libname _cstTemp "&studyrootpath/control";
```

```

data work.stdvalidation_sasrefs;
  set _cstTemp.stdvalidation_sasrefs;
  if type='control' and subtype='validation' then
  do;
    filetype='view';
    memname='validation_control_std.sas7bview';
  end;
run;

```

Note: Alternate views might be used. See [“Internal Validation Driver Modules That Are Provided by SAS”](#) on page 236.

- 3 Call the process setup macro to perform all CST-FRAMEWORK file and library allocations.

The returned &_cstSASRefs data set contains fully resolved path and memname values.

```

%cstutil_processsetup(_cstSASReferencesLocation=&workpath,
_cstSASReferencesName=stdvalidation_sasrefs);

```

- 4 (Optional) Re-create work.stdvalidation_sasrefs, and replace _srcfile='STDVAL' with _srcfile='FWVAL'

```

*****;
* work.stdvalidation_sasrefs will accumulate SASReferences records from all *;
* sources for later use by cstvalidate(). *;
*****;

data work.stdvalidation_sasrefs;
  set &_cstSASRefs
    attrib _srcfile format=$8. label='File source for record';
  *****;
  * Framework validation sasreferences: cstcntl.stdvalidation_sasrefs *;
  *****;
  _srcfile='STDVAL';
run;

```

Note: This step is optional because it merely provides an indication of the sources and purposes of specific SASReferences data set records.

- 5 Call the code-generator macro to build the job stream for each standard:

```

filename incCode CATALOG "work._cstCode.stds.source" LRECL=255;

```

```
%cstutilbuildstdvalidationcode(_cstStdDS=work._cstStandardsforIV,
 _cstSampleRootPath=_DEFAULT_, _cstSampleSASRefDSPath=_DEFAULT_,
 _cstSampleSASRefDSName=_DEFAULT_);
```

This macro call populates the `work._cstCode.stds.source` catalog entry with standard-specific code, which is subsequently `%included`. For information about macro parameters, see the `cstutilbuildstdvalidationcode()` macro header comments in the online API documentation.

The workflow of this catalog entry is summarized in the following steps:

- a** Initialize `work._cstTempSASRefDS` to accumulate SASReferences records from all of the standards of interest for later use by `cstvalidate()`.
- b** Look for the standard-specific `StandardSASReferences` data set from the global standards library. If found, run `cstutil_processsetup()` using this data set.
- c** Append the fully resolved `work._cstSASRefs` to the `work._cstTempSASRefDS` that was created in `validate_standard` driver workflow step 1. Set `_srcfile='STD'`.
- d** Look for the standard-specific `sdtvalidation_sasrefs` data set from the sample library. If found, run `cstutil_processsetup()` using this data set.
- e** Append the fully resolved `work._cstSASRefs` to the `work._cstTempSASRefDS` that was created in step a. Set `_srcfile='STUDY'`.
- f** Remove any duplicate records from `work._cstTempSASRefDS` using these key values: `standard`, `standardversion`, `type`, and `subtype`.

This significantly reduces the number of records given the commonalities of SASReferences data sets, but it is assumed that it is irrelevant which record is retained.

- g** Run `%cstutilbuildmetadatafromsasrefs(cstSRefsDS=work._cstTempSASRefDS, cstSrcTabDS=work.source_tables, cstSrcColDS=work.source_columns)`.

This macro dynamically builds `reference_tables` and `reference_columns` data sets from a SASReferences data set. For examples, see [Display 7.1 on page 241](#) and [Display 7.2 on page 242](#).

- h Set `_cstSASRefs=work._cstTempSASRefDS`, which is the cumulative ready-to-go SASReferences data set.
 - i Call `cstvalidate()`, which uses the `validation_control` view specific to the driver focus (in this case, `validation_control_std`) as specified in [“Internal Validation Driver Modules That Are Provided by SAS”](#) on page 236.
 - j Remove standard-specific records from `work._cstTempSASRefDS` to anticipate appending new records for the next standard to the remaining framework records.
- 6 For each standard selected in `validate_standard` driver workflow step 1, repeat steps a through j in step 5.

Results are collated in `cstrslt.validation_results`. For excerpts of the results, see [Display 7.3 on page 243](#).

Display 7.1 Sample of Dynamically Derived `work.reference_tables**`

	sasref	table	path	standard	standardversion	type	subtype
1	CSTCNTL	STDVALIDATION_SASREFS	%studyRootPath/control	CST-FRAMEWORK	1.2	control	reference
2	CSTLKUP	STANDARDLOOKUP	%_cstGRoot/standards/cst-framework-1.5/control	CST-FRAMEWORK	1.2	lookup	
3	CSTMETA	STANDARDLOOKUP	%_cstGRoot/standards/cst-framework-1.5/control	CST-FRAMEWORK	1.2	cstmetadata	lookup
4	CSTMETA	STANDARDS	%_cstGRoot/standards/cst-framework-1.5/control	CST-FRAMEWORK	1.2	cstmetadata	standard
5	CSTMETA	STANDARDSASREFERENCES	%_cstGRoot/standards/cst-framework-1.5/control	CST-FRAMEWORK	1.2	cstmetadata	sasreferences
6	CSTMSG	MESSAGES	%_cstGRoot/standards/cst-framework-1.5/messages	CST-FRAMEWORK	1.2	messages	
7	CSTRCNTL	VALIDATION_MASTER	%_cstGRoot/standards/cst-framework-1.5/validation/control	CST-FRAMEWORK	1.2	referencecontrol	validation
8	GLMETA	STANDARDS	c:/cstGlobalLibrary/metadata	CST-FRAMEWORK	1.2	globalmetadata	standard
9	GLMETA	STANDARDSASREFERENCES	c:/cstGlobalLibrary/metadata	CST-FRAMEWORK	1.2	globalmetadata	sasreferences
10	LOOKUP	STANDARDLOOKUP	%_cstGRoot/standards/cdisc-adam-2.1-1.5/control	CDISC-ADAM	2.1	lookup	
11	MESSAGES	MESSAGES	%_cstGRoot/standards/cdisc-adam-2.1-1.5/messages	CDISC-ADAM	2.1	messages	
12	REFCNTL	VALIDATION_CLASSBYCHECK	%_cstGRoot/standards/cdisc-adam-2.1-1.5/validation/control	CDISC-ADAM	2.1	referencecontrol	checktable
13	REFCNTL	VALIDATION_MASTER	%_cstGRoot/standards/cdisc-adam-2.1-1.5/validation/control	CDISC-ADAM	2.1	referencecontrol	validation
14	REFMETA	CLASS_COLUMNS	%_cstGRoot/standards/cdisc-adam-2.1-1.5/metadata	CDISC-ADAM	2.1	classmetadata	column
15	REFMETA	CLASS_TABLES	%_cstGRoot/standards/cdisc-adam-2.1-1.5/metadata	CDISC-ADAM	2.1	classmetadata	table
16	REFMETA	REFERENCE_COLUMNS	%_cstGRoot/standards/cdisc-adam-2.1-1.5/metadata	CDISC-ADAM	2.1	referencemetadata	column
17	REFMETA	REFERENCE_TABLES	%_cstGRoot/standards/cdisc-adam-2.1-1.5/metadata	CDISC-ADAM	2.1	referencemetadata	table
18	SRCNCNTL	STDVALIDATION_SASREFS	%studyRootPath/control	CDISC-ADAM	2.1	control	reference

Note: **This is an excerpt only. Not all records and columns are shown.

Display 7.2 Sample of Dynamically Derived `work.reference_columns`**

	sasref	table	column	standard	standardversion
122	GLMETA	STANDARDASREFERENCE	standard	CST-FRAMEWORK	1.2
123	GLMETA	STANDARDASREFERENCE	standardversion	CST-FRAMEWORK	1.2
124	GLMETA	STANDARDASREFERENCE	type	CST-FRAMEWORK	1.2
125	GLMETA	STANDARDASREFERENCE	subtype	CST-FRAMEWORK	1.2
126	GLMETA	STANDARDASREFERENCE	SASref	CST-FRAMEWORK	1.2
127	GLMETA	STANDARDASREFERENCE	reftype	CST-FRAMEWORK	1.2
128	GLMETA	STANDARDASREFERENCE	iotype	CST-FRAMEWORK	1.2
129	GLMETA	STANDARDASREFERENCE	filetype	CST-FRAMEWORK	1.2
130	GLMETA	STANDARDASREFERENCE	allowoverwrite	CST-FRAMEWORK	1.2
131	GLMETA	STANDARDASREFERENCE	relpathprefix	CST-FRAMEWORK	1.2
132	GLMETA	STANDARDASREFERENCE	path	CST-FRAMEWORK	1.2
133	GLMETA	STANDARDASREFERENCE	order	CST-FRAMEWORK	1.2
134	GLMETA	STANDARDASREFERENCE	memname	CST-FRAMEWORK	1.2
135	GLMETA	STANDARDASREFERENCE	comment	CST-FRAMEWORK	1.2
136	LOOKUP	STANDARDLOOKUP	SASref	CDISC-ADAM	2.1
137	LOOKUP	STANDARDLOOKUP	table	CDISC-ADAM	2.1
138	LOOKUP	STANDARDLOOKUP	column	CDISC-ADAM	2.1
139	LOOKUP	STANDARDLOOKUP	refcolumn	CDISC-ADAM	2.1
140	LOOKUP	STANDARDLOOKUP	refvalue	CDISC-ADAM	2.1
141	LOOKUP	STANDARDLOOKUP	value	CDISC-ADAM	2.1
142	LOOKUP	STANDARDLOOKUP	default	CDISC-ADAM	2.1
143	LOOKUP	STANDARDLOOKUP	nonnull	CDISC-ADAM	2.1
144	LOOKUP	STANDARDLOOKUP	order	CDISC-ADAM	2.1
145	LOOKUP	STANDARDLOOKUP	templatetype	CDISC-ADAM	2.1
146	LOOKUP	STANDARDLOOKUP	template	CDISC-ADAM	2.1

Note: **This is an excerpt only. Not all records and columns are shown.

Display 7.3 Sample Results Data Set: *validate_standard***

	resultid	checkid	resultseq	seqno	srcdata	message	resultseverity
5	CST0200		1	0	CDISC-ADAM 2.1	PROCESS WORKFLOW: Validating CDISC-ADAM 2.1	Info
9	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-adam-2.1-1.5/programs/initialize.properties	Info
11	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-adam-2.1-1.5/programs/validation.properties	Info
13	CST0200		1	2	CSTUTIL_ALLOCATESASREFERENCES	SASReferences data set was successfully validated	Info
16	CST0200		1	3	CSTUTILBUILDMETADATAFROMSASREFS	Reference metadata was successfully derived from work_cstTempSASRefDS	Info
17	CST0200		1	1	CSTVALIDATE	PROCESS STANDARD: CST-FRAMEWORK	Info
18	CST0200		1	2	CSTVALIDATE	PROCESS STANDARD/VERSION: 1.2	Info
19	CST0200		1	3	CSTVALIDATE	PROCESS DRIVER: validate_standardmetadata.sas	Info
20	CST0200		1	4	CSTVALIDATE	PROCESS DATE: 2013-01-20T18:54:37	Info
21	CST0200		1	5	CSTVALIDATE	PROCESS TYPE: VALIDATION	Info
22	CST0200		1	6	CSTVALIDATE	PROCESS SASREFERENCES: c:/cstSampleLibrary/cst-framework-1.5/control/stdvalidation_sasrefs.sas7bdat	Info
23	CST0200		1	7	CSTVALIDATE	PROCESS VALIDATION CONTROL DATA SET: c:/cstSampleLibrary/cst-framework-1.5/control/validation_control_std.sas7bview	Info
24	CST0200		1	8	CSTVALIDATE	PROCESS STUDYROOTPATH: c:/cstSampleLibrary/cst-framework-1.5	Info
25	CST0200		1	9	CSTVALIDATE	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info
26	CST0200		1	10	CSTVALIDATE	PROCESS STUDYLIBRARY: c:/cstSampleLibrary	Info
27	CST0200		1	11	CSTVALIDATE	PROCESS CSTVERSION: 1.5	Info
28	CST0200		1	10	CSTVALIDATE	PROCESS CONTROLLED TERMINOLOGY SOURCE: c:/cstGlobalLibrary/standards/cdisc-terminology-1.5/cdisc-adam/current/formats/cterns	Info
30	CST0002	CSTV251	1	1	CSTCHECK_COMPAREDOMAINS	No tables evaluated-check validation control data set	Warning: Check not run
31	CST0002	CSTV252	1	1	CSTCHECK_COMPAREDOMAINS	No tables evaluated-check validation control data set	Warning: Check not run
32	CST0002	CSTV252	2	1	CSTCHECK_COMPAREDOMAINS	No tables evaluated-check validation control data set	Warning: Check not run
33	CST0100	CSTV254	1	1	CSTCHECKENTITYNOTFOUND	No errors detected in assessment of target files	Info
34	CST0100	CSTV254	2	1	CSTCNTLSTDVALIDATION_SASREFS	No errors detected in CSTCNTLSTDVALIDATION_SASREFS	Info

Note: **This is an excerpt only. Not all records and columns are shown.

Validation Checks

validation_master Data Set

A total of 103 validation checks are provided in support of internal validation for the SAS Clinical Standards Toolkit 1.5. These can be found in

global standards library directory/standards/cst-framework-1.5/validation/control/validation_master.sas7bdat.

The validation_master data set column checktype is used to specify the primary focus of each check. This table shows the distribution of records by checktype:

Table 7.4 Distribution of Internal Validation Checks by Checktype

Focus	Checktype	Total Number of Checks (Unique)
Global standards library metadata	GLMETA	64 (62)

Focus	Checktype	Total Number of Checks (Unique)
Standard-specific metadata in global standards library and sample library	STDIQOQ	30 (22)
Standard-specific content	STD	9 (1)

The 103 validation checks use 11 of the SAS Clinical Standards Toolkit framework check macros. This table shows the distribution of these checks by check macro:

Table 7.5 *Distribution of Internal Validation Checks by Check Macro*

Check Macro	Number of Records
cstcheck_column	27
cstcheck_columncompare	49
cstcheck_comparedomains	4
cstcheck_dsmismatch	3
cstcheck_notconsistent	1
cstcheck_notincodelist	1
cstcheck_notunique	2
cstcheck_recmismatch	2
cstcheck_recnofound	7
cstcheck_zeroobs	2
cstcheckentitynotfound	5

A review of the validation_master tablescope and columnscope values shows a reference to the dynamically derived table and column metadata that is shown in [Display 7.1 on page 241](#) and [Display 7.2 on page 242](#).

Note: work.source_tables is a copy of the derived work.reference_tables.work.source_columns is a copy of the derived work.reference_columns.

For internal validation, using the SAS libref is usually required in the validation_master tablescope value. Each SAS libref is associated with a specific SAS library through the SASReferences record that identifies the library (or specific SAS file) as an input to the process.

As with all validation check data sets in the SAS Clinical Standards Toolkit, you can add your own checks or modify existing checks to meet your validation requirements.

validation_control SAS Views

As with any SAS Clinical Standards Toolkit validation process, a key step is the specification of a validation_control data set, which is the definition of a subset of defined validation checks that are the focus of that specific validation process. For internal validation, multiple SAS views have been defined against the superset of internal validation checks that are provided with the SAS Clinical Standards Toolkit 1.5.

These SAS views have been created with the code shown in [Example Code 7.1 on page 245](#), where SAS librefs have been defined based on the SASReferences data set references as follows:

```
libname refcntl 'c:/cstGlobalLibrary/standards/cst-framework-1.5/validation/
               control';
libname cstcntl 'c:/cstSampleLibrary/cst-framework-1.5/control';
```

(The SAS Clinical Standards Toolkit global standards library and sample study library have been set to the path that is indicated.)

Note: The SASReferences filetype column should be set to “view”.

Example Code 7.1 SAS Code to Build Internal Validation Views

```
proc sql;
  create view cstcntl.validation_control_glmata
  as select *
```

```
from cstrcntl.validation_master as a
where upcase(a.checktype)="GLMETA";

create view cstcntl.validation_control_std
as select *
from cstrcntl.validation_master as a
where upcase(a.checktype) in ("STD","STDIQOQ");

create view cstcntl.validation_control_stdqiq
as select *
from cstrcntl.validation_master as a
where upcase(a.checktype) in ("STDIQOQ");
quit;
```

The location of the views can vary based on where your global standards library and sample study library are located.

Example Internal Validation Check: CSTV026

Validation check CSTV026 reports the following condition:

Root path does not exist for standard as defined in metadata standards data set

This check reports each instance where the Standards data set column rootpath cannot be found. This value is important to support the use of relative paths, which are indicated by a non-null value in the SASReferences relpathprefix column.

A portion of the check metadata for this check follows:

Display 7.4 Internal Validation Check CSTV026 Metadata from validation_master

	checkid	checkseverity	checktype	codesource	usesourcemetadata	tablescoope	columnscope	codelogic
17	CSTV026	Error	GLMETA	cstcheck_columncompare	N	glmeta.standards	[rootpath][standard]	%cstcheckutilcheckfolder;

Each of the column values shown in [Display 7.4 on page 246](#) is explained in this table:

Table 7.6 Column Descriptions for Internal Validation Check CSTV026**

Column	Value	Description
checkid	CSTV026	Specifies the check identifier used to return the correct message from the CST-FRAMEWORK messages data set.
checkseverity	Error	Specifies that the condition is deemed to be serious, which warrants an Error condition.
checktype	GLMETA	Indicates that this check targets the global standards library metadata folder contents. This check is included in the validation_control_glmata SAS view.
codesource	cstcheck_columncompare	Indicates the check macro to use for processing. All check macros can be found in the primary SAS Clinical Standards Toolkit autocall library.
usesourcemetadata	N	Specifies that the check macro should use work.reference_tables and work.reference_columns to find the tablescope and columnscope values.
tablescope	glmeta.standards	Indicates the specific data set of interest. The SAS libref has been defined in the SASReferences data set (row 8 in Display 7.1 on page 241) and is included in work.reference_tables.
columnscope	[rootpath][standard]	Specifies the two columns of primary interest in glmeta.standards. The syntax matches what is expected by the cstcheck_columncompare check macro.
codeologic	%cstcheckutilcheckfolder;	Uses a new check utility macro included in Display 7.3 on page 243 .

Note: **Not all check metadata columns are described.

8

XML-Based Standards

<i>SAS Support of XML-Based Standards</i>	250
<i>Reading XML Files</i>	251
Overview	251
Basic Workflow	252
Reading CDISC ODM XML Files: odm_read Macro	252
Sample Driver Program: create_sasodm_fromxml.sas	256
Extracting Clinical Data and Reference Data from the SAS Representation of an ODM XML File: odm_extractdomaindata Macro	262
Reading CDISC ODM Controlled Terminology XML Files: ct_read Macro	269
Sample Driver Program: create_sasct_fromxml.sas	272
Creating a Format Catalog and a Controlled Terminology Data Set from the SAS Representation of a CDISC ODM Controlled Terminology XML File: ct_createformats Macro	276
Reading CDISC CRT-DDS define.xml Files: crtdds_read Macro	280
Sample Driver Program: create_sascrtdds_fromxml.sas	282
<i>Writing XML Files</i>	288
Overview	288
Basic Workflow	289
Creating the CDISC CRT-DDS 1.0 define.xml File	289
Sample Driver Program: create_crtdds_from_sdtm.sas	291

Sample Driver Program: create_crtds_define.sas	296
Creating a define.pdf File from the SAS	
Representation of the CDISC CRT-DDS 1.0 Standard	301
Creating a CDISC ODM XML File	304
Sample Driver Program: create_odmxml.sas	305
Validation of XML-Based Standards	309
XML Validation	309
Validating CDISC CRT-DDS 1.0 Files	310
Validating CDISC ODM Files	316
Special Topic: A Round Trip Exercise Involving the CDISC SDTM and CDISC CRT-DDS Standards	322
Overview	322
The Workflow	323
Running Multiple Driver Programs	326
Special Topic: A Round Trip Exercise Involving the CDISC CRT-DDS Standard: Importing and Exporting the define.xml File	327
Overview	327
Sample Driver Program:	
import_sascrtds_fromxml_export_toxml.sas	328
Special Topic: Identifying Unsupported Elements and Attributes in a CDISC ODM File	334
Overview	334
Sample Utility Program: find_unsupported_tags.sas	336

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 file (CDISC ODM 1.3.0, CDISC ODM 1.3.1, CDISC ODM controlled terminology, or CDISC CRT-DDS 1.0 XML [define.xml]) 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.

For ODM 1.3.1, there are 76 data sets containing 352 columns in the SAS representation of the model.

For CDISC CT 1.0, there are 15 data sets containing 73 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 a CDISC CRT-DDS 1.0 XML file, a CDISC ODM 1.3.0 file, a CDISC ODM 1.3.1 XML file, or a CDISC CT XML 1.0 file.

Reading XML Files

Overview

Support of CDISC XML-based standards, such as CDISC CRT-DDS (define.xml) and CDISC ODM, includes the ability to read XML files into SAS data set format. In the SAS Clinical Standards Toolkit, you can read these types of files:

- a CDISC CRT-DDS 1.0 define.xml file that references a CDISC SDTM study (version 3.1.1, 3.1.2, or 3.1.3) or an ADaM 2.1 study
- a CDISC ODM 1.3.0 or CDISC ODM 1.3.1 XML file
- the controlled terminology files as they are published by the NCI in ODM XML format

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

Note: The process for reading ODM XML files is the same for all ODM versions that are supported by the SAS Clinical Standards Toolkit. The process is explained using ODM version 1.3.0.

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-1.3.0-1.5/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 `/referencexml` as `odm.map`. The `odm.map` file is required to process the `cubeXML` file. If it does not exist, then the `odm_read` macro attempts to create one using the ODM reference metadata.

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 1.3.0 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 8.1 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. This parameter is optional.
_cstBuildFmtCat	Build format catalog(s), representing language-specific codelist TranslatedText, as a part of the Read operation. Default=Y (yes), otherwise leave blank. This parameter is optional.
_cstFmtLib	Where catalog(s) are written. This parameter is optional. If not specified, default first to the value derived from SASReferences, then Work.
_cstReplaceFmtCat	Indicates that an existing format catalog by the same name in _cstFmtLib is replaced. This parameter is optional. Values: N Y Default behavior: Y (overwrite existing catalog)
_cstFmtCatPrefix	The prefix to use for catalog names. This parameter is optional. If not specified, default is <standard mnemonic>FmtCat (such as ODMFmtCat). This default will produce an English format catalog name of ODMFmtCat_en.
_cstFmtCatLang	If specified, create a format catalog ONLY for the specified language. This parameter is optional. Example: _cstFmtCatLang=en. If no records exist for the specified language, an empty catalog is created.

Parameter	Description
<code>_cstFmtCatLangOption</code>	The action to take when no language tag is provided in the XML. This parameter is optional. Values: Ignore English Use <code>_cstFmtCatLang</code> . If Ignore, records are ignored (but reported in the SAS log). If English, records are added to the English catalog (default). If Use <code>_cstFmtCatLang</code> , 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 program that is provided by SAS. For reading ODM XML files, this module is `create_sasodm_fromxml.sas`.

The driver program is located at:

sample study library directory/cdisc-odm-1.3.0-1.5/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 93](#).

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 8.2 on](#)

page 257 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=&_cstSRoot/cdisc-odm-&_cstStandardVersion.  
-&_cstVersion
```

```
&studyOutputPath=&_cstSRoot/cdisc-odm-&_cstStandardVersion.  
-&_cstVersion
```

Table 8.2 Key Components of the SASReferences Data Set for the `create_sasodm_fromxml.sas` Macro

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				
externalxml	odmxml	fileref	&studyRootPath/ sourcexml	odm_sample.xml
referencexml	odmmap	fileref	&studyRootPath/ referencexml	odm.map
Output				
sourcedata	srcdata	libref	&studyOutputPath/ derived/data	*.*
sourcemetadata	srcmeta	libref	&studyOutputPath/ derived/metadata	source_ tables.sas7bdat
sourcemetadata	srcmeta	libref	&studyOutputPath/ derived/metadata	source_ columns.sas7bdat
targetdata	trgdata	libref	&studyOutputPath/ derived/formats	
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 `odmmap` 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 8.1 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\vfjans\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
7	CST0200	1	3	ODM_XMLVALIDATE	PROCESS DRIVER: CREATE_SASODM_FROMXML	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\vfjans\AppData\Local\Temp\SAS Temporary Files\TD7372_L72371_cstsasrefs.sas7bdat	Info	0	0
40	CST0200	1	7	ODM_READ	PROCESS STUDYROOTPATH: %sasroot%/../SAS/ClinicalStandardsToolkitODM130/1.4/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\SAS\ClinicalStandardsToolkitODM130\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\SAS\ClinicalStandardsToolkitODM130\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 8.2 Example of Partial Source_Tables Data Set Derived during odm_read

 VIEWTABLE: Srcmeta.Source_tables

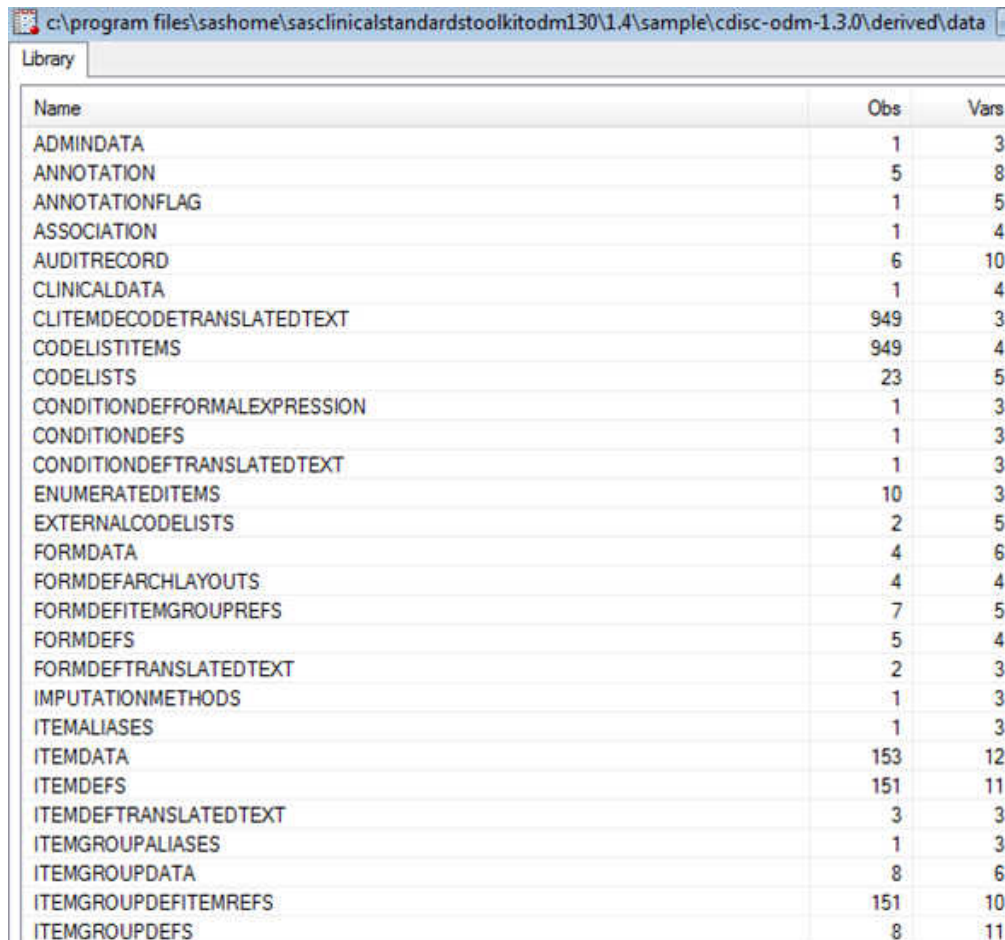
	SASreferences sourcedata	Table Name	Name of Standard	Version of Standard	Case-sensitive XML element name
1	SRCDATA	AdminData	CDISC-ODM	1.3.0	AdminData
2	SRCDATA	Annotation	CDISC-ODM	1.3.0	Annotation
3	SRCDATA	AnnotationFlag	CDISC-ODM	1.3.0	AnnotationFlag
4	SRCDATA	Association	CDISC-ODM	1.3.0	Association
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 8.3 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 for ODM 1.3.0. 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 8.4 Example of Partial Srcdata Library Derived during odm_read


The screenshot shows a SAS Library window titled "c:\program files\sas\home\sasclinicalstandardstoolkitodm130\1.4\sample\cdisc-odm-1.3.0\derived\data". The window displays a table with three columns: "Name", "Obs", and "Vars". The table lists various data sets and their corresponding number of observations and variables.

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
FORMDEFITEMGROUPREFS	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

Extracting Clinical Data and Reference Data from the SAS Representation of an ODM XML File: odm_extractdomaindata Macro

As the primary interchange format for CDISC, ODM XML is a common format for electronic data capture (EDC) data management views of clinical data. This format often does not closely approximate submission (SDTM) and analysis (ADaM) data structures unless the EDC views have been built using the CDISC-CDASH standard. From a SAS perspective, you might want to extract clinical data from an ODM XML file to serve as source data for transformations that derive SDTM domain data sets.

The `odm_extractdomaindata` macro supports extracting clinical data or reference data from the SAS data sets that were created by the `odm_read` macro.

The `odm_extractdomaindata` macro makes the following assumptions:

- An ODM XML file is available that contains sufficient metadata and content for extractable clinical data and reference data.
- A full SAS representation of an ODM XML file is available (for example, the `odm_read` macro has been run against the XML file).

- The SAS representation of an ODM XML file contains both metadata and data.

By default, the driver assumes all source data files reside in the sample derived folder or the data folder that is typically populated by running the `odm_read` macro. However, the source data files and the source metadata files can be in different folders.

- Any codelists defined in the ODM XML file and associated with extracted data set columns are available as part of the output of the `odm_read` macro.

ODM integer and float data types are converted to SAS numeric data. All other ODM data types are converted to SAS character data. If an integer or float data value cannot be converted, a warning appears in the SAS log and Results data set.

Here is a partial listing of the metadata in a sample ODM XML file:

```
<ItemGroupDef OID="ItemGroupDefs.OID.AE" Repeating="Yes"
  SASDatasetName="AE" Name="Adverse Events" Domain="AE"
  Comment="Some adverse events from this trial">
  <ItemRef ItemOID="ID.TAREA"      OrderNumber="1"  Mandatory="No" />
  <ItemRef ItemOID="ID.PNO"        OrderNumber="2"  Mandatory="No" />
  <ItemRef ItemOID="ID.SCTRY"      OrderNumber="3"  Mandatory="No" />
  <ItemRef ItemOID="ID.F_STATUS"   OrderNumber="4"  Mandatory="No" />
  <ItemRef ItemOID="ID.LINE_NO"    OrderNumber="5"  Mandatory="No" />
  <ItemRef ItemOID="ID.AE_TERM"    OrderNumber="6"  Mandatory="No" />
  <ItemRef ItemOID="ID.AE_STMON"   OrderNumber="7"  Mandatory="No" />
  <ItemRef ItemOID="ID.AE_STDAY"   OrderNumber="8"  Mandatory="No" />
  <ItemRef ItemOID="ID.AE_STYR"    OrderNumber="9"  Mandatory="No" />
  <ItemRef ItemOID="ID.AE_STDT"    OrderNumber="10" Mandatory="No" />
  <ItemRef ItemOID="ID.AE_ENMON"   OrderNumber="11" Mandatory="No" />
  <ItemRef ItemOID="ID.AE_ENDAY"   OrderNumber="12" Mandatory="No" />
  <ItemRef ItemOID="ID.AE_ENYR"    OrderNumber="13" Mandatory="No" />
  <ItemRef ItemOID="ID.AE_ENDT"    OrderNumber="14" Mandatory="No" />
  <ItemRef ItemOID="ID.AE_SEV"     OrderNumber="15" Mandatory="No" />
```

```

    <ItemRef ItemOID="ID.AEREL"    OrderNumber="16" Mandatory="No" />
    <ItemRef ItemOID="ID.AEOUT"    OrderNumber="17" Mandatory="No" />
    <ItemRef ItemOID="ID.AEACTTRT" OrderNumber="18" Mandatory="No" />
    <ItemRef ItemOID="ID.AECONTRT" OrderNumber="19" Mandatory="No" />
</ItemGroupDef>
...
<ItemDef OID="ID.AESTDT" SASFieldName="AESTDT"
    Name="Derived Start Date" DataType="date"/>
<ItemDef OID="ID.AEENMON" SASFieldName="AEENMON"
    Name="Stop Month - Enter Two Digits 01-12" DataType="integer" Length="2" />
<ItemDef OID="ID.AEENDAY" SASFieldName="AEENDAY"
    Name="Stop Day - Enter Two Digits 01-31" DataType="integer" Length="2" />
<ItemDef OID="ID.AEENYR" SASFieldName="AEENYR"
    Name="Stop Year - Enter Four Digit Year" DataType="integer" Length="4" />
<ItemDef OID="ID.AEENDT" SASFieldName="AEENDT"
    Name="Derived Stop Date" DataType="date"/>
<ItemDef OID="ID.AESEV" SASFieldName="AESEV"
    Name="Severity" DataType="text" Length="1">
<CodeListRef CodeListOID="CL.$AESEV" />
</ItemDef>
<ItemDef OID="ID.AEREL" SASFieldName="AEREL"
    Name="Relationship to study drug" DataType="text" Length="1">
    <CodeListRef CodeListOID="CL.$AEREL" />
</ItemDef>

```

Here is a partial listing of the data in the same sample ODM XML file:

```

<ClinicalData StudyOID="Study.OID" MetaDataVersionOID="MetaDataVersion.OID.1">
<SubjectData SubjectKey="S001P011" TransactionType="Insert">
    <StudyEventData StudyEventOID="StudyEventDefs.OID.6.AdverseEvent"
        StudyEventRepeatKey="1">
        <FormData FormOID="FormDefs.OID.AE" FormRepeatKey="1">
        <ItemGroupData ItemGroupOID="ItemGroupDefs.OID.AE"
            ItemGroupRepeatKey="1">
            <ItemData ItemOID="ID.TAREA" Value="ONC" />
            <ItemData ItemOID="ID.PNO" Value="143-02" />
            <ItemData ItemOID="ID.SCTRY" Value="USA" />
            <ItemData ItemOID="ID.F_STATUS" Value="V" />
            <ItemData ItemOID="ID.LINE_NO" Value="1" />
            <ItemData ItemOID="ID.AETERM" Value="HEADACHE" />
            <ItemData ItemOID="ID.AESTMON" Value="06" />
            <ItemData ItemOID="ID.AESTDAY" Value="10" />
            <ItemData ItemOID="ID.AESTYR" Value="1999" />
            <ItemData ItemOID="ID.AESTDT" Value="1999-06-10" />
            <ItemData ItemOID="ID.AEENMON" Value="06" />
            <ItemData ItemOID="ID.AEENDAY" Value="14" />
            <ItemData ItemOID="ID.AEENYR" Value="1999" />
            <ItemData ItemOID="ID.AEENDT" Value="1999-06-14" />

```

```

<ItemData ItemOID="ID.AESEV" Value="1" />
<ItemData ItemOID="ID.AEREL" Value="0" />
<ItemData ItemOID="ID.AEOUT" Value="1" />
<ItemData ItemOID="ID.AEACTTRT" Value="0" />
<ItemData ItemOID="ID.AECONTRT" Value="1" />
</ItemGroupData>

```

The `odm_extractdomaindata` macro creates the data set shown in [Display 8.5 on page 265](#) and [Display 8.6 on page 266](#). The first 12 columns in this data set are the data set keys. The macro parameter `_cstODMMinimumKeyset` determines whether these keys are part of the extracted data set.

Display 8.5 AE SAS Data Set (Unformatted) Created by the `odm_extractdomaindata` Macro

Obs	__StudyOID	__MetaDataVersionOID	__SubjectKey	__StudyEventOID	__StudyEventRepeatKey	__FormOID
1	Study.OID	MetaDataVersion.OID.1	S001P011	StudyEventDefs.OID.6.AdverseEvent	1	FormDefs.OID.AE
2	Study.OID	MetaDataVersion.OID.1	S001P011	StudyEventDefs.OID.6.AdverseEvent	1	FormDefs.OID.AE

__FormRepeatKey	__ItemGroupOID	__ItemGroupRepeatKey	__TransactionType	__LocationOID	__UserOID
1	ItemGroupDefs.OID.AE	1	Insert	Location.OID.S001	User.OID.I008
1	ItemGroupDefs.OID.AE	2	Insert	Location.OID.S001	User.OID.I008

TAREA	PNO	SCTRY	F_STATUS	LINE_NO	AETERM	AESTMON	AESTDAY	AESTYR	AESTDT
ONC	143-02	USA	V	1	HEADACHE	6	10	1999	1999-06-10
ONC	143-02	USA	V	2	CONGESTION	6	11	1999	1999-06-11

AEENMON	AEENDAY	AEENYR	AEENDT	AESEV	AEREL	AEOUT	AEACTTRT	AECONTRT
6	14	1999	1999-06-14	1	0	1	0	1
.	.	.	1999	1	0	2	0	1

Display 8.6 AE SAS Data Set (Formatted) Created by the `odm_extractdomaindata` Macro

Obs	__StudyOID	__MetaDataVersionOID	__SubjectKey	__StudyEventOID	__StudyEventRepeatKey	__FormOID
1	Study.OID	MetaDataVersion.OID.1	S001P011	StudyEventDefs.OID.6.AdverseEvent	1	FormDefs.OID.AE
2	Study.OID	MetaDataVersion.OID.1	S001P011	StudyEventDefs.OID.6.AdverseEvent	1	FormDefs.OID.AE

__FormRepeatKey	__ItemGroupOID	__ItemGroupRepeatKey	__TransactionType	__LocationOID	__UserOID
1	ItemGroupDefs.OID.AE	1	Insert	Location.OID.S001	User.OID.I008
1	ItemGroupDefs.OID.AE	2	Insert	Location.OID.S001	User.OID.I008

TAREA	PNO	SCTRY	F_STATUS	LINE_NO	AETERM	AESTMON	AESTDAY	AESTYR	AESTDT
Oncology	143-02	United States	Source verified, queried	1	HEADACHE	6	10	1999	1999-06-10
Oncology	143-02	United States	Source verified, queried	2	CONGESTION	6	11	1999	1999-06-11

AEENMON	AEENDAY	AEENYR	AEENDT	AESEV	AEREL	AEOUT	AEACTTRT	AECONTRT
6	14	1999	1999-06-14	Mild	None	Resolved, no residual effects	None	Medication required
-	-	-	1999	Mild	None	Continuing	None	Medication required

The `odm_extractdomaindata` macro has this signature:

```
%macro odm_extractdomaindata(
  _cstSourceMetadata=,
  _cstSourceData=,
  _cstIsReferenceData=No,
  _cstSelectAttribute=Name,
  _cstSelectAttributeValue=,
  _cstLang=en,
  _cstMaxLabelLength=256,
  _cstAttachFormats=Yes,
  _cstODMMinimumKeyset=No,
  _cstOutputLibrary=,
  _cstOutputDS=
);
```

Here are the parameters:

- `_cstSourceMetadata` and `_cstSourceData` contain the SAS libref for the SAS ODM metadata representation data.
If this is not specified, the macro looks for `type=sourcedata` in `SASReferences`. If this is not provided, the data set source is assumed to be in the SAS Work library.
- `_cstIsReferenceData` indicates whether the data to extract is reference data or clinical data. Examples of reference data are laboratory reference ranges or trial design data.
- `_cstSelectAttribute` contains the `ItemGroup` attribute that identifies which `ItemGroup` to extract. Valid values are `OID`, `Name`, `SASDatasetName`, and `Domain`.
- `_cstSelectAttributeValue` contains the value of the attribute defined by `_cstSelectAttribute` that identifies the `ItemGroup` to extract.
- `_cstLang` specifies a language identifier for the language tag attribute (`xml:lang`) in the ODM `TranslatedText` elements.
- `_cstMaxLabelLength` determines the maximum value of labels to be created.
If this is not provided, 256 is assumed. Formats are attached to the data set variables in case the parameter `_cstAttachFormats` has a value of 'Yes'.
- `_cstODMMinimumKeyset` determines the creation of data set keys. If this is not provided, 'No' is assumed.
- `_cstOutputLibrary` defines the SAS library where the extracted data sets are written.
If this is not specified, the macro looks for `type=targetdata` in `SASReferences`. If this is not provided, the data sets are written to the SAS Work library.
- `_cstOutputDS` contains the name of the extracted data set.
If this is an invalid SAS data set name, an error is generated. If the data set name is not provided, the macro looks for `type=targetdata` in `SASReferences`.

Two sample driver programs for ODM version 1.3.0 are provided by SAS to demonstrate the use of the `odm_extractdomaindata` macro:

```
sample study library directory/cdisc-odm-1.3.0-1.5/
programs/extract_domaindata_all.sas
```

```
sample study library directory/cdisc-odm-1.3.0-1.5/
programs/extract_domaindata.sas
```

Two sample driver programs for ODM version 1.3.1 are provided by SAS to demonstrate the use of the `odm_extractdomaindata` macro:

```
sample study library directory/cdisc-odm-1.3.1-1.5/
programs/extract_domaindata_all.sas
```

```
sample study library directory/cdisc-odm-1.3.1-1.5/
programs/extract_domaindata.sas
```

The `extract_domaindata_all.sas` sample driver programs demonstrate how all data sets can be extracted at once. The following shows a code fragment:

```
filename incCode CATALOG "work._cstCode.domains.source" lrecl=255;

data _null_;
  set srcdata.itemgroupdefs(keep=OID Name IsReferenceData SASDatasetName Domain);
  file incCode;
  length macrocall $400 _cstOutputName $100;

  _cstOutputName=SASDatasetName;
  * If we have to use the Name, Only use letters and digits;
  if missing(_cstOutputName) then _cstOutputName=cats(compress(Name, 'adk'));
  * If first character a digit, prepend an underscore;
  if anydigit(_cstOutputName)=1 then _cstOutputName=cats('_', _cstOutputName);
  * Cut long names;
  if length(_cstOutputName) > 32 then _cstOutputName=substr(_cstOutputName, 1, 32);

  macrocall=cats('%odm_extractdomaindata(_cstSelectAttribute=OID',
                                     ', _cstSelectAttributeValue=', OID,
                                     ', _cstIsReferenceData=', IsReferenceData,
                                     ', _cstMaxLabelLength=256',
                                     ', _cstAttachFormats=Yes',
                                     ', _cstODMMinimumKeyset=No',
                                     ', _cstLang=en',
                                     ', _cstOutputDS=', _cstOutputName, ');');

  put macrocall;
run;

%include incCode;
```

```
filename incCode clear;
```

Reading CDISC ODM Controlled Terminology XML Files: `ct_read` Macro

To read an ODM controlled terminology XML file as published quarterly by NCI, a specialized macro named `ct_read` is available in the CDISC controlled terminology 1.0 standards macros folder. This folder is located at:

```
global standards library directory/standards/cdisc-ct-1.0-1.5/  
macros
```

This macro is referenced from the `create_sasct_fromxml.sas` driver program. For more information, see [“Sample Driver Program: `create_sasct_fromxml.sas`” on page 272](#).

File references and other metadata that are required by the macro are set as global macro variable values. These global macro variable values are set through the framework initialization properties and the CDISC controlled terminology 1.0 initialization properties. Throughout the processing of the `ct_read` macro, the Results data set contains all framework-specific messages and CDISC controlled terminology 1.0-specific messages that were generated during run time.

Based on file references defined in the SASReferences data set, the `ct_read` macro accesses the ODM controlled terminology XML file.

Here is a partial listing of a sample ODM controlled terminology XML file:

Display 8.7 Partial Listing of a Sample ODM Controlled Terminology XML File

```
?xml version="1.0" encoding="UTF-8"?>
ODM xmlns="http://www.cdisc.org/ns/odm/v1.3"
  xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ncioid="http://ncicb.nci.nih.gov/xml/odm/EVS/CDISC"
  xs:schemaLocation="http://www.nci.nih.gov/EVS/CDISC ../schema/controlledterminology1-0-0.xsd"
  FileType="Snapshot" FileOID="CDISC_CT.SDM.2012-12-21"
  Granularity="Metadata" CreationDateTime="2012-12-18T15:58:26"
  AsOfDateTime="2012-12-21T00:00:00"
  ODMVersion="1.3.1">
<Study OID="CDISC_CT.SDM.2012-12-21">
  <GlobalVariables>
    <StudyName>CDISC SDTM Controlled Terminology</StudyName>
    <StudyDescription>CDISC SDTM Controlled Terminology, 2012-12-21</StudyDescription>
    <ProtocolName>CDISC SDTM Controlled Terminology</ProtocolName>
  </GlobalVariables>
  <MetadataVersion OID="CDISC_CT.MetadataVersion.SDM.2012-12-21"
    Name="CDISC SDTM Controlled Terminology"
    Description="CDISC SDTM Controlled Terminology, 2012-12-21">
    <CodeList OID="CL.C66767.ACN" Name="Action Taken with Study Treatment"
      DataType="text" ncioid:ExtCodeID="C66767" ncioid:CodeListExtensible="No">
      <Description>
        <TranslatedText xml:lang="en">Action Taken with Study Treatment</TranslatedText>
      </Description>
      <EnumeratedItem CodedValue="DOSE INCREASED" ncioid:ExtCodeID="C49503">
        <ncioid:CDISCDefinition>An indication that a medication schedule was modified by addition; either by changing the frequency, strength or amount. (NCI)
        </ncioid:CDISCDefinition>
        <ncioid:PreferredTerm>Dose Increased</ncioid:PreferredTerm>
      </EnumeratedItem>
      <EnumeratedItem CodedValue="DOSE NOT CHANGED" ncioid:ExtCodeID="C49504">
        <ncioid:CDISCDefinition>An indication that a medication schedule was maintained. (NCI)
        </ncioid:CDISCDefinition>
        <ncioid:PreferredTerm>Dose Not Changed</ncioid:PreferredTerm>
      </EnumeratedItem>
      <EnumeratedItem CodedValue="DOSE REDUCED" ncioid:ExtCodeID="C49505">
        <ncioid:CDISCDefinition>An indication that a medication schedule was modified by subtraction, either by changing the frequency, strength or amount. (NCI)
        </ncioid:CDISCDefinition>
        <ncioid:PreferredTerm>Dose Reduced</ncioid:PreferredTerm>
      </EnumeratedItem>
      <EnumeratedItem CodedValue="DRUG INTERRUPTED" ncioid:ExtCodeID="C49501">
        <ncioid:CDISCDefinition>An indication that a medication schedule was modified by temporarily terminating a prescribed regimen of medication. (NCI)
        </ncioid:CDISCDefinition>
        <ncioid:PreferredTerm>Drug Interrupted</ncioid:PreferredTerm>
      </EnumeratedItem>
    </CodeList>
  </MetadataVersion>
</Study>
</ODM>
```

After the `ct_read` macro confirms that the ODM controlled terminology XML file exists, a call is made to the SAS DATA step component `JavaObj`. `JavaObj` processing converts the ODM controlled terminology XML file into a cubeXML file through transformations using XSL files and processes.

The cubeXML file is created in the SAS 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 CDISC controlled terminology 1.0 study folder hierarchy (`referencexml/odm.map`). An `odm.map` file is required to process the cubeXML file. If it does not exist, the `ct_read` macro attempts to create one using the CDISC controlled terminology reference metadata.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<SXLEMAP name="CT100" version="1.2">

<TABLE name="CodeLists">
```

```

<TABLE-PATH syntax="XPath">/LIBRARY/CodeLists</TABLE-PATH>
<TABLE-DESCRIPTION>Codelist metadata</TABLE-DESCRIPTION>

<COLUMN name="OID">
  <PATH syntax="XPath">/LIBRARY/CodeLists/OID</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>Unique identifier for this codelist</DESCRIPTION>
  <LENGTH>64</LENGTH>
</COLUMN>
<COLUMN name="Name">
  <PATH syntax="XPath">/LIBRARY/CodeLists/Name</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>CodeList name</DESCRIPTION>
  <LENGTH>128</LENGTH>
</COLUMN>
<COLUMN name="DataType">
  <PATH syntax="XPath">/LIBRARY/CodeLists/DataType</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>CodeList item value data type (integer | float | text | string)</DESCRIPTION>
  <LENGTH>7</LENGTH>
</COLUMN>
<COLUMN name="SASFormatName">
  <PATH syntax="XPath">/LIBRARY/CodeLists/SASFormatName</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>SAS format name</DESCRIPTION>
  <LENGTH>8</LENGTH>
</COLUMN>
<COLUMN name="ExtCodeID">
  <PATH syntax="XPath">/LIBRARY/CodeLists/ExtCodeID</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>Unique numeric code randomly generated by NCI Thesaurus (NCIt)</DESCRIPTION>
  <LENGTH>64</LENGTH>
</COLUMN>
<COLUMN name="CodeListExtensible">
  <PATH syntax="XPath">/LIBRARY/CodeLists/CodeListExtensible</PATH>
  <TYPE>character</TYPE>
  <DATATYPE>character</DATATYPE>
  <DESCRIPTION>Defines if controlled terms may be added to the codelist (Yes | No)</DESCRIPTION>
  <LENGTH>3</LENGTH>
</COLUMN>
<COLUMN name="CDISCSubmissionValue">
  <PATH syntax="XPath">/LIBRARY/CodeLists/CDISCSubmissionValue</PATH>

```

```
<TYPE>character</TYPE>
<DATATYPE>character</DATATYPE>
<DESCRIPTION>Specific value expected for submissions</DESCRIPTION>
<LENGTH>512</LENGTH>
</COLUMN>
```

When the cubeXML file is processed, each of the 15 data sets (such as CodeLists) that are included in the SAS representation of the CDISC controlled terminology model is derived. One input parameter can be specified in the call to the ct_read macro. The parameter offers the option to create source metadata files.

The parameter is shown in this table:

Table 8.3 ct_read Macro Parameter

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.

By default, if a %ct_read() macro call is made with null parameters, source metadata is derived. The target location of the derived metadata files is defined in the SASReferences data set.

Sample Driver Program: create_sasct_fromxml.sas

Overview

Each primary SAS Clinical Standards Toolkit task, such as reading CDISC ODM controlled terminology XML files, is guided by a sample driver program that is provided by SAS. For reading ODM controlled terminology XML files, this driver program is create_sasct_fromxml.sas.

This driver program is located in:

```
sample study library directory/cdisc-ct-1.0-1.5/programs
```

The SASReferences Data Set

As part of each SAS Clinical Standards Toolkit process setup, a valid SASReferences data set is required. The SASReferences data set references the input files that are needed (such as the ODM controlled terminology XML file), the librefs and filenames to use, and the names and locations of the data sets to create. The SASReferences data set can be modified to point to study-specific files.

For more information, see [“SASReferences File” on page 93](#).

In the SASReferences data set, there are two input file references and five output data set references that are key to the successful completion of the driver program. [Table 8.4 on page 273](#) lists these files and data sets. In the sample `create_sasct_fromxml.sas` macro, the following values are set for `&studyRootPath` and `&studyOutputPath`:

`&studyRootPath=sample study library directory/cdisc-ct-1.0-1.5`

`&studyOutputPath=sample study library directory/cdisc-ct-1.0-1.5`

Table 8.4 Key Components of the SASReferences Data Set for the `create_sasct_fromxml.sas` Macro

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				
externalxml	crtxml	fileref	&studyRootPath/ sourcexml/sdtm/201212	sdtm_terminology.xml
referencexml	ctmap	fileref	&studyRootPath/ referencexml	ct-1.0.0.map
Output				
sourcedata	srcdata	libref	&studyOutputPath/data/ sdtm/201212	*.*
results	results	libref	&studyOutputPath/results	read_results_sdtm_2012.sas7bdat

Process Inputs

The metadata type `externalxml` refers to the ODM controlled terminology XML file to read. The filename reference `crtxml` is defined in the `SASReferences` data set. This filename reference is used in the submitted SAS code to refer to the ODM controlled terminology 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 `ctmap` is defined in the `SASReferences` data set. This filename reference is used in the submitted SAS code to refer to the SAS map file. If a path and filename for the map file are not specified, a temporary map file is created as part of the `ct_read` macro processing.

Process Outputs

When the driver program finishes, the `read_results_sdtm_201212` data set is created in the Results library. This data set contains informational messages, warnings, and error messages that were generated by the program.

This display shows an example of the contents of a Results data set that was created while reading the sample ODM controlled terminology XML file as released by NCI that was provided by SAS.

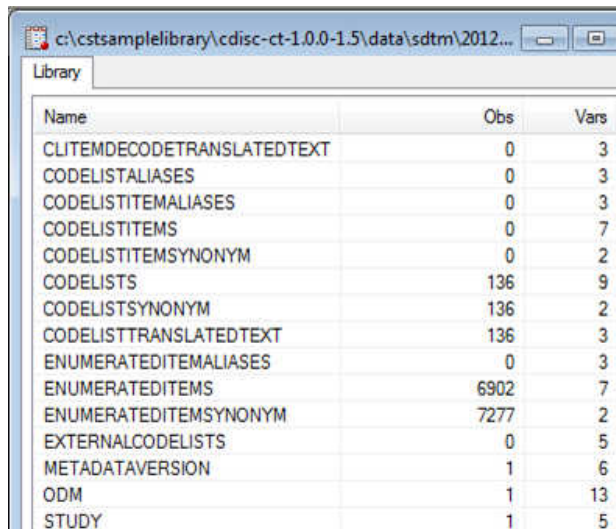
Display 8.8 Example of a Partial Results Data Set Created by the `create_sasct_fromxml.sas` Macro

VIEWTABLE: Results.Read_results_sdtm_201212				
	resultid	srcdata	message	resultseverity
9	CST0200	CT_XMLVALIDATE	PROCESS STANDARDVERSION: 1.0.0	Info
10	CST0200	CT_XMLVALIDATE	PROCESS DRIVER: CREATE_CTXML	Info
11	CST0200	CT_XMLVALIDATE	PROCESS DATE: 2013-01-10T11:22:38	Info
12	CST0200	CT_XMLVALIDATE	PROCESS TYPE: XMLVALIDATE CT	Info
13	CST0200	CT_XMLVALIDATE	PROCESS SASREFERENCES: C:\Users\FRJANS~1\CAR\AppData\Local\Temp\SAS Temporary Files\TD9072_L72371_cstsasrefs.sas7bdat	Info
14	CST0200	CT_XMLVALIDATE	PROCESS STUDYROOTPATH: c:/cstSampleLibraryTK15/cdisc-ct-1.0.0-1.5	Info
15	CST0200	CT_XMLVALIDATE	PROCESS GLOBALLIBRARY: c:/cstGlobalLibraryTK15	Info
16	CST0200	CT_XMLVALIDATE	PROCESS CSTVERSION: 1.5	Info
17	CST0200	JAVA CHECK	No Java issues	Info
18	CT0001	XML TRANSFORMER	Transform starting:	Info
19	CT0001	XML TRANSFORMER	Using JRE: C:\PROGRA~2\Java\JRE16~1.0_2	Info
20	CT0001	XML TRANSFORMER PARAMETER	Import Or Export: EXPORT	Info
21	CT0001	XML TRANSFORMER PARAMETER	Standards XML Path: c:/cstSampleLibraryTK15/cdisc-ct-1.0.0-1.5/sourcexml/sdtm/201212/sdtm_terminology.xml	Info
22	CT0001	XML TRANSFORMER PARAMETER	Fail on Validation Error: false	Info
23	CT0001	XML TRANSFORMER PARAMETER	Standard Name: CDISC-CT	Info
24	CT0001	XML TRANSFORMER PARAMETER	Standard Version: 1.0.0	Info
25	CT0001	XML TRANSFORMER PARAMETER	Schema Repository Location: c:/cstGlobalLibraryTK15/schema-repository	Info
26	CT0001	XML TRANSFORMER PARAMETER	XSL Repository Location: null	Info
27	CT0001	XML TRANSFORMER PARAMETER	Output Encoding: UTF-8	Info
28	CT0001	XML TRANSFORMER PARAMETER	Log File Location: C:\Users\FRJANS~1\CAR\AppData\Local\Temp\SAS Temporary Files\TD9072_L72371_log4774	Info
29	CT0001	XML TRANSFORMER PARAMETER	Header Comment Text: Produced from SAS data using the SAS Clinical Standards Toolkit	Info
30	CT0001	XML TRANSFORMER PARAMETER	Is Validating XML: true	Info
31	CT0001	XML TRANSFORMER PARAMETER	Creating Display Stylesheet: false	Info
32	CT0001	XML TRANSFORMER PARAMETER	Custom Stylesheet: null	Info
33	CT0001	XML TRANSFORMER PARAMETER	Custom Stylesheet Output Shortname: null	Info
34	CT0001	XML TRANSFORMER PARAMETER	Creating Output Folders: true	Info
35	CT0001	XML TRANSFORMER	The document validated successfully	Info
36	CT0115	CSTUTIL_APPENDRESULTDS	No errors were found in the ODM file.	Info
37	CST0102	CSTUTIL_SAVERESULTS	results read_results_sdtm_201212 was created as requested	Info
38	CST0200	CT_READ	PROCESS STANDARD: CDISC-CT	Info
39	CST0200	CT_READ	PROCESS STANDARDVERSION: 1.0.0	Info
40	CST0200	CT_READ	PROCESS DRIVER: CREATE_SASCT_FROMXML	Info
41	CST0200	CT_READ	PROCESS DATE: 2013-01-10T11:22:39	Info
42	CST0200	CT_READ	PROCESS TYPE: FILEIO	Info
43	CST0200	CT_READ	PROCESS SASREFERENCES: C:\Users\FRJANS~1\CAR\AppData\Local\Temp\SAS Temporary Files\TD9072_L72371_cstsasrefs.sas7bdat	Info
44	CST0200	CT_READ	PROCESS STUDYROOTPATH: c:/cstSampleLibraryTK15/cdisc-ct-1.0.0-1.5	Info
45	CST0200	CT_READ	PROCESS GLOBALLIBRARY: c:/cstGlobalLibraryTK15	Info
46	CST0200	CT_READ	PROCESS CSTVERSION: 1.5	Info
47	CST0200	JAVA CHECK	No Java issues	Info
48	CT0013	CT_READ	The CT map file was read from the following location: c:/cstSampleLibraryTK15/cdisc-ct-1.0.0-1.5/referencexml/vct-1.0.0.map	Info

The Srcdata library contains the SAS data sets that represent the ODM controlled terminology XML file metadata and content. By default, the `ct_read` macro creates 15 unique data sets in the SAS Clinical Standards Toolkit. Some of these data sets can be empty if no associated content was derived from the ODM controlled terminology 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 8.9 Example of Partial Srcdata Library Derived from the ct_read Macro



Name	Obs	Vars
CLITEMDECODETRANSLATEDTEXT	0	3
CODELISTALIASES	0	3
CODELISTITEMALIASES	0	3
CODELISTITEMS	0	7
CODELISTITEMSYNONYM	0	2
CODELISTS	136	9
CODELISTSYNONYM	136	2
CODELISTTRANSLATEDTEXT	136	3
ENUMERATEDITEMALIASES	0	3
ENUMERATEDITEMS	6902	7
ENUMERATEDITEMSYNONYM	7277	2
EXTERNALCODELISTS	0	5
METADATAVERSION	1	6
ODM	1	13
STUDY	1	5

Creating a Format Catalog and a Controlled Terminology Data Set from the SAS Representation of a CDISC ODM Controlled Terminology XML File: ct_createformats Macro

To use the NCI CDISC controlled terminology in a SAS Clinical Standards Toolkit process, the SAS data sets created by the ct_read macro must be converted to a SAS format catalog. To enable SAS Clinical Data Integration to import controlled terminology, the SAS data set representation created by the ct_read macro must be combined into one SAS data set.

This display shows an example of controlled terminology in ODM XML (the Action Taken with Study Treatment codelist):

Display 8.10 Example of Controlled Terminology in ODM XML

```
<CodeList OID="CL.C66767.ACN" Name="Action Taken with Study Treatment"
  DataType="text" nciidm:ExtCodeID="C66767" nciidm:CodeListExtensible="No">
  <Description>
    <TranslatedText xml:lang="en">Action Taken with Study Treatment</TranslatedText>
  </Description>
  <EnumeratedItem CodedValue="DOSE INCREASED" nciidm:ExtCodeID="C49503">
    <nciidm:CDISCDefinition>An indication that a medication schedule was modified by addition;
      either by changing the frequency, strength or amount. (NCI)</nciidm:CDISCDefinition>
    <nciidm:PreferredTerm>Dose Increased</nciidm:PreferredTerm>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="DOSE NOT CHANGED" nciidm:ExtCodeID="C49504"> [3 lines]
  <EnumeratedItem CodedValue="DOSE REDUCED" nciidm:ExtCodeID="C49505"> [3 lines]
  <EnumeratedItem CodedValue="DRUG INTERRUPTED" nciidm:ExtCodeID="C49501"> [3 lines]
  <EnumeratedItem CodedValue="DRUG WITHDRAWN" nciidm:ExtCodeID="C49502"> [3 lines]
  <EnumeratedItem CodedValue="NOT APPLICABLE" nciidm:ExtCodeID="C48660"> [5 lines]
  <EnumeratedItem CodedValue="UNKNOWN" nciidm:ExtCodeID="C17998">
    <nciidm:CDISCSynonym>U</nciidm:CDISCSynonym>
    <nciidm:CDISCSynonym>Unknown</nciidm:CDISCSynonym>
    <nciidm:CDISCDefinition>Not known, not observed, not recorded, or refused. (NCI)</nciidm:CDISCDefinition>
    <nciidm:PreferredTerm>Unknown</nciidm:PreferredTerm>
  </EnumeratedItem>
  <nciidm:CDISCSubmissionValue>ACN</nciidm:CDISCSubmissionValue>
  <nciidm:CDISCSynonym>Action Taken with Study Treatment</nciidm:CDISCSynonym>
  <nciidm:PreferredTerm>CDISC SDTM Action Taken with Study Treatment Terminology</nciidm:PreferredTerm>
</CodeList>
```

The `ct_createformats` macro creates the data set shown in this display:

Display 8.11 *Partial cterms SAS Data Set Created by the `ct_createformats` Macro*

VIEWTABLE: Trgdata.Cterms ("CDISC SDTM Controlled Terminology, 2012-12-21")											
	description	codelist	codelist_code	codelist_name	codelist_ext	datatype	type	fmtname	code	cdisc_submission_value	cdisc_synonym
1	CDISC SDTM Controlled Terminology, 2012-12-21	ACN	C66767	Action Taken with Study Treatment	No	text	C	ACN	C49503	DOSE INCREASED	
2	CDISC SDTM Controlled Terminology, 2012-12-21	ACN	C66767	Action Taken with Study Treatment	No	text	C	ACN	C49504	DOSE NOT CHANGED	
3	CDISC SDTM Controlled Terminology, 2012-12-21	ACN	C66767	Action Taken with Study Treatment	No	text	C	ACN	C49505	DOSE REDUCED	
4	CDISC SDTM Controlled Terminology, 2012-12-21	ACN	C66767	Action Taken with Study Treatment	No	text	C	ACN	C49501	DRUG INTERRUPTED	
5	CDISC SDTM Controlled Terminology, 2012-12-21	ACN	C66767	Action Taken with Study Treatment	No	text	C	ACN	C49502	DRUG WITHDRAWN	
6	CDISC SDTM Controlled Terminology, 2012-12-21	ACN	C66767	Action Taken with Study Treatment	No	text	C	ACN	C48660	NOT APPLICABLE	NA: Not Applicable
7	CDISC SDTM Controlled Terminology, 2012-12-21	ACN	C66767	Action Taken with Study Treatment	No	text	C	ACN	C17998	UNKNOWN	U: Unknown

The `ct_createformats` macro uses the data set to create the \$ACN SAS format shown in this display:

Display 8.12 *\$ACN SAS Format Created by the `ct_createformats` Macro*

#	Name	Type	Description
1	ACN	FORMATC	Action Taken with Study Treatment (ACN - C66767)

FORMAT NAME: \$ACN			
LENGTH: 16		NUMBER OF VALUES: 7	
MIN LENGTH: 1	MAX LENGTH: 40	DEFAULT LENGTH: 16	FUZZ: 0
START	END	LABEL (VER. V7 V8 11JAN2013:17:17:33)	
DOSE INCREASED	DOSE INCREASED	DOSE INCREASED	
DOSE NOT CHANGED	DOSE NOT CHANGED	DOSE NOT CHANGED	
DOSE REDUCED	DOSE REDUCED	DOSE REDUCED	
DRUG INTERRUPTED	DRUG INTERRUPTED	DRUG INTERRUPTED	
DRUG WITHDRAWN	DRUG WITHDRAWN	DRUG WITHDRAWN	
NOT APPLICABLE	NOT APPLICABLE	NOT APPLICABLE	
UNKNOWN	UNKNOWN	UNKNOWN	

The `ct_createformats` macro has this signature:

```
%macro ct_createformats(  
  _cstLang=en,           /* Language tag in TranslatedText to use */  
  _cstCreateCatalog=1,   /* Create format catalog */  
  _cstKillCatFirst=0,    /* Empty catalog first */  
  _cstUseExpression=,    /* Expression to create the SAS format name */  
)
```

```

_cstAppendChar=F,          /* Letter to append in case SAS format name
                           ends with digit                */
_cstDeleteEmptyColumns=1, /* Delete columns in output data set that are
                           completely missing              */
_cstTrimCharacterData=1   /* Truncate character data in output data set
                           to the minimum value needed.    */
);

```

The `ct_createformats` macro attempts to map the `CodeList/nciodm:CDISCSubmissionValue` in the `codelist` variable to the `fmtname` variable. The `fmtname` variable value must contain a valid SAS format name. The `ct_createformats` macro uses the following steps to create a valid SAS format name:

- 1 Apply a user-defined expression to create the `fmtname` variable.
- 2 If the value of `fmtname` is empty, use the `CodeList/SASFormatName` attribute (typically empty in NCI EVS ODM XML files).
- 3 If the value of `fmtname` is empty, use the `CodeList/nciodm:CDISCSubmissionValue` value in the `codelist` variable.
- 4 If the value of `fmtname` ends with a digit, add the character specified by the `_cstAppendChar` macro parameter (default=F).

After these steps, the value of the `fmtname` variable is validated against the following regular expression:

```
'm/^(?=.{1,32}$)([\\$a-zA-Z_][a-zA-Z0-9_]*[a-zA-Z_])$/'
```

If the value of the `fmtname` variable fails validation, the `fmtname` variable value does not contain a valid SAS format name. The value is set to missing. Then, the `codelist` is not used to create a SAS format.

Two sample driver programs are provided by SAS to demonstrate the use of the `ct_createformats` macro:

```
sample study library directory/cdisc-ct-1.0-1.5/programs/
create_ctformats.sas
```

```
sample study library directory/cdisc-ct-1.0-1.5/programs/
create_ctformats_qs.sas
```

Both of these sample driver programs demonstrate how the CDISCSubmissionValue can be mapped to a valid SAS format name.

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-1.0-1.5/macros*. This macro is referenced from the create_sascrtdds_fromxml.sas driver program. There are no input parameters in the call to the crtdds_read macro. File references and other metadata that are required by the macro are set as global macro variables. Currently, their values are set through the framework initialization properties and the CDISC CRT-DDS 1.0 initialization properties processes. Throughout processing of the crtdds_read macro, the Results data set contains all framework and CRT-DDS 1.0 specific messages generated during run time.

Based on file references defined in the SASReferences data set, the crtdds_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 `_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 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 `crtds_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 at:

sample study library directory/cdisc-crtdds-1.0-1.5/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 93](#).

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 8.5 on page 283](#) lists these files and data sets, and they are discussed in separate sections. In the sample `create_sascrtdds_fromxml.sas` driver program, these values are set for `&studyRootPath` and `&studyOutputPath` and are specific to a SAS release.

```
&studyRootPath=&_cstSRoot/cdisc-crtdds-1.0-&_cstVersion
```

```
&studyOutputPath=&_cstSRoot/cdisc-crtdds-1.0-&_cstVersion
```

Table 8.5 Key Components of the SASReferences Data Set for the `create_sascrtdds_fromxml.sas` Macro

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				
externalxml	crtxml	fileref	&studyRootPath/sourcexml	define.xml
referencexml	crtmap	fileref	&studyRootPath/ referencexml	define.map
Output				
sourcedata	srcdata	libref	&studyOutputPath/ deriveddata	*.*
sourcemetadata	srcmeta	libref	&studyOutputPath/ derivedmetadata	source_tables. sas7bdat

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
sourcemetadata	srcmeta	libref	&studyOutputPath/ derivedmetadata	source_ columns. sas7bdat
sourcemetadata	srcmeta	libref	&studyOutputPath/ derivedmetadata	source_study. sas7bdat
results	results	libref	&studyOutputPath/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 crtdds_read processing.

Process Outputs

The sourcedata type is the library where the metadata files are created. These metadata files are the data sets that comprise the CRT-DDS information.

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 8.13 Example of a Partial Results Data Set Created by the create_sascrtdds_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_PROCESSSETUP	Process setup is using this SASReferences: C:\Users\vfjans\AppData\Local\Temp\SAS Temporary Files\TD7552_L72371_sasreferences	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: %sasroot%\..\SAS\ClinicalStandardsToolkit\CRTDDS10\1.4/sample\cdisc-crtdd	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
31	CRT0001	1	17	XML TRANSFORMER PARAMETER	Creating Output folder: out	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: %sasroot%\..\SAS\ClinicalStandardsToolkit\CRTDDS10\1.4/sample\cdisc-crtdd	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\SAS\ClinicalStandardsToolkit\CRTDDS10\1.4\sample\cdisc-cr	Info	0	0
45	CRT0012	1	2	CRTDDS_READ	The CRT-DDS file C:\Program Files\SASHome\SAS\ClinicalStandardsToolkit\CRTDDS10\1.4\sample\cdisc-cr was read successfully.	Info	0	0

The `crtds_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 8.14 Example of Partial `Source_Tables` Data Set Derived during `crtds_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	ItemMURRefs	CDISC-CRTDDS	1.0	ItemMURRefs
20	SRCDATA	ItemQuestionExternal	CDISC-CRTDDS	1.0	ItemQuestionExternal
21	SRCDATA	ItemQuestionTranslatedText	CDISC-CRTDDS	1.0	ItemQuestionTranslatedText

Display 8.15 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 8.16 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
FORMDEFITEMGROUPREFS	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\sas\home\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 93.

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, 3.1.2, or 3.1.3 study or an ADaM 2.1 study. You can also create a CDISC ODM 1.3.0 XML file or a CDISC ODM 1.3.1 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_sdtmtodf` 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 19 CRT-DDS data sets.

The `crtds_adamtodefine` macro is similar to the `crtds_sdtmtodefine` macro but uses ADaM table and column metadata.

- 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. Several sample driver programs 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 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.

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.

The driver programs described in this section create a `define.xml` based on SDTM metadata. Similar programs are provided with the SAS Clinical Standards Toolkit for the creation of a `define.xml` based on ADaM metadata.

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, 3.1.2, or 3.1.3 metadata files. (For more information about the `source_tables` and `source_columns` data sets, see [“Source Metadata” on page 126](#).) 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 8.6 Parameters for the `create_crtds_from_sdtm.sas` Macro

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

Parameter	Required	Description
<code>_cstSourceValues</code>	No	The data set that contains the SDTM metadata for the Value Level columns to include in the CRT-DDS file.
<code>_cstSourceDocuments</code>	No	The data set that contains the SDTM metadata for the Document references to include 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,  
  _cstSourceValues=sampdata.source_values,  
  _cstSourceDocuments=sampdata.source_documents,  
  _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 `_cstSourceValues` parameter accesses the `source_values` data set that exists in the `sampdata` library (`sampdata.source_values`). The `_cstSourceDocuments` parameter accesses the `source_documents` data set that exists in the `sampdata` library (`sampdata.source_documents`).

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:

```
sample study library directory/cdisc-crtds-1.0-1.5/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 93](#).

In the SASReferences data set, there are five input file references and one output reference that are key to successful completion of the create_crtds_from_sdtm.sas driver program. [Table 8.7 on page 293](#) lists these files and data sets, and they are discussed in separate sections. In the sample create_crtds_from_sdtm.sas driver program, these values are set for &studyRootPath and &studyOutputPath:

```
&studyRootPath=sample study library directory/cdisc-sdtm-3.1.3-1.5/sascstdemodata
```

```
&studyOutputPath=sample study library directory/cdisc-crtds-1.0-1.5
```

Table 8.7 Key Components of the SASReferences Data Set for the create_crtds_from_sdtm.sas Macro

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				
sourcemetadata	sampdata	libref	&studyRootPath/ metadata	source_tables. sas7bdat
sourcemetadata	sampdata	libref	&studyRootPath/ metadata	source_columns. sas7bdat
sourcemetadata	sampdata	libref	&studyRootPath/ metadata	source_study. sas7bdat
sourcemetadata	sampdata	libref	&studyRootPath/ metadata	source_values. sas7bdat
sourcemetadata	sampdata	libref	&studyRootPath/ metadata	source_documents. sas7bdat

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Output				
sourcedata	srcdata	libref	&studyOutputPath/ data	

Process Inputs

The sourcedata type refers to three data sets that contain the SDTM domain metadata, source_table, source_columns, and source_values. These data sets are stored in the same library. Because the sample create_crtds_from_sdtm.sas driver program provided with the SAS Clinical Standards Toolkit references a source CDISC SDTM 3.1.3 study, the source_tables data set contains SDTM 3.1.3 metadata about each standard domain defined in the CDIC SDTM 3.1.3 Implementation Guide and includes any customizations that you have added. The source_columns type contains similar metadata, but it is at the column level. The source_values data set contains Value Level metadata. This source metadata is read from this location:

```
sample study library directory/cdisc-sdtm-3.1.3-1.5/  
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 8.8 Variables Required in the Source Study Data Set (source_study.sas)

Variable*	Required	Description
StudyName	Yes	The name of the study. This value is used to populate the srcdata.study.studyname column.
DefineDocumentName	Yes	The name of the define document to create. This value is used to populate the srcdata.definedocument.description and srcdata.definedocument.id columns.

Variable*	Required	Description
SASref	Yes	The 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	The name of the protocol for the study. This value is used to populate the srcdata.study.protocolname column.
StudyDescription	Yes	The description of the study. This value is used to populate the srcdata.study.studydescription column. Note: You cannot use commas, semicolons, or quotation marks in the description.
Standard	Yes	The name of the standard in the SAS Clinical Standards Toolkit. (For example, CDISC-SDTM.)
StandardVersion	Yes	The version of the standard in the SAS Clinical Standards Toolkit. (For example, 3.1.3.)
FormalStandard	Yes	The formal name of the standard as used in CRT-DDS. (For example, CDISC SDTM.)
FormalStandardVersion	Yes	The formal version of the standard as used in CRT-DDS. (For example, 3.1.3.)

*All variables are required to be non-blank.

Only a single study can be referenced in the source data sets.

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 *sample study library directory/cdisc-crtds-1.0-*

1.5/data directory. This location is represented in the driver program by the srcdata library name.

Process Results

When the driver program finishes running, the sdtmtodefine_results data set is created. This data set contains informational, warning, and any error messages that were generated by the submitted driver program.

Display 8.17 Example of a Partial Results Data Set from CRT-DDS Sample Study

VIEWTABLE: Results.Sdtmtodefine_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_CRDTDDS_FROM_SDTM	PROCESS STUDYROOTPATH: Isasroot/././SASClinicalStandardsToolkitSDTM312/1.4/sample/cdisc-sdtm-	Info	0	0
12	CST0200	8	CREATE_CRDTDDS_FROM_SDTM	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary	Info	0	0
13	CST0200	9	CREATE_CRDTDDS_FROM_SDTM	PROCESS CSTVERSION: 1.4	Info	0	0
14	CST0200	10	CREATE_CRDTDDS_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 srcdata	Info	0	0
16	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.definestructure was created as requested	Info	0	0
17	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.study was created as requested	Info	0	0
18	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.metadataversion was created as requested	Info	0	0
19	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.itemgroupdefs was created as requested	Info	0	0
20	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.codelist was created as requested	Info	0	0
21	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.codelistitems was created as requested	Info	0	0
22	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.citemdecodetranslatedtext was created as requested	Info	0	0
23	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.itemdefs was created as requested	Info	0	0
24	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.itemgroupdefitemrefs was created as requested	Info	0	0
25	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.itemgroupleaf was created as requested	Info	0	0
26	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.itemgroupleaftitles was created as requested	Info	0	0
27	CST0102	1	CRDTDDS_SDTMTODEFINE	srcdata.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 8.9 *Parameters for the crtdds_write.sas Macro*

Parameter	Required	Description
<code>_cstCreateDisplayStyleSheet</code>	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 <code>referencexml</code> and a subtype of <code>stylesheet</code> , 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.
<code>_cstOutputEncoding</code>	Optional	XML encoding to use for the CRT-DDS file that is created. By default, UTF-8 is used.
<code>_cstHeaderComment</code>	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.
<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 `crtdds_write.sas` macro:

```
%crtdds_write(_cstCreateDisplayStyleSheet=1,
              _cstOutputEncoding=UTF-16,
              _cstResultsOverrideDS=&_cstResultsDS);
```

In this example, a default style sheet is generated in the same directory as the XML output based on the information in the SASReferences data set. XML encoding is set to UTF-16, and process results are written to the default `&_cstResultsDS` data set.

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:

sample study library directory/cdisc-crtds-1.0-1.5/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 309](#).

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 93](#).

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 8.10 on page 299](#) 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=sample study library directory/cdisc-crtds-1.0-1.5
```

```
&studyOutputPath=sample study library directory/cdisc-crtds-1.0-1.5
```

Table 8.10 Key Components of the SASReferences Data Set for the crtdds_write.sas Macro

Metadata Type	LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				
control	control	libref	&workpath	sasreferences.sas7bdat
sourcedata	srcdata	libref	&studyRootPath/data	
referencexml	odmmmap	fileref	&studyRootPath/ referencexml	define.map
Output				
referencexml	xslt01	filename	&studyOutputPath/ sourcexml	define-v1-updated- html.xml
results	results	LIBNAME	&studyOutputPath/ results	write_results.sas7bdat
externalxml	extxml	filename	&studyOutputPath/ sourcexml	define.xml

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_crtdds_from_sdtm.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 *sample study library directory/cdisc-crtdds-1.0-1.5/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 *sample study library directory/cdisc-crtdds-1.0-1.5/sourcexml* directory.

The referencexml type can serve as either an input or output file reference. Because the path and filename are not provided, the crtdds_write macro interprets the _cstCreateDisplayStyleSheet=1 parameter to use the default style sheet that is provided by the SAS Clinical Standards Toolkit in the global standards library. Had a path and filename been provided, the referencexml type would serve as an output file reference for the crtdds_write macro to copy the default style sheet from the global standards 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 *sample study library directory/cdisc-crtdds-1.0-1.5/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 8.18 Example of a Partial Results Data Set from the CRT-DDS Sample Study

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)
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.xsl	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 a define.pdf File from the SAS Representation of the CDISC CRT-DDS 1.0 Standard

The *CDER Data Standards Common Issues Document* (Version 1.1/December 2011) states:

“A critical component of data submission is the define file. A properly functioning define.xml file is an important part of the submission of standardized electronic datasets and should not be considered optional. As a transition step, CDER prefers that sponsors submit both the define.pdf and define.xml formats. The define.pdf is primarily for printing purposes and need not include hyperlinks. CDER will advise when it is ready to only receive define.xml.”

The SAS Clinical Standards Toolkit has a macro that supports the creation of a define.pdf file from the SAS representation of a CDISC CRT-DDS 1.0 standard. This macro is called `crtds_writepdf` and is located at:

global standards library directory/standards/cdisc-crtds-1.0-1.5/macros

The `crtds_writepdf` macro supports the creation of a define.pdf file for the CDISC ADaM, SDTM, and SEND standards. The contents of the sections (which attributes are printed) is based on the Study Data Tabulation Model Metadata Submission Guidelines (SDTM-MSG) (<http://www.cdisc.org/sdtm>, 2011-12-31).

The define.pdf file has an optional table of contents and these sections:

- Dataset level metadata
- Variable level metadata
- Value level metadata
- Algorithms (Computational Methods)
- Controlled Terminology

These are the most important parameters for the `crtds_writepdf` macro:

- `_cstCDISCStandard`
The CDISC standard for which the `define.pdf` is created. Valid values: SDTM, SEND, and ADAM. The default is SDTM.
- `_cstSourceLib`
The library that contains the CRT-DDS SAS data sets. If not provided, the code looks in `SASReferences` for `type=sourcedata`.
- `_cstReportOutput`
The name of the PDF to create. If not provided, the code looks in `SASReferences` for `type=report`.
- `_cstLinks`
Indicates whether the macro creates internal hyperlinks in the PDF. Valid values: Y or N. The default is N.
- `_cstTOC`
Indicates that the macro creates a table of contents in the PDF. Valid values: Y or N. The default is N.

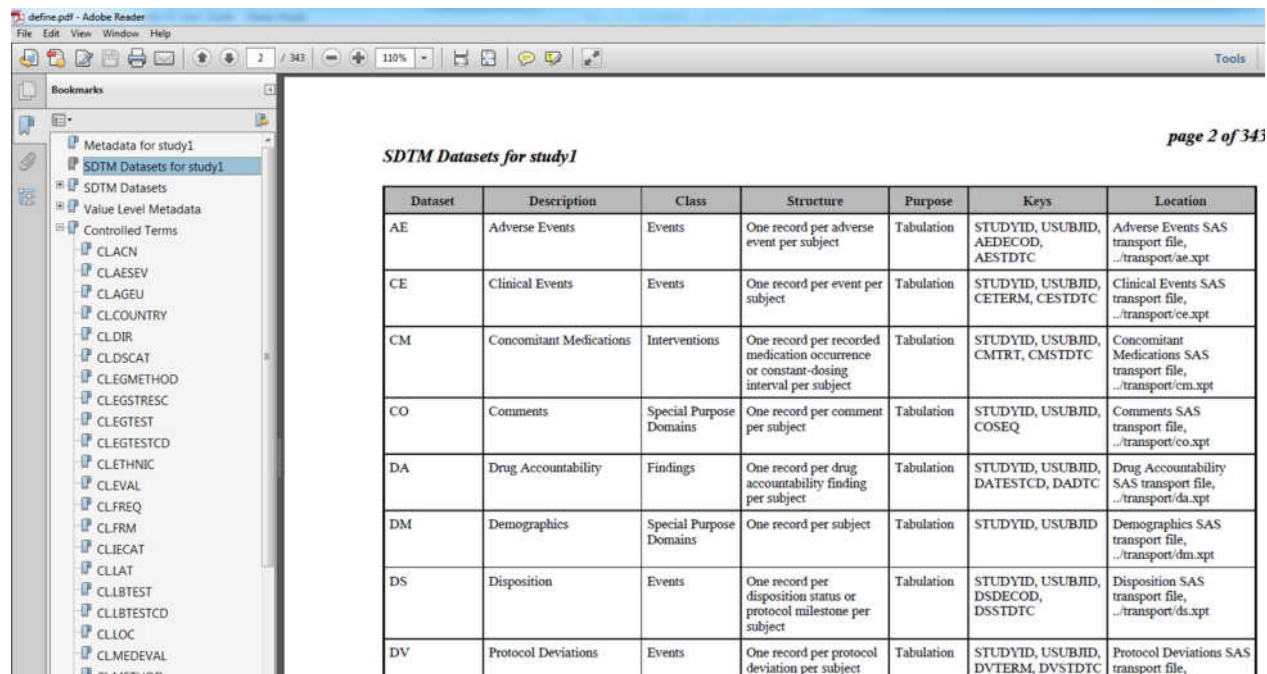
Two sample driver programs are provided by SAS to demonstrate the use of the `crtds_writepdf` macro:

```
sample study library directory/cdisc-crtds-1.0-1.5/programs/
create_crtds_define_pdf.sas
```

```
sample study library directory/cdisc-crtds-1.0-1.5/programs/
create_crtds_define_pdf_adam.sas
```

These displays show examples of `define.pdf` files that were created by the `crtds_writepdf` macro.

Display 8.19 Example define.pdf File for SDTM



SDTM Datasets for study1

Dataset	Description	Class	Structure	Purpose	Keys	Location
AE	Adverse Events	Events	One record per adverse event per subject	Tabulation	STUDYID, USUBJID, AEDECOD, AESTDTC	Adverse Events SAS transport file, .../transport/ae.xpt
CE	Clinical Events	Events	One record per event per subject	Tabulation	STUDYID, USUBJID, CETERM, CESTDTC	Clinical Events SAS transport file, .../transport/ce.xpt
CM	Concomitant Medications	Interventions	One record per recorded medication occurrence or constant-dosing interval per subject	Tabulation	STUDYID, USUBJID, CMTRT, CMSTDTC	Concomitant Medications SAS transport file, .../transport/cm.xpt
CO	Comments	Special Purpose Domains	One record per comment per subject	Tabulation	STUDYID, USUBJID, COSEQ	Comments SAS transport file, .../transport/co.xpt
DA	Drug Accountability	Findings	One record per drug accountability finding per subject	Tabulation	STUDYID, USUBJID, DATESTCD, DADTC	Drug Accountability SAS transport file, .../transport/da.xpt
DM	Demographics	Special Purpose Domains	One record per subject	Tabulation	STUDYID, USUBJID	Demographics SAS transport file, .../transport/dm.xpt
DS	Disposition	Events	One record per disposition status or protocol milestone per subject	Tabulation	STUDYID, USUBJID, DSDECOD, DSSTDTC	Disposition SAS transport file, .../transport/ds.xpt
DV	Protocol Deviations	Events	One record per protocol deviation per subject	Tabulation	STUDYID, USUBJID, DVTERM, DVSTDTC	Protocol Deviations SAS transport file,

Display 8.20 Example define.pdf File for ADaM

define_adam.pdf - Adobe Reader

File Edit View Window Help

19 / 28 111%

Tools

Bookmarks

Metadata for Derived Study built by SAS Clinical Standards Toolkit

Analysis Datasets for Derived Study built by SAS Clinical Standards Toolkit

Analysis Datasets

Adverse Event Analysis Dataset (ADAE)

ADaM Questionnaire Analysis Data Set (ADQS)

Subject-Level Analysis Data Set (ADSL)

Time to Event Analysis Dataset (ADTTE)

Parameter Value Level Metadata

VLADQS.AVAL

Controlled Terms

CLACN

CLAESEV

CLAGEU

CLNY

CLRACE

CLSEX

CLTIMEFL

Analysis Derivations

Analysis Derivations

page 19 of 28

Parameter Value Level Metadata - Parameter Value List VLADQS.AVAL

Source Variable	Where PARAMCD=	Where PARAM=	Type	Display Format	Controlled Terms or Format	Comments / Derivations
AVAL	ACITM01	Word Recall Task	integer			Origin:Derived Derivation:QS.QSSTRESN where QSTESTCD=PARAMCD
AVAL	ACITM02	Naming Objects and Fingers	integer			Origin:Derived Derivation:QS.QSSTRESN where QSTESTCD=PARAMCD
AVAL	ACITM03	Delayed Word Recall	integer			Origin:Derived Derivation:QS.QSSTRESN where QSTESTCD=PARAMCD
AVAL	ACITM04	Commands	integer			Origin:Derived Derivation:QS.QSSTRESN where QSTESTCD=PARAMCD
AVAL	ACITM05	Constructional Praxis	integer			Origin:Derived Derivation:QS.QSSTRESN where QSTESTCD=PARAMCD
AVAL	ACITM06	Ideational Praxis	integer			Origin:Derived Derivation:QS.QSSTRESN where QSTESTCD=PARAMCD

Creating a CDISC ODM XML File

Note: The process to create a CDISC ODM XML files is the same for all ODM versions that are supported by the SAS Clinical Standards Toolkit. The process is explained using ODM version 1.3.0.

There are several key macros that are provided with the SAS Clinical Standards Toolkit that support the creation of an 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 programs 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 8.11 Parameters for the `odm_write.sas` Macro

Parameter	Required	Description
<code>_cstCreateDisplayStyleSheet</code>	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 <code>SASReferences</code> file for a record with a type and subtype of <code>referencexml</code> and <code>stylesheet</code> and uses that file. If the value is 0, then the macro does not create the XSL, even if one is specified in the <code>SASReferences</code> file. The default setting is 0.
<code>_cstOutputEncoding</code>	Optional	XML encoding to use for the ODM XML file that is created. By default, UTF-8 is used.
<code>_cstHeaderComment</code>	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.
<code>_cstResultsOverrideDS</code>	Optional	Provides the opportunity to designate <code>[LIBNAME.]member</code> 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 `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:

sample study library directory/cdisc-odm-1.3.0-1.5/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 93](#).

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 8.12 on page 307](#) 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=sample study library directory/cdisc-odm-1.3.0-1.5
```

```
&studyOutputPath=sample study library directory/cdisc-odm-1.3.0-1.5
```

Table 8.12 Key Components of the SASReferences Data Set for the odm_write.sas Macro

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				
sourcedata	srcdata	libref	&studyRootPath/data	
Output				
results	results	libref	&studyOutputPath/results	write_results.sas7bdat

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
externalxml	extxml	filename	&studyOutputPath/ 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 *sample study library directory/cdisc-odm-1.3.0-1.5/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 *sample study library directory/cdisc-odm-1.3.0-1.5/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:

sample study library directory/cdisc-odm-1.3.0-1.5/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 8.21 Example of a Partial Results Data Set from the ODM Sample Data Hierarchy

	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: lsasroot/.../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/rjfans/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\SASClinicalStandardsToolkit\OD	Info	0	0

Validation of XML-Based Standards

XML Validation

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

The first methodology is described in [Chapter 6, “Compliance Assessment Against a Reference Standard,” on page 115](#). 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 and 1.3.1 files.

Validating CDISC CRT-DDS 1.0 Files

The `crtds_xmlvalidate` Macro

The `crtds_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 `crtds_xmlvalidate` macro immediately following the call to the `crtds_write` macro as the last step of the `create_crtds_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 `crtds_xmlvalidate` macro immediately before the call to the `crtds_read` macro in the `create_crtds_fromxml.sas` sample driver program.

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

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

In this example, the `%crtds_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 8.13 Parameters for the *crtdds_xmlvalidate.sas* Macro

Parameter	Required	Description
<code>_cstLogLevel</code>	Yes	Identifies the log level. Valid values are Info, Warning, Error, and Fatal Error. The default value is Info.
<code>_cstResultsOverrideDS</code>	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 <code>&_cstResultsDS</code> 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 8.14 Log Levels for the *crtdds_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, “Compliance Assessment Against a Reference Standard,” on page 115](#). Metadata about each check is provided in the Validation Master data set in *global standards library directory/standards/cdisc-crtds-1.0-1.5/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 `crtds_validate` macro.

Sample Driver Program: validate_crtds_data.sas

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

The driver program is located in:

```
sample study library directory/cdisc-crtds-1.0-1.5/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 93.

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 8.15 on page 313 lists these libraries and data sets, and they are discussed in separate sections. In the sample validate_crtds_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=sample study library directory/cdisc-crtds-1.0-1.5
&studyOutputPath=sample study library directory/cdisc-crtds-1.0-1.5
```

Table 8.15 Key Components of the SASReferences Data Set for the validate_crtds_data.sas Macro

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
control	cntl_s	libref	&workpath	sasreferences.sas7bdat
control	cntl_v	libref	&studyRootPath/ control	validation_control. sas7bdat
sourcemetadata	srcmeta	libref	&studyRootPath/ metadata	source_tables.sas7bdat
sourcemetadata	srcmeta	libref	&studyRootPath/ metadata	source_columns. sas7bdat
sourcedata	srcdata	libref	&studyRootPath/ data	
Output				
results	results	libref	&studyOutputPath/ results	validation_results. sas7bdat

Process Inputs

The use of the cntl_s LIBNAME that points to the &workpath path illustrates a technique of documenting the derivation of the SASReferences data set in the SAS Work library. The driver program initiates the macro variable &workPath with 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 *sample study library directory/cdisc-crtdds-1.0-1.5/control* directory.

The sourcemetadata type references two metadata data sets that describe the table (source_tables) and column (source_columns) metadata for the 39 data sets that comprise the SAS representation of the CRT-DDS model. Both data sets are stored in the same library. In the SAS Clinical Standards Toolkit, this source metadata is read from the *sample study library directory/cdisc-crtdds-1.0-1.5/*

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 *sample study library directory/cdisc-crtdds-1.0-1.5/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 8.22 Example of a CDISC CRT-DDS Results Data Set

VIEWTABLE: Results.Validation_results										
	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.CLItemDecodeTranslatedText 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 Files

XML Schema Validation

Note: The process for validating ODM XML files is the same for all ODM versions that are supported by the SAS Clinical Standards Toolkit. The process is explained using ODM version 1.3.0.

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

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)
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\source	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\schemas-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/fjans/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 8.16 *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.
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, “Compliance Assessment Against a Reference Standard,” on page 115](#). Metadata about each check is provided in the Validation Master data set in *global standards library directory/standards/cdisc-odm-1.3.0-1.5/validation/control1*. 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:

```
sample study library directory/cdisc-odm-1.3.0-1.5/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 93](#).

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 8.17 on page 319](#), 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=sample study library directory/cdisc-odm-1.3.0-1.5
&studyOutputPath=sample study library directory/cdisc-odm-1.3.0-1.5
```

Table 8.17 Key Components of the SASReferences Data Set for the validate_odm_data.sas Macro

Metadata Type	LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				

Metadata Type	LIBNAME or Fileref to Use	Reference Type	Path	Name of File
control	cntl_v	libref	&studyRootPath/ control	validation_ control.sas7bdat
sourcemetadata	srcmeta	libref	&studyRootPath/ metadata	source_tables.sas7bdat
sourcemetadata	srcmeta	libref	&studyRootPath/ metadata	source_ columns.sas7bdat
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 *sample study library directory/cdisc-odm-1.3.0-1.5/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 *sample study library directory/cdisc-odm-1.3.0-1.5/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 *sample study library directory/cdisc-odm-1.3.0-1.5/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 8.24 Example of a CDISC ODM Validation Results Data Set

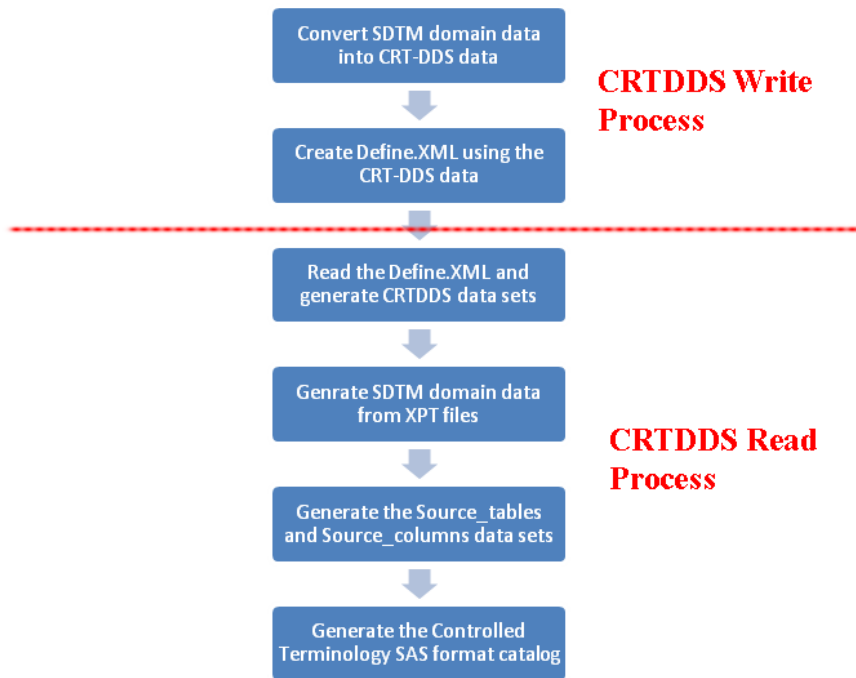
VIEWTABLE: Results.Validation_results									
	Result identifier	Validation check identifier	Unique invocation of resultid	Source data	Resolved message text from message file	Result severity (e.g., warning, error)	Problem detected? (0=no, otherwise yes)	Process status (Non-zero, aborted)	Actual value 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.ITEMDEFTRANSLATEDT (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 re-creating the original source SDTM study. Re-creating 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 `crtds_write` and `crtds_read` SAS Clinical Standards Toolkit macros and allows a comparison of original and re-created SDTM metadata and data. This display details the high-level workflow for this exercise.

Figure 8.1 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_sdmtodefine` 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 *sample study library directory/cdisc-crtds-1.0-1.5/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 *sample study library directory/cdisc-crtds-1.0-1.5/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 re-created using the XML read process.

- 6 Submit the `create_sasrtds_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 `crtds_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 *sample study library directory/cdisc-crtds-1.0-1.5/deriveddata* directory. This driver program generates the `source_tables` and `source_columns` data sets in the *sample study library directory/cdisc-crtds-1.0-1.5/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 *sample study library directory/cdisc-sdm-3.1.3-1.5/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 *sample study library directory/cdisc-crtdds-1.0-1.5/transport* directory. The generated data sets are written to the *sample study library directory/cdisc-sdtm-3.1.3-1.5/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 *sample study library directory/cdisc-sdtm-3.1.3-1.5/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 *sample study library directory/cdisc-sdtm-3.1.3-1.5/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 *sample study library directory/cdisc-*

`sdtm-3.1.3-1.5/sascstdemodata/programs` directory. The driver program accesses the `sdtmutil_createformatsfromcrtds` 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 *sample study library directory/cdiscsdtm-3.1.3-1.5/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-1.5/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 `crtds_sdtmtodefine` macro.

Depending on the completeness of this source data, the crtdds_sdtmtodefine macro can (partially) populate these 19 of the 39 CRT-DDS SAS tables:

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

The externalcodelists table will not be automatically populated by the SAS Clinical Standards Toolkit. The remaining tables are typically not used for a define.xml in the context of an electronic data submission.

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:

sample study library directory/cdisc-crtdds-1.0-1.5/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 `crtds_xmlvalidate` macro to validate the CRT-DDS file (`define_import.xml`) to be imported.

- 3 Call the `crtds_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 `crtds_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 `crtds_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 93](#).

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

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

[Table 8.18 on page 330](#) and [Table 8.19 on page 331](#) 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=sample study library directory/cdisc-crtdds-1.0-1.5
```

```
&studyOutputPath=sample study library directory/cdisc-crtdds-1.0-1.5
```

Table 8.18 Key Components of the SASReferences Data Set for the `import_sasreference` Macro

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

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
sourcemetadata	srcmeta	libref	&workpath	source_columns.sas7bdat
sourcemetadata	srcmeta	libref	&workpath	souce_study
results	results	libref	&studyOutputPath/results	import_results.sas7bdat

Table 8.19 Key Components of the SASReferences Data Set for the export_sasreferences Macro

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				
sourcedata	srcdata	libref	&workpath	*.*
Output				
externalxml	crtxml	fileref	&studyRootPath/sourcexml	define_export.xml
referencexml	xslt01	fileref	&studyRootPath/referencexml	
results	results	libref	&studyOutputPath/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	metadataversion
codelistitems	study

codelists	annotatedcrfs
computationmethods	supplementaldocs
definedocument	mdvleaf
itemdefs	mdvleaftitles
itemgroupdefitemrefs	itemvaluelistrefs
itemgroupdefs	valuelists
itemgroupleaf	valuelistitemrefs
itemgroupleaftitles	

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 8.25 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.FXC.I.USION.IFTEST	FXCI 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.IF.IFCAT	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.PULSE	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

Note: The process explained below is the same for all ODM versions that are supported by the SAS Clinical Standards Toolkit. The process is explained using ODM version 1.3.0.

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 8.20 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>

Parameter	Required	Description
<code>_cstxmlattrds</code>	No	Libref.dataset for file attributes. Default=work.cstodmattributes

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:

sample study library directory/cdisc-odm-1.3.0-1.5/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 93](#).

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 8.21 on page 337](#) 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=sample study library directory/cdisc-odm-1.3.0-1.5
&studyOutputPath=sample study library directory/cdisc-odm-1.3.0-1.5
```

Table 8.21 Key Components of the SASReferences Data Set for the find_unsupported_tags.sas Macro

Metadata Type	SAS LIBNAME or Fileref to Use	Reference Type	Path	Name of File
Input				
externalxml	odmxml	fileref	&studyRootPath/ sourcexml	odm_extended.xml
standardmetadata (element)	odmmeta	libref		
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-1.3.0-1.5/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 8.26 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 *sample study library directory/cdisc-odm-1.3.0-1.5/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 8.27 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

9

Working with CDISC ADaM Data

Overview	341
SAS Representation of CDISC ADaM Metadata	342
ADaM Data Set Templates	353
Validation of ADaM Data Sets	354
Overview	354
Specific Check Implementation Details	355
Unique Validation Properties	356
Validation Check Macros	357
Cross-Standard Validation Checks	357
Sample Data for Validation and Reporting	358
Validation Results	359
Sample Reporting Methodology	361
Overview	361
TLF Metadata	364
Analysis Programs	367
Analysis Results (Tables, Listings, and Figures)	370
Analysis Results Metadata	371

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 342](#).
- 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 ADaM Adverse Event (ADAE) data set.

Note: Templates for additional ADaM data structures will be provided in future releases after the CDISC ADaM team approves them for use.

- Implementation of version 1.2 CDISC ADaM validation checks as prepared by the CDISC ADaM team.

In addition, SAS has provided validation checks for the ADAE and ADaM Time-to-Event (ADTTE) domains. These validation checks are derived from individual implementation guides provided by CDISC. For the ADAE domain, the release of the implementation guide is *Analysis Data Model (ADaM) Data Structure for Adverse Event Analysis, Version 1.0*. For the ADTTE domain, the release of the implementation guide is *ADaM Basic Data Structure for Time-to-Event Analyses, Version 1.0*.

- 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 (ADaM), Version 2.1* document and the *Analysis Data Model (ADaM) 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 9.1 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-2.1-1.5/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 9.2 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 (that is, 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 (that is, XPT file)	xmlpath
DATASET STRUCTURE	The level of detail represented by individual records in the dataset (for example,, “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 (that is, “ADAE”, “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 might be provided. Links or references to external documents (for example, protocol, statistical analysis plan, software code) might be used.	documentation

**Source: *Analysis Data Model (ADaM), Version 2.1*, Section 5.1, Analysis Dataset Metadata, Table 5.1.1

The reference_tables data set provided with the SAS Clinical Standards Toolkit 1.5 contains three records for the ADaM ADAE data set, ADaM ADSL data set, and a representative ADaM BDS 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.

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-2.1-1.5/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 9.3 *Analysis Variable Metadata*

Analysis Variable Metadata Field**	Description**	reference_columns Column Mapping
DATASET NAME	The filename of the analysis dataset	table
VARIABLE NAME	The name of the variable	column
VARIABLE LABEL	A brief description of the variable	label
VARIABLE TYPE	The variable type. Valid values are as defined in the Case Report Tabulation Data Definition Specification Standard (for example, in version 1.0.0 they include “text,” “integer,” and “float”)	xmldatatype
DISPLAY FORMAT	The variable display information (that is, 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 application used for analysis (for example, \$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

Analysis Variable Metadata Field**	Description**	reference_columns Column Mapping
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 (for example, ADSL.AGEGR) identifying the data file and variable that is the source of the variable (that is, a variable copied with no change). It can be a simple description of a derivation and the variable used in the derivation (for example, “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.	origin comment (supplemented by origin and algorithm from the source metadata, such as SDTM)

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

The reference_columns data set provided with the SAS Clinical Standards Toolkit 1.5 contains one record for each column in each of the three data sets (ADSL, BDS, and ADAE) in the reference_tables data set. This results in 63 records (columns) for ADSL, 142 records (columns) for BDS, and 85 records (columns) for the ADAE data set.

Core reference_columns metadata for each column is in the *Analysis Data Model (ADaM) Implementation Guide, Version 1.0*. [Figure 9.1 on page 347](#) provides an excerpt of ADSL column metadata as itemized in Table 3.1.1 of the *Analysis Data*

Model (ADaM) Implementation Guide, Version 1.0. This metadata has been translated into the SAS representation of ADSL as shown in [Figure 9.2 on page 347](#).

Figure 9.1 ADSL Columns as Specified in the Analysis Data Model (ADaM) Implementation Guide

Variable Name	Variable Label	Type	Codelist / Controlled Terms	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 9.2 ADSL Columns as Defined in reference_columns Data Set

sasref	table	column	label	class	order	type	length	displayformat	xmldatatype	xmlcodelist	core	role	term
REFMETA	ADSL	STUDYID	Study Identifier	ADSL	1	C	40		text		Req	StudyIdentifier	
REFMETA	ADSL	USUBJID	Unique Subject Identifier	ADSL	2	C	40		text		Req	StudyIdentifier	
REFMETA	ADSL	SUBJID	Subject Identifier for the Study	ADSL	3	C	40		text		Req	StudyIdentifier	
REFMETA	ADSL	SITEID	Study Site Identifier	ADSL	4	C	40		text		Req	StudyIdentifier	
REFMETA	ADSL	SITEGRy	Pooled Site Group y	ADSL	5	C	80		text		Perm	StudyIdentifier	
REFMETA	ADSL	SITEGRyN	Pooled Site Group y (N)	ADSL	6	N	8		integer		Perm	StudyIdentifier	
REFMETA	ADSL	AGE	Age	ADSL	7	N	8 8.1		float		Req	SubjectDemographic	
REFMETA	ADSL	AGEU	Age Units	ADSL	8	C	10		text	AGEU	Req	SubjectDemographic	(AGEU)
sasref	table	column	algorithm	qualifiers	standard	standardversion	standardref	comment					
REFMETA	ADSL	STUDYID		UPPERCASE	CDISC-ADAM	2.1		Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.					
REFMETA	ADSL	USUBJID		UPPERCASE	CDISC-ADAM	2.1		Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.					
REFMETA	ADSL	SUBJID		UPPERCASE	CDISC-ADAM	2.1		Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.					
REFMETA	ADSL	SITEID		UPPERCASE	CDISC-ADAM	2.1		Must be identical to the SDTM variables DM.STUDYID, DM.USUBJID, DM.SUBJID and DM.SITEID.					
REFMETA	ADSL	SITEGRy		MIXEDCASE	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.					
REFMETA	ADSL	SITEGRyN			CDISC-ADAM	2.1		The numeric code for SITEGRy. One-to-one map to SITEGRy.					
REFMETA	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.					
REFMETA	ADSL	AGEU		UPPERCASE	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 *Analysis Data Model (ADaM), Version 2.1* document and the *Analysis Data Model (ADaM) 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 (ADaM), Version 2.1* (see [Table 9.1 on page 343](#)) 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 SAS Clinical Standards Toolkit CDISC ADaM sample study provides a `source_values` data set that captures analysis parameter information. This data set offers a consistent approach for all CDISC standards that contribute metadata to the derivation of CRT-DDS (ADaM, SDTM, and SEND).

This display shows an excerpt of the sample ADaM source_values data set.

Display 9.1 Excerpt of the Sample source_values Data Set

SASref	table	column	value	label	algorithm	order	type	xmldatatype
SRCDATA	ADQS	AVAL	ACITM01	WORD RECALL TASK		1	N	integer
SRCDATA	ADQS	AVAL	ACITM02	NAMING OBJECTS AND FINGERS		2	N	integer
SRCDATA	ADQS	AVAL	ACITM03	DELAYED WORD RECALL		3	N	integer
SRCDATA	ADQS	AVAL	ACITM04	COMMANDS		4	N	integer
SRCDATA	ADQS	AVAL	ACITM05	CONSTRUCTIONAL PRAXIS		5	N	integer
SRCDATA	ADQS	AVAL	ACITM06	IDEATIONAL PRAXIS		6	N	integer
SRCDATA	ADQS	AVAL	ACITM07	ORIENTATION		7	N	integer
SRCDATA	ADQS	AVAL	ACITM08	WORD RECOGNITION		8	N	integer
SRCDATA	ADQS	AVAL	ACITM09	ATTENTION/VISUAL SEARCH TASK		9	N	integer
SRCDATA	ADQS	AVAL	ACITM10	MAZE SOLUTION		10	N	integer
SRCDATA	ADQS	AVAL	ACITM11	SPOKEN LANGUAGE ABILITY		11	N	integer
SRCDATA	ADQS	AVAL	ACITM12	COMPREHENSION OF SPOKEN LANGUAGE		12	N	integer
SRCDATA	ADQS	AVAL	ACITM13	WORD FINDING DIFFICULTY IN SPONTANEOUS SPEECH		13	N	integer
SRCDATA	ADQS	AVAL	ACITM14	RECALL OF TEST INSTRUCTIONS		14	N	integer
SRCDATA	ADQS	AVAL	ACTOT	ADAS-COG(11) Subscore	ACTOT = Sum of ADAS scores for items 1,2,4,5,6,7,8,11,12,13,and 14	15	N	integer
SRCDATA	ADQS	AVAL	CIBIC	EXTENT OF CHANGE, IF ANY, SINCE BASELINE CIBIC		16	N	integer

This data set can be found in *sample study library directory/cdisc-adam-2.1-1.5/sascstdemodata/metadata*.

For more information about analysis parameter value-level metadata, see sections 5.2.1 and 5.2.2 of the *Analysis Data Model (ADaM) Version 2.1* document.

The final set of metadata prescribed by the *Analysis Data Model (ADaM) Version 2.1* 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 9.4 on page 350](#). The analysis results metadata is illustrated in the SAS Clinical Standards Toolkit CDISC ADaM sample study analysis_results.sas7bdat data set found in *sample study library directory/cdisc-adam-2.1-1.5/sascstdemodata/*

metadata. This sample file can serve as a template to initialize your analysis results data set, or see [“ADaM Data Set Templates” on page 353](#).

Table 9.4 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 (for example, 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 identifierprovides 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
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

Analysis Results Metadata Field**	Description**	reference_columns (value of column where table='RESULTS')
REASON	The rationale for performing this analysis. It indicates when the analysis was planned (for example, "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 (for example, "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.	datasets
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

Analysis Results Metadata Field**	Description**	reference_columns (value of column where table='RESULTS')
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 (that is, 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 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
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 (for example, covariance structure). The name and version of the applicable software application 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 (ADaM), Version 2.1*, 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 (ADaM) 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 ADAE, ADSL, and BDS 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.ADAE has 0 observations and 85 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-2.1-1.5/metadata
```

A zero-observation template data set for the `analysis_results` data set can be found in **global standards library directory/standards/cdisc-adam-2.1-1.5/templates**.

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, “Compliance Assessment Against a Reference Standard,”](#) on page 115.

ADaM validation of ADSL and BDS data sets is based on the *CDISC ADaM Validation Checks Version 1.2 Maintenance Release* (dated and released July 15, 2012 to correct errors and remove duplicate checks). This documentation was prepared by the CDISC ADaM team. The version 1.2 release identifies 223 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-2.1-1.5/validation/control*)

This data set contains 264 records, 212 of which are CDISC validation checks.

Note: This is fewer checks than what is provided by CDISC because some of the CDISC checks are combined in the SAS Clinical Standards Toolkit and are handled by a single validation check.

There are 52 checks provided by SAS that address the addition of the two new domains (ADAE and ADTTE).

- the Messages data set (located at *global standards library directory/standards/cdisc-adam-2.1-1.5/messages*)

This data set contains 257 observations. Some messages in this data set are used across several checks in the Validation Master data set.

Several 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 1.2 release.

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 9.5 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 code logic 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 code logic might be required if your values are of greater precision.</p>

Check	Details
ADAM0133	R2AyLO is not equal to AVAL divided by AyLO 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.
ADAM0134	R2AyHI is not equal to AVAL divided by AyHI 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.

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	cstcheckcompareallcolumns*
cstcheck_columncompare	cstcheck_crossstdcomparedomains*
cstcheck_columnexists*	cstcheck_crossstdmetamismatch*
cstcheck_columnvarlist	cstcheck_metamismatch
cstcheck_comparedomains	cstcheck_notincodelist
cstcheck_dsmismatch	cstcheck_notunique
cstcheck_notconsistent	cstcheck_zeroobs

* These macros 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 the online macro API reference documentation.

Cross-Standard Validation Checks

Twenty-two 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 check macros (cstcheck_crossstdcomparedomains and cstcheck_crossstdmetamismatch) are available in the SAS Clinical Standards Toolkit 1.5. Part of the metadata available in the Validation Master data set for the 22 ADaM cross-standard validation checks is shown in [Figure 9.3 on page 358](#).

Figure 9.3 Partial Metadata for the CDISC ADaM Cross-Standard Validation Checks

checkid	codesource	tablescope	columnscope	code logic
2	ADAM0002	cstcheck_crossstdmetamismatch	_ALL_	<pre>"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;</pre>
3	ADAM0003	cstcheck_crossstdmetamismatch	_ALL_	<pre>"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;</pre>
4	ADAM0004	cstcheck_crossstdmetamismatch	_ALL_	<pre>"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;</pre>
51	ADAM0053	cstcheck_crossstdcomparedomains	_ALL_	STUDYID+USUBJID <pre>proc sql noprint;create table work_ _cstproblems as select &_cstSQLColList from &_cstDSName except select &_cstSQLColList from &_cstCrosDataLib. dm; quit;</pre>
56	ADAM0061	cstcheck_crossstdmetamismatch	ADSL	TRTSDT+TRTSDTM <pre>proc sql noprint;select count(distinct column) into _cstColFound from work_ _cstcolumnmetadata;create table work_ _cstProblems as select count(table) as exFound, cabs(" " &_cstCrMnemonic. " table, "table found") as &_cstCrMnemonic. _value, "&_cstColFound column(s) found" as &_cstStMnemonic. _value, "&_cstTableScope" as table, "&_cstColumnScope" as column from work_ _cstcrosstabmetadata (where=(table="EX")) having exFound=1 and &_cstColFound=0;quit;</pre>
156	ADAM0180	cstcheck_crossstdcomparedomains	_ALL_	ADSL SRCDOM <pre>proc sql noprint;create table work_ _cstproblems as select * from &_cstDSName where SRCDOM not in (select table from work_ _cstcrosstabmetadata);quit;</pre>

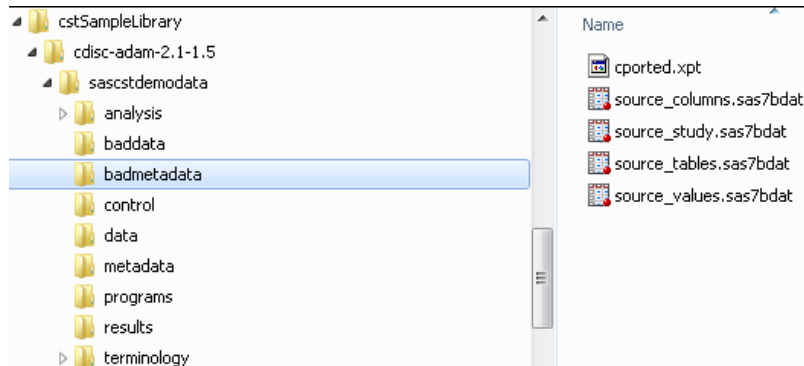
Sample Data for Validation and Reporting

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 programs (analyze_data.sas and

validate_data.sas, respectively), which are located in the programs folder (as shown in [Figure 9.4 on page 358](#)) point to the correct source data and metadata folders.

Figure 9.4 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 9.5 on page 360](#) and [Figure 9.6 on page 360](#). The first 15 records of the data set shown in [Figure 9.5 on page 360](#) 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 9.5 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 9.6 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=lcoun	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 9.7 on page 361](#). 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 9.7 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 (ADaM), Version 2.1* 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 9.6 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.

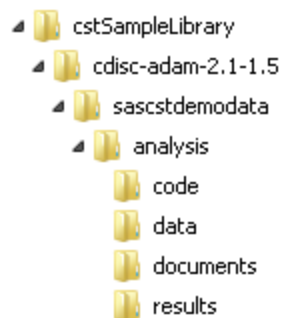
Reporting Component	Comments
<p>Analysis Results (tables, listings, and figures)</p> <p>For more information, see “Analysis Results (Tables, Listings, and Figures)” on page 370.</p>	<p>The set of statistical displays (for example, text, tabular, or graphical presentation of results) or inferential statements (such as p-values or estimates of treatment effect).</p>
<p>TLF Metadata (to include table shells)</p> <p>For more information, see “TLF Metadata” on page 364.</p>	<p>Commonly provided as table shells. Can also include display-specific metadata (often as Microsoft Excel files) used by the analysis programs to generate the displays.</p>
<p>Analysis Results Metadata</p> <p>For more information, see “Analysis Results Metadata” on page 371.</p>	<p>Defined by the <i>Analysis Data Model (ADaM), Version 2.1</i> document, Section 5.3. For more information, see Table 9.4 on page 350.</p>
<p>Analysis Programs</p> <p>For more information, see “Analysis Programs” on page 367.</p>	<p>Programming code that uses the analysis data sets (and, optionally, TLF metadata) to create the analysis results.</p>
<p>Submission Package (for example, eCTD)</p>	<p>The structured submission used to package data, metadata, code, and results in a standard form to facilitate review.</p>
<p>Define.xml</p>	<p>A metadata format that documents each tabulation (SDTM) or analysis (ADaM) data set, ancillary documents, and controlled terminology for a study or submission.</p>
<p>CSR/ISS/ISE</p>	<p>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.</p>

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:

sample study library directory/cdisc-adam-2.1/sascstdemodata/analysis

Here is an illustration of the ADaM analysis folder hierarchy:

Figure 9.8 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 9.6 on page 362](#).
- The data folder contains the display-specific metadata noted in the TLF Metadata component of [Table 9.6 on page 362](#).
- The documents folder contains table shells for the TLF Metadata component. For more information about table shells, see “[TLF Metadata](#)” on page 364.
- 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:

sample study library directory/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 9.9 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 1.5, 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:

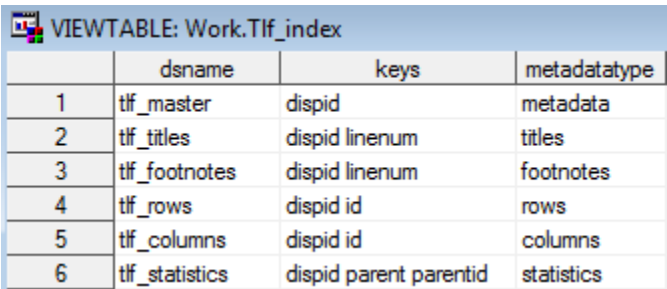
*sample study library directory/cdisc-adam-2.1/sascstdemodata/
 metadata/tlfdtd.xml*

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:

```
sample study library directory/cdisc-adam-2.1/sascstdemodata/analysis/data
```

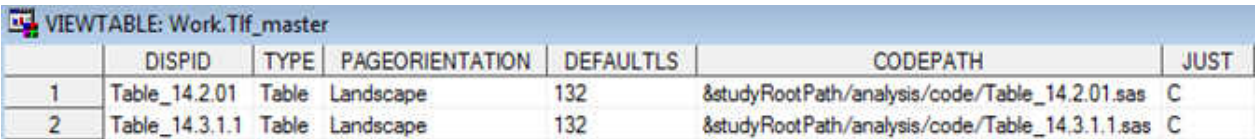
The following figures provide examples of some of the metadata available in the source XML file. This metadata has been extracted into SAS data sets.

Figure 9.10 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 9.11 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 9.12 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 9.9 on page 365](#).

Analysis Programs

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

sample study library directory/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 program 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 program is located in this folder:

sample study library directory/cdisc-adam-2.1/sascstdemodata/programs/analyze_data.sas

In the sample driver program, 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 9.11 on page 366](#)) 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
```

Creates an analysis result display from ADaM analysis data sets.

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 is %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.

```
@macvar studyRootPath Root path to the sample source study
@macvar _cstCTDescription Description of controlled terminology packet
@macvar _cstDebug Turns debugging on or off for the session
@macvar cstDefaultReportFormat Specifies the SAS ODS report destination
@macvar _cstGRoot Root path of the Toolkit Global Library
@macvar _cstResultsDS Results data set
@macvar _cstResultSeq Results: Unique invocation of check
@macvar _cstSASRefs Run-time SASReferences data set derived in process setup
@macvar _cstSeqCnt Results: Sequence number within _cstResultSeq
@macvar _cstSrcData Results: Source entity being evaluated
@macvar _cstStandard Name of a standard registered to Toolkit
@macvar _cstStandardVersion Version of the standard referenced in _cstStandard
@macvar _cst_rc Task error status
@macvar _CSTTLF_MASTERCODEPATH Dynamically derived code segment path from
    TLF metadata.
@macvar workpath Path to the SAS session work library

@param _cstDisplaySrc - required - Where information comes from to generate
    the result.
    Values: Code | Metadata
    Default: Metadata
@param _cstDisplayCode - conditional - Either a valid filename or the fully
    qualified path to code that produces an analysis result. If
    _cstDisplaySrc=Code, this parameter is used and is required. All of
    the remaining parameters are ignored.
@param _cstUseAnalysisResults - conditional - The study-specific analysis
    results metadata are used to provide report metadata.
    If _cstDisplaySrc=Metadata, either this parameter or _cstUseTLFddt
    must be set to Y. If both _cstUseAnalysisResults and _cstUseTLFddt
```

are set to Y, `_cstUseAnalysisResults` takes precedence.

Values: N | Y

Default: Y

@param `_cstUseTLFddt` - conditional - The study-specific mock table shell metadata (known as Tables, Listings, and Figures data definition metadata (TLFDDT)) are used to provide report metadata.

If `_cstDisplaySrc=Metadata`, either this parameter or `_cstUseAnalysisResults` must be set to Y. If both `_cstUseAnalysisResults` and `_cstUseTLFddt` are set to Y, `_cstUseAnalysisResults` takes precedence.

Values: N | Y

Default: Y

@param `_cstDisplayID` - conditional - The ID of the display from the designated metadata source. If `_cstDisplaySrc=Metadata`, this parameter is required.

@param `_cstDisplayPath` - optional - A valid filename or the fully qualified path to the generated display. If not provided, the 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 9.13 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-1.5/programs/initialize.properties
2	CST0200	1	CST_CREATEDFROMTEMPLATE	The SAS libref cstmplit was allocated to c:/cstGlobalLibrary/standards/cst-framework-1.5/templates to perform the template look-up
3	CST0102	2	CST_CREATEDFROMTEMPLATE	work.validation_sasrefs was created as requested
4	CST0200	1	CSTUTIL_PROCESSETUP	Process setup is using this SASReferences: C:\Users\geligh\AppData\Local\Temp\SAS Temporary Files\TD8128_L73859_validation_sasref
5	CST0200	1	CST_INSERTSTANDARDASREFS	SASReferences data set was successfully validated
6	CST0200	2	CSTUTIL_ALLOCATESASREFERENCE	SASReferences data set was successfully validated
7	CST0108	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-adam-2.1-1.5/programs/initialize.properties
8	CST0108	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstSampleLibrary/cdisc-adam-2.1-1.5/sascstdmodata/programs/validation.properties
9	CST0200	1	ADAM_VALIDATE	PROCESS STANDARD: CDISC-ADAM
10	CST0200	2	ADAM_VALIDATE	PROCESS STANDARDVERSION: 2.1
11	CST0200	3	ADAM_VALIDATE	PROCESS DRIVER: Unspecified
12	CST0200	4	ADAM_VALIDATE	PROCESS DATE: 2013-03-27T15:24:52
13	CST0200	5	ADAM_VALIDATE	PROCESS TYPE: VALIDATION
14	CST0200	6	ADAM_VALIDATE	PROCESS SASREFERENCES: C:\Users\geligh\AppData\Local\Temp\SAS Temporary Files\TD8128_L73859_validation_sasrefs.sas7bdat
15	CST0200	7	ADAM_VALIDATE	PROCESS STUDYROOTPATH: c:/cstSampleLibrary/cdisc-adam-2.1-1.5/sascstdmodata
16	CST0200	8	ADAM_VALIDATE	PROCESS GLOBALLIBRARY: c:/cstGlobalLibrary
17	CST0200	9	ADAM_VALIDATE	PROCESS CSTVERSION: 1.5

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 9.9 on page 365.](#))

For example, the Summary of Demographic and Baseline Characteristics provided in Table 14.2.01 is shown in this figure:

Figure 9.14 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%)

Analysis Results Metadata

The *Analysis Data Model (ADaM)*, Version 2.1 document provides specifications for capturing analysis results. As a result, traceability back to the contributing source data is possible. [Table 9.4 on page 350](#) 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 9.15 Analysis Results Metadata

	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"
DISPID	DOCUMENT		PROGSTMT		XMLPATH		XMLTITLE		
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.		%_cst\$Root/cdisc-adam-2.1-1.5/sascstdemodata/analysis/code/Table_14.2.01_nomd.sas		../Table_14.2.01.pdf		Table 14.2.01 - Summary of Demographics and Baseline Characteristics, ITT Population		
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.		%_cst\$Root/cdisc-adam-2.1-1.5/sascstdemodata/analysis/code/Table_14.3.1.1_nomd.sas		../Table_14.3.1.1.pdf		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:

```
sample study library directory/cdisc-adam-2.1/sascstdemodata/  
metadata/analysis_results.sas7bdat
```


10

Reporting

<i>Sample Reports</i>	373
Overview	373
<i>Process Results Reporting</i>	374
<i>Validation Check Metadata Reporting</i>	386

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 10.1 on page 376](#).

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 1.5, 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 10.1 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: <div><div>1</div>The data set referenced in &_cstSASRefs with type=results and subtype is either results or validationresults. <div>2</div>The data set referenced by &_cstResultsDS.</div>	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 &_cstSASRefs with type=results and subtype is either metrics or validationmetrics. 2 The data set referenced by &_cstMetricsDS. 	The data set referenced in &_cstSASRefs with type=results and subtype is either metrics or validationmetrics.
Validation_Control	The data set referenced in &_cstSASRefs with type=control and subtype=validation.	The data set referenced in &_cstSASRefs with type=control and subtype=validation.
Messages	&_cstMessages used by the prior task.	&_cstMessages built by a call to %cstutil_allocatesasreferences.

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 7.

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 375](#), 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(_cstsasreferencesdset=&_cstSASRefs,_cstreportbydomain=Y,
_cstreportoutput=&studyrootpath/results/cstchecktablereport.pdf);
```

This table describes all supported parameters in the sample %cstutil_createreport macro.

Table 10.2 Supported Parameters for the %cstutil_createreport Macro

Parameter	Description
_cstsasreferencesdset	The libref.dataset of SASReferences data set used for a specific process. This parameter is optional. If it is specified, then _cstresultsdset and _cstmetricsdset parameters are ignored. Either _cstsasreferencesdset or _cstresultsdset must be provided.
_cstresultsdset	The libref.dataset of the SAS Clinical Standards Toolkit process Results data set. This parameter is optional. Either _cstsasreferencesdset or _cstresultsdset must be provided. This parameter is ignored if _cstsasreferencesdset is specified.
_cstmetricsdset	The libref.dataset of the SAS Clinical Standards Toolkit process Metrics data set. This parameter is optional. This parameter is ignored if _cstsasreferencesdset is specified.
_cstreporterrorsonly	If N (default), then this parameter reports all records in the Results data set, including information and non-error results. If Y, then this parameter reports only records in error (where the Results data set field results.resultflag=1).
_cstreportobs	If null (default), then this parameter reports all records in error (where results.resultflag=1) in the Results data set. Otherwise, set this parameter to any integer value > 0, signifying the number of records to print per checkid (where results.checkid is non-null). If _cstreportobs > 0 excludes any records, then a footnote is printed, noting that not all records were printed.
_cstreportbytable	If N (default), then this parameter does not report results by table (that is, run report 1). If Y, then this parameter reports results by table (that is, run report 2).
_csttablechecksdsdset	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: CDISC SDTM Domains by Check” on page 141 . This parameter is optional. If this parameter is not used, then the data set is created.

Parameter	Description
<code>_csttablechecksdset</code>	Report 2 parameter. The code module (macro) to build <code>_csttablechecksdset</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>_csttablechecksdset</code> is not provided.
<code>_cstkeepablechecklist</code>	Report 2 parameter. The value is Y or N (default). If running report 2, then keep the derived list of tables (<code>_csttablechecklist</code>) to reuse in subsequent report requests. Building this file takes a while.
<code>_csttablesubset</code>	Report 2 parameter. This parameter is optional. It produces a report based on a specific table, indicated by <code>libref.data</code> set. 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(
```

```

_cstsasreferencesdset=&_cstSASRefs,
_cstresultsdset=&_cstRptResultsDS,
_cstmetricsdset=&_cstRptMetricsDS,
_cstreportbytable=N,
_cstreporterroronly=Y,
_cstreportobs=50,
_cstreportoutput=%nrbrquote(&_cstRptOutputFile),
_cstsummaryReport=Y,
_cstioReport=Y,
_cstmetricsReport=Y,
_cstgeneralResultsReport=Y,
_cstcheckIdResultsReport=Y);

```

Interpretation of this request (based on the parameter descriptions in [Table 10.2 on page 379](#)) 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 10.1 *Example of Report Summary*

SAS Clinical Standards Toolkit 1.5 CDISC-SDTM 3.1.3 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	2013-03-31T21:28:43
Report only errors, warnings & notes?	Yes
# records to report	50
Report results by table	No
Report output file	c:/cstSampleLibrary/cdisc-sdtm-3.1.3-1.5/sascstdemodata/results/cstreport.pdf

Display 10.2 Example of Process Inputs and Outputs

SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 VALIDATION

Process Inputs/Outputs

Type	Path
Autocall Libraries	(sdtmauto sasautos)
	sdtmauto: c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.3-1.5/macros
Format Search Path Libraries	(srcfmt cstfmt)
	srcfmt: c:/cstSampleLibrary/cdisc-sdtm-3.1.3-1.5/sascstdemodata/terminology/formats
	cstfmt: c:/cstGlobalLibrary/standards/cdisc-terminology-1.5/cdisc-sdtm/201212/formats
Reference Metadata	c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.3-1.5/metadata
Source Data	c:/cstSampleLibrary/cdisc-sdtm-3.1.3-1.5/sascstdemodata/data
Source Metadata	c:/cstSampleLibrary/cdisc-sdtm-3.1.3-1.5/sascstdemodata/metadata

Display 10.3 Example of Process Metrics (Report 1)

SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 VALIDATION

Process Metrics

Summary Metrics		Check Metrics				
Metric	#	Check ID	# Check Invocations	# Recs (if available)	# Errors	# Check Invocations Not Run
# of distinct check invocations	15	SDTM0003	1	1	0	0
# check invocations not run	1	SDTM0017	1	1	0	0
Errors (severity=High) reported	1	SDTM0033	1	1	0	0
Warnings (severity=Medium) reported	91	SDTM0580	1	25	24	0
Notes (severity=Low) reported	118	SDTM0601	1	36	0	0
Structural errors, warnings and notes	0	SDTM0606	1	36	0	0
Content errors, warnings and notes	210	SDTM0673	1	1	0	1
		SDTM0807	1	1	0	0
		SDTM0809	1	118	118	0
		SDTM0816	1	1	1	0
		SDTM0841	1	1	1	0
		SDTM0843	1	1	0	0
		SDTM0846	1	64	64	0
		SDTM0860	1	2	2	0
		SDTM0872	1	1	0	0

Note: "# Check Invocations Not Run" includes both checks that did not run and checks that failed to complete successfully.

Display 10.4 Example of Process Metrics by Domain (Report 2)

SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 VALIDATION

Process Metrics

Summary Metrics		Table Metrics				
Metric	#	Table	# Check Invocations	# Recs (if available)	# Errors	# Check Invocations Not Run
# of distinct check invocations	15	AE	3	3	0	0
# check invocations not run	1	CE	3	3	0	0
Errors (severity=High) reported	1	CM	3	3	0	0
Warnings (severity=Medium) reported	91	CO	4	4	0	0
Notes (severity=Low) reported	118	DA	3	3	0	0
Structural errors, warnings and notes	0	DM	6	6	0	1
Content errors, warnings and notes	210	DS	4	27	24	0
		DV	3	3	0	0
		EG	3	3	0	0
		EX	3	3	0	0
		FA	3	3	0	0
		IE	3	3	0	0
		LB	3	3	0	0
		MB	3	3	0	0
		MH	3	3	0	0
		MS	3	3	0	0
		PC	3	3	0	0
		PE	3	3	0	0
		POOLDEF	3	3	0	0
		PP	3	3	0	0
		QS	3	3	0	0
		RELREC	4	5	2	0
		RS	3	3	0	0
		SC	3	3	0	0
		SE	4	4	0	0

Display 10.5 Example of General Process Reporting

**SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 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-1.5/programs/initialize.properties
1	CST_CREATEDSFROMTEMPLATE	CST0200	Info	No	The SAS libref cstmplit was allocated to c:/cstGlobalLibrary/standards/cst-framework-1.5/templates to perform the template lookup
2	CST_CREATEDSFROMTEMPLATE	CST0102	Info	No	work.sasreferences was created as requested
1	CSTUTIL_PROCESSETUP	CST0200	Info	No	Process setup is using this SASReferences: C:\Users\sasses\AppData\Local\Temp\SAS Temporary Files\TD5816_D72872_sasreferences
1	CST_INSERTSTANDARDASREFS	CST0200	Info	No	SASReferences data set was successfully validated
2	CSTUTIL_ALLOCATESASREFERENCES	CST0200	Info	No	SASReferences data set was successfully validated
1	CST_SETPROPERTIES	CST0108	Info	No	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.3-1.5/programs/initialize.properties
1	CST_SETPROPERTIES	CST0108	Info	No	The properties were processed from the PATH c:/cstSampleLibrary/cdisc-sdtm-3.1.3-1.5/sascstdemodata/programs/validation.properties
1	SDTM_VALIDATE	CST0200	Info	No	PROCESS STANDARD: CDISC-SDTM
2	SDTM_VALIDATE	CST0200	Info	No	PROCESS STANDARDVERSION: 3.1.3
3	SDTM_VALIDATE	CST0200	Info	No	PROCESS DRIVER: Unspecified
4	SDTM_VALIDATE	CST0200	Info	No	PROCESS DATE: 2013-03-31T21:28:43
5	SDTM_VALIDATE	CST0200	Info	No	PROCESS TYPE: VALIDATION
6	SDTM_VALIDATE	CST0200	Info	No	PROCESS SASREFERENCES: C:\Users\sasses\AppData\Local\Temp\SAS Temporary Files\TD5816_D72872_sasreferences.sas7bdat

Display 10.6 Example of Validation Results by CheckID (Report 1)

**SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 VALIDATION**

Process Results, CheckID: SDTM0580

Description: Variable values should be populated with terms found in Completion/Reason for Non-Completion (C66727) CDISC controlled terminology codelist

Check scope: (Tables) DS, (Columns) DSDECOD

Source: WebSDM (R5112)

Lookup type: FORMAT, **Lookup source:** \$NCOMPLT

Validation check macro: cstcheck_notincodelist, using source metadata

Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
1	2	SRCDATA.DS. DSDECOD	SDTM0580	Value of DSDECOD not found in NCOMPLT Controlled Terminology Codelist	Warning	Yes	DSDECOD=LOST TO FOLLOW UP	STUDYID=SASCSTDDEMADATA, USUBJID=S003P011, DSDECOD=LOST TO FOLLOW UP,DSSTDTC=2008-07-01
1	3	SRCDATA.DS. DSDECOD	SDTM0580	Value of DSDECOD not found in NCOMPLT Controlled Terminology Codelist	Warning	Yes	DSDECOD=LOST TO FOLLOW UP	STUDYID=SASCSTDDEMADATA, USUBJID=S003P007, DSDECOD=LOST TO FOLLOW UP,DSSTDTC=2008-05-06
1	4	SRCDATA.DS. DSDECOD	SDTM0580	Value of DSDECOD not found in NCOMPLT Controlled Terminology Codelist	Warning	Yes	DSDECOD=LOST TO FOLLOW UP	STUDYID=SASCSTDDEMADATA, USUBJID=S003P008, DSDECOD=LOST TO FOLLOW UP,DSSTDTC=2008-08-16

Display 10.7 Example of Validation Results by Domain (Report 2)

SAS Clinical Standards Toolkit 1.5 CDISC-SDTM 3.1.3 VALIDATION									
Process Results, Table: DS									
Check ID	Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
SDTM0580	1	2	SRCDATA.DS. DSDECOD	SDTM0580	Value of DSDECOD not found in NCOMPLT Controlled Terminology Codelist	Warning	Yes	DSDECOD=LOST TO FOLLOW UP	STUDYID=SASCSTDDEMODATA, USUBJID=S003P011, DSDECOD=LOST TO FOLLOW UP.DSSTDTC=2008-07-01
SDTM0580	1	3	SRCDATA.DS. DSDECOD	SDTM0580	Value of DSDECOD not found in NCOMPLT Controlled Terminology Codelist	Warning	Yes	DSDECOD=LOST TO FOLLOW UP	STUDYID=SASCSTDDEMODATA, USUBJID=S003P007, DSDECOD=LOST TO FOLLOW UP.DSSTDTC=2008-05-06
SDTM0580	1	4	SRCDATA.DS. DSDECOD	SDTM0580	Value of DSDECOD not found in NCOMPLT Controlled Terminology Codelist	Warning	Yes	DSDECOD=LOST TO FOLLOW UP	STUDYID=SASCSTDDEMODATA, USUBJID=S003P009, DSDECOD=LOST TO FOLLOW UP.DSSTDTC=2008-06-16

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 10.3 *Supported Parameters for the %cstutil_createmetadatareport Macro*

Parameter	Description
<code>_cstStandardTitle</code>	This parameter is optional. Title that defines the title2 statement.
<code>_cstValidationDS</code>	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.
<code>_cstValidationDSWhClause</code>	Optional WHERE clause applied to <code>_cstValidationDS</code> .
<code>_cstMessagesDS</code>	This parameter is required. The Messages data set used by a SAS Clinical Standards Toolkit process.
<code>_cstStdRefDS</code>	The Validation StdRef data set created for a SAS Clinical Standards Toolkit standard. This file is required if <code>_cstStdRefReport=Y</code> .
<code>_cstReportOutput</code>	This parameter is required. The path and filename where the report output is to be written. File types HTML, RTF, and PDF are supported.
<code>_cstCheckMDReport</code>	Specifies whether panel 2 additional check details is run. The default value is N.
<code>_cstMessageReport</code>	Specifies whether panel 3 message details is run. The default value is N.
<code>_cstStdRefReport</code>	Specifies whether panel 4 reference information is run. The default value is N.
<code>_cstRecordView</code>	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 3.1.3 Validation Check Metadata),
  _cstvalidationds=refcntl.validation_master,
  _cstvalidationdswclause=,
  _cstmessagesds=&_cstMessages,
  _cststdrefds=refcntl.validation_stdref,
  _cstreportoutput=%nrbrquote(&studyOutputPath/results/cstcheckmetadatareport.pdf),
  _cstcheckmdreport=Y,
  _cstmessagereport=Y,
  _cststdrefreport=Y,
  _cstrecordview=N);
```

Interpretation of this request, based on the parameter descriptions in [Table 10.3 on page 388](#), produces a validation check metadata report (cstcheckmetadatareport.pdf) that contains all four report sections for the CDISC SDTM 3.1.3 validation checks.

Display 10.8 Example of Check Overview

SAS Clinical Standards Toolkit 1.5 CDISC-SDTM 3.1.3 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	***	WebSDM	IR5000	Identifies domain table that has zero rows and therefore contains no data	Warning	_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.	Error	DM+DS+EX	
SDTM0003	***	SAS	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_	

Display 10.9 Example of Additional Check Details (Panel 2) [_cstCheckMDReport=Y]

SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 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	WebSDM	Metadata	ostocheck_zeroobs	Yes	No code logic for this check			Active	Yes
SDTM0002	SAS	Metadata	ostocheck_zeroobs	No	No code logic for this check			Active	Yes
SDTM0003	SAS	Metadata	ostocheck_zeroobs	No	No code logic for this check			Active	Yes
SDTM0004	SAS	Metadata	ostocheck_dsmismatch	Yes	proc sql noprint; create table work_ostproblems as select src.sasref, src.table from work_osttablemetadata src left join work_ostrefablemetadata ref on upcase(src.table)=upcase(ref.table) where ref.table=""; quit;			Active	Yes
SDTM0005	SAS	Metadata	ostocheck_dsmismatch	Yes	proc sql noprint; create table work_ostproblems as select src.sasref, src.table from work_osttablemetadata (where=(ksubstr(kleft(upcase(table)),1,4) ^= "SUPP" and (ksubstr(kleft(upcase(table)),1,1) not in ("X" "Y" "Z") or length(table) ne 2))) src left join work_ostrefablemetadata ref on upcase(src.table)=upcase(ref.table) where ref.table=""; quit;			Active	Yes
SDTM0006	SAS	Metadata	ostocheck_dsmismatch	Yes	proc sql noprint; select upcase(data.sasref) into _cstSourceData from work_osttablemetadata data; create table work_ostproblems as select "&_cstSourceData" as sasref, memname as table from sashelp.vtable data left join work_osttablemetadata src on data.memname=upcase(src.table) where src.table="" and data.libname="&_cstSourceData"; quit;			Active	Yes

Display 10.10 Example of Message Details (Panel 3) [_cstMessageReport=Y]

SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 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	WebSDM	Domain &_ostparm1 contains 0 observations or is missing			
SDTM0002	SAS	Missing (or empty) DM, DS or EX domain			
SDTM0003	SAS	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			

Display 10.11 Example of Reference Information (Panel 4) [*cstSTDRefReport=Y*]

SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 Validation Check Metadata

Reference Information

Validation Check Identifier	Source of Information	Reference in Source Supporting Check	Source Text that Supports Check
SDTM0001	SDTM 3.1.2 Implementation Guide	3.2, page 17	In the event that no records are present in a dataset (e.g., a small PK study where no subjects took concomitant medications), the empty dataset should not be submitted and should not be described in the define.xml document.
	SDTM 3.1.2 Implementation Guide	2.5, page 13	A sponsor should only submit domain datasets that were actually collected (or directly derived from the collected data) for a given study.
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, v 0.8, 13-August-2007	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.
	SDTM 3.1.2 Implementation Guide	Appendix C2, page 274	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	SDTM 3.1.2 Implementation Guide	4.1.1.6, page 22	In some cases, sponsors may need to define new custom domains other than those represented in the SDTMIG or listed in Appendix C2, and may be concerned that CDISC domain codes defined in the future will conflict with those they choose to use. To eliminate any risk of a sponsor using a name that CDISC later determines to have a different meaning, domain codes beginning with the letters X, Y, or Z have been reserved for the creation of custom domains. Any letter or number may be used in the second position. Note the use of codes beginning with X, Y, or Z is optional, and not required for custom domains.
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.

Display 10.12 Example of Using WHERE Clause
[*cstValidationDSWhClause=checkid='SDTM0801'*]

SAS Clinical Standards Toolkit 1.5
CDISC-SDTM 3.1.3 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	***	WebSDM	IR5500	Identifies non-Demographics domain subjects (USUBJID) not found in the Demographics domain	Error	[ALL_DM][DM]	STUDYID+USUBJID

Display 10.13 Example of by Record View [*_cstRecordView=Y*]

SAS Clinical Standards Toolkit 1.5 CDISC-SDTM 3.1.3 Validation Check Metadata

Full Metadata Listing for Checkid SDTM0001

Metadata Item	Value
Validation check identifier	SDTM0001
Standard version	3.1.2
Source of check	WebSDM
Record identifier used by checksource	IR5000
Severity of check	Warning
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	SDTM000101CST150SDTM3132013-03-07T18:15:21CST
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) SDTM 3.1.2 Implementation Guide, 3.2, page 17: In the event that no records are present in a dataset (e.g., a small PK study where no subjects took concomitant medications), the empty dataset should not be submitted and should not be described in the define.xml document.
Basis for check from information source	(2) SDTM 3.1.2 Implementation Guide, 2.5, page 13: A sponsor should only submit domain datasets that were actually collected (or directly derived from the collected data) for a given study.

Appendix 1

Global Macro Variables

Overview 393

Global Macro Variables and Their Associated Metadata 394

Overview

Most of the SAS Clinical Standards Toolkit global macro variables that are provided by SAS are defined in properties files in the form of name and value pairs, such as:

```
_cstDebug=0
```

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 process that references the standard. Each registered standard might have an action-related properties file that specifies global macro variables that are needed for processes that perform the action. An example of this type of action-related properties file is validation.properties.

A properties file is processed in one of two ways:

- 1 A direct call is made to the SAS Clinical Standards Toolkit utility macro `%cst_setstandardproperties` in a code module, such as a driver program like `validate_data.sas`. The `%cst_setstandardproperties` macro calls `%cst_setproperties`.
- 2 The file is included in the SASReferences data set (with type=properties), in which the `%cstutil_allocatesasreferences` macro calls `%cst_setproperties`.

Global macro variables can be deleted at the end of a process if the SAS Clinical Standards Toolkit utility macro %cstutil_cleanupcstsession is called with the _cstDeleteGlobalMacroVars parameter set to 1.

Global Macro Variables and Their Associated Metadata

Global macro variables and their associated metadata can be found in the standardmacrovariables and standardmacrovariabledetails data sets in the standard control folder.

These displays show examples of the standardmacrovariables data set and the standardmacrovariabledetails data set.

Display A1.1 Example of the standardmacrovariables Data Set

	macrovariable	label	description	required	casesensitive	valueset	property/filetype
1	_cstCheckSortOrder	Specifies the order in which validation checks are to be run	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).	N	N	ANY	validation
2	_cstColumnMetadata	Data set containing column-level metadata supporting validation	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.	Y	N	DATASET	validation
3	_cstDebug	Turns debugging on or off for the session	If on, then _cstDebugOptions are set. Many files remain in the Work library at process conclusion. Note that when _cstDebug=1, the size of the SAS log is significantly larger.	Y	N	DISCRETE	initialize
4	_cstDebugOptions	SAS session debugging options when _cstDebug=1	SAS system options set when _cstDebug=1.	N	N	ANY	initialize
5	_cstFMTLibraries	Modify format search path	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.	N	N	ANY	initialize
6	_cstMessageOrder	Merge or append message data sets?	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.	N	N	DISCRETE	initialize

Display A1.2 Example of the `standardmacrovariabledetails` Data Set

	macrovariable	macrovalue	macrovaluelabel	default
1	<code>_cstCheckSortOrder</code>	<code>_DATA_</code>	Use order defined in the Validation Control data set	Y
2	<code>_cstColumnMetadata</code>	<code>work._cstcolumnmetadata</code>		Y
3	<code>_cstDebug</code>	0	Off	Y
4	<code>_cstDebug</code>	1	On	N
5	<code>_cstDebugOptions</code>	<code>mprint mlogic symbolgen mautolocdisplay</code>		Y
6	<code>_cstFMTLibraries</code>			Y
7	<code>_cstMessageOrder</code>	APPEND	Append records from multiple message data sets	Y
8	<code>_cstMessageOrder</code>	MERGE	Merge records from multiple message data sets	N
9	<code>_cstMessages</code>	<code>work._cstmessages</code>		Y
10	<code>_cstMetrics</code>	0	Off	N
11	<code>_cstMetrics</code>	1	On	Y
12	<code>_cstMetricsCntNumBadChecks</code>	0	counter initialized to 0	Y
13	<code>_cstMetricsCntNumChecks</code>	0	counter initialized to 0	Y

The `standardmacrovariables` and `standardmacrovariabledetails` data sets can be easily merged with the following SAS code:

```
proc sql;
  select smv.*, smvd.macrovalue, smvd.macrovaluelabel, smvd.default
  from control.standardmacrovariables smv,
       control.standardmacrovariabledetails smvd
  where smv.macrovariable = smvd.macrovariable;
quit;
```

Here are several commonly used global macro variables that are not defined in the properties files previously described:

Global Macro Variable	Example	Comments
<code>_cstGRoot</code>	<code>C:\cstGlobalLibrary</code>	This variable is required. It defines the location of <code>_cstGlobalLibrary</code> . It is set with the autocall macro <code>%cstutil_setcstgroot</code> , which is called in most framework macros. It is used most often in SASReferences paths to enable relative path mobility.

Global Macro Variable	Example	Comments
<code>_cstSRoot</code>	<code>C:\cstSampleLibrary</code>	This variable is optional. It defines the location of <code>_cstSampleLibrary</code> . It is set with the autocall macro <code>%cstutil_setcstsroot</code> , which is called in most sample driver programs to derive the <code>studyRootPath</code> and <code>studyOutputPath</code> global macro variables.
<code>studyRootPath</code>	<code>C:\Study1</code>	This variable is optional. It defines the location of study data and metadata. It is often set in user-defined driver programs (for example, <code>validate_data.sas</code>). It is used in <code>SASReferences</code> paths to limit the changes that are required when changing input data sources, which facilitates portability.
<code>studyOutputPath</code>	<code>C:\Study1\output</code>	This variable is optional. It defines the location of generated output. It is often set in user-defined driver programs (for example, <code>validate_data.sas</code>). It is used in <code>SASReferences</code> paths to limit the changes that are required when changing output locations, which facilitates portability.

Index

C

CDISC 1

CDISC ADaM

Analysis data set metadata
343

analysis results metadata 350

analysis variable metadata
345

cross-standard validation 357

data set templates 353

key clinical reporting
components 362

overview 341

sample data 358

sample reporting 361

SAS representation 342

TLF metadata 364

unique validation properties
356

validation check macros 357

validation of analysis data sets
354

CDISC ADaM 2.1 69

purpose 69

reference standard 71

regulatory basis 70

release date 70

CDISC CRT-DDS 1.0 74

purpose 74

reference standard 75

regulatory basis 74

release data 74

CDISC CRT-DDS standard

sample XML style sheet 55

CDISC ODM 79

release dates 79

CDISC ODM 1.3.0

purpose 79

reference standard 80

CDISC ODM 1.3.1

reference standard 86

CDISC SDTM

description 63

CDISC SDTM 3.1.1 62

purpose 62

reference standard 66

release dates 63

CDISC SDTM 3.1.2

reference standard 67

CDISC SDTM 3.1.3

reference standard 68

CDISC SEND standard 87

CDISC Terminology standard
89

purpose 89

reference standard 89

clinical

- defined 1
- Clinical Data Interchange
Standards Consortium
- See [CDISC](#)
- clinical research activities 1
- columns
 - in data tables 55
- common framework metadata 13
- controlled terminology 119
 - alternatives 222
 - defined 119

D

- data set templates
 - for CDISC ADaM 353
- data sets
 - creating data sets used by
framework 20
 - list of data sets associated
with registered standard 18
- data standards
 - creating table shells based on 20
 - getting a copy of the reference
metadata for 21
- data tables 54
 - columns in 55
- default version for a standard
setting 27
- default version of standards
referencing 17

F

- files
 - list of files associated with
registered standard 18
- folder hierarchy
 - global standards library 59
- framework
 - creating data sets used by 20
 - creating table shells based on
a data standard 20
 - determining which revision of
a standard version is
installed 18
 - getting a copy of the reference
metadata for a data
standard 21
 - getting a list of files and data
sets associated with a
registered standard 18
 - getting a list of installed
standards 17
 - initializing global macro
variables 16
 - inserting information from
registered standards into
SASReferences files 22
 - referencing default version of
standards 17
 - usage scenarios 16
- framework metadata 13
- Framework module 8

G

global macro variables
 initializing 16
 global standards library 8
 directories in 9
 directory structure 10
 folder hierarchy 59

I

initializing global macro
 variables 16
 installed standards
 getting a list of 17
 internal validation
 checks 243
 defined 229
 driver modules supplied by
 SAS 236
 example check 246
 macros 231
 sample driver programs 235
 validation_control SAS views
 245
 validation_master data set
 243

L

list of files and data sets
 associated with registered
 standard 18

list of installed standards 17

M

macro variables
 initializing framework's global
 macro variables 16
 macros
 for internal validation 231
 utility macros for metadata
 files 95
 maintenance usage scenarios
 25
 Messages data set 15
 file content and structure 47
 metadata
 getting a copy of reference
 metadata 21
 metadata directory 9
 metadata files
 additional files 54
 common framework metadata
 13
 descriptions of 34
 SASReferences files 93
 metadata repository
 See [global standards library](#)

P

process controls 119
 defined 119
 properties 15, 119

defined 119
 properties files
 structure of 45

R

reference_columns data set 55
 reference_tables data set 54
 reference metadata 119
 defined 119
 getting a copy of 21
 reference standards 59
 references 2
 referencing default version of
 standards 17
 registered standards
 inserting information from
 SASReferences files into
 22
 list of files and data sets
 associated with 18
 registering
 new standards 25
 new version of a standard 25
 unregistering a standard
 version 27
 unregistering an old version of
 a standard, then registering
 a new version of a standard
 28
 releases
 determining which release is
 installed 18
 results 119

defined 119

Results data set 15
 file content and structure 50
 revisions
 determining which revision is
 installed 18

S

SAS Clinical Standards Toolkit
 1
 SAS sessions
 translating content of
 SASReferences file for 113
 SASReferences data set 15
 file content and structure 42
 validating 233
 SASReferences file
 assessing structural integrity
 and content 108
 communicating filename and
 location to SAS Clinical
 Standards Toolkit 106
 how it's used 106
 translating content for SAS
 sessions 113
 SASReferences files 93
 building 94
 inserting information from
 registered standards into
 22
 sample files 94
 templates 94
 utility macros 95

- scenarios
 - maintenance usage scenarios 25
- scenarios for framework usage 16
- schema-repository directory 12
- set of checks to run 119
 - defined 119
- source_columns data set 55
- source_tables data set 54
- source data 118
 - defined 118
- source metadata 118
 - defined 118
- standard versions
 - unregistering 27
- Standardlookup data set 14, 95
 - file content and structure 39
 - type and subtype values 96
- standards 1
 - CDISC ADaM 69
 - CDISC CRT-DDS 1.0 74
 - CDISC ODM 79
 - CDISC SDTM 62
 - CDISC SDTM 3.1.2 67
 - CDISC SDTM 3.1.3 68
 - CDISC SEND 87
 - CDISC Terminology 89
 - creating table shells based on
 - a data standard 20
 - defined 13
 - determining which revision is installed 18
 - getting a copy of the reference metadata for a data standard 21
 - getting a list of installed standards 17
 - inserting information from registered standards into SASReferences files 22
 - list of files and data sets associated with registered standard 18
 - reference standards 59
 - referencing default version of 17
 - registering a new standard 25
 - registering a new version 25
 - SAS representation of 58
 - setting the default version for a standard 27
 - supported 58
 - unregistering an old version of a standard, then registering a new version of a standard 28
- Standards data set 14
 - file content and structure 34
- standards directory 10
- StandardSASReferences data set 14
 - file content and structure 37
- style sheet 55
- Summary data set 55
- supported standards 58

T

table shells
 creating, based on a data
 standard 20
 defined 364
 TLF metadata
 CDISC ADaM 364
 toolkit
 defined 2
 translating content of
 SASReferences file 113

U

unregistering
 a standard version 27
 an old version of a standard,
 and then registering a new
 version of a standard 28
 usage scenarios
 maintenance scenarios 25
 usage scenarios 16
 utility macros 95

V

validation checks 54
 Validation Control data set 54
 validation framework 117
 building a validation process
 153
 components of 118

cross-standard validation 150
 debugging validation
 processes 204
 how SAS Clinical Standards
 Toolkit interprets validation
 check metadata 191
 messages 145
 metadata requirements 120
 performance considerations
 227
 reference metadata 121
 running a validation process
 160
 sample CDISC SDTM 3.1.3
 driver program:
 validate_data.sas 160
 SAS implementation of ISO
 8601 196
 SASReferences customization
 154
 setting properties for the
 validation process 159
 source metadata 126
 supplemental validation check
 metadata: CDISC SDTM
 class by check 143
 supplemental validation check
 metadata: CDISC SDTM
 domains by check 141
 supplemental validation check
 metadata: validation
 standard references 138
 validation check macros 185

- validation check metadata:
 - Validation Master data set [127](#)
- validation checks by standard [172](#)
- validation control: specification of run-time checks [156](#)
- validation customization [212](#)
- validation metrics [146](#)
- validation properties [143](#)
- validation results and metrics [168](#)
- Validation Master data set [54](#)
- validation metrics [55](#)
- variables
 - initializing framework's global macro variables [16](#)
- versions
 - determining which revision is installed [18](#)
 - referencing default version of a standard [17](#)
 - registering a new version [25](#)
 - setting the default version for a standard [27](#)
 - unregistering a standard version [27](#)
 - unregistering an old version of a standard, then registering a new version of a standard [28](#)

X

- XML style sheet [55](#)
- xsl-repository directory [13](#)

