THE POWER TO KNOW.

SAS® Clinical Standards Toolkit 1.2

User's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2009. *SAS® Clinical Standards Toolkit 1.2: User's Guide*. Cary, NC: SAS Institute Inc.

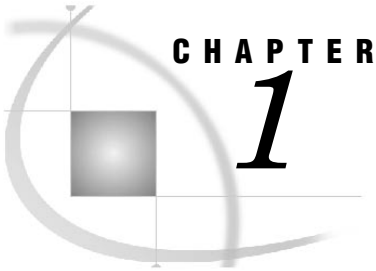
SAS® Clinical Standards Toolkit 1.2: User's Guide

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

1st printing, August 2009

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents



CHAPTER 1

Introduction to SAS Clinical Standards Toolkit

How to Use This Document

The following list provides suggestions for using this document:

- Chapter 1 provides an introduction to the software.
- Chapter 2 provides an overview of the Toolkit framework including how standards are defined, registered, and managed.
- Chapter 3 summarizes the Toolkit metadata supporting key framework functions and common tasks across multiple standards.
- Chapter 4 provides an overview of the standards supported in Toolkit 1.2.
- Chapter 5 describes a key metadata file, SASReferences, which itemizes all inputs and outputs of a Toolkit process.
- Chapter 6 provides detailed information about the first of two key features of Toolkit 1.2—validation of user metadata and data against a registered Toolkit standard.
- Chapter 7 provides detailed information about the second of two key features of Toolkit 1.2—creation of a CDISC-CRTDDS (define.xml) file.
- Appendix 1 provides a list of all global macro variables.
- Appendix 2 provides framework messages.
- Appendix 3 provides sample CDISC-SDTM SAS representations of a reference_tables record and a reference_columns record.
- Appendix 4 provides the macro application programming interface (API) reference.
- Appendix 5 provides the CDISC-SDTM 3.1.1 validation checks.
- Appendix 6 provides sample CDISC-CRTDDS SAS representations of a reference_tables record and a reference_columns record.
- Appendix 7 provides the CDISC-CRTDDS 1.0 validation checks.

What Is SAS Clinical Standards Toolkit?

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

Clinical

SAS Clinical Standards Toolkit focuses primarily on support of clinical research activities. That is, those activities that are involved with the discovery and development of new pharmaceutical and biotechnology products and medical devices, from project initiation through product submission, and the full product lifecycle. This does not include non-research patient records or healthcare, pharmacy, hospital, and insurance electronic records.

Standards

The product initially focuses on standards defined by the Clinical Data Interchange Standards Consortium (CDISC), a global, open, multidisciplinary, non-profit organization that has established standards to support the acquisition, exchange, submission, and archive of clinical research data and metadata. The CDISC mission is to develop and support global, platform-independent data standards that enable information system interoperability to improve medical research and related areas of healthcare. The Toolkit is not limited to support of CDISC standards, but, with time, will include other evolving industry-standard data models. The Toolkit framework is designed to support specification and use of any user-defined standard.

Toolkit

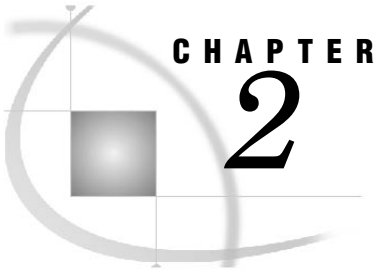
The term “toolkit” connotes a collection of tools, products, and solutions. SAS Clinical Standards Toolkit 1.2 provides an initial set of standards and functionality that will evolve and grow with future product updates and releases. Customer requirements and expectations of Toolkit will play a key role in the functionality provided in future releases.

References

Table 1.1 References

Reference	Web Address	Description
CDISC SDTM 3.1.1	http://www.cdisc.org/models/sdtm /v1.1/index.html	Provides access to *CDISC SDTM Implementation Guide V3.1.1 Final* and *CDISC Study Data Tabulation Model Version 1.1 Final*
CDISC CRTDDS 1.0	http://www.cdisc.org/models/def /v1.0/index.html	Provides access to *Case Report Tabulation Data Definition Specification (CRT-DDS, also called define.xml) Final Version 1.0*
CDISC SDTM 3.1.2	http://www.cdisc.org/models/sdtm /v1.1/index.html	Provides access to Release of SDTM V1.2 & SDTM IG V3.1.2
CDISC Controlled Terminology	http://www.cancer.gov/cancertopics /terminologyresources/page6 /print?page=&keyword	Provides access to FTP directory of documents, text listings, and spreadsheets of supported CDISC terminology. http://evs.nci.nih.gov/ftp1/CDISC /SDTM/ (*Note:* This site offers a current, cumulative set of terminology supporting CDISC-SDTM.)

Reference	Web Address	Description
Validation Checks Performed by WebSDM™ Version 2.6 on SDTM version 3.1.1 Datasets	http://phaseforward.com/resource/whitepapers/Validation Checks 2.6/WebSDM V2.6 Validation Checks FINAL.pdf	WebSDM Version 2.6 checks
Janus Operational Pilot	http://www.fda.gov/ForIndustry/DataStandards/CDISCDataStandards/ucm155327.htm	Provides information about operational pilots to date, including error checks run as a part of the load process into Janus.
ISO 8601:2004 Data elements and interchange formats--Information interchange--Representation of dates and times	http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40874	The official ISO 8601 standard.
SAS Customer Support site for SAS Clinical Standards Toolkit	http://support.sas.com	Find current information about SAS Clinical Standards Toolkit, including SAS Notes detailing usage notes and their current status.



CHAPTER 2

Framework

Overview

The framework module of SAS Clinical Standards Toolkit provides functionality to manage the registration of standards and also general cross-model functionality.

To understand framework, it is necessary to understand the fundamentals of how the files are structured and used. The framework consists of two distinct pieces:

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

The following sections describe the structure of the global standards library and use some of the framework macros as an example of how the files are used.

Global Standards Library

During installation and configuration, the user is prompted for the location where the global standards library should be located. The configuration process creates a series of directories under this location.

- **`metadata`**: contains data sets that have information relating to the registered standard versions. For details, see "Common Framework Metadata" on page 7.
- **`schema-repository`**: contains the schemas for XML-based standards that are supported.

- **`standards`**: contains directories for each of the supported standards.
- **`xsl-repository`**: contains directories and XSL files used in reading and writing XML files.

The metadata directory contains two data sets: standards and standardsasreferences. The standards data set contains a list of the registered standards along with basic information relating to the standard. An example of the main columns from standards is shown below:

Figure 2.1 Key Columns, Standards Data Set

	mnemonic	standardversion	comment	rootpath	isstandarddefault	iscstframework	isdatastandard	supportsvalidation	isxmlstandard
1	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1	Y	N	Y	Y	Y
2	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.	Y	N	Y	Y	N
3	CT	200810	CDISC Terminology, Packages 1, 2A, 2B, LABTEST	&_cstGRoot/standards/cdisc-terminolo	Y	N	N	N	N
4	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework	Y	Y	N	N	N

The standardsasreferences data set contains pointers to artifacts shipped with the standard version. Selected columns for CDISC-CRTDDS version 1.0 are shown below:

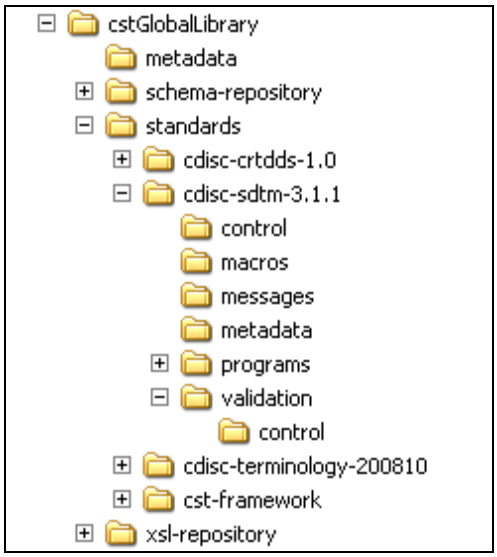
Figure 2.2 StandardSASReferences Data Set, CDISC- CRTDDS V1.0

	standard	standardversion	type	subtype	SASref	reftype	path	order	memname
1	CDISC-CRTDDS	1.0	autocall		crtauto	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0	1	
2	CDISC-CRTDDS	1.0	fmtsearch		ctfmt	libref	&_cstGRoot./standards/cdisc-crtdds-1.0	1	crtddsct.sas7bcat
3	CDISC-CRTDDS	1.0	lookup		crtctrl	libref	&_cstGRoot./standards/cdisc-crtdds-1.0	.	standardlookup.sas7bdat
4	CDISC-CRTDDS	1.0	messages		crtmsg	libref	&_cstGRoot./standards/cdisc-crtdds-1.0	1	messages.sas7bdat
5	CDISC-CRTDDS	1.0	properties	initialize	crtprop	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0	1	initialize.properties
6	CDISC-CRTDDS	1.0	referencemetadata	column	crtref	libref	&_cstGRoot./standards/cdisc-crtdds-1.0	.	reference_columns.sas7bdat
7	CDISC-CRTDDS	1.0	referencemetadata	table	crtref	libref	&_cstGRoot./standards/cdisc-crtdds-1.0	.	reference_tables.sas7bdat
8	CDISC-CRTDDS	1.0	referencexml	stylesheet	xslt01	fileref	&_cstGRoot./standards/cdisc-crtdds-1.0	1	define1-0-0.xsl

The type and subtype columns are a way to reference the information that Toolkit needs to access from the directory structures and file naming standards that are used by the customer. A full list of valid types and subtypes are provided later in this document.

The **`standards`** directory contains subdirectories for each of the standard versions provided by SAS (and optional customer-provided standard versions). Each subdirectory should be considered a stand-alone module. This is how the Toolkit can keep parallel standards while reducing the need for revalidation. Within each subdirectory, there might be one or more directories that group the files, data sets, and housekeeping programs. The following shows the directory structure for a Microsoft Windows global standards library with the CDISC-SDTM version 3.1.1 tree expanded.

Figure 2.3 Directory Structure for a Microsoft Windows Global Standards Library



The **`schema-repository`** directory contains XML schema definitions that are used to validate XML files. Standards that use XML should put their schemas into this directory so that they can be found.

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

What Is a Standard?

The answer depends on what the standard is meant to do. In the case of terminology, it might be as little as a format catalog and data set. In the case of an XML-based standard, it might be metadata describing the SAS representation of the XML; data sets to control validation of the SAS representation of the XML; routines to convert the SAS representation to the actual XML files, as well as initialization files for standard-specific properties.

The minimum items needed to register a standard to the framework are the data sets that define the standard and the standard's SASReferences. The macro to register the standard is described in "Registering a New Version of a Standard" on page 13.

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

Common Framework Metadata

The following SAS Clinical Standards Toolkit metadata files support the following:

- key framework functions
- common tasks across multiple standards

File structure and content for each of these metadata files is fully described in Chapter 3, “Metadata File Descriptions.” Use of these files is documented in sections that use this Toolkit metadata.

standards–This data set contains a list of the registered standards (for example, CDISC-SDTM-3.1.1), along with basic information relating to the standard.

standardsasreferences–This data set contains pointers to generic system input and output files shipped with each standard version that are used by each SAS Clinical Standards Toolkit process.

standardlookup–This data set contains valid values for discrete variables in the various SAS Clinical Standards Toolkit metadata files.

sasreferences–Defines both generic system and study-specific input and output files required by each SAS Clinical Standards Toolkit process. A sample SASReferences data set is provided with each supported standard.

properties–One or more files that provide the set of name-value pairs required to establish the environment for each SAS Clinical Standards Toolkit process. These properties are translated into SAS global macro variables at the start of each process.

messages–This data set contains a list of codes and associated text, specific to each standard. And, in some cases, it contains specific actions such as validation that are used to report process results.

results–This data set summarizes each SAS Clinical Standards Toolkit process, capturing the outcome of specific process actions and using the messages data set to standardize output.

Other SAS Clinical Standards Toolkit metadata files specific to supported standards or specific to Toolkit actions such as validation are described in Chapter 3, “Metadata File Descriptions,” and discussed in this document.

Common Usage Scenarios for the Framework

The following paragraphs describe various usage scenarios that the framework accommodates and shows the code required to complete the scenario. Each section shows an example of macro calls. Full macro documentation is available in Appendix 4, “Macro Application Programming Interface.”

Initializing the Framework’s Global Macro Variables

The framework requires certain global macro variables to exist in order to execute properly. A user should initialize these variables at the start of the session. The same might be true for a standard, in that it needs macro variables to exist in order to call its macros. The framework provides a macro to help with this.

```
/*
initialize the global variables needed by the framework
*/
%cst_setStandardProperties(
    _cstStandard=CST-FRAMEWORK
    ,_cstStandardVersion=1.2
    ,_cstSubType=initialize
    );
```

This code looks at the global SASReferences information for a properties entry with a subtype of initialize. It uses the referenced file to load the properties from. After this macro has been called once, a user does not need to call it again during the SAS session unless he wants to override macro variables or reset them.

Referencing the Default Version of a Standard

In places that require a version to be specified, this information can usually be omitted if the default version is to be used. The default version is specified in the global standards library metadata. For example, the code to initialize the framework's properties could be written as:

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

In this case, the initialization properties for the default version are always used without the need to change the program.

Getting a List of the Standards That Are Installed

When a user is in SAS, he might want to know which standards are registered to the framework. The following code can be used to do this:

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

The data set work.regStds contains a copy of the information from the registered standards table. The main columns are shown below. See the full column listing in Table 3.1 in Chapter 3, "Metadata File Descriptions."

Figure 2.4 Key Columns, work.regStds Data Set

	mnemonic	standardversion	comment	rootpath	isstanderddefault	iscstframework	isdatastandard	supportsvalidation	isxmlstandard
1	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1	Y	N	Y	Y	Y
2	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.	Y	N	Y	Y	N
3	CT	200810	CDISC Terminology, Packages 1, 2A, 2B, LABTEST	&_cstGRoot/standards/cdisc-terminolo	Y	N	N	N	N
4	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework	Y	Y	N	N	N

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

When standards are registered, information about the files and data sets that make up the standard is registered also. If the user wants to see this information, he can use the following:

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

The parameters used in this example specify the standard (CST-FRAMEWORK), the standard version (1.2), and the data set that should be created to contain the information ([work.]sasrefs). If the standard version is omitted, the default version is used. The data set that is returned is a SASReferences file. For the call above, the first few columns of data are shown below:

Figure 2.5 Column Subset, work.sasrefs Data Set

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

Creating Data Sets Used by the Framework

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

```
/*
Create the empty SASReferences data set used in the next step
 */
%cst_createDS(
    _cstStandard=CST-FRAMEWORK,
    _cstStandardVersion=1.2,
    _cstType=control,
    _cstSubType=reference,
    _cstOutputDS=work.sasrefs
    );
```

The type and subtype identify that it is a SASReferences table. The standard and standard version information identify the module to be used. The output is a data set named work.sasrefs.

Creating Table Shells Based on a Data Standard

Data standards, like CDISC-SDTM, have reference metadata that describes the tables and columns that make up the standard. It can be useful and time-saving to create table shells using this metadata. The framework has a mechanism to do this.

```
/*
Create the table shells for the default version
of CDISC SDTM in the work library.
```

```
*/
%cst_createTablesForDataStandard(
    _cstStandard=CDISC-SDTM
    ,_cstOutputLibrary=work
    );
```

This creates the 26 domains described by SDTM V3.1.1 (the default version) in the Work library.

Getting a Copy of the Reference Metadata for a Data Standard

If the user wants a copy of the reference metadata, the framework has a mechanism to do this:

```
/*
Step 1. Create the empty SASReferences data set used in the next step
 */
%cst_createDS(
    _cstStandard=CST-FRAMEWORK,
    _cstStandardVersion=1.2,
    _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='3.1.1';
    * ----- REFERENCE METADATA -----;
    * tables metadata;
    type='referencemetadata';
    subType='table';
    sasRef='work';
    refType='libref';
    memname='refTables';
    output;
    * columns metadata;
    type='referencemetadata';
    subType='column';
    sasRef='work';
    refType='libref';
    memname='refColumns';
    output;
run;
/*
Step 3. Call the macro to get the metadata.
 */
%cst_getStandardMetadata(
    _cstSASReferences=work.sasrefs,
    _cstResultsOverrideDS=&testResultsDS
    );
```

Step 1 uses another macro to create an empty SASReferences data set as work.sasrefs. Step 2 fills in the information that should be returned: the standard (and version) is SDTM V3.1.1; the type and subtypes identify the types of metadata to be returned, and the sasref and memname identify where each data set should be copied to. Step 3 is the macro call that actually does the processing: work.sasrefs is read and the global metadata is used to fulfill the request.

The outcome of this is two data sets: work.reftables and work.refcolumns.

Inserting Information from Registered Standards into a SASReferences File

When a standard version is registered, information about the data sets and files that make up the standard are registered. When a user wants to reference the files that make up the standard, he can have the Toolkit fill in missing information in the SASReferences file. This is useful in a number of cases.

- It removes the need for the programmer to know all of the locations.
- If the global standards library needs to move locations, this can be done without changing all of the SASReferences files that use a standard.
- To change standard versions, the only information that needs to change is the contents of the version column.

For example, the following code creates a partially filled in SASReferences file:

```
/*
Step 1. Initialize the global variables needed by the framework.
*/
%cst_setStandardProperties(
    _cstStandard=CST-FRAMEWORK
    ,_cstStandardVersion=1.2
    ,_cstSubType=initialize
    );
/*
Step 2. Create the empty sasreferences table.
*/
%cst_createDS(
    _cstStandard=CST-FRAMEWORK,
    _cstStandardVersion=1.2,
    _cstType=control,
    _cstSubType=reference,
    _cstOutputDS=sasrefs
    );
/*
Step 3. Fill in minimal information for a series of records
*/
data sasrefs;
    if 0 then set sasrefs;

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

```
    output;
run;
```

The data set looks like:

Figure 2.6 Example SASReferences Data Set

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

Notice that the path and memname columns are missing. The user has specified the standard, version, type, subtype, SASref, and reftype. This information is enough, and the rest is filled in from the registered standards.

The following macro call has Toolkit insert the missing information if it is found in a registered standard:

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

The output data set looks like the following:

Figure 2.7 work.outSASRefs Data Set, with Added Content

	standard	standardversion	type	subtype	SASref	reftype	path	order	memname	comment
1	CST-FRAMEWORK	1.2	lookup		template	libref	&_cstGRoot./standards/cst-framework/c	1	standardlookup	
2	CST-FRAMEWORK	1.2	messages		messages	libref	&_cstGRoot./standards/cst-framework/m	1	messages	
3	CST-FRAMEWORK	1.2	results	results	template	libref	&_cstGRoot./standards/cst-framework/t	1	results	

Maintenance Usage Scenarios

The following paragraphs describe various usage scenarios that the framework accommodates for standards management.

Registering a New Version of a Standard

If the user has a new standard, or he has a new version of an existing standard to be registered to the framework, then the following code shows how this is done:

```
/*
Step 1. Ensure that the macro var pointing to the global library exists.
*/
%cstutil_setcstgroot;
/*
Step 2. Register the standard to the framework's global library
*/
%cst_registerStandard(
    _cstRootPath=%nrstr(&_cstGRoot./standards/myStandard),
    _cstControlSubPath=control,
```

```
    _cstStdDSName=standards,
    _cstStdSASRefsDSName=standardsasreferences);
```

Step 1 ensures that the macro variable that contains the global standards library's path is set. Step 2 performs the registration by passing in the following information:

- The main path to the directory that contains the standard version's files.
- The path to the registration data sets that are used to populate the global standards library's metadata data sets.
- The names of the standard and standardsasreferences data sets. These files are the same structure as the ones in the global standards library's metadata directory.

In this case, the root path uses %nrstr(&_cstGroot) so that the &_cstGroot is registered as a macro variable. This allows the global standards library to be moved or copied without needing to reregister the full path of the new standard.

Setting the Default Version for a Standard

When multiple versions of a standard exist, the first one installed is the default, unless the user opts to set another as the default. The default version is used when the user opts not to provide the information in macro calls or in a SASReferences file. The following code can be used to set the default version:

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

Unregistering a Standard Version

If standards eventually become obsolete and need to be unregistered, the framework provides a way to do this. Unregistering a standard can be necessary during development of custom standards. The following shows an example of how to do this:

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

CHAPTER 3

Metadata File Descriptions

Overview

SAS Clinical Standards Toolkit provides and uses a series of metadata files to support the basic core functions of the Toolkit, as well as supporting specific functionality within the Toolkit. The file content and structure are described below. Use of these files is described in this document.

Standards

The standards data set is used by the framework to store information about a standard version.

Table 3.1 Global Standards Library: Metadata/Standards Data Set

Column	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.
comment ($200)	A description of the registered standard version.
rootpath	The root path for the standard version's directory.

($200)	
isstandarddefault ($1)	Identifies whether the version is the default for the standard. Allows more than one version to be registered while still maintaining a default. Valid values are Y/N.
iscstframework ($1)	Identifies whether the standard version is part of the framework. It can be used to subset the list of registered modules. Valid values are Y/N.
isdatastandard ($1)	Identifies whether the standard version is a data standard. For example, CDISC SDTM versions are data standards, whereas CDISC Terminology is not. Valid values are Y/N.
supportsvalidation ($1)	Identifies whether the standard version supports validation. Valid values are Y/N.
isxmlstandard ($1)	Identifies whether the standard version is based on XML. CDISC-SDTM is not, whereas CDISC- CRTDDS is. Valid values are Y/N.
importxsl ($200)	If the standard version is XML based, then this is the path to the XSL file to import the XML into the SAS representation.
exportxsl ($200)	If the standard version is XML based, then this is the path to the XSL file to export the XML file.
schema ($200)	If the standard is XML based, this is path to the XML schema document that can be used to validate the XML file of that type.

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

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

The global standards library data set contains the following default records:

Figure 3.1 Global Standards Library: Metadata/Standards Data Set Content

	standard	mnemonic	standardversion	comment	rootpath	isstandarddefault	iscstframework
1	CDISC-CRTDDS	CRT	1.0	CDISC CRT-DDS V1.0	&_cstGRoot./standards/cdisc-crtdds-1.0	Y	N
2	CDISC-SDTM	SDTM	3.1.1	CDISC SDTM V3.1.1	&_cstGRoot./standards/cdisc-sdtm-3.1.1	Y	N
3	CDISC-TERMINOLOGY	CT	200810	CDISC Terminology, Packages 1, 2A, 2B, LABTEST	&_cstGRoot/standards/cdisc-terminology	Y	N
4	CST-FRAMEWORK	CST	1.2	Clinical Standards Toolkit Framework	&_cstGRoot./standards/cst-framework	Y	Y

	standard	isdatastandard	supportsvalidation	isxmlstandard	importxsl	exportxsl	schema
1	CDISC-CRTDDS	Y	Y	Y	CRT-DDS/1.0/import/Root.xsl	CRT-DDS/1.0/export/Root.xsl	cdisc-crtdds-1.0.0/define1-0-0.xsd
2	CDISC-SDTM	Y	Y	N			
3	CDISC-TERMINOLOGY	N	N	N			
4	CST-FRAMEWORK	N	N	N			

An example of the global standards library data set used to register a standard can be found at the following location:

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

StandardSASReferences

This table has the same structure as the SASReferences table, which is documented below.

The global standardsasreferences data set provided with the SAS Clinical Standards Toolkit can be found at the following location:

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

A sample of records in the global standardsasreferences data set is provided in the following figure:

Figure 3.2 Global Standards Library: Metadata/StandardSASReferences Data Set Content

	standard	standardversion	type	subtype	SASref	reftype	path
5	CDISC-CRTDDS	1.0	referencemetadata	column	crtref	libref	&_cstGRoot/standards/cdisc-crtdds-1.0/metadata
6	CDISC-CRTDDS	1.0	referencemetadata	table	crtref	libref	&_cstGRoot/standards/cdisc-crtdds-1.0/metadata
7	CDISC-CRTDDS	1.0	referencexml	stylesheet	xslt01	fileref	&_cstGRoot/standards/cdisc-crtdds-1.0/stylesheet
8	CDISC-SDTM	3.1.1	autocall		sdtmauto	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/macros

	standard	standardversion	type	subtype	order	memname	comment
5	CDISC-CRTDDS	1.0	referencemetadata	column	.	reference_columns.sas7bdat	
6	CDISC-CRTDDS	1.0	referencemetadata	table	.	reference_tables.sas7bdat	
7	CDISC-CRTDDS	1.0	referencexml	stylesheet	1	define1-0-0.xsl	Default stylesheet to display define.xml
8	CDISC-SDTM	3.1.1	autocall		1		

An example of the standardsasreferences data set used to register a standard can be found at the following location:

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

Standardlookup

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

Table 3.2 Global Standards Library: Standardlookup Data Set Structure

Column	Description
sasref ($8)	SAS libref
table ($32)	CST table name
column ($32)	CST column name
refcolumn	Associated CST column name

Column	Description
($32)	
refvalue	Associated CST column value
($200)	
value	Unique CST column value
($200)	
default	Default CST column value
($200)	
nonnull	Specify whether CST column value must be nonnull
($1)	
order	CST column value order
(8.)	
comment	Explanatory comments
($200)	

A standardlookup data set is provided for most standards with this release of the SAS Clinical Standards Toolkit, and users might find this data set valuable when defining and registering custom standards to the SAS Clinical Standards Toolkit.

An example of the standardlookup data set can be found at the following location:

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

A sample of records in this example standardlookup data set is provided in the following figure:

Figure 3.3 Global Standards Library: Framework Standardlookup Data Set Content

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

These records show that the SASReferences data set allows a value of referencemetadata for the type column, type must be non-null, and two subtype values (table and column) are allowed when type is referencemetadata. See discussions of the SASReferences data set below for more information on these columns and values.

SASReferences

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

Table 3.3 SASReferences Data Set Structure

Column	Description
standard ($20)	Captures the standard name.. This value should match the standard field in the standards data set found in **C:\cstGlobalLibrary\metadata** and in other metadata files referenced within SASReferences. Examples include CDISC-SDTM, CDISC-CRTDDS. Required.
standardversion ($20)	Captures a specific version of a standard. This value should match one of the standardversion values associated with the standard field in the standards data set found in **C:\cstGlobalLibrary\metadata** and in other metadata files referenced within SASReferences. Examples include 3.1.1, 1.0. Required.
type ($40)	Describes the type of input and output data or metadata. For SAS Clinical Standards Toolkit 1.2, this is a predefined set of values documented in the **cstGlobalLibrary\standards\cst-framework\control\standardlookup** data set and also itemized in Table 5.1. Required.
subtype ($40)	Describes the specific subtype within type for input and output data or metadata. For SAS Clinical Standards Toolkit 1.2, this is a predefined set of values documented in the **cstGlobalLibrary\standards\cst-framework\control\standardlookup** data set and also itemized in Table 5.1. Optional, depending on type.
SASref ($8)	The SAS libref or fileref that is used to refer to the library or file within the SAS Clinical Standards Toolkit SAS process. Should match the values of sasref used in other associated metadata files. (For example, in the Source Columns data set, type=srcmeta.) Required. Must conform to SAS libref or fileref naming conventions.
reftype ($8)	Reference type (libref or fileref). Required.
path ($200)	The path of the library or the path portion of the file reference. If you want to use the default value for a given standard, standardversion, type, or subtype, leave the path blank and the value is added to the &_cstSASRefs working version of SASReferences 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 some environment variable (for example, !sasroot) or SAS macro variable (for example, &studyrootpath).
order (8.)	Processing or concatenation order within type. If this exists, it should be a positive integer with no duplicates within type. Optional, depending on type. Order should be specified if multiple records exist within these types—autocall, fmtsearch, messages; if library concatenation is wanted (multiple librefs within the same value of sasref for a given type); or if there is a need to establish precedence within a given type (for example, look first in this library, then in another library).

Column	Description
memname ($48)	The name of a specific SAS file (data set, catalog) or file that is not SAS (for example, properties or XML file). Filename (null for libraries). Optional, depending on type. As a general rule, memname should be provided if path is provided, except for cases where individual file references are not appropriate (for example, type=autocall and type=sourcedata). If you want to use the default value for a given standard, standardversion, type, or subtype, leave memname blank and the value is added to the &_cstSASRefs working version of SASReferences from the standard-specific standardsasreferences data set. The file suffix for SAS files is optional.
comment ($200)	Explanatory comments. Optional.

The following figure provides a glimpse of the information in a typical SAS Clinical Standards Toolkit SASReferences data set.

Figure 3.4 A Sample SASReferences Data Set

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

From this figure, it is evident that the file contains information about types of data and metadata and where they are located. SAS Clinical Standards Toolkit 1.2 imposes a rigid SASReferences file structure. No additional or fewer columns are allowed, and no changes to column attributes are permitted (for example, altering a column length).

Properties

SAS Clinical Standards Toolkit uses properties files to set default preferences for each process. These properties are a series of name-value pairs that are translated into SAS global macro variables available for the duration of a SAS Clinical Standards Toolkit process. Properties can be defined in any number of files. Both text file and SAS data set formats are supported. All SAS Clinical Standards Toolkit global macro variables, derived from all properties files provided by SAS, are documented in Appendix 1, “Global Macro Variables.”

The table below describes the contents of an example properties file found in `<global standards library directory>/standards/cst-framework/programs/initialize.properties`. Each property (global macro variable) is fully described in Appendix 1.

Table 3.4 Framework Initialize.Properties

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

Messages

SAS Clinical Standards Toolkit provides a messages data set for most supported standards. Each messages data set provides a list of codes and associated text, specific to each standard, and, in some cases, specific actions such as validation, used to report process results. The structure of all message files is described in the following table.

Table 3.5 Messages Data Set Structure

Column	Description
resultid ($8)	The message ID. SAS Clinical Standards Toolkit has adopted a naming convention matching each standard. Resultid values are prefixed with up to a 4-byte prefix (CST for framework messaging, CDISC examples: ODM, SDTM, ADAM, CRT). By convention, the prefix matches the mnemonic field in the standards data set found in `C:\cstGlobalLibrary\metadata`. This prefix is followed by a 4-byte numeric unique within the standard (for example, SDTM1234). Customers can use any other naming convention limited to 8-characters. For CDISC standards supporting validation, resultid should match the checkid from the Validation Master data set. Required.

Column	Description
standardversion ($20)	Captures a specific version of a standard. This value must match one of the standard versions 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 that standard. Examples: 3.1.1, 1.0, ***. If a subsequent version of the standard is released, *** should still be applicable if the check is also valid for the new version. Required.
checksource ($40)	String identifying the source of the message. SAS Clinical Standards Toolkit is used for messages created and generated within SAS Clinical Standards Toolkit. CDISC examples: Janus, JanusFR (FAIL-REJECT), SAS, WebSDM. This field can contain any user-defined value. Required.
sourceid ($8)	Contains a reference identifier for this message from the checksource. Optional.
checkseverity ($40)	The severity as assigned by the checkSource, mapped to the following standardized values: Note (Low), Warning (Medium), Error (High). A value is expected, though not technically required. It is used in reporting.
sourcedescription ($500)	A full description of the validation check associated with the checksource if the source is external to SAS Clinical Standards Toolkit. If checksource is set to CST, 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, these are _cstParm1 and _cstParm2, but the code recognizes any _cst prefix parameter. 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, this default is used. Optional (can be null).
parameter2 ($100)	The message parameter 2 (_cstParm2) default value. If the code using the message does not provide a parameter value, this default is used. Optional (can be null).
messagedetails ($200)	Any additional information that explains the message in greater detail. Optional.

The messages data set provided to support the SAS Clinical Standards Toolkit framework can be found in:

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

The following figure provides an excerpt of records and columns from the Framework messages data set:

Figure 3.5 Framework Messages Data Set

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

For a complete list of messages supporting the SAS Clinical Standards Toolkit framework, see Appendix 2, "Framework Messages." Other message data sets that support the non-framework standards are described in greater detail in this document.

Results

Each SAS Clinical Standards Toolkit process generates a results data set that can optionally be persisted beyond the SAS session based on SASReferences data set settings. Each results data set captures the outcome of specific process actions, using the messages data set to standardize output.

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

Table 3.6 Results Data Set Structure

Column	Description
resultid ($8)	Result identifier. The resultid is a message ID from the standard messages data set (for example, Framework or CDISC SDTM). SAS Clinical Standards Toolkit has adopted a naming convention for resultid matching each standard. Resultid values are prefixed with up to a 4-byte prefix (CST for framework messaging, CDISC examples: ODM, SDTM, ADAM, CRT). By convention, the prefix matches the mnemonic field in the standards data set found in `C:\cstGlobalLibrary\metadata`. This prefix is followed by a 4-byte numeric unique within the standard (for example, SDTM1234). Customers can use any other naming convention limited to 8-characters. Values should be non-null.
checkid ($8)	Validation check identifier. SAS Clinical Standards Toolkit has adopted a naming convention matching the standard to be validated. Checkid values are prefixed with up to a 4-byte prefix (CDISC examples: ODM, SDTM, ADAM, CRT). By convention, the prefix matches the mnemonic field in the standards data set found in `C:\cstGlobalLibrary\metadata`. This prefix is followed by a 4-byte numeric unique within the standard (for example, SDTM1234). Customers can use any other naming convention limited to 8-characters. Values should be non-null for validation processes, otherwise, optional.

Column	Description
resultseq (8.)	Unique invocation of resultid. For validation processes, a counter to indicate the record number within checkid in the Validation Control run-time set of checks. Set to 1, this is incremented only with each repeat invocation of a check. For non-validation processes, generally a constant 1, but reset to 1 with each new invocation of the SAS Clinical Standards Toolkit macro being run when the result record is generated. Values should be non-null positive integers.
seqno (8.)	Sequence number within resultseq. Unique sequence number for results record within each unique value of resultseq. Values should be non-null positive integers.
srcdata ($200)	Source data. String that generally specifies: (for validation) the domains evaluated or check macro used (otherwise) SAS Clinical Standards Toolkit macro being run when the result record is generated Values should be non-null.
message ($500)	Resolved message text from message file. Message includes up to 2 run-time parameter values in message text. Values should be non-null.
resultseverity ($40)	Result severity (for example, warning, error). Info–Informational note Note–Problem detected, low severity Warning–Problem detected, medium severity Warning: Check not run–no assessment able to be made Error–Problem detected, high severity Values should be non-null.
resultflag (8.)	Problem detected? (0=no, otherwise yes) -1=Validation check not run 0=no problem detected (value always 0 when resultseverity=Info) 1=Validation check run, error detected Values should be non-null.
_cst_rc (8.)	Process status (Nonzero, aborted). Values should be non-null.
actual ($240)	Actual value observed. Generally used for validation reporting, provides actual column values found to be in error. Optional.
keyvalues ($2000)	Record-level keys+values. Generally used for validation reporting, provides domain key values for records found to be in error. Optional.
resultdetails ($200)	Basis or explanation for result. Optional.

For examples of a SAS Clinical Standards Toolkit results data set, see Figures 6.9 and 6.10 in Chapter 6, “Validation.”

Additional Metadata Files

The following metadata files, provided by SAS Clinical Standards Toolkit, can be used for specific tasks, and, in some cases (where indicated), the file structures might be unique to the supported or referenced standard. These files are described in detail in this document as indicated.

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. (See Table 3.1 on page 15 for a description of the standards.supportsvalidation field.) This data set should have the columns as defined in Table 6.3 in Chapter 6, "Validation," though additional columns are permitted reflecting user customizations. For each SAS Clinical Standards Toolkit validation process, the set of run-specific checks is captured in a validation_control data set, identical in structure to the validation_master data set, but potentially different only in the number of records (that is, checks) included.

Reference_tables (Source_tables)

Part of the definition of each standard is the itemization of the data tables that define the SAS representation of that standard (and version). The reference_tables data set captures table-level metadata about each reference standard data set. The structure of this data set can be standard specific. For example, Table 6.1 in Chapter 6, "Validation," describes the table metadata for the CDISC SDTM standard. SAS Clinical Standards Toolkit also requires for selected actions a similarly structured source_tables data set that defines study-specific tables. For example, a Toolkit validation process compares the study metadata contained in the source_tables data set with the reference standard metadata contained in the reference_tables data set.

Reference_columns (Source_columns)

As with table metadata, part of the definition of each standard is the itemization of the columns within each data table that define 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 in Chapter 6, "Validation," describes the column metadata for the CDISC SDTM standard. SAS Clinical Standards Toolkit also requires for selected actions a similarly structured source_columns data set that defines study-specific columns. For example, a Toolkit validation process compares the study metadata contained in the source_columns data set with the reference standard metadata contained in the reference_columns data set.

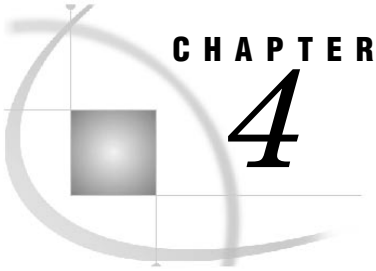
Validation_metrics

Each SAS Clinical Standards Toolkit validation process, based on validation property settings, can optionally generate a summary data set that serves to provide a meaningful denominator for most validation checks so that the relative scope of errors

detected can be more accurately assessed. This data set can optionally be persisted beyond the SAS session based on SASReferences data set settings. For example, Table 6.6 in Chapter 6, “Validation,” describes the metrics metadata for the CDISC-SDTM standard, and Figure 6.2 in Chapter 6 provides sample content for the CDISC-SDTM standard.

CDISC-CRTDDS Style Sheet

A sample XML style sheet (define1-0-0.xsl), copied from http://www.cdisc.org/models/def/v1.0/define1-0-0.xsl, is provided as part of the CDISC-CRTDDS standard supplied by SAS. A define.xml document can be rendered in a human-readable form if it contains an explicit XML style sheet reference, such as to the default style sheet. Alternative style sheets can be used to provide metadata support for CDISC-CRTDDS.



CHAPTER 4

Supported Standards

SAS Representation of Standards

SAS Clinical Standards Toolkit is designed to support any number of varied clinical standards. While initially built to support CDISC (Clinical Data Interchange Standards Consortium) standards, the generic framework allows definition of any type of standard, including HL7 (Health Level 7) messages.

Each SAS Clinical Standards Toolkit standard provides a SAS interpretation of the source guidelines or specification. As such, it is designed to serve as a representative model or template of the source specification.

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

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

  the use of SAS routines to assess how well any user-defined set of data and metadata conforms to the standard

  the use of SAS code to read and derive files in other formats (for example, XML)

  So, each SAS Clinical Standards Toolkit standard serves as an optimized reference standard (from a SAS perspective).
- Users are able to define their own customized standards or are able to modify the existing SAS standards. See “Registering a New Standard” in Chapter 2 for details on how new standards are registered to SAS Clinical Standards Toolkit.

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

Generally, this guide uses the term “reference standard” to refer to the SAS representation of each source specification.

What exactly is a reference standard? The answer depends on several factors, including the complexity of the external source standard, the intended use of the standard, and the user’s preferred implementation methodology. Here are three alternative answers:

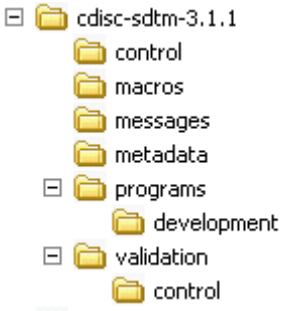
- A limited SAS interpretation of an external standard, defined as one or more SAS files.

  Let’s look at two of the CDISC standards supported in SAS Clinical Standards Toolkit 1.2 as examples. The CDISC-Terminology standard can be represented in its simplest form as either a SAS data set or SAS format catalog of acceptable values. The CDISC-SDTM standard can be represented as a set of domains (SAS data sets) and 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 those standards needed from SAS Clinical Standards Toolkit.

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

  By default, reference standards are specified within the global standards library created with product deployment. Each standard can be unique with regard to the folder hierarchy and set of supporting files. Let’s look again at the CDISC-SDTM standard. The following global standards library folder hierarchy is provided for SDTM:

Figure 4.1 Global Standards Library Folder Hierarchy



The metadata folder contains the data set and column metadata for each supported domain. SAS Clinical Standards Toolkit provides a utility macro (cst_createTablesForDataStandard) that reads this metadata and builds an empty data set for each supported SDTM domain. This information fulfills the requirements specified in the previous bullet. All supporting files required by SAS Clinical Standards Toolkit to support the CDISC-SDTM standard are provided in the remaining folders.

The control folder provides these data sets:

standards–a single-record file providing metadata about the standard

standardlookup–provides acceptable values for many discrete-value columns for a number of standard metadata files

standardsasreferences–a sample or template specification of records describing input or output files relevant to using the standard

The macros folder contains any SAS code specific to the SDTM standard.

The messages folder contains messages associated with the primary CDISC-SDTM task supported by SAS Clinical Standards Toolkit (validation).

The metadata folder provides these data sets:

class_tables–identifies a limited set of column collections specific to one or more SDTM domains; this data set is unique to SDTM

class_columns–identifies the full set of unique column definitions used within the SDTM domains; this data set is unique to SDTM

reference_tables–provides metadata for the 25 data sets (domains) supported for CDISC-SDTM 3.1.1

reference_columns–provides metadata for the 495 columns in the 25 domains supported for CDISC-SDTM 3.1.1

The **`programs`** folder contains several properties files that specify both generic SAS Clinical Standards Toolkit and specific CDISC-SDTM properties translated into SAS global macro variables for a given toolkit process

The **`programs/development`** folder contains a set of utility code modules for building and populating various files for the standard

The **`validation/control`** folder provides check metadata associated with the primary CDISC-SDTM task supported by SAS Clinical Standards Toolkit (validation)

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

- A logical set of files from multiple SAS libraries and optionally multiple standards as defined in the previous two bullets, collated within a single SASReferences data set.

  Each reference standard can also be defined by the files itemized in a SASReferences data set used to perform some specific standard task. The SASReferences data set documents all the input and output files associated with a given 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 supporting a process that builds a CDISC-CRTDDS define.xml file. That SASReferences file might point to some set of 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 provided for the generated define.xml file. In this instance, a broader view of what comprises the CDISC-CRTDDS reference standard must include recognition that the standard must also reference data and metadata from other standards.

Best Practice Recommendation: A sponsor wanting to alter any SAS standard is encouraged to define a new standard instead. This allows seamless updates to SAS standards facilitating Operational Qualification and demo scripts and Technical Support debugging against a fixed standard. Mechanisms are provided for customer requests for updates to be made by SAS for errors in SAS implementation. Sponsors are instead encouraged to define a new standard (which might be an alteration of the SAS standard or an entirely new standard) that can be installed as described in Chapter 2, “Framework.”

CDISC-SDTM 3.1.1

Purpose

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

Released

CDISC-SDTM Model, Final Version 1.1, May 4, 2005

CDISC-SDTM Implementation Guide, Final Version 3.1.1, September 8, 2005

Reference Standard

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

Table 4.1 CDISC-SDTM 3.1.1 Reference Standard Domains

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

Within these 25 domains, 495 columns have been defined. CDISC standards allow for the inclusion and exclusion of some columns (for example, timing variables) at the sponsor's discretion. In addition, CDISC standards do not specify a length for most columns. Therefore, any implementation of a CDISC standard requires interpretation of that standard that might lead to differences in the implementation of that standard. Each sponsor derives a set of reference standards based on internal conventions and experiences and discussions with regulatory authorities.

The domain and column metadata that constitute the SAS representation of CDISC-SDTM 3.1.1 are provided in (or derivable from) the global standards library in these formats:

- as derivable empty data sets (using the utility macro cst_createTablesForDataStandard)
- as SAS DATA step code (see the cstutil_createSDTMdomains.sas code module in the **`programs/development`** folder)
- as table metadata (reference_tables in the standard metadata folder)
- as column metadata for each domain (reference_columns in the standard metadata folder)

A secondary, though equally important goal of the SAS Clinical Standards Toolkit CDISC-SDTM reference standard is to provide metadata and code to validate both the structure and content of the SDTM domains. To enable this, supplemental files supporting SDTM validation processes include the following global standards library files:

- The validation_master data set in the **`validation/control`** folder contains the superset of checks validating domain structure and content
- The messages data set in the messages folder provides error messaging for all validation_master checks
- SAS code in the macros folder provides SDTM-specific code that augments that supplied in the primary SAS Clinical Standards Toolkit autocall library

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

CDISC-CRTDDS 1.0

Purpose

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

Released

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 Harmonization (ICH) electronic Common Technical Document (eCTD) specifications. Within these new specifications, the FDA expanded the acceptable file types to include XML.

Reference Standard

The domain and column metadata that constitute the SAS representation of CDISC-CRTDDS 1.0 are provided in (or derivable from) the global standards library in these formats:

- as derivable empty data sets (using the utility macro cst_createTablesForDataStandard)
- as table metadata for 39 data sets (reference_tables in the standard metadata folder)
- as column metadata for 176 columns within the 39 data sets (reference_columns in the standard metadata folder)

As a general statement, the SAS representation of the CDISC-CRTDDS standard is patterned to match the XML element (data sets) and attribute (columns) structure of define.xml. For example, (CDISC-SDTM) domain-level metadata is represented by a define.xml ItemGroupDef element and is captured in the ItemGroupDefs SAS data set. This is illustrated in the following code and table that represent the TE domain metadata.

```
<ItemGroupDef OID="docroot.IG.TE"
    Name="TE"
    Repeating="Yes"
    IsReferenceData="Yes"
    Purpose="Tabulation"
    def:Label="Trial Elements"
    def:Structure="One record per element"
    def:DomainKeys="STUDYID, DOMAIN, ETCD"
    def:Class="Trial Design"
    def:ArchiveLocationID="ArchiveLocation.te">
    !-- All ItemRefs would be listed here -->
       <def:leaf ID="ArchiveLocation.te"
       xlink:href="te.xpt"> <def:title>te.xpt</def:title>
    </def:leaf>
</ItemGroupDef>
```

Table 4.2 Sample Data Set Representation: ItemGroupDefs.sas7bdat

Column	Value
OID	docroot.IG.TE
Name	TE
Repeating	Yes
IsReferenceData	Yes
SASDatasetName	
Domain	
Origin	
Role	
Purpose	Tabulation
Comment	
Label	Trial Elements
Class	Trial Design
Structure	One record per element
DomainKeys	STUDYID, DOMAIN, ETCD
ArchiveLocationID	ArchiveLocation.te
FK_MetaDataVersion	

The complete set of 39 tables forming the SAS Clinical Standards Toolkit 1.2 SAS representation of CDISC-CRTDDS is presented in Table 4.3 on page 34.

Table 4.3 Data Sets in SAS Representation, CDISC-CRTDDS 1.0 Standard

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

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

Field lengths within the 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 according to the table below, which has distilled the standard data types into core types. When in doubt, larger lengths have been chosen in an attempt to ensure that no data loss occurs with the Toolkit's pre-installed data sets. Production tables might be compressed using SAS mechanisms to preserve disk space, if the user prefers.

Table 4.4 CDISC-CRTDDS Default Lengths by Data Type

Type Name	Length	Description
oid	64	A unique object identifier or a reference
text	2000	A character field capable of accommodating 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

Below is the complete listing of the data sets with member columns that will comprise the CRT-DDS 1.0.0 data in SAS Clinical Standards Toolkit 1.2.

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

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

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

Per the standard, which is very flexible, only the DefineDocument data set, containing valid values for the FileOID and FileType variables, is needed to create a minimal but valid CRT-DDS-compliant XML document.

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

Table 4.5 CRT-DDS SAS Table Construction

Data Set Name	Variable Name	SAS Data Type	Length (if char)
DefineDocument			
	**FileOID [X]	character	64 (oid)
	Archival	character	3
	AsOfDateTime	Numeric (dateTime)	8
	Description	character	2000 (text)
	FileType [X]	character	13
	Granularity	character	15
	Id	character	64 (oid)
	ODMVersion	character	2000 (text)
	Originator	character	2000 (text)
	PriorFileOID	character	64 (oid)
	SourceSystem	character	2000 (text)
	SourceSystemVersion	character	2000 (text)
Study			
	**OID [X]	character	64 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	StudyName [X]	character	128 (name)
	StudyDescription [X]	character	2000 (text)
	ProtocolName [X]	character	128 (name)
	^^FK_DefineDocument [DefineDocument] [X]	character	64 (oid)
MeasurementUnits			
	**OID [X]	character	64 (oid)
	Name [X]	character	128 (name)
	^^FK_Study [Study] [X]	character	64 (oid)
MUTranslatedText			
	TranslatedText	character	2000 (text)
	lang	character	128 (name)
	^^FK_MeasurementUnits [MeasurementUnits][X]	character	64 (oid)
MetaDataVersion			
	**OID [X]	character	64 (oid)
	Name [X]	character	128 (name)
	Description	character	2000 (text)
	IncludedOID	character	64 (oid)
	IncludedStudyOID	character	64 (oid)
	DefineVersion [X]	character	2000 (text)
	StandardName [X]	character	2000 (text)
	StandardVersion [X]	character	2000 (text)
	^^FK_Study [Study] [X]	character	64 (oid)
AnnotatedCRFs			
	DocumentRef	character	2000 (text)
	^^leafID [MDVLeaf] [X]	character	64 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
SupplementalDocs			
	DocumentRef	character	2000 (text)
	^^leafID [MDVLeaf] [X]	character	64 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
MDVLeaf			
	**ID [X]	character	64 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	href	character	512 (path)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
MDVLeafTitles			
	title	character	2000 (text)
	^^FK_MDVLeaf [MDVLeaf] [X]	character	64 (oid)
ComputationMethods			
	**OID [X]	character	64 (oid)
	method	character	2000 (text)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
ValueLists			
	**OID [X]	character	64 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
ValueListItemRefs			
	^^ItemOID [ItemDefs] [X]	character	64 (oid)
	OrderNumber	numeric	8
	Mandatory [X]	character	3
	KeySequence	numeric	8
	^^ImputationMethodOID [ImputationMethods]	character	64 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	Role	character	128 (name)
	^^RoleCodeListOID [CodeLists]	character	64 (oid)
	^^FK_ValueLists [ValueLists] [X]	character	64 (oid)
ProtocolEventRefs			
	Mandatory [X]	character	3
	OrderNumber	numeric	8
	^^StudyEventOID [StudyEventDefs] [X]	character	64 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
StudyEventDefs			
	**OID [X]	character	64 (oid)
	Category	character	2000 (text)
	Name [X]	character	128 (name)
	Repeating [X]	character	3
	Type [X]	character	11
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
StudyEventFormRefs			
	^^FormOID [FormDefs] [X]	character	64 (oid)
	Mandatory [X]	character	3
	OrderNumber	numeric	8
	^^FK_StudyEventDefs [StudyEventDefs] [X]	character	64 (oid)
FormDefs			
	**OID [X]	character	64 (oid)
	Name [X]	character	128 (name)
	Repeating [X]	character	3
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
FormDefItemGroupRefs			
	^^ItemGroupOID [ItemGroupDefs] [X]	character	64 (oid)
	Mandatory [X]	character	3
	OrderNumber	numeric	8
	^^FK_FormDefs [FormDefs] [X]	character	64 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
FormDefArchLayouts			
	**OID [X]	character	64 (oid)
	PdfFileName [X]	character	512 (path)
	^^PresentationOID [Presentation]	character	64 (oid)
	^^FK_FormDefs [FormDefs] [X]	character	64 (oid)
ItemGroupDefs			
	**OID [X]	character	64 (oid)
	Name [X]	character	128 (name)
	Repeating [X]	character	3
	IsReferenceData	character	3
	SASDatasetName	character	8
	Domain	character	2000 (text)
	Origin	character	2000 (text)
	Role	character	128 (name)
	Purpose	character	2000 (text)
	Comment	character	2000 (text)
	Label [X]	character	2000 (text)
	Class	character	2000 (text)
	Structure	character	2000 (text)
	DomainKeys	character	2000 (text)
	^^ArchiveLocationID [ItemGroupLeaf] [X]	character	2000 (text)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
ItemGroupDefItemRefs			
	^^ItemOID [ItemDefs] [X]	character	64 (oid)
	Mandatory [X]	character	3
	OrderNumber	numeric	8
	KeySequence	numeric	8
	^^ImputationMethodOID [ImputationMethods]	character	64 (oid)
	Role [X]	character	128 (name)
	^^RoleCodeListOID [CodeLists]	character	64 (oid)
	^^FK_ItemGroupDefs [ItemGroupDefs][X]	character	64 (oid)
ItemGroupAliases			
	Context [X]	character	2000 (text)
	Name [X]	character	2000 (text)
	^^FK_ItemGroupDefs	character	64 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	[ItemGroupDefs] [X]		
ItemGroupLeaf			
	**ID [X]	character	64 (oid)
	href	character	512 (path)
	^^FK_ItemGroupDefs [ItemGroupDefs] [X]	character	64 (oid)
ItemGroupLeafTitles			
	title	character	2000 (text)
	^^FK_ItemGroupLeaf [ItemGroupLeaf] [X]	character	64 (oid)
ItemDefs			
	**OID [X]	character	64 (oid)
	Name [X]	character	128 (name)
	DataType [X]	character	8
	Length	numeric	8
	SignificantDigits	numeric	8
	SASFieldName	character	8
	SDSVarName	character	8
	Origin	character	2000 (text)
	Comment	character	2000 (text)
	^^CodeListRef [CodeLists]	character	64 (oid)
	Label	character	2000 (text)
	DisplayFormat	character	2000 (text)
	^^ComputationMethodOID[ComputationMethods]	character	64 (oid)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
ItemQuestionTranslated Text			
	TranslatedText	character	2000 (text)
	lang	character	17
	^^FK_ItemDefs [ItemDefs] [X]	character	64 (oid)
ItemQuestionExternal			
	Dictionary	character	2000 (text)
	Version	character	2000 (text)
	Code	character	2000 (text)
	^^FK_ItemDefs [ItemDefs] [X]	character	64 (oid)

Data Set Name	Variable Name	SAS Data Type	Length (if char)
ItemMURefs			
	^^MeasurementUnitOID [MeasurementUnits] [X]	character	64 (oid)
	^^FK_ItemDefs [ItemDefs] [X]	character	64 (oid)
ItemRangeChecks			
	**OID [X]	character	64 (oid)
	Comparator [X]	character	5
	SoftHard [X]	character	4
	^^MURefOID [MeasurementUnits]	character	64 (oid)
	^^FK_ItemDefs [ItemDefs] [X]	character	64 (oid)
ItemRangeCheckValues			
	CheckValue	character	512 (value)
	^^FK_ItemRangeChecks [ItemRangeChecks] [X]	character	64 (oid)
RCErrorTranslatedText			
	TranslatedText	character	2000 (text)
	lang	character	17
	^^FK_ItemRangeChecks [ItemRangeChecks] [X]	character	64 (oid)
ItemRole			
	Name	character	2000 (text)
	^^FK_ItemDefs [ItemDefs] [X]	character	64 (oid)
ItemAliases			
	Context [X]	character	2000 (text)
	Name [X]	character	2000 (text)
	^^FK_ItemDefs [ItemDefs] [X]	character	64 (oid)
ItemValueListRefs			
	^^ValueListOID [ValueLists] [X]	character	64 (oid)
	^^FK_ItemDefs [ItemDefs] [X]	character	64 (oid)
CodeLists			
	**OID [X]	character	64 (oid)
	Name [X]	character	128 (name)
	DataType [X]	character	7
	SASFormatName	character	8

Data Set Name	Variable Name	SAS Data Type	Length (if char)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
ExternalCodeLists			
	Dictionary	character	2000 (text)
	Version	character	2000 (text)
	^^FK_CodeLists [CodeLists] [X]	character	64 (oid)
CodeListItems			
	**OID [X]	character	64 (oid)
	CodedValue	character	512 (value)
	^^FK_CodeLists [CodeLists] [X]	character	64 (oid)
	Rank	numeric	8
CLItemDecodeTranslatedText			
	TranslatedText	character	2000 (text)
	lang	character	17
	^^FK_CodeListItems [CodeListItems] [X]	character	64 (oid)
ImputationMethods			
	**OID [X]	character	64 (oid)
	method	character	2000 (text)
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)
Presentation			
	**OID [X]	character	64 (oid)
	presentation	character	2000 (text)
	lang	character	17
	^^FK_MetaDataVersion [MetaDataVersion] [X]	character	64 (oid)

The CDISC-CRTDDS reference standard is designed to support reading (and representing) a define.xml file, build a define.xml file, and to validate both the structure and content of the SAS representation of a given define.xml, as well as the structural integrity of the define.xml file itself. To support this functionality, the following supplemental files are included in the global standards library CDISC-CRTDDS folder hierarchy:

- A SAS format catalog (crtddsct.sas7bcat) in the formats folder, providing valid values for selected columns in the 39-tables 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 tables.

- The messages data set in the messages folder provides error messaging for all validation_master checks.
- SAS code in the macros folder provides CRT-DDS-specific code that augments that supplied in the primary SAS Clinical Standards Toolkit autocall library.
- The style sheet folder contains the file define1-0-0.xsl (copied from `http://www.cdisc.org/models/def/v1.0/define1-0-0.xsl`). A define.xml document can be rendered in a human-readable form if it contains an explicit XML style sheet reference, such as to the default style sheet.

It is this set of files in whole or part that define the CDISC-CRTDDS reference standard.

CDISC-Terminology-200810

Purpose

CDISC terminology is designed to support standardization of values for designated columns in data submitted to the regulatory agencies. Such standardization facilitates loads into regulatory databases, data review, and analysis. Initial standardization of values has primarily been in support of SDTM submission data and CDISC CDASH (Clinical Data Acquisition Standards Harmonization) development of standardized data collection instruments.

Released

SDTM Package 1, SDTM Package 2A, Labtest Package 1, SDTM Package 2B, and Labtest Package 2: September 24, 2008

Reference Standard

CDISC terminology is maintained and distributed as part of the National Cancer Institute (NCI) Enterprise Vocabulary Services Thesaurus. See the “References” section in Chapter 1 for more information. Periodically, the set of support CDISC terminology is updated to include the work of numerous terminology project teams, in the form of new packages or sets of terminology. In September 2008, the CDISC Terminology Team finalized the production release of SDTM Package 2B and Labtest Package 2. This terminology set was combined with prior production releases of SDTM Package 1, SDTM Package 2A, and Labtest Package 1, bringing the total number of CDISC (primarily SDTM) production terms to nearly 2300.

A snapshot of the NCI Thesaurus was taken in October 2008 in support of SAS Clinical Standards Toolkit 1.2. This snapshot, representing the above packages, has been assigned the standard name of CDISC-Terminology, with a standardversion value of 200810.

The CDISC-Terminology standard includes the following files, all distributed in the global standards library (paths assume deployment of the global standards library to `C:\cstGlobalLibrary`):

Table 4.6 Files Supporting the CDISC Terminology Standard

Path	File/Description
C:\cstGlobalLibrary\standards\cdisc-terminology-200810\programs\development	
	SDTM_ProductionTerminologyupdated_24Sept2008.xls–the source Excel spreadsheet from NCI
	createstandarddatasets.sas–sample SAS code to create files for the standard
C:\cstGlobalLibrary\standards\cdisc-terminology-200810\formats	
	cterms.sas7bcat–a SAS catalog representation of the spreadsheet
	cterms.sas7bdat-a SAS data set representation of the spreadsheet

The following codelists or SAS formats are represented in the snapshot:

Table 4.7 Supported Codelists/Formats

Codelist/Format Name	Codelist/Format Description	Unique Values
ACN	Action Taken with Study Treatment	7
AESEV	Severity/Intensity Scale for Adverse Events	3
AGESPAN	Age Span	8
AGEU	Age Unit	5
COUNTRY	Country	243
DICTNAM	Dictionary Name	7
DOMAIN	Domain Abbreviation	43
DSCAT	Category for Disposition Event	3
EGMETHOD	ECG Test Method	22
EGSTRESC	ECG Result	107
EGTEST	ECG Test Name	46
EGTESTCD	ECG Test Code	46
ETHNIC	Patient Ethnic Group	4
FREQ	Frequency	49
FRM	Pharmaceutical Dosage Form	168
IECAT	Category for Inclusion/Exclusion	2
LBTEST	Laboratory Test Name	271
LBTESTCD	Laboratory Test Code	271
LOC	Anatomical Location	303
MARISTAT	Marital Status	9
NCOMPLT	Completion/Reason for Non-Completion	16
ND	Not Done	1
NY	No Yes Response	4
OUT	Outcome of Event	6
POSITION	Position	10
RACE	Race	5
ROUTE	Route of Administration	112

Codelist/Format Name	Codelist/Format Description	Unique Values
SCCD	Subject Characteristic Code	7
SEX	Sex	4
SEXPOP	Sex of Participants	3
SIZE	Size	3
SKINCLAS	Skin Classification	6
SKINTYP	Skin Type	3
SOC	CDISC System Organ Class	26
STENRF	Relation to Reference Period	5
TBLIND	Trial Blinding Schema	3
TCNTRL	Control Type	3
TDIGRP	Diagnosis Group	1
TINDTP	Trial Indication Type	5
TOXGR	Common Terminology Criteria for Adverse Events	5
TPHASE	Trial Phase	12
TSPARM	Trial Summary Parameter Test Name	22
TSPARMCD	Trial Summary Parameter Test Code	22
TTYPE	Trial Type	8
UNIT	Unit	304
VSRESU	Units for Vital Signs Results	14
VSTEST	Vital Signs Test Name	13
VSTESTCD	Vital Signs Test Code	13

Support for Upcoming Standards

From a CDISC perspective, the following standards are candidates for future SAS Clinical Standards Toolkit support. Check with your on-site SAS support personnel and with SAS product management for more information.

CDISC-SDTM 3.1.2

CDISC announced in March 2009 the availability of the SDTM 1.2 and the SDTM Implementation Guide for Human Clinical Trials (SDTMIG v.3.1.2). CDISC also posted with this announcement the following statement:

> “As most of you will be aware the FDA are not able to immediately accept this new version. The FDA has stated that "we intend to move to 3.1.2 and that the timing on that will be determined by the availability of updated software". CDISC expects this to be around the middle of 2009 but it is for FDA to make further statements on this. In the interim users are advised to discuss potential use of V3.1.2 with the FDA as part of their normal discussions surrounding a regulatory submission.”

CDISC ADAM 2.1

The CDISC Analysis Dataset Model (ADaM) defines a standard for analysis data sets that are to be submitted in support of the statistical analyses performed by the sponsor. Once ADaM 2.1 and ADaM Implementation Guide, Version 1.0 are finalized, provision of a SAS representation of ADaM will be considered.

CDISC-Terminology 2009*xx*

Updates to the NCI Thesaurus for CDISC-Terminology were posted on May 1, 2009, adding support of SDTM Package 3 and Labtest Package 3. Subsequent updates to existing terminology were posted in July 2009, and future updates are likely. Snapshots of the NCI Thesaurus will be taken before future SAS Clinical Standards Toolkit releases to capture the cumulative NCI Thesaurus support of CDISC-Terminology.

CHAPTER 5

SASReferences File

Overview

The primary purpose of SAS Clinical Standards Toolkit is to support submission of SAS processes optimized to use the SAS Clinical Standards Toolkit metadata framework and infrastructure. The key metadata file supporting this functionality is the SASReferences file. This SAS data set essentially identifies all the key inputs and outputs for any given SAS Clinical Standards Toolkit process. As such, each unique process can have an associated unique SASReferences file. However, SAS Clinical Standards Toolkit offers a number of standardization aids that make the use of more generic SASReferences files preferable.

The required SASReferences file structure is provided in Table 2.3 and example content is provided in Figure 2.3.

Building a SASReferences File

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

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

1. Use sample SASReferences files provided with SAS Clinical Standards Toolkit. These demonstrate required and optional contents for specific tasks. For example, the sample process that demonstrates CDISC-SDTM validation functionality uses the SASReferences file found at the following location in SAS 9.1.3:

   ```
   !sasroot/../CSTSDTM311\9.1.3\sample\cdisc-sdtm-3.1.1\sascstdemodata\control
   ```

   An excerpt of this sample file is provided in Figure 3.4 in Chapter 3, “Metadata File Descriptions.”

2 SAS Clinical Standards Toolkit provides SAS code used to derive template and sample metadata files, including SASReferences. Look in the standard-specific (signified by <*standard*>) folder hierarchy:

```
C:\cstGlobalLibrary\standards\<standard>\programs\development
```

3 SAS Clinical Standards Toolkit provides SASReferences templates (either zero-observation data sets or data sets containing records that must be modified) for use. An empty SASReferences data set can be found in:

```
C:\cstGlobalLibrary\standards\cst-framework\templates
```

SAS Clinical Standards Toolkit also provides default SASReferences data sets for each supported standard. These defaults contain records commonly required for certain SAS Clinical Standards Toolkit tasks (such as validation). However, all records required might not be present; or all records provided might not be required for selected tasks; and SAS librefs and filerefs, paths, and memname values might require modification. For example, see the standardsasreferences data set found in:

```
C:\cstGlobalLibrary\standards\cdisc-sdtm-3.1.1\control
```

4 SAS Clinical Standards Toolkit does provide utility macros to build and return the data sets described in step 3 above:

%cst_getStandardSASReferences returns the requested standardsasreferences data set

%cst_createds can be used to return an empty SASReferences data set

Use of these utility macros is illustrated in the following discussion.

Two SASReferences fields—type and subtype—are used to define each type of input and output file. The values for these fields are restricted for SAS Clinical Standards Toolkit to those itemized in Table 5.1.

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

Type	Subtypes	Comments
autocall		One record for each library containing macros to be included in the SAS autocall path. Typically, this would include one record for each standard referenced in the SASReferences file, excluding SAS Clinical Standards framework (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, would require an autocall record for each library.
control	validation, sasreferences, or reference	This type identifies any run-time process control file, including the SASReferences data set itself (that is, this is a self-documentation record). For SAS Clinical Standards Toolkit Validation processes, the Validation Control data set specifying the validation checks to be run is identified with subtype=validation.
cterms	format, code, or data	Not used in SAS Clinical Standards Toolkit 1.2, but designed to provide a run-time specification of controlled terminology (most often a specific data set or catalog), much like type=control provides run-time specification of control files. Under review for deprecation.

Type	Subtypes	Comments
fmtsearch		Provides a way to build the format search path for a validation process. SAS Clinical Standards Toolkit sets the SAS fmtsearch option based on each record, specifying a SAS catalog using the order=n sequence. Not provided by default in standardsasreferences; user specification is required. Type=fmtsearch is optional unless one or more checks are to be run assessing value compliance against a SAS format.
messages		Identifies one or more messages data sets associated with each SAS Clinical Standards Toolkit standard. Provided by default in standardsasreferences; user specification is necessary only with user customizations requiring new or modified messages. SAS Clinical Standards Toolkit populates the data set referenced by the global macro variable &_cstMessages with all messages data sets included in SASReferences. Required for each standard.
referencecontrol		Identifies the standard-supplied master superset of supported validation checks. While this is key metadata, it is not typically referenced at run time and need not be included. It is the Validation Control file identified with type-control subtype=validation that must be present.
referencemetadata	column or table	Identifies the SAS data sets (sasref.memname) containing the column and tables metadata for a standard version. Provided by default in standardsasreferences; user specification required only to override the default for the standard. Records for both subtypes are required.
classmetadata	column or table	Identifies the SAS data sets (sasref.memname) containing the column and tables metadata for a set of specific CDISC-SDTM template data sets used to build standard SDTM-compliant data sets. Provided by default in standardsasreferences. Optional.
referencecterm		Identifies a SAS data set (sasref.memname) most often containing controlled terminology in a data set rather than SAS format. Example: medDRA. Type=referencecterm is optional unless one or more checks are to be run assessing value compliance against a SAS data set.
results	results or validationresults, metrics or validationmetrics	Specifies the persisted location of the results and metrics data sets generated by the SAS Clinical Standards Toolkit process. Metrics is specific only to SAS Clinical Standards Toolkit validation processes and is optional depending on property settings. A **`results/validationresults`** record is required.
resultspackage	xml or log	Not used in SAS Clinical Standards Toolkit 1.2, but designed to bundle a set of process inputs and outputs together for later access.
sourcedata		Defines the location of the data for a specific study. Required for validation processes if one or more checks are to be run accessing a specific source data domain.
sourcemetadata	column or table	Identifies the SAS data sets (sasref.memname) containing the column and tables metadata for a given study or set of source data. Not provided by default in standardsasreferences; user specification is required. Records for both subtypes are required.
standards	registeredstandards or registeredsasreferences	Identifies the template for the registered standards and sasreferences data sets, respectively. Used by the framework

Type	Subtypes	Comments
		when the global metadata library is created. Not used in post-deployment processes.
properties	validation or initialize	Used to initialize a standard version's required macro variables. Specification within SASReferences is optional. (These can be defined with calls to %cst_setstandardproperties or %cst_setproperties instead.) Each standard should have at least one (initialize) properties file, and can have any number of additional files as needed. A subtype=validation is specific to SAS Clinical Standards Toolkit validation processes.
lookup		Identifies a data set (standardlookup) associated with each SAS Clinical Standards Toolkit standard containing valid values for discrete metadata fields. Provided by default in standardsasreferences; required for each standard. For example, the valid values for type and subtype documented in this table have been defined in one or more SAS Clinical Standards Toolkit standardlookup data sets.
externalxml	xml	Identifies an external XML file. Depending upon the standard version and the subsequent macro that is called, this file can be read or written. Using CDISC-CRTDDS as an example, this specifies the define.xml that is created when the %crtdds_write() macro is called. When %crtdds_read() is supported, this will identify the XML file to be read.
referencexml	stylesheet	Identifies the directory and filename of an XML style sheet. In the production of CDISC-CRTDDS XML files, this should point to the style sheet to be copied into the directory with the XML file.

The primary function of the SASReferences file is to define SAS Clinical Standards Toolkit process inputs and outputs. What information does the process need to reference, what does the process produce, and where does the information come from and go? The "what" information is governed by the use of the type and subtype fields. The "where" part of the equation is provided in the path and memname fields.

Does every instance of the SASReferences file require provision of a specific path and filename? While this is certainly allowed, it is not required. In the options described above on how to build a SASReferences file, a call to the following macro was described:

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

This call produces this SASReferences file:

Figure 5.1 Standard SASReferences for CST-FRAMEWORK

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

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

Now, let's look at the information returned in a call to %cst_getStandardSASReferences for the CDISC-SDTM standard, as shown in Figure 5.2 on page 51.

```
%cst_getStandardSASReferences(_cstStandard=CDISC-
SDTM,_cstStandardVersion=3.1.1, _cstOutputDS=sasreferences);
```

Figure 5.2 Standard SASReferences for CDISC-SDTM

	standard	standardversion	type	subtype	SASref	reftype	path	order	memname
1	CDISC-SDTM	3.1.1	autocall		sdtmauto	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/macros	1	
2	CDISC-SDTM	3.1.1	classmetadata	column	sdtmcls	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/metadata	.	class_columns.sas7bdat
3	CDISC-SDTM	3.1.1	classmetadata	table	sdtmcls	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/metadata	.	class_tables.sas7bdat
4	CDISC-SDTM	3.1.1	lookup		sdtmctrl	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/control	.	standardlookup.sas7bdat
5	CDISC-SDTM	3.1.1	messages		sdtmmsg	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/messages	1	messages.sas7bdat
6	CDISC-SDTM	3.1.1	properties	initialize	sdtmprop	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/programs	1	initialize.properties
7	CDISC-SDTM	3.1.1	properties	validation	sdtmprp2	fileref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/programs	1	validation.properties
8	CDISC-SDTM	3.1.1	referencecontrol	validation	sdtmcntl	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/validation\control	.	validation_master.sas7bdat
9	CDISC-SDTM	3.1.1	referencemetadata	column	sdtmref	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/metadata	.	reference_columns.sas7bdat
10	CDISC-SDTM	3.1.1	referencemetadata	table	sdtmref	libref	&_cstGRoot/standards/cdisc-sdtm-3.1.1/metadata	.	reference_tables.sas7bdat

A comparison of Figure 5.1 on page 50 and Figure 5.2 suggest 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 to the task (validation of SDTM domains). SAS Clinical Standards Toolkit validation processes require specification of a comparative reference standard, and we now see reference 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.

Let's now look at an actual SASReferences file built to support CDISC-SDTM validation. As noted above, the sample process that demonstrates CDISC-SDTM validation functionality uses the SASReferences file found in the following location in SAS 9.1.3:

```
!sasroot/../SASClinicalStandardsToolkitSDTM311\9.1.3\sample\cdisc-sdtm-
3.1.1\sascstdemodata\control
```

The complete file contents are shown in Figure 5.3 on page 52.

Figure 5.3 Sample SASReferences for CDISC-SDTM Validation

	standard	standardversion	type	subtype	SASref	reftype	path	order	memname
1	CDISC-SDTM	3.1.1	sourcedata		srcdata	libref	&studyRootPath\data	.	
2	CDISC-SDTM	3.1.1	sourcemetadata	table	srcmeta	libref	&studyRootPath\metadata	.	source_tables.sas7bdat
3	CDISC-SDTM	3.1.1	sourcemetadata	column	srcmeta	libref	&studyRootPath\metadata	.	source_columns.sas7bdat
4	CDISC-SDTM	3.1.1	referencecontrol	validation	refcntl	libref		.	
5	CDISC-SDTM	3.1.1	referencemetadata	table	refmeta	libref		.	
6	CDISC-SDTM	3.1.1	referencemetadata	column	refmeta	libref		.	
7	CDISC-SDTM	3.1.1	autocall		sdtmcode	fileref	&_cstGRoot\standards\cdisc-sdtm-3.1.1\	1	
8	CDISC-SDTM	3.1.1	fmtsearch		srcfmt	libref	&studyRootPath\terminology\formats	1	formats.sas7bcat
9	CDISC-TERMINOL	200810	fmtsearch		cstfmt	libref	&_cstGRoot\standards\cdisc-terminology	2	cterms.sas7bcat
10	CUSTOM		referencecterm		ctref	libref	&studyRootPath\terminology\coding-dicti	1	meddra.sas7bdat
11	CDISC-SDTM	3.1.1	control	validation	control	libref	&studyRootPath\control	.	validation_control.sas7bdat
12	CDISC-SDTM	3.1.1	control	reference	control	libref	&studyRootPath\control	.	sasreferences.sas7bdat
13	CDISC-SDTM	3.1.1	messages		sdtmmsg	libref	&_cstGRoot\standards\cdisc-sdtm-3.1.1\	1	messages.sas7bdat
14	CST-FRAMEWORK	1.2	messages		cstmsg	libref	&_cstGRoot\standards\cst-framework\m	2	messages.sas7bdat
15	CDISC-SDTM	3.1.1	properties	validation	valprop	fileref	&studyRootPath\programs	1	validation.properties
16	CDISC-SDTM	3.1.1	results	validationresults	results	libref	&studyRootPath\results	.	validation_results.sas7bdat
17	CDISC-SDTM	3.1.1	results	validationmetrics	results	libref	&studyRootPath\results	.	validation_metrics.sas7bdat

Table 5.2 Explanation of Sample SASReferences for CDISC-SDTM Validation Data

Lines	Comment
1	This is a new record (type) not present in the template files (standardsasreferences). It defines the location of the study (source) data. Note the use of &studyRootPath, which, coupled with the assumption of a fixed folder hierarchy, allows portability across studies. Memname is not relevant for a library of SAS data sets.
2-3	Source metadata references are also new and follow the style used in line 1 for source data. Note the same SASref is used for multiple subtypes within a single type because we are referencing two different named SAS data sets from the same folder.
4-6	These records point to the reference standard for CDISC-SDTM 3.1.1, but note that unlike the template defaults shown in Figure 5.2 on page 51, path and memname are blank. This convention tells SAS Clinical Standards Toolkit to do a look up to the CDISC-SDTM 3.1.1 standardsasreferences file and to 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. That lookup results in the inclusion of the path and memname values as defined in Figure 5.2 on page 51.
7	This record instructs SAS Clinical Standards Toolkit to add any SDTM-specific macros to the autocall path.
8-9	Fmtsearch records are also new. Two standards are referenced to create a format search path, looking first at the SDTM study-specific formats catalog, and then the more general CDISC-Terminology cterms catalog. Note the precedence is set by the order column.
10	This record references a medDRA data set that is maintained in the study-specific hierarchy. The standard field value CUSTOM indicates this standard is not supported by SAS Clinical Standards Toolkit.
11	This record points to the set of validation checks to be run in this specific validation assessment. The framework default values for SASref, path, and memname have been overridden.
12	This record should document the name and location of this file. This information is used in sample reports discussed in this guide.
13-14	These records are identical to the FRAMEWORK and CDISC-SDTM standardsasreferences records.
15	The validation properties path has been modified to point to a location in the study hierarchy rather than the global standards library defined in the standardsasreferences file.
16-17	Results are to be persisted to a location in the study hierarchy.

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

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

This macro clones the template. New records can be added using a variety of coding methods, including the PROC SQL technique illustrated above. There is no requirement that the SASReferences file live outside the SAS Work area and be persisted beyond the conclusion of a SAS Clinical Standards Toolkit process. Taking this approach, however, limits future capabilities such as process reruns and reporting.

How is a SASReferences File Used?

Once a SASReferences file has been created for a given task, three key steps occur:

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

Upon completion of these steps, a SAS environment has been established to support subsequent SAS Clinical Standards Toolkit tasks.

Communicating the Filename and Location to SAS Clinical Standards Toolkit

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

- _cstSASRefsLoc–provides the path to the SAS library containing the file
- _cstSASRefsName–provides the SASReferences data set name in _cstSASRefsLoc
- _cstSASRefs–provides the libref.dset for the SASReferences data set as returned from the call to the cst_insertstandardsasrefs macro and as used in SAS Clinical Standards Toolkit code for the remainder of the process

From various sample driver modules provided with SAS Clinical Standards Toolkit, here are several sample code excerpts that set the values for these macro variables:

```
data _null_;
    select("&sysver");
```

```
when("9.1") call
symput('studyRootPath','!sasroot/../SASCstCdiscSdtm311/9.1.3/sample/cdisc
-sdtm-3.1.1/sascstdemodata');
when("9.2") call
symput('studyRootPath','!sasroot/../SASCstCdiscSdtm311/9.2/sample/cdisc-
sdtm-3.1.1/sascstdemodata');
otherwise;
end;
run;
%let _cstSASRefsLoc=&studyrootpath/control;
%let _cstSASRefsName=sasreferences;
```

This example references a permanent SASReferences data set called sasreferences in a study-specific folder hierarchy. If there is no any additional information, the _cstSASRefs macro variable value is used as set in the initialize.properties file. (For example, _cstSASRefs = work._cstsasrefs.)

```
%let workPath=%sysfunc(pathname(work));
%let _cstSASRefsLoc=&workpath;
%let _cstSASRefsName=sasreferences;
%let _cstSASRefs=work.sasreferences;
```

This example references a SASReferences data set called sasreferences created in the SAS Work library. The value of _cstSASRefs is set to work.sasreferences (that is, the same file) and overrides any default value of _cstSASRefs that might have been set in the initialize.properties file.

Assessing Structural Integrity and Content

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

The cst_insertstandardsasrefs macro performs the simple task of looking up missing paths and memnames in the constructed SASReferences file from each standardsasreferences data set. For example, this is the method that sets the path and memname values for lines 4-6 in the example illustrated above in Figure 5.3 on page 52. This method attempts only to update records for supported standards (and standardversions) that having missing path and memname information. It does not modify records with non-null values and does not add any records from the standardsasreferences data sets. If this macro runs successfully, the resulting data set (which, by default, is referenced by the &_cstSASRefs global macro variable) will have paths and memnames for all records that require them (that is, not autocall and sourcedata records).

The cstutil_chkds macro is provided to check the structure and content of the data sets used by SAS Clinical Standards Toolkit, including SASReferences. This macro validates SASReferences against the expected structure and content as defined by the standardsasreferences and standardlookup data sets.

The syntax of this macro is as follows,

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

_cstDSname specifies a two-level name of the data set to be validated (required).

_cstType specifies the type of data set to be validated (required). This value comes from the TYPE column in the registered sasreferences for the standard-version combination.

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

_cstStandard specifies the name of the data standard to validate against (optional). By default, all standards will be included.

_cstStandardVersion specifies the version of the data standard to validate against (optional). By default, all standardVersions will be included.

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

The most common errors detected by the cstutil_checkds macro are described in Table 5.3 on page 55, along with suggested solutions.

Table 5.3 Debugging Problems with SASReferences

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

Error	Location Where It Is Reported	Possible Cause and Corrective Action
		the list of valid values and update the column value as required.

Translating Content for a SAS Session

Once the SASReferences file has been built, its content must be translated for use by a SAS Clinical Standards Toolkit process. This is accomplished with a call to the SAS Clinical Standards Toolkit framework utility macro %cstutil_allocatesasreferences. If this macro runs successfully, the SAS session is properly configured for the planned primary task (such as validation) that follows.

What happens when this macro is called?

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

At the conclusion of this step, all libraries should be allocated, all paths and global macros set, and the global status macro variable _cst_rc should be set to 0, indicating the process is ready to proceed with the following step.

This is a common process failure point, given the importance of the SASReferences file and the strict structural and content expectations of the file. SASReferences is key to the process and any errors will generally cause the process to fail. See Table 5.3 on page 55 for tips on debugging problems with SASReferences.

Best Practice Recommendation: Each SASReferences file is customized to the specific task to be completed. See later sections of this guide for more detailed SASReferences implementations required by these tasks.

CHAPTER 6

Validation

Validation Framework Overview

SAS Clinical Standards Toolkit validation is designed to assess the compliance of data and the metadata describing the data against an accepted standard. It is also designed to enable an assessment of the consistency of values within a specific column, between columns, across records within a specific data set, and across data sets. The primary

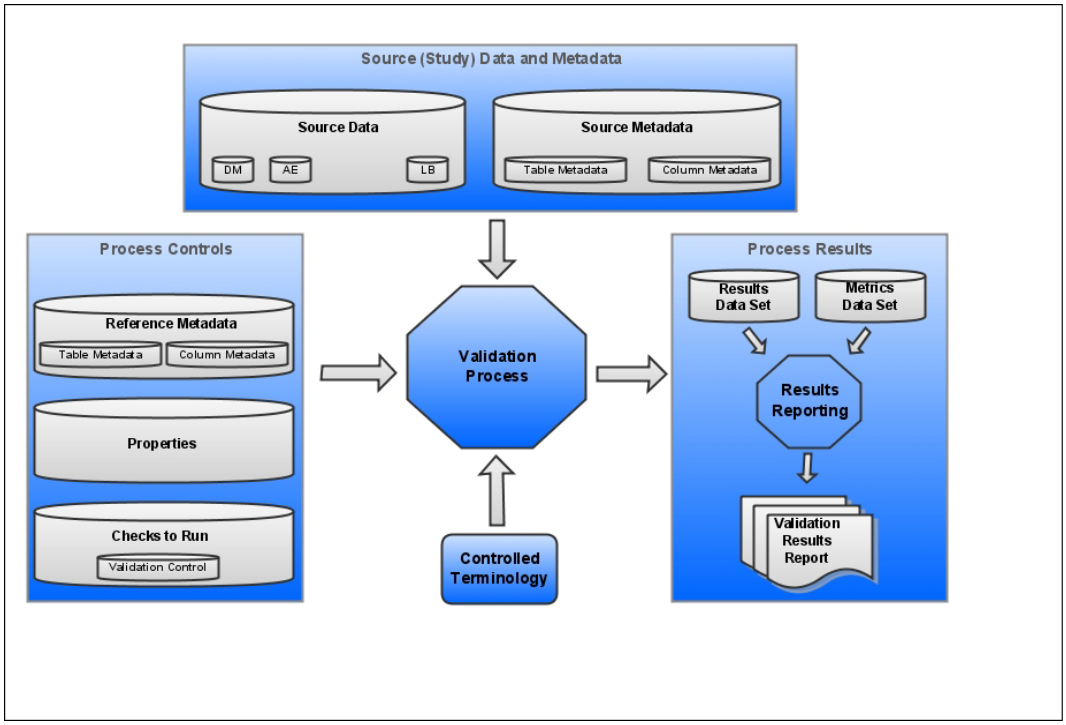
output is a results data set itemizing the process findings and an optional metrics data set summarizing the results.

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

Generally, validation is performed using a series of SAS macros running against the standard represented as SAS files. Validation of some standards, such as CDISC-CRTDDS, might include validation of files that are not SAS (such as define.xml).

Figure 6.1 provides a representation of a SAS Clinical Standards Toolkit validation process. Each element is fully described below.

Figure 6.1 Components of a SAS Clinical Standards Toolkit Process



- Source Data–a set of SAS data sets in one or more libraries that collectively represents a clinical study. These are often 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 the assessment to a subset of validation checks.
- Source Metadata–a set of SAS data sets in one or more libraries that provide metadata about the source data, typically in a format specific to a given standard. For example, metadata about source data sets might be captured in a source_tables data set, and metadata about columns in those source data sets might be captured in a source_columns data set.

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

  Reference metadata-a set of SAS data sets that provide metadata that defines a specific standard, typically in a format specific to a given standard. For example, metadata about data sets might be captured in a reference_tables data set, and metadata about columns might be captured in a reference_columns data set. See Appendix 3, "CDISC SDTM SAS Representation," for an example.

  Properties–a series of name-value pairs that are translated into SAS global macro variables available for the duration of a given SAS Clinical Standards Toolkit process. Properties might be defined in any number of files; both text file and SAS data set formats are supported. A sample validation.properties file is discussed in "Validation Check Metadata: Validation Master" on page 63, and the SAS Clinical Standards Toolkit global macro variables are documented in Appendix 1, "Global Macro Variables."

  Set of checks to run–a set of checks that represent all or some subset of checks defined for a given standard. Each check provides metadata used by the validation code to perform a specific compliance assessment.
- Controlled Terminology–an optional set of lookup values against which source data columns can be evaluated. These can be in the form of SAS format catalogs or SAS data sets.
- Results-a results data set itemizing the process findings and an optional metrics data set summarizing the results. The results data set usually contains a record indicating that each check was run successfully without error or itemizes the errors detected. Informational items about the process might also be included. Generation of metrics is conditional based on property file settings.

SAS Clinical Standards Toolkit validation makes the following basic assumptions:

1. There is some combination of source data and metadata available as SAS files that the user wants to validate.
2. A reference standard has been defined against which the source data and metadata are to be compared. SAS Clinical Standards Toolkit supplies representative reference metadata for each supported standard.
3. The source data can be in any number of SAS files, and those files can have any structure. However, the metadata describing the source data must accurately represent the source data and must be in a structure specific to a supported standard and defined by SAS Clinical Standards Toolkit.
4. A set of validation checks must be defined and conform to a generic SAS Clinical Standards Toolkit SAS data set structure. SAS Clinical Standards Toolkit supplies a representative set of checks for each supported standard.

Metadata Requirements

As noted in Chapter 4, "Supported Standards," a standard consists of a set of properties, messages, and metadata files that collectively represent that industry-recognized standard in SAS Clinical Standards Toolkit. Each SAS Clinical Standards

Toolkit registered standard can be set to support validation by setting the standards.supportsvalidation flag to Y. This action signals that the required set of validation files describing that standard exist. By default, the set of files supplied by SAS that supports the standards supplied by SAS can be found in the cstGlobalLibrary folder hierarchy.

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

```
C:\cstGlobalLibrary\standards\cdisc-sdtm-3.1.1
```

This assumes that cstGlobalLibrary is the name and it is installed in **`c:\`**.

The following subsections describe each type of file that defines metadata, either entirely unique to a SAS Clinical Standards Toolkit validation process, or having validation-specific elements. See Chapter 3, "Metadata File Descriptions," for details on metadata files common to all SAS Clinical Standards Toolkit processes.

Reference Metadata

For CDISC standards, reference metadata refers to metadata about data sets, defined in a reference_tables data set, and metadata about columns, defined in a reference_columns data set. A description of the columns in each data set is provided below. Examples of a reference_tables record and a reference_columns record are provided in Appendix 3, "CDISC SDTM SAS Representation." This metadata is required and effectively serves as the gold standard specifically describing the tables and columns for this version of the standard.

Table 6.1 Reference_Tables Data Set

Column	Description
sasref ($8)	The SAS libref that is used to refer to the table within the SAS Clinical Standards Toolkit SAS process. Should match the value of the SASReferences.sasref field where type=referencemetadata and subtype=table. Required.
table ($32)	The name of the domain being defined within the standard. The value must conform to SAS data set naming conventions. Required.
label ($40)	The label of the domain being defined within the standard. The value must conform to SAS naming conventions. Optional.
class ($40)	The observation class within the standard. Example CDISC SDTM values: Events, Findings, Interventions, Relates, Special Purpose, Trial Design. Optional; 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 for this field can be used in the creation of define.xml to populate the value for the def:leaf xlink:href link to the domain file. Should be the pathname and filename of the SAS transport file relative to the location of the define.xml. Optional; not relevant for all standards.
xmltitle ($200)	The title of the SAS transport file. The value for this field can be used in the creation of define.xml to populate the value for the def:leaf def:title value. It can be used to provide a meaningful description, label, or location of the domain leaf (for example, **`crt/datasets/Protocol 1234/AE.xpt`**). Optional; not relevant for all standards.

Column	Description
structure ($200)	Describes the general structure of the table. Example value: One record per event per subject. Optional; not relevant for all standards.
purpose ($20)	Describes the general purpose of the table. Examples: Tabulation (required for CDISC-SDTM), Analysis (required for CDISC-ADAM). Optional; not relevant for all standards.
keys ($200)	A space-delimited string of keys that capture the set of table columns that uniquely define records within the table. Normally, this set of keys also defines the sort order of records within the table. Example: STUDYID USUBJID. Required.
state ($20)	An optional description of the table state, such as Draft or Final.
date ($20)	A meaningful, distinguishing date (for example, release date, creation date, or modified date) that describes the table. Optional.
standard ($20)	Captures the standard name. This value must match the name of a registered standard in the SAS Clinical Standards Toolkit framework. See Chapter 2, "Framework," for a discussion of registered standards. This value must also match the standard field in the SASReferences data set. Examples: CDISC SDTM, CDISC-CRTDDS. Required.
standardversion ($20)	Captures a specific version of a standard. This value must match one of the standard versions associated with a registered standard. This value must also match the standardversion field in the SASReferences data set. Examples: 3.1.1, 1.0. 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. Optional.
comment ($200)	Any character string providing comments relevant to the table. Optional.

Table 6.2 Reference_Columns Data Set

Column	Description
sasref ($8)	The SAS libref that is used to refer to the table containing the column within the SAS Clinical Standards Toolkit SAS process. Should match the value of the SASReferences.sasref field where type=referencemetadata and subtype=table. Required.
table ($32)	The name of the containing domain being defined within the standard. The value must conform to SAS data set naming conventions. Required.
column ($32)	The name of the column within the table. The value must conform to SAS column naming conventions. Required.
label ($200)	The label of the column. The value must conform to SAS naming conventions. Optional.
order (8.)	Provides an ordering to the columns within each table. Values must be integers >0 and unique within each table. Required.
type ($1)	SAS type, N for numeric, C for character. Required.
length	Length of the column. Numeric columns have a length of 8. Required.

Column	Description
(8.)	
displayformat ($32)	Display format for numeric variables. (For example, 8.2 means display floating-point variable values to the second decimal place.) Optional; not relevant for all standards.
xmldatatype ($8)	The data type of the column as it is defined within the define.xml file. Values: integer | float | date | datetime | time | text. Optional; not relevant for all standards.
xmlcodelist ($32)	A SAS format name (without a $ prefix for character formats and without the trailing period) used to assess conformance to controlled terminology. Also as the codelist name within the define.xml file. This SAS format name must be in the format search path for successful column-value validation to occur. Optional; not relevant for all standards.
core ($10)	An indicator of whether the column is required. Sample CDISC-SDTM values: Req (required), Exp (expected), Perm (permissible). Optional; not relevant for all standards.
origin ($40)	Information about the source or origin of the column. Values can include CRF Page numbers, derived, or variable references, and are user extensible. Optional; not relevant for all standards.
role ($200)	Space-delimited column classification (for example, Identifier, Topic, Qualifier, Timing, Selection, Analysis). Columns can have multiple roles. Optional; not relevant for all standards.
term ($80)	Indicates whether the column is subject to controlled terminology as defined within each standard source specification. Optional; not relevant for all standards.
algorithm ($1000)	Imputation or computation method or code to derive the column value. Optional; might not be relevant for all standards.
qualifiers ($200)	Space-delimited string containing supplemental column attributes. Example (CDISC SDTM) values: MIXEDCASE, UPPERCASE, DATETIME, DURATION. Optional; not relevant for all standards.
standard ($20)	Captures the standard name. This value must match the name of a registered standard in the SAS Clinical Standards Toolkit framework. See Chapter 2, "Framework," for a discussion of registered standards. This value must also match the standard field in the SASReferences data set. Examples: CDISC SDTM, CDISC-CRTDDS. Required.
standardversion ($20)	Captures a specific version of a standard. This value must match one of the standard versions associated with a registered standard. This value must also match the standardversion field in the SASReferences data set. Examples: 3.1.1, 1.0. Required.
standardref ($200)	Any reference to an associated standard definition, implementation guide, schema, and so on that provides additional information about the column or describes the column in greater detail. Optional.
comment ($1000)	Any character string providing comments relevant to the column. Optional.

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

```
C:\cstGlobalLibrary\standards\<specific standard>\metadata
```

For example, for the CDISC SDTM 3.1.1 standard:

```
C:\cstGlobalLibrary\standards\cdisc-sdtm-3.1.1\metadata
```

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

Source Metadata

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

SAS Clinical Standards Toolkit also assumes that source_tables and source_columns accurately reflect and are consistent with the source data they describe. While some standard-specific validation checks might specifically look for such discrepancies and report those inconsistencies in detail, failure to meet this assumption often leads to errors in the SAS Clinical Standards Toolkit validation process and halts execution of the process.

Validation Check Metadata: Validation Master

The validation master data set contains the full set of validation checks defined for any given standard. This file, by default, is deployed to the following directory within each supported standard:

```
C:\cstGlobalLibrary\standards\<standard>\validation\control
```

This assumes that cstGlobalLibrary is the name and it is installed in **`c:\`**. Also, by default, the validation master SAS data set name is validation_master.sas7bdat.

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

Table 6.3 Validation Master Data Set

Column	Description
checkid ($8)	SAS Clinical Standards Toolkit has adopted a naming convention matching the standard to be validated. Checkid values are prefixed with up to a 4-byte prefix (CDISC examples: ODM, SDTM, ADAM, CRT). By convention, the prefix matches the mnemonic field in the standards data set found in **`C:\cstGlobalLibrary\metadata`**. This prefix is followed by

Column	Description
	a 4-byte numeric unique within the standard (for example, SDTM1234). Customers can use any other naming convention limited to 8-characters. By default, the checkid field is the first (primary) sort field in the validation master data set supplied by SAS. Sorting by checkid is not required. Required.
standard ($20)	Captures the standard name. This value must match the name of a registered standard in the SAS Clinical Standards Toolkit framework. See Chapter 2, "Framework," for a discussion of registered standards. This value must also match the standard field in the SASReferences data set. Examples: CDISC-SDTM, CDISC-CRTDDS. Required.
standardversion ($20)	Captures a specific version of a standard. This value must match one of the standard versions 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 *** might be used to signify that the check applies to all supported versions of that standard. Examples: 3.1.1, 1.0, ***. If a subsequent version of the standard is released, *** should still be applicable if the check is also valid for the new version. Required.
checksource ($40)	String identifying the source of the check. CDISC examples: Janus, JanusFR (FAIL-REJECT), SAS, WebSDM. 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. Required.
sourceid ($8)	Contains 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 supplied by SAS with no external source. Example: IR4000 (WebSDM identifier). Optional.
checkseverity ($40)	The severity as assigned by the checkSource, mapped to the following standardized values: Note (Low), Warning (Medium), Error (High). A value is expected, though not technically required. It is used in messages and reporting when provided.
checktype ($20)	General type of check used to categorize checks and to aid in registration of 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: Metadata–structural; checks some metadata-only property (no data access required). ColumnValue–content; checks a column value or compares two column values. Date–content; checks ISO 8601 compliance or compares two date values. Multirecord–content; looks across multiple records within a single domain. Multitable–content; looks across multiple domains. Controlterm–content; assesses whether column value is consistent with controlled terminology. Optional.
codesource ($32)	The name of the check macro; must conform to SAS macro naming conventions and be found in the SAS autocall path. Example: cstcheck_notunique. Required.
usesourcemetadata ($1)	Provides an indicator to the program code about whether to use source metadata rather than reference metadata to control derivation of to be validated domains and columns lists, program flow and looping. Values: Y, N (default). Optional.
tablescope	Specifies the domains to be tested by the check. The domains must exist in

Column	Description
($200)	either or both the reference or source table metadata. Can be in the form: _ALL_—all domains (equivalent to **). DM—any single domain; can be specified as libref.domain. DM+AE—multiple domains delimited with a +. _ALL_-DM—multiple domains that exclude specific domains delimited with a –. SUPP**—wildcard to include multiple domains. CLASS:EVENTS—all domains capturing event results (this syntax means use table metadata column CLASS for EVENTS as the value—similar syntax used for all other fields and values). [_ALL_-DM][DM]—bracket syntax to define sublists for comparative purposes; in this example, all non-DM domains compared with DM. See the validation master data set for a full set of values. Required.
columnscope ($200)	Specifies one or more columns, space delimited, identified for inclusion or exclusion for the specified check. Can be in the form: _ALL_—all columns (equivalent to ** or a null value). _NA_—not applicable (that is, domain-level check). AGE—any single column; can be specified as libref.domain.column or domain.column. ARM+ARMCD—multiple columns delimited with a +. **BLFL-LBBLFL—multiple columns that exclude specific columns delimited with a –. **DTC—wildcard to include multiple columns with ** representing the domain name. xxx**—(for example, AE**, where ** is a column wildcard). [**STDTC][**ENDTC]—bracket syntax to define sublists for comparative purposes; in this example, all start dates compared with all end dates. The number of columns in each sublist must be equivalent. See the validation master data set for a full set of values. Optional (if null, equivalent to _ALL_).
codelogic ($2000)	Check-specific code segment inserted into the check macro defined in codesource and consistent with codetype. Codelogic enables check-level customization, allowing reuse of more general check macros. The field length ($2000) limits the code to generally short code segments, although reference to another macro or use of %include expands this capability. Codelogic can use global and local macro variables (for example, those provided as macro input parameters and those set within the calling code). Examples: If (. < &_cstColumn1 < &_cstColumn2), then _cstError=1; %include <*fileref*>, limited to filerefs set externally to SAS Clinical Standards Toolkit or within the SASReferences control data set. %sdtmcheckutil_recordlookup data _cstProblems;set&_cstDSName;if <*some condition*>;run; Optional.
codetype (8.)	Defines whether and what type of codelogic can be used within the validation code. Values: 0—No codelogic used.

Column	Description
	1—DATA step statement level (for example, if &_cstColumn <0 then _cstError=1)
	2—Full DATA step or PROC SQL step or multiple steps
	3—Calls a SAS macro or %include that can contain only DATA step statement level code (like codetype=1)
	4—Calls a SAS macro or %include that can contain only full DATA step or PROC SQL step code (like codetype=2)
	Required.
lookuptype ($20)	Defines the type of information to use for value comparison to some standard. Values:
	Metadata—use SAS Clinical Standards Toolkit metadata; specifically, the value of the column metadata field xmlcodelist is to be used to identify the codelist (rendered as a SAS format).
	Format—use a SAS format to be found from the SAS format search path.
	Dataset—use a reference SAS data set (for example, medDRA); there are no SAS Clinical Standards Toolkit requirements for the structure and content of the reference data set.
	<extensible>—other user-defined values can be used if explicitly referenced within user-written code.
	Optional.
lookupsource ($32)	The specific SAS format or file associated with lookuptype. If lookuptype is:
	Metadata—lookupsource should be blank; the code gets the value from the source_columns.xmlcodelist field.
	Format—SAS format that must be in the format search path (if specified). Should generally match any value in source_columns.xmlcodelist for the columns specified in columnscope, though this field allows a run-time validation check against another format.
	Dataset—should be the name of a SAS data set; specified as data set name (for example, meddra) or libref.dataset; if a value is provided without a libref, SAS Clinical Standards Toolkit looks for any SASReferences type=referencecterm records for the sasref value.
	Optional.
standardref ($200)	Any reference to an associated standard definition, implementation guide, schema, and so on that provides additional information about the check or describes the basis for the check in greater detail. Optional.
reportingcolumns ($200)	Allows inclusion of columns not included in columnscope for code-processing purposes and to aid in error resolution. If specified, should be a space-delimited list of columns found in the domains specified in the tablescope field. Values of these columns can be reported in the results data set. Optional.
checkstatus (8.)	Is the check ready to be consumed and included in any validation_control run-time data set? If ready, set value to any positive integer. Values:
	0—(inactive, default)
	>0—(active)
	<0—(deprecated, archived, and so on)
	Optional, though expected.
reportall ($1)	Allows for more concise reporting of errors. Values:
	Y—(yes, report all records, default)

Column	Description
	N—(no)
	Required, although not all check macro modules support abbreviated (N) reporting.
uniqueid ($48)	Provides a unique ID for the check. Ensures uniqueness within the data set and within SAS Clinical Standards Toolkit. Designed to allow any shipped or derived check to be uniquely identifiable over time. Example: SDTM000100CST120SDTM3112009-05-12T12:00:00CDI.
	Legend:
	bytes 1-8 checkid
	bytes 9-10 checkid repeat indicator (00 unless multiple invocations of checkid are included)
	bytes 11-16 SAS Clinical Standards Toolkit version
	bytes 17-23 Standard version
	bytes 24-42 implementation datetime
	bytes 43-48 assigning authority
	Optional, though expected.

Content of the validation master data set is based on a combination of compliance requirements and the SAS representation of the standard.

The following table describes a sample validation_master record for the CDISC-SDTM 3.1.1 standard.

Table 6.4 Sample CDISC-SDTM 3.1.1 Validation_Master Record

Column Name	Column Value	Comment
checked	SDTM0208	The SAS Clinical Standards Toolkit check identifier used in validation results and reports.
standard	CDISC-SDTM	
standardversion	***	
checksource	WebSDM	This check originated as a WebSDM check.
sourceid	IR4009	WebSDM check IR4009.
checkseverity	Warning	
checktype	Column	
codesource	cstcheck_columncompare	This check uses the cstcheck_columncompare macro.
usesourcemetadata	Y	
tablescope	CLASS:FINDINGS-IE	This check is to be run on all findings domains except IE.
columnscope	[**ORRES][**STAT]	This check compares two column values from each

		domain.
codelogic	%let _cstExtraColumn=&_cstDomainOnly.DRVFL;data work._cstProblems;set &_cstDSName.; if ((&_cstColumn1 ne "" and &_cstColumn2 ne "") or (&_cstColumn1 = "" and &_cstColumn2 = "" and upcase (&_cstExtraColumn) ne "Y"));run;	This logic is used within cstcheck_columncompare; errors are documented in a work._cstProblems data set.
lookuptype		
lookupsource		
standardref		
reportingcolumns		
checkstatus	1	
reportall	Y	This check reports all errors identified.
uniqueid	SDTM020801CST120SDTM3112009-05-13T15:57:59CST	

While the validation master data set contains the full set of all validation checks for a given standard, the validation control data set is the run-time equivalent, containing just those checks to be run in any given 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 SAS Clinical Standards Toolkit validation processes, see "Special Topic: How SAS Clinical Standards Toolkit Interprets Validation Check Metadata" on page 102.

Validation.Properties

A set of properties specific to validation processes is provided with SAS Clinical Standards Toolkit. These properties allow specification of how validation checks are to be processed, as well as whether metrics are to be reported for the validation process.

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

The following table describes these properties.

Table 6.5 Validation Properties

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, the data set order is assumed. Alternatively, _cstCheckSortOrder can be set to sort the validation control file at run time by any fields within that data set. For example: CHECKSOURCE CHECKID.
_cstMetrics	Calculate and report metrics? 1=Yes.
_cstMetricsDS	Set the SAS data set name to use to accumulate metrics during

Property Name	Description
	the process. Default=work._cstmetrics.
_cstMetricsNumSubj _cstMetricsCntNumSubj	Calculate/report subject-level counts? 1=Yes, initialize _cstMetricsCntNumSubj to 0. Cannot be valid for all check macros.
_cstMetricsNumRecs _cstMetricsCntNumRecs	Calculate/report record-level counts? 1=Yes, initialize cstMetricsCntNumRecs to 0.
_cstMetricsNumChecks _cstMetricsCntNumChecks	Summarize/report the number of checks run? 1=Yes, initialize cstMetricsCntNumChecks to 0.
_cstMetricsNumBadChecks _cstMetricsCntNumBadChecks	Summarize/report the number of check invocations that failed? 1=Yes, initialize cstMetricsCntNumBadChecks to 0.
_cstMetricsNumErrors _cstMetricsCntNumErrors	Summarize/report the total number of errors (resultseverity=Error) found? 1=Yes, initialize cstMetricsCntNumErrors to 0.
_cstMetricsNumWarnings _cstMetricsCntNumWarnings	Summarize/report the total number of warnings (resultseverity=Warning) found? 1=Yes, initialize cstMetricsCntNumWarnings to 0.
_cstMetricsNumNotes _cstMetricsCntNumNotes	Summarize/report the total number of notes (resultseverity=Note) found? 1=Yes, initialize cstMetricsCntNumNotes to 0.
_cstMetricsNumStructural _cstMetricsCntNumStructural	Summarize/report the total number of structural (metadata) errors found? 1=Yes, initialize cstMetricsCntNumStructural to 0.
_cstMetricsNumContent _cstMetricsCntNumContent	Summarize/report the total number of structural (data) errors found? 1=Yes, initialize cstMetricsCntNumContent to 0.
_cstMetricsTimer	Report the elapsed time for each check invocation? 1=Yes.

By default, the validation properties can be found at the following location:

```
C:\cstGlobalLibrary\standards\<standard>\programs
```

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

Messages

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

The standard-specific messages data set, by default, is deployed to the following directory within each supported standard:

```
C:\cstGlobalLibrary\standards\<standard>\messages
```

This assumes that cstGlobalLibrary is the name and it is installed in **c:**.

No differences in structure are expected for all messages files used within SAS Clinical Standards Toolkit. That structure is defined in Chapter 3, "Metadata File Descriptions."

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

Complete message lists supporting the SAS Clinical Standards Toolkit standards are provided in the following appendices:

- Appendix 2, "Framework Messages
- Appendix 5, "CDISC-SDTM 3.1.1 Validation Checks"
- Appendix 7, "CDISC-CRTDDS 1.0 Validation Checks"

Validation Metrics

The generation of SAS Clinical Standards Toolkit validation metrics serves to provide a meaningful denominator for most validation checks so that the relative scope of errors detected can be more accurately assessed. Generally, the calculated denominator is a count of the number of records processed in any given domain.

The following code segment extracted from a validation check macro illustrates a typical calculation of the number of records in a domain and the macro call to add the count to the metrics data set:

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

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

However, some checks can evaluate multiple columns within any given domain so the counts will be greater. In addition, the metadata-level checks that do not access the domain data directly might report the number of metadata records evaluated instead.

Metrics processing is enabled based on property file settings. See Table 6.5 on page 68.

A description of the validation metrics data set, including the meaning of each field, is provided in Table 6.6 on page 71.

Table 6.6 Validation Metrics Data Set

Column	Description
metricparameter ($40)	A descriptive text string that specifies the metric of interest. This string is hardcoded within the check macro and cannot be modified without code changes within the check macro. Values should be non-null.
reccount (8.)	Generally, a count of the number of records specific to the combination of metricparameter and resultid. This number is derived within the check macro and cannot be modified without code changes. Can also contain summary counts 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.
resultid ($8)	The resultid is either the checkid or a hardcoded constant such as METRICS. SAS Clinical Standards Toolkit has adopted a naming convention for checkid matching each standard. Checkid (resultid) values are prefixed with up to a 4-byte prefix (CST for framework messaging, CDISC examples: ODM, SDTM, ADAM, CRT). By convention, the prefix matches the mnemonic field in the standards data set found in **`C:\cstGlobalLibrary\metadata`**. This prefix is followed by a 4-byte numeric unique within the standard (for example, SDTM1234). Customers can use any other naming convention limited to 8-characters. Values should be non-null.
srcdata ($200)	String that specifies the domain or check macro to which the metricparameter applies. Values should be non-null.
resultseq (8.)	A counter to indicate the record number within checkid in the validation control run-time set of checks. Set to 1, this is incremented only with each repeat invocation of a check. This value enables linkage with both the validation control and results data sets. Values should be non-null.

The following figure illustrates validation metrics output from a SAS Clinical Standards Toolkit validation process running CDISC-SDTM 3.1.1 validation. The validation control data set contained three records: two SDTM0451 checks and one SDTM0623 check.

Figure 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 1-2 are produced to document that SDTM0451 check was invoked twice. The missing recount value and the absence of other metrics reporting indicate that the two check invocations failed. This should be reported within the validation results data set.

Lines 3-7 provide metrics information about the SDTM0623 check. SDTM0623 checks that multiple standard units do not exist for any give test within findings domains. The 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 is reported.

Lines 8-14 are summary metrics reported at the end of the SDTM validation process within the sdtm_validate macro. No errors, other than noting that two checks could not be run (Lines 9 and 14), are reported.

See Table 6.6 on page 71 and the following discussion for more information about the validation metrics data set.

Building a Validation Process

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

SASReferences Customizations

A SAS Clinical Standards Toolkit validation process requires specification of a reference standard against which the source data and metadata can be compared. The following three records should be included in the SASReferences data set:

Figure 6.3 Defining the Reference Standard in the SASReferences Data Set

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

Note that the empty path signals that the path and memname information should be derived from the standardsasreferences data set associated with the specified standard and standardversion. Inclusion of these referencecontrol and referencemetadata records is currently unique to SAS Clinical Standards Toolkit validation.

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

1 A validation-specific properties file.

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

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

Validation.properties sets process global macro variables specific to validation, such as metrics. See "Validation.Properties" on page 68 for a complete discussion of these properties and see Appendix 1, "Global Macro Variables," for details about the derived global macro variables. This is a required file to support SAS Clinical Standards Toolkit validation.

Note that for CDISC-CRTDDS, the validation properties have been included in the standard-specific initialize.properties file and need not be separately referenced within SASReferences.

2 Specification of the output location of any process-generated metrics data set.

Figure 6.5 Defining the Validation Metrics Output Location in the SASReferences Data Set

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

The metrics data set provides a summary of the validation process, including error counts, processing time, and denominators for specific checks. See "Validation Metrics" on page 70 and "Validation Results and Metrics" on page 81 for a complete discussion of validation metrics, and see Appendix 1, "Global Macro Variables," for details about the global macro variables that govern metrics output. The metrics data set is typically output to the same location (SAS libref) as the validation results data set common to all SAS Clinical Standards Toolkit processes.

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

Figure 6.6 Defining Controlled Terminology in the SASReferences Data Set

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

The type=fmtsearch records allow specification of multiple format catalogs (for example, company-wide, compound, or group-level and study-level). Order within the format search path is set by the order field. The type=referencecterm record provides an opportunity to specify one or more lookup data sets (such as dictionary lookups like Loinc and MedDRA). Such lookup data sets need not conform to a specific structure or be in a structure that can be read into a SAS format. Customized code (typically in the validation master codeLogic field) is required to meaningfully join domain data with each associated lookup data set.

4 The location of the run-time validation check control file.

Figure 6.7 Defining the Run-Time Validation Check Control File Location in the SASReferences Data Set

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

This is a required file discussed in the following section.

Validation Control: Specification of Run-Time Checks

Each SAS Clinical Standards Toolkit validation process requires the specification of the validation checks to be run. This is accomplished by cloning, subsetting, or otherwise building a set of checks based on the validation master data set (see "Validation Check Metadata: Validation Master" on page 63). SAS Clinical Standards Toolkit assumes that each validation control data set is structurally equivalent to the validation master data set.

A sample CDISC-SDTM 3.1.1 validation control file is deployed to the following SAS 9.1.3 directory (the deployed location for SAS 9.2 is different, but similar):

```
!sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-
3.1.1/sascstdemodata/control
```

By default, the validation control SAS 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 control file. The following image illustrates how this file, as well as the validation master data set used in the following table, are defined in the sample CDISC-SDTM 3.1.1 SASReferences data set:

Figure 6.8 Defining Validation Check Master and Run-Time File Locations in the SASReferences Data Set

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

Note that &studyRootPath is assumed to have been set to **`!sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-3.1.1/sascstdemodata`**.

The following table provides several examples of how to create a validation control data set from the validation master data set. Note that the sample code is written assuming that the code is submitted in a context where libraries have been allocated and the format search and autocall paths have been set.

Table 6.7 Sample Code to Create Validation Control Data Set

Check Subset	Sample Code to Derive
All checks provided with SAS Clinical Standards Toolkit	data control.validation_control; set refcntl.validation_master; run;
Structural checks (that is, metadata-only checks not requiring access to the domain data)	data control.validation_control; set refcntl.validation_master (where=(upcase(checktype)="METADATA")); run;
Content checks (that is, checks that require access to the domain data)	data control.validation_control; set refcntl.validation_master (where=(upcase(checktype) ne "METADATA")); run;

Check Subset	Sample Code to Derive
Checks with a production status	data control.validation_control; set refcntl.validation_master (where=(checkstatus>0)); run;
WebSDM checks (CDISC SDTM only)	data control.validation_control; set refcntl.validation_master (where=(upcase(checksource)= “WEBSDM”)); run;

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

CDISC-SDTM 3.1.1 checks	data control.validation_control; set refcntl.validation_master (where=(standardVersion = "3.1.1" or standardVersion = "***")); run;
All codelist-related checks (that is, those that use the cstcheck_notincodelist macro)	data control.validation_control; set refcntl.validation_master (where=(upcase(checksource)=“CSTCHECK_NOTINCODELIST”)); run;

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

Check Subset	Sample Code to Derive

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

Check Subset	Sample Code to Derive

	set work._csttempds;
	if _n_=1 then stop;
	run;
	%end;
	
	%if %length(&_cstDomain)>0 %then %do;
	data &_cstChecksForDomain;
	set &_cstOutputDS (where=(table="&_cstDomain"));
	run;
	%end;
	%mend;
	
	%* Run this only once per stable reference validation_master - it takes a while... ;
	%buildcheckdomainlist(_cstCheckDS=refcntl.validation_master);
	
	%macro
	subsetdomainlist(_cstInputDS=work._cstcheckdomains,_cstOutputDS=control.validation_control,
	_cstDomain=);
	proc sql noprint;
	create table &_cstOutputDS as
	select vm.* from refcntl.validation_master vm
	right join &_cstInputDS dom
	on vm.checkid=dom.checkid and vm.standardversion=dom.standardversion and
	vm.checksource=dom.checksource
	where table="&_cstDomain";
	quit;
	%mend;
	
	%* Example call: subset validation data set just to those checks for the specified domain ;
	%* Returns all records for checkid/standardversion/checksource if any matches domain - needs tweaking... ;
	%subsetdomainlist(_cstDomain=AE);

Generally, 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 (global macro variable) _cstCheckSortOrder. If this property is not set, the data set order is assumed. As a part of the validation control derivation, checks can be sorted or ordered in any user-defined order. Alternatively, _cstCheckSortOrder can be set to sort the validation control file at run time by any fields within that data set.

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

Setting Properties for the Validation Process

While the set of properties (across all standards) available for any given process is extensive (see the full list in Appendix 1, "Global Macro Variables"), only a few are likely to be modified on a regular basis. These include:

- _cstSASRefsLoc–Point to another location for the SASReferences file.
- _cstSASRefsName–Point to another SASReferences filename.
- _cstSASRefs–Point to a specific libref.sasreferences file to use (typically in Work).
- _cstSubjectColumns–Reset the columns that identify a subject (if any).
- _cstReallocateSASRefs–Choose to reallocate SAS librefs and filerefs within the same SAS session, typically when changing studies or standards.
- _cstFMTLibraries–Modify the format search path built from SASReferences, most often to add a reference to a Work format catalog.
- _cstCheckSortOrder–Provide a set of validation control columns to resort the check processing order.
- _cstMetrics-Set to 1 to enable metrics calculations and reporting.
- _cstDebug–Turn on or off debugging for the session.
- _cstDebugOptions–Alter the SAS options when debugging.

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

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

Running a Validation Process

Sample CDISC-SDTM 3.1.1 Driver Program: validate_data.sas

Each SAS Clinical Standards Toolkit process uses a SAS driver module to set up the program execution flow. The following steps illustrate the execution sequence in a typical driver module to perform SAS Clinical Standards Toolkit validation.

Step 1: Define the study (data and metadata) location.

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

 This driver program illustrates use of the last mechanism, setting the
global macro variable studyRootPath which is referenced:
```

```
    (1) below to override the default global macro variable
_cstSASRefsLoc
    (2) within the sample study sasreferences data set path column.

 Note this example is dependent upon the SAS version and installation
folder structure. */
data _null_;
 select("&sysver");
    when("9.1") call symput('studyRootPath',

'!sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-
3.1.1/sascstdemodata');
    when("9.2") call symput('studyRootPath',

'!sasroot/../../SASClinicalStandardsToolkitSDTM311/9.2/sample/cdisc-sdtm-
3.1.1/sascstdemodata');
    otherwise;
 end;
```

&studyRootPath is a convenience macro variable that is not required.

Step 2: Set process properties (and global macro variables).

```
* Set properties supplied as part of the CST-FRAMEWORK standard. ;
%cst_setStandardProperties( _cstStandard=CST-
FRAMEWORK,_cstStandardVersion=1.2,_cstSubType=initialize);
* Set properties supplied as part of the CDISC-SDTM standard.;
%cst_setStandardProperties( _cstStandard=CDISC-
SDTM,_cstStandardVersion=3.1.1,_cstSubType=initialize);
```

Each registered standard should have its own initialize.properties. For each standard included in a specific process, %cst_setStandardProperties can be called at this point in the process flow. Alternatively, type=properties records can be added to the SASReferences data set, and properties are processed when %cstutil_allocatesasreferences is called in Step 4 below.

Step 3: Tell the process where to find the SASReferences data set.

```
* Set the location of the sasreferences file (overrides default
properties -- see comments above) ;
%let _cstSASRefsLoc=&studyrootpath/control;
%let _cstSASRefsName=sasreferences;
```

An alternative method of specifying the SASReferences data set to the process is to use the _cstSASRefs global macro variable. For example, issuing the following statement tells the process that SASReferences has been built in the SAS Work library as the SASReferences data set:

```
%let _cstSASRefs=work.sasreferences;
```

Step 4: Allocate SASReferences.

```
* Allocate all the SAS references specified in the sasreferences data
set;
* Set autocall and fmtsearch paths ;
* Build message lookup data set defined by _cstMessages global macro
variable ;
%cstutil_allocatesasreferences;
```

This is the final setup step for the process. The SASReferences data set specified in the previous step is now interpreted by SAS Clinical Standards Toolkit. These actions occur:

5 The macro cst_insertstandardsasrefs is called to insert paths into any records missing that information from the standardsasreferences data set for each standard. See “Inserting Information from Registered Standards into a SASReferences File” in Chapter 2 for more details on how this works.

6 The cstutil_checkds macro is called to perform internal validation on the SASReferences data set updated in step 1.

7 All filerefs and librefs are allocated (contingent on the _cstReallocateSASRefs property or global macro variable value).

8 Any property files are passed to %cst_setProperties to create global macro variables.

9 The format search path is set if any type=fmtsearch records are found, based on the order specified.

10 The autocall path is set if any type=autocall records are found, based on the order specified.

11 A messages data set is created to contain records from each referenced standard, based on the properties or global macro variables _cstMessages and _cstMessageOrder. This data set is used for the duration of the process to add fully resolved messages to the results data set.

At the conclusion of this step, all libraries should be allocated, all paths and global macros set, and the global status macro variable _cst_rc should be set to 0, indicating that the process is ready to proceed with the following step.

This is a common process failure point. SASReferences is key to the process and any errors generally cause the process to fail. See “Special Topic: Debugging a Validation Process” on page 112 for tips on debugging these process failures.

Step 5: Run the actions of interest.

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

What does this macro do?

12 Look up the validation control data set reference from SASReferences.

13 Optionally resort the validation control data set based on the _cstCheckSortOrder property or global macro variable value.

14 For each check in the validation control data set, call the check macro specified in the validation control codesource field, passing all of the check metadata to the codesource macro.

15 After all checks are run:

- Persist the results to the file specified in SASReferences (type=results, subtype=validationresults).
- Summarize any process results to the metrics data set, if requested.
- Persist the metrics to the file specified in SASReferences (type=results, subtype=validationmetrics).
- Clean up various SAS Work files as needed.

See “Special Topic: Debugging a Validation Process” on page 112 for tips on debugging if unexpected errors occur.

Step 6: Session cleanup.

```
 * Clean-up the CST process files, macro variables and macros.;
 %cstutil_cleanupcstsession(_cstClearCompiledMacros=0, _cstClearLibRefs=0;
 _cstResetSASAutos=0, _cstResetFmtSearch=0,
 _cstResetSASOptions=1,_cstDeleteFiles=1,_cstDeleteGlobalMacroVars=0);
```

This step is optional and unnecessary with batch processing. Care should be taken in cleaning up prematurely or too aggressively if additional SAS Clinical Standards Toolkit processes are to be run in the same interactive SAS session.

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

Table 6.8 Sample Code to Create Validation Control Data Set

Parameter	Action attempted
_cstClearCompiledMacros	Delete all macros from the work.sasmacr catalog.
_cstResetSASAutos	Reset the SASAutos path based on the value of the macro variable cstInitSASAutos. This macro is set, typically in the driver module, to capture the SASAutos value at the start of the SAS Clinical Standards Toolkit process, before calling %cstutil_allocatesasreferences. This parameter is ignored if _cstInitSASAutos does not exist.
_cstClearLibRefs	Clear all filerefs and librefs included in SASReferences, except any autocall filerefs.
_cstResetFmtSearch	Reset the fmtsearch path based on the fmtsearch value at the start of the SAS Clinical Standards Toolkit process. This parameter is ignored if the data set work._cstsessionoptions does not exist. To support this functionality, this data set should be created, typically in the driver module, before calling %cstutil_allocatesasreferences.
_cstResetSASOptions	Reset all SAS options back to their status at the start of the SAS Clinical Standards Toolkit process. This parameter is ignored if the data set work._cstsessionoptions does not exist. To support this functionality, this data set should be created, typically in the driver module, before calling %cstutil_allocatesasreferences.
_cstDeleteFiles	Delete the following files if the global macro variable _cstDebug=0: &_cstsasrefs, &_cstmessages, and work._cstsessionoptions.
_cstDeleteGlobalMacroVars	Call %symdel for all macro variables found in sashelp.vmacro (where=(lowcase(name) =: "_cst" and scope="GLOBAL")).

Validation Results and Metrics

For SAS Clinical Standards Toolkit validation processes, the primary products of each process are the results data set and an optional metrics data set. These data sets itemize and summarize, respectively, the findings of the validation effort.

The following sdisplays summarize a sample validation process that:

16 Was run on CDISC-SDTM 3.1.1 source data.

17 Referenced a validation control file containing metadata for five checks.

18 Included SASReferences records to persist the process results as results.validation_results and results.validation_metrics.

Figure 6.9 Validation Results Data Set (#1)

VIEWTABLE: Results.Validation_results

	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-framew	Info	0	0
2	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-sdtm-	Info	0	0
3	CST0075		1	1	WORK._CSTSASREFS	Unable to allocate libref for SRCFMT	Error	-1	0
4	CST0075		1	2	WORK._CSTSASREFS	Unable to allocate libref for CTREF	Error	-1	0
5	CST0108		1	1	CST_SETPROPERTIES	The properties were processed from the PATH !sasroot/../SASClinicalStandardsToolkitS	Info	0	0
6	CST0200		1	1	SDTM_VALIDATE	PROCESS STANDARD: CDISC-SDTM	Info	0	0
7	CST0200		1	2	SDTM_VALIDATE	PROCESS STANDARDVERSION: 3.1.1	Info	0	0
8	CST0200		1	3	SDTM_VALIDATE	PROCESS DRIVER: SDTM_VALIDATE	Info	0	0
9	CST0200		1	4	SDTM_VALIDATE	PROCESS DATE: 2009-05-21T09:17:57	Info	0	0
10	CST0200		1	5	SDTM_VALIDATE	PROCESS TYPE: VALIDATION	Info	0	0
11	CST0200		1	6	SDTM_VALIDATE	PROCESS SASREFERENCES: !sasroot/../SASClinicalStandardsToolkitS	Info	0	0
12	CST0100	SDTM0011	1	1	WORK._CSTSRCCOLUMNMETADATA	No errors detected in source data	Info	0	0
13	CST0029	SDTM0218	1	1	CSTCHECK_NOTINCODELIST	Format catalog WORK.FORMATS in fmtsearch could not be found	Info	0	0
14	CST0029	SDTM0218	1	2	CSTCHECK_NOTINCODELIST	Format catalog SRCFMT.FORMATS in fmtsearch could not be found	Info	0	0
15	CST0033	SDTM0218	1	3	CSTCHECK_NOTINCODELIST	Format search path has been set to WORK.FORMATS SRCFMT.FORMATS CSTFMT.CTERMS	Info	0	0
16	SDTM0218	SDTM0218	1	4	SRCDATA.LB.LBSTAT	Invalid NOTDONE code	Warning	1	0
20	SDTM0804	SDTM0804	1	1	SRCDATA.AE (SRCDATA.SV)	Invalid Subject Visit/Visit Number	Note	1	0
21	SDTM0804	SDTM0804	1	2	SRCDATA.AE (SRCDATA.SV)	Invalid Subject Visit/Visit Number	Note	1	0
22	SDTM0804	SDTM0804	1	3	SRCDATA.AE (SRCDATA.SV)	Invalid Subject Visit/Visit Number	Note	1	0
23	SDTM0804	SDTM0804	1	4	SRCDATA.DS (SRCDATA.SV)	Invalid Subject Visit/Visit Number	Note	1	0
32	CST0004	SDTM0851	1	1	CSTCHECK_RECMISMATCH	No columns evaluated - check validation_control specification	Warning: Check not run	-1	0
33	CST0004	SDTM0851	2	1	CSTCHECK_RECMISMATCH	No columns evaluated - check validation_control specification	Warning: Check not run	-1	0

Figure 6.10 Validation Results Data Set (#2)

VIEWTABLE: Results.Validation_results

	resultid	checkid	resultseq	seqno	actual	keyvalues	resultdetails
1	CST0108		1	1			
2	CST0108		1	1			
3	CST0075		1	1	type=fmtsearch,SASref=srcfmt,reftype=libref,path=&stu		Check the SAS log for errors
4	CST0075		1	2	type=referencecterm,SASref=ctref,reftype=libref,path=		Check the SAS log for errors
5	CST0108		1	1			
6	CST0200		1	1			
7	CST0200		1	2			
8	CST0200		1	3			
9	CST0200		1	4			
10	CST0200		1	5			
11	CST0200		1	6			
12	CST0100	SDTM0011	1	1			
13	CST0029	SDTM0218	1	1			
14	CST0029	SDTM0218	1	2			
15	CST0033	SDTM0218	1	3			
16	SDTM0218	SDTM0218	1	4	LBSTAT=LB01	STUDYID=SASDEMO,USUBJID=S002P008,LBTES	
20	SDTM0804	SDTM0804	1	1	USUBJID=S001P003,VISIT=UNSCHEDULED,VISIT	STUDYID=SASDEMO,USUBJID=S001P003,AETE	
21	SDTM0804	SDTM0804	1	2	USUBJID=S001P007,VISIT=WEEK 3,VISITNUM=3	STUDYID=SASDEMO,USUBJID=S001P007,AETE RASH,AESTDTC=2007-03-16	
22	SDTM0804	SDTM0804	1	3	USUBJID=S002P008,VISIT=WEEK 3,VISITNUM=3	STUDYID=SASDEMO,USUBJID=S002P008,AETE HEADACHES,AESTDTC=2007-03-31T16:15	
23	SDTM0804	SDTM0804	1	4	USUBJID=S001P002,VISIT=WEEK 3,VISITNUM=3	STUDYID=SASDEMO,USUBJID=S001P002,DSST	
32	CST0004	SDTM0851	1	1	tableScope=CO,columnScope=RDOMAIN		Tablescope and columnScope should resolve to at least one column
33	CST0004	SDTM0851	2	1	tableScope=CO,columnScope=RDOMAIN		Tablescope and columnScope should resolve to at least one column

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

Lines	Comment
1,2,5	Informational notes reporting processing of properties files.
3,4	Error records noting that SAS Clinical Standards Toolkit is unable to allocate librefs in SASReferences (srcdata=WORK._CSTSASREFS); the specific records are identified in the actual field.
6-11	Informational process summary, providing internal documentation about the process.
12	Check SDTM0011 ran successfully. This is a metadata-only check that runs against the source_columns metadata (WORK._CSTSRCCOLUMNMETADATA).
13-15	Check SDTM0218–informational notes, produced by the check macro cstcheck_notincodelist, regarding the availability of fmtsearch format catalogs.
16	Check SDTM0218–check error that LBSTAT contains an invalid value (LB01) and the keyvalues column identifies the record in error.
20-23	Check SDTM0804–4 records found that have invalid VISIT/VISITNUM values, relative to records in the SV domain; the actual values in error are listed in the actual column and the keyvalues column identifies the specific records in error.
32-33	Check SDTM0851–both (resultseq 1 & 2) check invocations failed attempting to evaluate RDOMAIN and the CO domain. The resultdetails suggests why. In this case, we have no CO data or metadata at this point in the source data and metadata.

Figure 6.11 Validation Metrics Data Set

VIEWTABLE: Results.Validation_metrics

	metricparameter	reccount	resultid	srcdata	resultseq
1	# of records tested	245	SDTM0011	WORK._CSTSRCCOLUMNMETADATA	1
2	Elapsed time to run check: 0:00:01	.	SDTM0011	CSTCHECK_METAMISMATCH	1
3	# of records tested	8	SDTM0218	SRCDATA.LB	1
4	# of records tested	7	SDTM0218	SRCDATA.MH	1
5	# of records tested	21	SDTM0218	SRCDATA.PF	1
6	# of records tested	14	SDTM0218	SRCDATA.VS	1
7	Elapsed time to run check: 0:00:02	.	SDTM0218	CSTCHECK_NOTINCODELIST	1
8	# of records tested	24	SDTM0804	SRCDATA.SV	1
9	# of records tested	7	SDTM0804	SRCDATA.AE	1
10	# of records tested	18	SDTM0804	SRCDATA.DS	1
11	# of records tested	6	SDTM0804	SRCDATA.IE	1
12	# of records tested	8	SDTM0804	SRCDATA.LB	1
13	# of records tested	7	SDTM0804	SRCDATA.MH	1
14	# of records tested	21	SDTM0804	SRCDATA.PF	1
15	# of records tested	14	SDTM0804	SRCDATA.VS	1
16	Elapsed time to run check: 0:00:03	.	SDTM0804	CSTCHECK_COMPAREDOMAINS	1
17	Elapsed time to run check: 0:00:01	.	SDTM0851	CSTCHECK_RECMISMATCH	1
18	Elapsed time to run check: 0:00:01	.	SDTM0851	CSTCHECK_RECMISMATCH	2
19	# of distinct check invocations	5	METRICS	SDTM_VALIDATE	2
20	# check invocations not run	2	METRICS	SDTM_VALIDATE	2
21	Errors (severity=High) reported	0	METRICS	SDTM_VALIDATE	2
22	Warnings (severity=Medium) reported	1	METRICS	SDTM_VALIDATE	2
23	Notes (severity=Low) reported	7	METRICS	SDTM_VALIDATE	2
24	Structural errors, warnings and notes	0	METRICS	SDTM_VALIDATE	2
25	Content errors, warnings and notes	10	METRICS	SDTM_VALIDATE	2

Table 6.10 Comments on Validation Metrics Data Set

Lines	Comment
1	Check SDTM0011-245 columns were evaluated.
2	Check SDTM0011–check took one second to run using cstcheck_metamismatch.
3-6	Check SDTM0218–check ran against four domains; record counts provided for each.
17-18	Check SDTM0851–because check did not run (using the cstcheck_recmismatch check macro), no recount was available; note separate metrics are produced for each check invocation (resultseq 1 & 2).
19	A summary metric of unique check invocations.
20	A summary metric of the number of checks that failed to run (defined as distinct checkid and resultseq combinations in the results data set where resultflag=-1).
21-25	Summary metric counts of the number of records, by type of metric, in the results data set.

Note: Some records in the validation results data set described in Figure 6.9 and Figure 6.10 on page 83 have been deleted for the sake of brevity, creating inconsistencies with the metrics listed in Figure 6.11 on page 84.

General Observations:

- The absence of a value in the results.checkid field can be used as a rough indicator of process setup messaging. Any results records where checkid is non-missing indicate messaging related to a specific validation check.
- Any resultseq value > 1 indicates a repeat invocation of a specific validation check with presumably some differences in the validation control check metadata.
- The seqno field is intended to be a record (message) counter within 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 to this is the case where 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, the seqno continues to increment in this scenario, 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. See "Special Topic: Validation Reporting" on page 121 for more information.

Sample CDISC-CRTDDS 1.0 Driver Program: validate_crtdds_data.sas

SAS Clinical Standards Toolkit validation of the SAS representation of the CDISC-CRTDDS standard follows the same pattern used for CDISC-SDTM validation described above. A sample driver module, validate_crtdds_data.sas, has been provided to perform process setup steps and to call the crtdds_validate.sas macro. Validation of the SAS representation of the CDISC-CRTDDS standard is described more fully in Chapter 7, "Creating CDISC-CRTDDS Define.xml," as is the use of the crtdds_xmlvalidate.sas macro, which performs XML validation of the generated define.xml.

Validation Checks by Standard

CDISC SDTM 3.1.1

SAS Clinical Standards Toolkit 1.2 provides 143 unique SDTM 3.1.1 validation checks. These checks are derived from three sources.

- The SAS interpretation of the CDISC-SDTM WebSDM 2.6 documented checks; see the Phase Forward White Paper at **`http://phaseforward.com/resource/whitepapers/Validation Checks 2.6/WebSDM V2.6 Validation Checks FINAL.pdf`**.
- Checks supporting loads into the Janus study data repository being developed by the FDA and the NCI as documented in the *SDTM Validation Specification, v1.0, November 2007* available at **`http://www.fda.gov/ForIndustry/DataStandards/CDISCDataStandards/ucm155327.htm`**.
- SAS checks based on SAS data management and cleaning experiences building CDISC-SDTM domains.

The CDISC SDTM 3.1.1 validation master data set, as defined in SAS Clinical Standards Toolkit 1.2, contains 246 records. If the Toolkit provides 143 unique CDISC SDTM 3.1.1 checks, why are there 246 records in the validation master data set? The validation master data set was built with multiple instances of the checks both to better support check selection by version or checksource (that is, WebSDM, Janus, or customer-defined checks) and to enable unique check logic and messaging by version or checksource.

The following table provides the distribution of all 246 CDISC-SDTM validation checks by the original source of the check (the validation master checksource field).

Table 6.11 Distribution of CDISC SDTM 3.1.1 Validation Checks (All)

Check Source	Count
WebSDM	105
Janus	51
JanusFR	58
SAS	32
Total	246

Note that this does not mean that SAS Clinical Standards Toolkit 1.2 supports 105 different WebSDM checks or 32 unique SAS checks. There are several duplicate invocations of specific checks to handle different sets of SDTM domains. For example, check SDTM0604 assesses whether the sequence numbers (**SEQ) are consecutively numbered. For most domains, this is assessed within each patient (USUBJID). However, the trial summary (TS) domain does not contain patient-level data, so the check logic differs. The validation master metadata differs for these two instances, but reports the same error message for the check.

Information about the 246 records in the CDISC-SDTM 3.1.1 validation master data set is itemized in Appendix 5, “CDISC-SDTM 3.1.1 Validation Checks.” Only selected columns are listed in the appendix. See Table 6.4 for a full description of a sample validation master record for the CDISC-SDTM 3.1.1 standard.

Consider the interrelationships among SAS Clinical Standards Toolkit validation check metadata. All run-time validation control data sets (and any programs that build or derive these data sets), corresponding message data sets (see “Messages” on page 69 describing this relationship), and the validation_stdref data set (found, by default, in the `C:\cstGlobalLibrary\standards\cdisc-sdtm-3.1.1\validation\control` folder, assuming that cstGlobalLibrary is the named and it is installed in `c:\`) are examples of the interconnectedness of many SAS Clinical Standards Toolkit metadata files.

Note: SAS Clinical Standards Toolkit does not currently fully support all WebSDM checks. Those not supported require a comparison between SDTM metadata and an associated define.xml. Loads into the Janus repository require the presence and use of a define.xml file. However, SAS Clinical Standards Toolkit 1.2 does not require an associated define.xml file for SDTM validation. Visit the SAS site support.sas.com for SAS Notes detailing this and other usage notes and their current status.

Note: Looking forward to SAS Clinical Standards Toolkit support of CDISC-SDTM 3.1.2, changes to the validation master data set are expected only as changes to check sources external to SAS (that is, WebSDM or Janus) require changes. The addition of new domains for SDTM 3.1.2 do not necessarily require validation check changes as

many checks are specific to all domains or domains of certain classes (for example, interventions) consistent across SDTM versions. As of the posting of this document, work is underway to develop and release the CDISC-SDTM 3.1.2 standard.

CDISC-CRTDDS 1.0

SAS Clinical Standards Toolkit provides a collection of check macros that validate the data contained within the SAS data sets representing CDISC-CRTDDS data. The goal of these checks is to ensure that all data is correctly specified and that referential integrity is maintained so that a standards-compliant CDISC define.xml file can be produced from those data sets.

The formal definition of the validity of CRTDDS data is given by the standards body in the form of XML schema definitions and must be translated into checks appropriate for the relational and tabular format.

The checks fall into these general categories:

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

These formats are specified in the standard in one of two ways:

- an enumeration
- a regular expression

The next table enumerates the types of checks to be done on the CRTDDS data.

Each check type is assumed to operate on data that exists in a source column within a source data set. A check type can reference one or more parameters that are used to validate the source column data. These parameters can be character strings or can be 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 any leading or trailing white space.

Table 6.12 CRTDDS Validation Check Types

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

Check Type	Check ID	Category	Description
			string comparisons are case sensitive.
Foreign key(targetColumn)	CRT0110	Structural	Each existing value in this column must have an equivalent value in the given target column.
Foreign key required(targetColumn)	(1)	Structural	A value is required for this column in every row and each value must have an equivalent value in the given target column. This check is the equivalent of running the required character value check, and failing if that check fails. If required character value passes, the foreign key() check is run.
Character format: language	CRT0106	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 digits after that hyphen. For example, e is a legal value, as is en-us and english and english-d842. Illegal values include 1en, mumblespeak, and en_us. The hyphen character sequence can be repeated any number of times also making a value such as english-mumbly-growly-47 a legal value. Regular expression: "[a-zA-Z]{1,8}(-[a-zA-Z0-9]{1,8})*".
Character format: fileName	CRT0107	Data	The character data must not contain any characters other than upper- and lower-case letters of the alphabet, numeric digits, the underscore (_) character, or a period. Regular expression: [A-Za-z0-9_.]+.
Character format: sasFormat	CRT0109	Data	The first character must be either a lower- or upper-case letter, an underscore (_), or the dollar sign ($). Any subsequent character must be either an upper- or lowercase letter, a numeric digit, the underscore (_), or a period. Regular expression: [A-Za-z_$][A-Za-z0-9_.]*.
Character format: sasName	CRT0108	Data	The first character must be either a lower- or upper-case letter or an underscore (_). Any subsequent character must be either an upper- or lowercase letter, a numeric digit, or the underscore (_). Regular expression: [A-Za-z_][A-Za-z0-9_]*.
Unique across data sets(targetcolumn0,…)	CRT0112	Structural	No value in this column can be equal to any value in any of the given data set columns.

Check Type	Check ID	Category	Description
Primary key	(2)	Data	Must satisfy the Unique in data set check type and the required character value check type.
Must Have Corresponding Value(targetColumn)	CRT0111	Structural	For each distinct value in this column, there must be at least one equivalent value in the supplied target column.
No Duplicates Per Unique Value(targetColumn)	CRT0113	Structural	For each distinct value in the target column, each value in the source column must be unique. That is, the same value cannot appear more than once in the source column for each distinct value in the target column.

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

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

Each check type belongs to one of two categories.

Data—Checks of this type have no dependencies on data outside of the source table. An example of such a check is ensuring that a value exists in a column for which values cannot be missing.

Structural—Checks of this type deal with relationships and data integrity between tables. Foreign key enforcement is an example of a structural check. These conditions must be satisfied for successful generation of a define.xml file. A user might want to defer structural checks until later in the process of populating the CRTDDS data sets, because foreign key relationships require that the data be made available in a particular order (that is, a referenced key must be available before the foreign key to it can exist).

Table 6.13 CRTDDS Validation Checks

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
0000	DefineDocument	CRT0100, CRT0101	FileOID	Primary key
0001	DefineDocument	CRT0101	FileType	Required character value
0002	DefineDocument	CRT0100	ID	Unique in data set
0003	DefineDocument	CRT0112	ID	Unique across data sets (MDVLeaf.ID, ItemGroupLeaf.ID)
0005	DefineDocument	CRT0114	FileType	Enumeration (“Snapshot”, “Transactional”)
0006	DefineDocument	CRT0114	Archival	Enumeration (“Yes”)
0007	DefineDocument	CRT0114	Granularity	Enumeration (“All”, “Metadata”, “AdminData”,

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
				"ReferenceData", "AllClinicalData", "SingleSite", "SingleSubject")
0008	Study	CRT0101, CRT0110	FK_DefineDocument	Foreign key required (DefineDocument.File OID)
0009	Study	CRT0100, CRT0101	OID	Primary key
0147	Study	CRT0101	StudyName	Required character value
0148	Study	CRT0101	StudyDescription	Required character value
0149	Study	CRT0101	ProtocolName	Required character value
0010	MeasurementUnits	CRT0100, CRT0101	OID	Primary key
0011	MeasurementUnits	CRT0101	Name	Required character value
0012	MeasurementUnits	CRT0101, CRT0110	FK_Study	Foreign key required (Study.OID)
0013	MUTranslatedText	CRT0106	lang	Character format: language
0014	MUTranslatedText	CRT0101, CRT0110	FK_MeasurementUnits	Foreign key required (MeasurementUnits.OID)
0015	MetaDataVersion	CRT0100, CRT0101	OID	Primary key
0016	MetaDataVersion	CRT0101	Name	Required character value
0017	MetaDataVersion	CRT0101, CRT0110	FK_Study	Foreign key required (Study.OID)
0150	MetaDataVersion	CRT0101	DefineVersion	Required character value
0151	MetaDataVersion	CRT0101	StandardName	Required character value
0152	MetaDataVersion	CRT0101	StandardVersion	Required character value
0018	AnnotatedCRFs	CRT0101, CRT0110	leafID	Foreign key required (MDVLeaf.ID)
0019	AnnotatedCRFs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0020	SupplementalDocs	CRT0101, CRT0110	leafID	Foreign key required (MDVLeaf.ID)

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
0021	SupplementalDocs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0022	MDVLeaf	CRT0100, CRT0101	ID	Primary key
0023	MDVLeaf	CRT0111	ID	Must have corresponding value (MDVLeafTitles.FK_MDVLeaf)
0024	MDVLeaf	CRT0112	ID	Unique across data sets (DefineDocument.ID, ItemGroupLeaf.ID)
0025	MDVLeaf	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0026	MDVLeafTitles	CRT0101, CRT0110	FK_MDVLeaf	Foreign key required (MDVLeaf.ID)
0027	ComputationMethods	CRT0100, CRT0101	OID	Primary key
0028	ComputationMethods	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0029	ValueLists	CRT0100, CRT0101	OID	Primary key
0030	ValueLists	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0031	ValueListItemRefs	CRT0101, CRT0110	ItemOID	Foreign key required (ItemDefs.OID)
0032	ValueListItemRefs	CRT0114	Mandatory	Enumeration (“Yes”, “No”)
0033	ValueListItemRefs	CRT0110	ImputationMethodOID	Foreign key (ImputationMethods.OID)
0034	ValueListItemRefs	CRT0110	RoleCodeListOID	Foreign key (CodeLists.OID)
0035	ValueListItemRefs	CRT0101, CRT0110	FK_ValueLists	Foreign key required (ValueLists.OID)
0036	ValueListItemRefs	CRT0101	Mandatory	Required character value
0037	ProtocolEventRefs	CRT0101, CRT0110	StudyEventOID	Foreign key required (StudyEventDefs.OID)
0038	ProtocolEventRefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
0039	ProtocolEventRefs	CRT0114	Mandatory	Enumeration (“Yes”, “No”)
0040	ProtocolEventRefs	CRT0101	Mandatory	Required character value
0041	ProtocolEventRefs	CRT0113	StudyEventOID	No duplicates per unique value (FK_MetaDataVersion)
0042	ProtocolEventRefs	CRT0113	OrderNumber	No duplicates per unique value (FK_MetaDataVersion)
0043	StudyEventDefs	CRT0100, CRT0101	OID	Primary key
0044	StudyEventDefs	CRT0101	Name	Required character value
0045	StudyEventDefs	CRT0114	Repeating	Enumeration (“Yes”, “No”)
0046	StudyEventDefs	CRT0101	Repeating	Required character value
0047	StudyEventDefs	CRT0114	Type	Enumeration (“Scheduled”, “Unscheduled”, “Common”)
0048	StudyEventDefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0153	StudyEventDefs	CRT0101	Type	Required character value
0049	StudyEventFormRefs	CRT0101, CRT0110	FormOID	Foreign key required (FormDefs.OID)
0050	StudyEventFormRefs	CRT0114	Mandatory	Enumeration (“Yes”, “No”)
0051	StudyEventFormRefs	CRT0101	Mandatory	Required character value
0052	StudyEventFormRefs	CRT0101, CRT0110	FK_StudyEventDefs	Foreign key required (StudyEventDefs.OID)
0053	StudyEventFormRefs	CRT0113	FormOID	No duplicates per unique value (StudyEventFormRefs. FK_StudyEventDefs)
0054	StudyEventFormRefs	CRT0113	OrderNumber	No duplicates per unique value (StudyEventFormRefs. FK_StudyEventDefs)
0055	FormDefs	CRT0100, CRT0101	OID	Primary key

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
0056	FormDefs	CRT0101	Name	Required character value
0057	FormDefs	CRT0114	Repeating	Enumeration (“Yes”, “No”)
0058	FormDefs	CRT0101	Repeating	Required character value
0059	FormDefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0060	FormDefItemGroupRefs	CRT0101, CRT0110	ItemGroupOID	Foreign key required (ItemGroupDefs.OID)
0061	FormDefItemGroupRefs	CRT0114	Mandatory	Enumeration (“Yes”, “No”)
0062	FormDefItemGroupRefs	CRT0101	Mandatory	Required character value
0063	FormDefItemGroupRefs	CRT0101, CRT0110	FK_FormDefs	Foreign key required (FormDefs.OID)
0064	FormDefItemGroupRefs	CRT0113	OrderNumber	No duplicates per unique value (FormDefItemGroupRefs.FK_FormDefs)
0065	FormDefItemGroupRefs	CRT0113	ItemGroupOID	No duplicates per unique value (FormDefItemGroupRefs.FK_FormDefs)
0066	FormDefArchLayouts	CRT0100, CRT0101	OID	Primary key
0067	FormDefArchLayouts	CRT0101	PdfFileName	Required character value
0068	FormDefArchLayouts	CRT0107	PdfFileName	Character format: filename
0069	FormDefArchLayouts	CRT0110	PresentationOID	Foreign key (Presentation.OID)
0070	FormDefArchLayouts	CRT0101, CRT0110	FK_FormDefs	Foreign key required (FormDefs.OID)
0071	ItemGroupDefs	CRT0100, CRT0101	OID	Primary key
0072	ItemGroupDefs	CRT0111	OID	Must have corresponding value (ItemGroupDefItemRefs.ItemOID)
0073	ItemGroupDefs	CRT0101	Name	Required character value
0074	ItemGroupDefs	CRT0114	Repeating	Enumeration (“Yes”, “No”)

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
0075	ItemGroupDefs	CRT0101	Repeating	Required character value
0076	ItemGroupDefs	CRT0114	IsReferenceData	Enumeration (“Yes”, “No”)
0077	ItemGroupDefs	CRT0108	SASDatasetName	Character Format: sasName
0078	ItemGroupDefs	CRT0101	Label	Required character value
0079	ItemGroupDefs	CRT0101	ArchiveLocationID	Required character value
0080	ItemGroupDefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0081	ItemGroupDefItemRefs	CRT0101, CRT0110	ItemOID	Foreign key required (ItemDefs.OID)
0082	ItemGroupDefItemRefs	CRT0114	Mandatory	Enumeration (“Yes”, “No”)
0083	ItemGroupDefItemRefs	CRT0101	Mandatory	Required character value
0084	ItemGroupDefItemRefs	CRT0110	ImputationMethodOID	Foreign key (ImputationMethods.OID)
0085	ItemGroupDefItemRefs	CRT0110	RoleCodeListOID	Foreign key (CodeLists.OID)
0086	ItemGroupDefItemRefs	CRT0101, CRT0110	FK_ItemGroupDefs	Foreign key required (ItemGroupDefs.OID)
0087	ItemGroupDefItemRefs	CRT0113	OrderNumber	No duplicates per unique value (ItemGroupDefItemRefs.FK_ItemGroupDefs)
0088	ItemGroupDefItemRefs	CRT0113	ItemOID	No duplicates per unique value (ItemGroupDefItemRefs.FK_ItemGroupDefs)
0154	ItemGroupDefItemRefs	CRT0101	Role	Required character value
0089	ItemGroupAliases	CRT0101	Context	Required character value
0090	ItemGroupAliases	CRT0101	Name	Required character value
0091	ItemGroupAliases	CRT0101, CRT0110	FK_ItemGroupDefs	Foreign key required (ItemGroupDefs.OID)
0092	ItemGroupLeaf	CRT0100, CRT0101	ID	Primary key
0093	ItemGroupLeaf	CRT0112	ID	Unique across data sets

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
				(DefineDocument.ID, MDVLeaf.ID)
0094	ItemGroupLeaf	CRT0101, CRT0110	FK_ItemGroupDefs	Foreign key required (ItemGroupDefs.OID)
0095	ItemGroupLeafTitles	CRT0101, CRT0110	FK_ItemGroupLeaf	Foreign key required (ItemGroupLeaf.ID)
0096	ItemDefs	CRT0100, CRT0101	OID	Primary key
0097	ItemDefs	CRT0101	Name	Required character value
0098	ItemDefs	CRT0114	DataType	Enumeration (“integer”, “float”, “date”, “datetime”, “time”, “text”, “string”)
0099	ItemDefs	CRT0101	DataType	Required character value
0100	ItemDefs	CRT0108	SASFieldName	Character format: sasName
0101	ItemDefs	CRT0108	SDSVarName	Character format: sasName
0102	ItemDefs	CRT0110	CodeListRef	Foreign key (CodeLists.OID)
0103	ItemDefs	CRT0110	ComputationMethodOID	Foreign key (ComputationMethods.OID)
0104	ItemDefs	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0105	ItemQuestionTranslatedText	CRT0106	lang	Character format: language
0106	ItemQuestionTranslatedText	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0107	ItemQuestionExternal	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0108	ItemMURefs	CRT0101, CRT0110	MeasurementUnitOID	Foreign key required (MeasurementUnits.OID)
0109	ItemMURefs	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0110	ItemRangeChecks	CRT0100, CRT0101	OID	Primary key
0111	ItemRangeChecks	CRT0111	OID	Must have corresponding value (ItemRangeCheckValues.OID)

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
0112	ItemRangeChecks	CRT0101	Comparator	Required character value
0113	ItemRangeChecks	CRT0114	Comparator	Enumeration (“LT”, “LE”, “GT”, “GE”, “EQ”, “NE”, “IN”, “NOTIN”)
0114	ItemRangeChecks	CRT0101	SoftHard	Required character value
0115	ItemRangeChecks	CRT0114	SoftHard	Enumeration (“Soft”, “Hard”)
0116	ItemRangeChecks	CRT0101, CRT0110	MURefOID	Foreign key required (MeasurementUnits.OID)
0117	ItemRangeChecks	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0118	ItemRangeCheckValues	CRT0101, CRT0110	FK_ItemRangeChecks	Foreign key required (ItemRangeChecks.OID)
0119	RCErrorTranslatedText	CRT0106	lang	Character format: language
0120	RCErrorTranslatedText	CRT0101, CRT0110	FK_ItemRangeChecks	Foreign key required (ItemRangeChecks.OID)
0121	ItemRole	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0122	ItemAliases	CRT0101	Context	Required character value
0123	ItemAliases	CRT0101	Name	Required character value
0124	ItemAliases	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0125	ItemValueListRefs	CRT0101, CRT0110	ValueListOID	Foreign key required (ValueLists.OID)
0126	ItemValueListRefs	CRT0101, CRT0110	FK_ItemDefs	Foreign key required (ItemDefs.OID)
0127	CodeLists	CRT0100, CRT0101	OID	Primary key
0128	CodeLists	CRT0101	Name	Required character value
0129	CodeLists	CRT0114	DataType	Enumeration (“integer”, “float”, “text”)
0130	CodeLists	CRT0101	DataType	Required character value
0131	CodeLists	CRT0109	SASFormatName	Character format:

CRTDDS Validation Number*	Source Data Set	Check ID	Variable Being Checked	Check
				sasFormat
0132	CodeLists	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0133	ExternalCodeLists	CRT0101, CRT0110	FK_CodeLists	Foreign key required (CodeLists.OID)
0134	ExternalCodeLists	CRT0112	FK_CodeLists	Unique across data sets (CodeListItems.FK_CodeLists)
0135	CodeListItems	CRT0100, CRT0101	OID	Primary key
0137	CodeListItems	CRT0101, CRT0110	FK_CodeLists	Foreign key required (CodeLists.OID)
0138	CodeListItems	CRT0112	FK_CodeLists	Unique across data sets (ExternalCodeLists.FK_CodeLists)
0139	CodeListItems	CRT0113	CodedValue	No duplicates per unique value (FK_CodeLists)
0140	CLItemDecodeTranslatedText	CRT0106	lang	Character format: language
0141	CLItemDecodeTranslatedText	CRT0101, CRT0110	FK_CodeListItems	Foreign key required (CodeListItems.OID)
0142	ImputationMethods	CRT0100, CRT0101	OID	Primary key
0143	ImputationMethods	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)
0144	Presentation	CRT0100, CRT0101	OID	Primary key
0145	Presentation	CRT0106	lang	Character format: language
0146	Presentation	CRT0101, CRT0110	FK_MetaDataVersion	Foreign key required (MetaDataVersion.OID)

*The CRTDDS validation numbers are not sequential because of modification of the model during development.

Special Topic: Validation Check Macros

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

19 Code modules should be generic and reusable across standards; 13 check macros support CDISC-SDTM 3.1.1 validation; CDISC-CRTDDS 1.0 uses six of these macros.

20 Code must run with SAS 9.1.3 (that is, no functionality new to SAS 9.2).

21 Code should be written as SAS macros.

22 SAS macros should have simple parameter signatures; all macros accept a single parameter, _cstControl, a single-observation data set containing check-specific metadata.

23 SAS macros should be implemented as non-compiled open code.

24 SAS macros should be callable using the SAS autocall facility; CST Framework supports a single sasmacros library, and each SAS Clinical Standards Toolkit standard supports a single macros library, each available using the SAS autocall path.

25 Code modules should be generic and reusable with multiple validation checks; the SAS Clinical Standards Toolkit check macros support 143 unique CDISC-SDTM 3.1.1 validation checks and 11 unique CDISC-CRTDDS validation checks.

26 To support code generalization, use metadata-driven techniques to provide check-specific information to the check macros, even including which check macro to call.

27 Code should write processing results to a single validation results data set, available for post-process review and reporting.

These design requirements, where appropriate, should be honored when developing custom validation check macros. The following table identifies and briefly describes the purpose of each of the 13 check macros provided with SAS Clinical Standards Toolkit.

Table 6.14 Overview, SAS Clinical Standards Toolkit Validation Check Macros

Check Macro	CodeLogic Style	Description/Purpose
cstcheck_notsorted	(not used)	Identifies any domain that is not sorted by the keys defined in the metadata.
cstcheck_zeroobs	(not used)	Identifies any data set with zero observations.
cstcheck_metamismatch	Step	Identifies inconsistencies between study and reference column metadata.
cstcheck_column	Statement	Identifies any invalid column values or attributes.
cstcheck_columncompare	Step	Supports comparison of column values.
cstcheck_dsmismatch	Step	Identifies any data set mismatches between study and template metadata and the source data library.

Check Macro	CodeLogic Style	Description/Purpose
cstcheck_violatesstd	Statement	Identifies any invalid column values defined within a reference standard.
cstcheck_notunique	Not used for Functions 1-3; DATA step for Function 4	A multi-function macro that assesses the uniqueness of data sets, columns, or value-pairs from two columns. Function 1: Is data set unique by a set of columns? Function 2: For any given subject, are column values unique? Function 3: Does a combination of two columns have unique values? Function 4: Are the values in one column (Column2) consistent within each value of another column (Column1)?
cstcheck_notconsistent	Step	Identifies any inconsistent column values across records.
cstcheck_recnotfound	Step	Generally compares the consistency of one or more columns across two tables or allows comparison of the consistency of one <*table*>.<*column*> with another <*table*>.<*column*>.
cstcheck_comparedomains	Step	Generally compares values for one or more columns in one domain with values for those same columns in another domain.
cstcheck_recmismatch	Step	Identifies any record mismatches across domains (domain as referenced within another domain).
cstcheck_notincodelist	If lookuptype=DATASET, DATA step codeLogic required Else, DATA step codeLogic 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: a **STAT value is found other than 'NOT DONE.'

Each validation check macro follows a standard basic workflow. Several of the 13 check macros perform more complex operations and multiple functions. The basic workflow includes the following:

28 Call the utility macro %cstutil_readcontrol, which translates the validation check metadata passed as the input parameter into local macro variables for the check macro processing.

29 Evaluate required check macro-specific metadata values.

30 Call the utility macro %cstutil_buildcollist (or, if domain-only processing, %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).

31 Loop through the target tables and columns identified in the prior step.

32 Perform the logic required to properly assess the validation check. This might be the check macro code itself or the code contained in the validation check metadata codeLogic field.

33 Write any informational or error messages to the results data set; optional metrics are written to the metrics data set.

34 Clean up any work files local to the check macro processing.

The distribution of validation checks, by check macro, is provided in Table 6.15 for CDISC-SDTM 3.1.1 and Table 6.16 on page 101 for CDISC-CRTDDS 1.0.

Table 6.15 CDISC-SDTM 3.1.1 Validation Check Grid

Check Macro (codesource)	Type of Check (checktype)	Unique Check Identifier (checkid)–Add SDTM Prefix
cstcheck_notsorted	Multirecord	0601
cstcheck_zeroobs	Metadata	0001, 0002, 0003
cstcheck_metamismatch	Metadata	0011, 0012, 0013, 0014, 0015, 0019, 0020, 0022, 0023, 0030, 0031, 0032
cstcheck_column	ColumnAttribute	0124, 0125, 0126, 0127, 0128, 0129
	ColumnValue	0204, 0205, 0206, 0207, 0217, 0220, 0222, 0251, 0271, 0352, 0354, 0355, 0452, 0506, 0521
	Date	0101, 0102
cstcheck_columncompare	Column	0208, 0209, 0211, 0212, 0213, 0214, 0215, 0216, 0219, 0223, 0351, 0353, 0405, 0406, 0408, 0409, 0410, 0411, 0412, 0413, 0414, 0462, 0463, 0501, 0502, 0503, 0507, 0534, 0541, 0551, 0561
	Date	0210, 0407
cstcheck_dsmismatch	Metadata	0004, 0005, 0006, 0017
cstcheck_violatesstd	Column	0201, 0202, 0203, 0606
cstcheck_notunique	Multirecord	0602, 0603, 0622, 0623, 0631, 0641, 0642, 0651, 0661, 0671
cstcheck_notconsistent	Multirecord	0604, 0605, 0607, 0621, 0643
	Multitable	0644, 0807
cstcheck_recnotfound	Multitable	0802, 0803, 0805, 0806, 0811, 0821, 0822, 0831, 0836, 0841
cstcheck_comparedomains	Multitable	0801, 0804
cstcheck_recmismatch	Multitable	0851, 0861, 0862, 0863, 0864, 0865, 0866, 0871
cstcheck_notincodelist	Controlterm	0218, 0221, 0301, 0302, 0303, 0401, 0402, 0403, 0450, 0451, 0453, 0454, 0455, 0456, 0457, 0458, 0459, 0460, 0461, 0464, 0504, 0505, 0508, 0531, 0532, 0533

Table 6.16 CDISC-CRTDDS V1.0 Validation Check Grid

Check Macro (codesource)	Check Type (checktype)	Unique Check Identifier (checkid))	Corresponding CRTDDS validation number*
cstcheck_column	ColumnValue	CRT0106	0013, 0105, 0119, 0140, 0145
		CRT0107	0068
		CRT0108	0077, 0100, 0101
		CRT0109	0131
cstcheck_violatesstd	Column	CRT0101	0000, 0001, 0008, 0009, 0010, 0011, 0012, 0014, 0015, 0016, 0017, 0018, 0019, 0020, 0021, 0022, 0025, 0026, 0027, 0028, 0029, 0030, 0031, 0035, 0036, 0037, 0038, 0040, 0043, 0044, 0046, 0048, 0049, 0051, 0052, 0055, 0056, 0058, 0059, 0060, 0062, 0063, 0066, 0067, 0070, 0071, 0073, 0075, 0078, 0079, 0080, 0081, 0083, 0086, 0089, 0090, 0091, 0092, 0094, 0095, 0096, 0097, 0099, 0104, 0106, 0107, 0108, 0109, 0110, 0112, 0114, 0116, 0117, 0118, 0120, 0121, 0122, 0123, 0124, 0125, 0126, 0127, 0128, 0130, 0132, 0133, 0135, 0137, 0141, 0142, 0143, 0144, 0146, 0147, 0148, 0149, 0150, 0151, 0152, 0153, 0154
cstcheck_notunique	Multirecord	CRT0100	0000, 0002, 0004, 0009, 0010, 0015, 0022, 0027, 0029, 0043, 0055, 0066, 0071, 0092, 0096, 0110, 0127, 0135, 0142, 0144,
cstcheck_recnotfound	Multitable	CRT0110	0008, 0012, 0014, 0017, 0018, 0019, 0020, 0021, 0025, 0026, 0028, 0030, 0031, 0033, 0034, 0035, 0037, 0038, 0048, 0049, 0052, 0059, 0060, 0063, 0069, 0070, 0080, 0081, 0084, 0085, 0086, 0091, 0094, 0095, 0102, 0103, 0104, 0106, 0107, 0108, 0109, 0116, 0117, 0118, 0120, 0121, 0124, 0125, 0126, 0132, 0133, 0137, 0141, 0143, 0146

Check Macro (codesource)	Check Type (checktype)	Unique Check Identifier (checkid))	Corresponding CRTDDS validation number*
		CRT0111	0023, 0072, 0111
		CRT0112	0003, 0024, 0093, 0134, 0138
cstcheck_recmismatch	Multitable	CRT0113	0041, 0042, 0053, 0054, 0064, 0065, 0087, 0088, 0139
cstcheck_notincodelist	Controlterm	CRT0114	0005, 0006, 0007, 0032, 0039, 0045, 0047, 0050, 0057, 0061, 0074, 0076, 0082, 0098, 0113, 0115, 0129

See Appendix 7, “CDISC-CRTDDS 1.0 Validation Checks,” for a full listing of validation checks.

More complete documentation, derived from the code header, is provided for each check macro in Appendix 4, “Macro Application Programming Interface.” See also “Special Topic: Validation Customization” on page 116 for tips on building new check macros.

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

Overview

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

SAS Clinical Standards Toolkit interprets usesourcemetadata to point to the correct set of metadata. If usesourcemetadata is set to Y, this tells SAS Clinical Standards Toolkit that the source metadata (source_tables and source_columns) is to be used to derive the set of 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.

SAS Clinical Standards Toolkit then interprets the tablescope and columnscope values to build the work._csttablemetadata and work._cstcolumnmetadata data sets. Based on the values of these scoping fields, SAS Clinical Standards Toolkit creates a subset of either source metadata or reference metadata (see point about usesourcemetadata) that represents the union of tablescope and columnscope. That is, SAS Clinical Standards Toolkit builds a set of columns specified in columnscope that also exist in the tables specified in tablescope.

For those checks that use codelogic, SAS Clinical Standards Toolkit builds local macro variables to communicate tablescope and columnscope settings to the codelogic code.

Simple examples: each domain is interpreted as &_cstDSName and each column as &_cstColumn.

Codelogic is run, generally setting _cstError=1 if codelogic is a statement (codetype=1 or 3) or creating work.cstproblems if codelogic is a DATA step or PROC SQL code segment (codetype=2 or 4).

Case Study 1: CDISC-SDTM, Check SDTM0604

Are the sequence numbers (**SEQ) used in various domains consecutively incremented beginning at 1 for each USUBJID?

What values should be assigned to usesourcemetadata, tablescope, and columnscope to set up a proper test of sequence numbers? First, we are interested in the domains we actually have (that is, source data and metadata), so usesourcemetadata should be set to Y. Next, we want to test all domains that contain sequence number, so tablescope should be set to _ALL_. Then, because each domain uses a domain-specific name for sequence number, we will set columnscope to "**SEQ".

The codelogic for CDISC-SDTM, check SDTM0604 is listed below:

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

Note the use of the following five macro variables, which are representative of variables set in many of the check macros before calling codelogic. See each validation check macro for local macro variables available to codelogic.

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

Processing based on this set of validation master metadata field values results in records being added to work._cstproblems for any record that does not match the record counter within the subject.

Note, however, that there are two records in the validation master check data set for CDISC-SDTM, check SDTM0604, and that the tablescope and columnscope settings for each record differ from the description above. The CDISC-SDTM TS (Trial Summary) domain does not contain the subject key USUBJID. While the codelogic above will run against TS without failing (the SAS log does provide an indication of a problem: NOTE: Variable first.USUBJID is uninitialized.), a more correct solution is offered in the validation master check data set with the two records.

Table 6.17 Example of Multiple Validation CheckIDs

checkid	tablescope	columnscope	codelogic

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

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

Case Study 2: CDISC-SDTM, Check SDTM0623

In the CDISC-SDTM findings domains, are the values for standard units (**STRESU) consistent within each test code (**TESTCD) across all records?

As in case study 1, we are interested only in the domains we actually have (that is, source data and metadata), so usesourcemetadata should be set to Y. Next, we want to test all findings domains (which typically contain these two domain columns of interest), so tablescope might be set to CLASS:FINDINGS. Then, because we want to compare two columns in each domain, we set columnscope to "[**TESTCD][**STRESU]". (For more information about tablescope and columnscope syntax, see Table 6.3 on page 63.)

The codelogic for CDISC-SDTM, check SDTM0623 is listed below:

```
data work._cstunique;
    set work._cstunique;
          &_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 illustrates the way SAS Clinical Standards Toolkit uses local macro variables for column comparisons. The columnscope syntax [**TESTCD][**STRESU] tells SAS Clinical Standards Toolkit to create two sublists, the first for all TESTCD columns, the second for all STRESU columns. These are referenced as &_cstColumn1 and &_cstColumn2 in codelogic, respectively.

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

This check, as we have defined it above, will fail. The generated results data set contains the following record excerpt:

Figure 6.12 Results Data Set Excerpt, 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 as to the problem. SAS Clinical Standards Toolkit resolves the columnscope sublist [**TESTCD] to five columns and the sublist [**STRESU] to four columns. 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 intended comparison.

So what is the solution? Recall that the tablescope and columnscope syntax is intended to support various wildcarding, as well as addition and subtraction operators. This functionality is not required and can be replaced with explicit table and column references. CDISC-SDTM, check SDTM0623 could be defined in the validation master data set as:

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

But, tablescope and columnscope syntax also allows:

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 also excluded from tablescope is that 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 IR4006). SDTM0623 is simply a generalization of SDTM0631 to include all findings domains, and there is no reason to redundantly test LB.

Case Study 3: CDISC-SDTM, Check SDTM0452

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

This is an example of a validation check with a very specific implementation using no wild carding and hardcoded column references.

Figure 6.13 Validation Check Metadata, Check SDTM0452

checkid	codesource	tablescope	columnscope	codelogic	reportingcolumns

SDTM0452	cstcheck_column	AE	AESER	if (upcase(&_cstColumn)='Y' and
				(upcase(AESCAN) ne 'Y' and
				upcase(AESCONG) ne 'Y' and
				upcase(AESDISAB) ne 'Y' and
				upcase(AESDTH) ne 'Y' and
				upcase(AESHOSP) ne 'Y' and
				upcase(AESLIFE) ne 'Y' and
				upcase(AESMIE) ne 'Y' and
				upcase(AESOD) ne 'Y')) then
				_cstError=1;
					AESCAN
					AESCONG
					AESDISAB
					AESDTH
					AESHOSP
					AESLIFE
					AESMIE
					AESOD

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

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

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

What are our options?

35 Don't run the check at all. Disabling a check can be done by removing it from the validation master data set or by setting the checkstatus flag to some value other than 1 (assuming the process you use to extract checks into the run-time validation control data set references the checkstatus field).

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

37 Can columnscope be modified to include the qualifier columns we do have? No. Remember that columnscope defines the scope of the primary columns to be tested. The check macro code loops through the resolved set of columns from columnscope, evaluating the validity of each. In this case, we would not want to test each of the qualifier columns against each other.

38 Modify the current check record metadata with updated codelogic and reportingcolumns values. This seems simple enough: just remove the columns not collected in the source study. Your best practices will control this process, including whether:

existing checks can be modified

any change in the validation master superset of checks requires a new validation master

all changes made require assignment of a new checkid (for example, CUST0452) and other new metadata (for example, checktype=CUSTOM)

how the uniqueid field is modified to reflect the new check

Users should consider the implications of modifying existing SAS checks with regard to future product updates and synchronization of changes. See the Special Topic "Validation Customization" for more information.

39 Submit a bug fix request to SAS to modify the existing check to be more generic and flexible to accommodate your scenario. For example, modify the codelogic to check that each variable exists first before attempting to evaluate its value.

Special Topic: SAS Implementation of ISO 8601

ISO 8601 is a widely used data standard for dates, times, durations, and intervals. The values are stored as text strings 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 3.1.1.

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

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

- Consistent with CDISC-SDTM 3.1.1 guidelines, SAS Clinical Standards Toolkit does not support the ISO 8601 basic format. This means that the strings must contain the dash (-) delimiter for parts of the dates and the colon (:) delimiter for parts of the time.
- SAS Clinical Standards Toolkit does not support some of the rarely used formats allowed by the ISO 8601 specification. The week (W) formats for dates, Julian dates, and extended dates (used to denote years greater than 9999) are not supported.
- SAS Clinical Standards Toolkit requires a SAS hot fix for the ISO informats.

  Several enhancements have been made to the SAS informats $N8601B. and $N8601E. to enable them to provide even better support for CDISC use of ISO 8601. This includes backporting the SAS informats for use with SAS 9.1.3. These enhancements are available as a free download as a SAS hot fix

(http://ftp.sas.com/techsup/download/hotfix/hotfix.html). See the SAS Clinical Standards Toolkit installation instructions for more details.

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

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

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

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

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

The follow table provides an overview of some of the more commonly found values, grouping the comments according to the ISO 8601 string type.

Table 6.18 Example ISO 8601 Values

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

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

String	Interpretation	Comments
	of an unknown year	
2009-03	Month of March 2009	Trailing partial values can be omitted (truncated). If the time is omitted, then the T must also be omitted.
2009-03--T12	The 12th hour of an unknown day in March 2009	Missing day of month.
	Durations: Template	
PnYnMnDTnHnMnS	Duration	A span of time where n is the number of the unit that follows the unit. P: indicates the value is a duration (period) nY: n elapsed years nM: n elapsed months nD: n elapsed days T: What follows is the elapsed time in hours, minutes, and seconds nH: n elapsed hours nM: n elapsed minutes nS: n elapsed seconds Typically, only the units with actual values are given. For example, P0Y1M would be given as just P1M.
	Durations: Examples	
P1D	The span of one day	Durations always start with P for a period of time. Units of time that are not known are usually omitted. If time is completely omitted, the T must also be omitted.
P0000-00-01	The span of Zero years + Zero Months + One Day	Durations can also be expressed in an alternative format. When expressed, the length of time is stored in the same format as dates and times, but preceded by a P. Instead of expressing a specific point in time, it is giving 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 should not be given if no time values are given.

String	Interpretation	Comments
	Intervals: Template	
PnYnMnDTnHnMnS/ YYYY-MM-DDTHH:MM:SS or YYYY-MM-DDTHH:MM:SS/PnYnMnDTnHnMnS or YYYY-MM-DDTHH:MM:SS/ PnYnMnDTnHnMnS or YYYY-MM-DDTHH:MM:SS/ YYYY-MM-DDTHH:MM:SS	Intervals	This is a duration that is anchored to a specific point in time.
	Intervals: Examples	
2009-03-25T22:29/P1Y	The span of one year starting at 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.

Table 6.19 SAS ISO 8601 References

Topic	Link
SAS 9.2 Language Reference: Dictionary	**http://support.sas.com/documentation/cdl/en/lrdict/61724/HTML/default/a002295669.htm**
Working with Dates and Times Using the ISO 8601 Basic and Extended Notations	**http://support.sas.com/documentation/cdl/en/lrdict/61724/HTML/default/a003169814.htm**
CALL IS8601_CONVERT Routine	**http://support.sas.com/documentation/cdl/en/lrdict/61724/HTML/default/a003156604.htm**
$N8601Bw.d Informat	**http://support.sas.com/documentation/cdl/en/lrdict/61724/HTML/default/a003170563.htm**
$N8601Ew.d Informat	**http://support.sas.com/documentation/cdl/en/lrdict/61724/HTML/default/a003170574.htm**
Reading Dates and Times Using the ISO 860 Basic and Extended Notations	**http://support.sas.com/documentation/cdl/en/lrdict/61724/HTML/default/a003169817.htm**
SAS Hot Fix for the ISO Informats	**http://ftp.sas.com/techsup/download/hotfix/hotfix.html**

Special Topic: Debugging a Validation Process

SAS Clinical Standards Toolkit provides two properties or global macro variables relevant to debugging problems occurring with all processes: _cstDebug and _cstDebugOptions.

_cstDebug toggles debugging options on and off. Many SAS Clinical Standards Toolkit code modules have conditional branching such as:

```
%if &_cstDebug %then
%do;
    /* perform some action */
end;
```

If debugging is toggled on (_cstDebug=1), several things can happen:

- If code is in place, like the following excerpt from the sample driver module (validate_data.sas) documented in “Running a Validation Process” on page 78, additional messaging to the SAS log can be enabled.

  ```
  data _null_;
      _cstDebug = input(symget('_cstDebug'),8.);
  if _cstDebug then call execute("options source source2
  &_cstDebugOptions;");
  else call execute("options source source2 nomlogic nomprint
  nosymbolgen;");
  run;
  ```

  By default, the &_cstDebugOptions global macro variable is set to:

  ```
  mprint mlogic symbolgen mautolocdisplay
  ```

  Be aware that these SAS options generate a great deal of information, quickly filling the SAS log when running interactively. It might be advisable in these cases to either run the process in batch or to use PROC PRINTTO to redirect the SAS log to a file.

- Many work files created during the process are not deleted, but remain available in the Work library to help with debugging.

Each SAS Clinical Standards Toolkit process consists of two primary tasks: setup routines to establish the SAS Clinical Standards Toolkit environment, and a task to perform some primary SAS Clinical Standards Toolkit action. Debugging focus is different for these two tasks.

In SAS Clinical Standards Toolkit setup, errors most often occur because of problems with the SASReferences data set. The following table lists some of the commonly occurring errors with possible causes. See "Building a SASReferences File" in Chapter 5 for recommendations on configuring the SASReferences file appropriately.

Table 6.20 Debugging Process Setup Errors

Error	Location Where Error Is Reported	Possible Cause/Corrective Action
Expected libraries are not allocated	SAS Log, Libraries window, SAS DMS	(1) An invalid physical name for the libref has been used. Is the libref a valid SAS name? A SAS name can contain from one to 32 characters. It must start with a letter or an underscore (_), not a number. Subsequent characters must be letters, numbers, or underscores. Blanks cannot appear in SAS names. Is the libref a reserved SAS libref name? You should not use Work, Sasuser, or Sashelp. (2) The path specified for the libref is invalid; it points to a nonexistent directory. Check the path in your SASReferences data set.
Error: SAS system library WORK cannot be reassigned	SAS log	Work used as a sasref value with or without a path being designated. A similar error appears 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.
Warning: Process ending prematurely for CST0090–there were problems with the sasreferences data set.	SAS log	There is a problem with the sasreferences data set being used. Check for these potential causes: The sasreferences data set does not exist. The sasreferences data set exists but is empty. The structure of the sasreferences data set is incorrect; for example, it might have an extra column that is not required or an expected column that is missing. A column type might be incorrect; for example,

Error	Location Where Error Is Reported	Possible Cause/Corrective Action
		the Order column might be character instead of numeric.
		An invalid TYPE or SUBTYPE or combination is used in the sasreferences data set. Valid TYPE and SUBTYPE values are provided in the standardlookup data set found in **`C:\cstGlobalLibrary\standards\cst-framework\control`** (assuming that cstGlobalLibrary is the name and it is installed to **`c:\`**).
		A TYPE value is missing.
		A SASREF value is missing or invalid.
		A REFTYPE value is missing or is not equal to libref or fileref (case insensitive).
Error: Physical file does not exist.	SAS log	(1) The sasreferences data set references a file that does not exist.
		(2) The filename is not a valid SAS name.
WARNING: Apparent invocation of macro SDTM_VALIDATE not resolved.	SAS log	(1) The macro is misnamed or has not been added to the expected autocall library.
		Does the macros folder for this standard exist in the cstGlobalLibrary, in the !sasroot hierarchy, or in some correctly designated custom location?
		(2) The expected autocall path was not created correctly in the call to %cstutil_allocatesasreferences.
		Check that the sasreferences data set contains a type=autocall record, defined as a fileref, and pointing to the correct folder location.
		Check for an error occurring earlier in the SAS log suggesting %cstutil_allocatesasreferences failed before setting the autocall path.

If the task to perform the primary SAS Clinical Standards Toolkit action begins (that is, the standard-specific validation macro, such as %sdtm_validate or %crtdds_validate, is found and begins processing), setup has usually completed successfully and remaining process failures are likely due to problems with the various validation components.

Most errors that halt a validation process are reported in the results data set. As a general rule, the following results data set fields signal process failures and provide information about the cause of the failure:

- the Process status field (_cst_rc), when the value is set to a nonzero value
- the Problem detected field (resultflag), when 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 have been saved to the persisted location if that has been 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 information itemized above. By default, &_cstResultsDS is set to work._cstresults.

Generally, 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 as well.

Some of the more common causes for premature process failure or failure of specific checks to run are itemized in the following table:

Table 6.21 Debugging Validation Process Errors

Error	Resultid in Results Data Set	Possible Cause/Corrective Action
No tables evaluated-check validation control data set	CST0002	No tables interpreted from the tablescope value could be found in the work._csttablemetadata data set.
	CST0003	This error usually indicates that a specific source column or data set could not be found. The code loops through a set of domains or columns built from the source metadata data sets. This error might result when the source metadata does not accurately reflect the source data.
No columns evaluated-check validation_control specification	CST0004	No columns interpreted from the columnscope value could be found in the work._cstcolumnmetadata data set. Remember that 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 specified in tablescope.
Lookup to SASreferences control data set failed	CST0006	The SAS Clinical Standards Toolkit code has a call to the cstutil_getsasreference utility macro for a type or type and subtype combination that cannot be found in the SASReferences data set. This indicates SASReferences has been incompletely defined for the SAS Clinical Standards Toolkit validation process.
Validation control parsing of tablescope/column results in inconsistent sublist lengths	CST0023	This check involves a comparison of tables or columns, as indicated by multiple sets of brackets in tablescope or columnscope. Each set of brackets constitutes a sublist. However, the number of items in the specified sublists 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_-domain)

Error	Resultid in Results Data Set	Possible Cause/Corrective Action
One or more check metadata column values is invalid	CST0026	A value in the validation control data set for the check being run is invalid in the context of the specific check macro. Examples include conditions required by the check macro but not found, such as no codelogic found, an unexpected usesourcemetadata value, or no lookuptype or lookupsource for valid value assessments.
Code failed due to SAS error–see log	CST0050	A SAS DATA step or SAS procedure failed, with the cause reported in the SAS log. This most commonly occurs because of missing data sets, missing columns, incorrectly sorted data sets, and unexpected macro variable values.
"<Message lookup failed to find matching record>"	<varies>	The check macro code generates a resultid value that does not find a match in the messages data set. Either the wrong resultid has been specified or the standard-specific messages data set has not been updated to include the resultid.

Other Debugging Tips

- Review available Work files for clues to the error condition (for example, _cstresults, _csttablemetadata, and _cstcolumnmetadata). These files might remain in the Work directory at the conclusion of some processes by default (generally for more serious errors), but toggling the _cstDebug global macro variable to 1 also retains many Work files at the end of a process.
- When debugging, avoid setting the parameter flags in cstutil_cleanupcstsession to 1 (if that cleanup macro is called):

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

- Use work._cstcolumnmetadata and work._csttablemetadata to resolve missing domain and column issues and sublist length differences (for checks using sublist syntax [] in tableScope and columnScope).
- Use the resultid code (for example, CST0003) found in the results data set to search through the check macro code module used for a specific check (as set in the validation control codesource field) for clues as to the possible cause of the error.

Special Topic: Validation Customization

Overview

One of the significant benefits of SAS Clinical Standards Toolkit is that users can customize the solution to meet their needs. From a validation perspective, this includes:

- modifying an existing standard or defining a new reference standard
- using any set of source data and metadata
- modifying the SAS validation checks for supported standards

- adding new validation checks for supported standards
- modifying existing validation check macros or adding new macros
- modifying SAS Clinical Standards Toolkit messaging, including internationalization

Each of these is described in the case studies below.

Case Study 1: Modifying an Existing Standard or Defining a New Reference Standard

Source data and metadata are validated in SAS Clinical Standards Toolkit against a reference standard. For CDISC standards, SAS Clinical Standards Toolkit supplies 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 have additional domains reflecting the focus of the clinical investigation. In addition, CDISC- SDTM domain classes (findings, events, and interventions) allow the inclusion and exclusion of most columns, depending on the clinical data points collected in the study. Finally, CDISC guidelines generally do not specify column lengths.

Each of these factors suggests that the SAS Clinical Standards Toolkit CDISC reference standards will usually be modified or replaced entirely with customer-derived standards. SAS Clinical Standards Toolkit offers the option of building a reference standard to encompass domain and column customizations, or to customize check macros and check logic to perform specific compliance assessments to a referenced 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, SUPPQUAL). It is at the customer's discretion whether the reference standard is modified to include each unique supplemental qualifier domain, or to alternatively use either 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 further in the case studies below.

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 within the SASReferences file (as type=referencemetadata).

See "Registering a New Version of a Standard" in Chapter 2 for details on registering a new standard to SAS Clinical Standards Toolkit.

Case Study 2: Using Any Set of Source Data and Metadata

From a SAS Clinical Standards Toolkit perspective, what defines a source study? It is the study domains, the study metadata represented in the source_tables and source_columns data sets, and anything else that might be unique to a specific study, including controlled terminologies, properties, validation checks, and associated messages.

A key SAS Clinical Standards Toolkit requirement is that the information about source studies be kept in sync and that all relevant elements of each study be accurately represented in a SASReferences file. The synchronization of study elements is a task that must occur outside SAS Clinical Standards Toolkit. That is, the study data libraries

must contain the domains of interest, the study metadata must provide the complete set of table- and column-level metadata necessary to describe the source data, and any format catalogs and coding dictionaries supporting the study must be available.

Best Practice Recommendation: If a standard folder hierarchy is adopted for source studies, such as that provided with the SAS Clinical Standards Toolkit CDISC-SDTM sample study (SAS 9.1.3: `!sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-3.1.1/sascstdemodata`), using more 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 includes adding multiple instances of existing checks. The most common reasons for modifying the SAS validation checks include:

- Altering the scope of the domains and columns to be validated. This change should not be required frequently, as many of the checks are defined to be run against specific domains or columns, against specific classes of domains (for example, CDISC-SDTM findings, events, or interventions), or against all available domains or columns. Changes are likely to involve alterations to the validation control tablescope or columnscope fields.
- Changing the validation control codelogic field to alter the logic used to identify error conditions. This might be a necessary change if a check needs to be generalized to accommodate new domains or columns, or as customer conventions differ from those in the SAS Clinical Standards Toolkit checks.
- If customer code changes are sufficiently significant, it might be advisable to create a new validation check macro. (See “Case Study 5: Modifying Existing Validation Check Macros or Adding New Macros” on page 119.) The validation control codesource field would then be modified to contain the name of the new macro.
- The validation control uniqueid field is designed to provide a means to uniquely identify a specific validation check for reference at some future point. Any change to any validation control data set check field would normally lead to the derivation of a new uniqueid. See Table 6.3 on page 63 for details on the structure of uniqueid.
- The validation control field checkstatus provides an easy way to identify selected checks with some user-defined status (for example, draft, deprecated, or not available for a given study). Note that SAS Clinical Standards Toolkit does not reference this field within any validation check macro.
- Should there be a need to reference a different SAS format or lookup data set (for example, a new version of meddra), this change would be noted in the validation control lookupsource field.

Case Study 4: Adding New Validation Checks for Supported Standards

Should the need arise to add a new validation check, the following checklist itemizes steps to consider:

- Check metadata must conform to the validation master structure (see Chapter 2, "Framework," for details).
- Certain validation master fields accept any user-defined value (for example, checksource, sourceid, checktype, standardref, checkstatus), and are not referenced by the validation check macros. The remaining fields are used in the check macros, so users must abide by some conventions imposed by SAS Clinical Standards Toolkit. These are also described in Chapter 2, "Framework."
- Following a typical development process, a new check would be added to the (run time) validation control data set for testing, and then promoted to the validation master when the check is to be made available to consuming applications and processes.
- For each new validation check, a matching message is required. This is the message to be written to the results data set when an error condition is detected. For details, see "Messages" on page 69.

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

SAS Clinical Standards Toolkit provides 13 validation check macros that offer a variety of code samples that function within the general SAS Clinical Standards Toolkit framework. These 13 macros support CDISC-SDTM 3.1.1 validation; CDISC-CRTDDS 1.0 uses six of these macros. (See the Special Topic "Validation Check Macros" for a full description of these macros.)

There are a number of validation scenarios that might require modifications to the SAS Clinical Standards Toolkit check macros or the derivation of new macros. When this becomes necessary or simply wanted for any reason, the following guidelines should be followed to facilitate use within both the general SAS Clinical Standards Toolkit framework and the specific SAS Clinical Standards Toolkit validation framework:

- Follow the current, or adopt a consistent naming convention that conforms to SAS naming conventions.
- Follow the current, or use a customized autocall library that has been defined in the SASReferences data set (type=autocall).
- Conform to the basic check macro workflow, as described in "Special Topic: Validation Check Macros" on page 98.
- Ensure that the macro correctly accepts and interprets the metadata provided as input from the validation control data set. If any new check macros do not do so, specific macros can be written to address specific validation checks (that is, they are hardcoded to provide very specific functionality).
- Ensure that the macro writes appropriate output to the results and metrics data sets.

Case Study 6: Modifying SAS Clinical Standards Toolkit Messaging, Including Internationalization

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

40 Maintaining the relationship between SAS Clinical Standards Toolkit standard-specific messages and standard-specific validation checks

41 Maintaining the relationship between messages and validation check macro code

(Deviations possible to the extent that missing parameters have suitable defaults)

42 Internationalization of messages

A SAS Clinical Standards Toolkit message is created for each distinct combination of the validation master standard and checksource fields. This allows SAS Clinical Standards Toolkit to support checksource-specific messaging and severity. A unique SAS Clinical Standards Toolkit message is also required for each value of the validation master standardversion field, if that value is not the wildcard ***.

Consider the following CDISC-SDTM 3.1.1 validation master record excerpts:

Figure 6.14 Validation Master Data Set Excerpt, 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 this check in the messages data set is:

Figure 6.15 Messages Data Set Excerpt, 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 within the validation master checkid.

Now, consider the following CDISC-SDTM validation master record excerpts:

Figure 6.16 Validation Master Data Set Excerpt, 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 pointing to a different domain (tablescope). This example assumes 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 only in CDISC-SDTM 3.1.2 (as CE is a new domain added in SDTM 3.1.2).

What should the messages data set look like for this check? Only two messages data set records are required:

Figure 6.17 Messages Data Set Excerpt, 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

In summary, it is the distinct combinations of the validation master checkid, standardversion, and checksource fields that control the associated messages records.

It is also important to maintain the relationship between messages and validation check macro code. If the validation macro code references an unknown resultid, the text <Message lookup failed to find matching record> is written to the results data set.

As defined above, CUST0073 defines a substitution parameter (&_cstParm1). (Note that SAS Clinical Standards Toolkit code assumes that message substitution parameters begin with the string "&_cst".) But, does the calling check macro support parameters when writing output to the results data set? If so, the parameter passed from the check macro should be syntactically consistent with the messagetext field in the messages data set.

In this case, building the message record to use a default value (as specified in the parameter1 field) covers the case of when the calling code fails to pass a substitution value. Use of parameters is optional and only necessary if the message is to be used in multiple contexts where substitution of one or two parameter values helps interpretation of the message.

Finally, SAS Clinical Standards Toolkit supports internationalization of messages through the specification of message file references in the SASReferences data set (type=messages). As long as the referenced message files conform to the structure expected by SAS Clinical Standards Toolkit, any text, including internationalized text, can be included.

Special Topic: Validation Reporting

To illustrate how SAS Clinical Standards Toolkit metadata and results can be summarized in a report format, several sample reports are available with SAS Clinical Standards Toolkit. As with all SAS Clinical Standards Toolkit content, these reports are offered as templates that can be modified as appropriate to facilitate data review. The report templates are PROC REPORT implementations that use ODS to generate report output in a variety of ODS-supported formats. Two sample reports are initially provided:

- Report 1: A report applicable to most SAS Clinical Standards Toolkit processes, itemizing records as they 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: A report specific to SAS Clinical Standards Toolkit validation processes for standards that have the concept of source data domains (for example, CDISC-SDTM and CDISC-ADAM). Results are summarized by domain.

Each report consists of multiple sections or panels, each of which 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 file, a summary of validation metrics, and a general process messaging panel.

As with other SAS Clinical Standards Toolkit processes, a sample driver code module is provided to define the SAS Clinical Standards Toolkit environment and to call the primary action framework macro (%cstutil_createreport). The following excerpt from the driver module 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 below:
    (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 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:
       - a SASReferences data set
       - 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
```

Generally, the reporting as implemented in SAS Clinical Standards Toolkit attempts to address the following two common scenarios:

43 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), generation of one or both reports is wanted. In this scenario, the reporting process uses the SASReferences file 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 as created and (optionally, as specified in the SASReferences data set) persisted to some location that is not the SAS Work location. Other files required by the report are identified in Table 6.22 on page 123.

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.

44 The results data set created in some prior SAS Clinical Standards Toolkit session is available and generation of one or both reports is desired. 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 containing the following sample message text:

```
Message
PROCESS STANDARD: CDISC-SDTM
PROCESS STANDARDVERSION: 3.1.1
PROCESS DRIVER: SDTM_VALIDATE
PROCESS DATE: 2009-05-21T08:17:40
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.

Table 6.22 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 is used to attempt to use the referenced file. &_cstSASRefs is set to this file.
Results	Precedence: The file referenced in &_cstSASRefs with type=results and subtype either results or validationresults The file referenced by &_cstResultsDS	As provided by the user in the driver module (cst_report.sas) _cstRptResultsDS macro variable.
Metrics	Precedence: The file referenced in &_cstSASRefs with type=results and subtype either metrics or validationmetrics The file referenced by &_cstMetricsDS	The file referenced in &_cstSASRefs with type=results and subtype either metrics or validationmetrics
Validation Control	The file referenced in &_cstSASRefs with type=control and subtype=validation	The file 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: Ideally, report output locations can be defined in the SASReferences data set. This is a planned enhancement to the supported SASReferences types as documented in the framework standardlookup data set as discussed in Chapter 2, “Framework.”

The reporting driver module makes one or more calls to the utility reporting macro. At a minimum (using default parameter values), a simple macro call to create the domain report 2 might include:

```
%cstutil_createreport(_cstsasreferencesdset=&_cstSASRefs,_cstreportbydoma
in=Y, _cstreportformat=PDF,_cstreportoutput=&studyrootpath/reports/);
```

The full set of supported parameters in the sample cstutil_createreport macro is described in the following table:

Table 6.23 Create Report Macro Parameters

Parameter	Description
_cstsasreferencesdset	libref.dataset of SASreferences data set used for a specific process. Optional, if specified, the _cstresultsdset and _cstmetricsdset parameters will be ignored. Either _cstsasreferencesdset or _cstresultsdset must be provided.
_cstresultsdset	libref.dataset of SAS Clinical Standards Toolkit process results data set. Optional. Either _cstsasreferencesdset or _cstresultsdset must be provided. Ignored if _cstsasreferencesdset specified.
_cstmetricsdset	libref.dataset of SAS Clinical Standards Toolkit process metrics data set. Optional. Ignored if _cstsasreferencesdset specified.
_cstreporterrorsonly	If N (default), report all records in the results data set, including informational and non-error results. If Y, report only records in error (where the results data set field resultflag=1).
_cstreportobs	If null (default), report all records in error (where results.resultflag=1) in the results data set. Otherwise, set to any integer value > 0, signifying the number of records to print per checkid (where results.checkid is non-null). If _cstreportobs >0 results in excluding any records, a footnote is printed, noting that not all records were printed.
_cstreportbydomain	If N (default), do not report results by domain (that is, run report 1). If Y, report results by domain (that is, run report 2).
_cstdomaindset	Optionally produce report based on a specific domain, indicated by libref.data set. If blank or the keyword _ALL_ is specified, all domains are included in the report. Ignored if _cstreportbydomain=N.
_cstreportformat	HTML (default). Allows specification of other ODS-supported formats (for example, PDF and RTF).
_cstreportoutput	The directory location for the generated report output. The value should not include the filename, which is currently hardcoded to either CSTCheckIDReport + the _cstreportformat (report 1) or CSTDomainReport + the _cstreportformat (report 2). Required.
_summaryReport	(Y/N) If set to Y, generate the report summary panel. Default=Y.
_ioReport	(Y/N) If set to Y, generate the process inputs/outputs panel. Default=Y.
_metricsReport	(Y/N) If set to Y, generate the process metrics panel. Default=Y. This should be set to N for any non-validation reports and cases where metrics were not generated.
_generalResultsReport	(Y/N) If set to Y, generate the general process reporting panel. Default=Y.
_checkIdResultsReport	(Y/N) If set to Y, generate the process results panel. Default=Y.

The following figures illustrate report content and apply to report 1 (by checkid) unless otherwise indicated.

Figure 6.18 Sample Report Summary

SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Report Summary

Report Parameter	Value
SASReferences data set	work._cstsasrefs
Results data set	results.validation_results
Metrics data set	results.validation_metrics
CST Process datetime	2009-06-01T14:22:14
Report only errors?	Yes
# records to report	50
Report results by domain	No
Report format	pdf
Report output file	C:\Playpen\Validation\Reports\Output\CSTCheckIDReport.pdf

Figure 6.19 Sample Report, Process Inputs/Outputs

SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Inputs/Outputs

Type	Path
Autocall Libraries	(sdtmcode sasautos)
	sdtmcode: c:/cstGlobalLibrary\standards\cdisc-sdtm-3.1.1\macros
Format Search Path Libraries	(srcfmt cstfmt)
	srcfmt: !sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-3.1.1/sascstdemodata\terminology\formats
	cstfmt: c:/cstGlobalLibrary\standards\cdisc-terminology-200810\formats
Reference Metadata	c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.1/metadata
Source Data	!sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-3.1.1/sascstdemodata\data
Source Metadata	!sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-3.1.1/sascstdemodata\metadata

Figure 6.20 Sample Report 1, Process Metrics

SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Metrics

Summary Metrics		Check Metrics				
Metric	#	Check ID	# Check Invocations	# Recs (if available)	# Errors	# Check Invocations Not Run
# of distinct check invocations	14	SDTM0001	1	16	0	0
# check invocations not run	1	SDTM0004	1	1	1	0
Errors (severity=High) reported	0	SDTM0011	1	1	0	0
Warnings (severity=Medium) reported	26	SDTM0101	1	10	0	0
Notes (severity=Low) reported	1939	SDTM0202	1	1927	1918	0
Structural errors, warnings and notes	1	SDTM0208	1	4	0	0
Content errors, warnings and notes	1965	SDTM0218	1	7	0	0
		SDTM0601	1	16	0	0
		SDTM0602	1	20	6	0
		SDTM0604	1	8	0	0
		SDTM0802	1	1	0	0
		SDTM0804	2	50	40	0
		SDTM0851	1	1	0	1

Figure 6.21 Sample Report 2, Process Metrics (by Domain)

SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Metrics

Summary Metrics		Domain Metrics				
Metric	#	Domain	# Check Invocations	# Recs (if available)	# Errors	# Check Invocations Not Run
# of distinct check invocations	14	AE	7	15	9	0
# check invocations not run	1	DM	7	30	24	0
Errors (severity=High) reported	0	DS	8	10	2	0
Warnings (severity=Medium) reported	26	EG	10	459	447	0
Notes (severity=Low) reported	1939	IE	8	9	0	0
Structural errors, warnings and notes	1	LB	10	248	236	0
Content errors, warnings and notes	1965	SC	9	14	4	0
		SU	9	50	40	0
		SUPPAE	6	6	1	0
		SV	6	6	0	0
		TA	5	13	9	0
		TE	5	5	0	0
		TI	5	5	0	0
		TS	5	5	0	0
		TV	5	9	5	0
		VS	10	1200	1188	0

Figure 6.22 Sample Report, General Process Reporting

SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

General Process Reporting

Seq #	Source Data	Result Identifier	Severity	Problem Detected?	Message
1	CST_SETPROPERTIES	CST0108	Info	No	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cst-framework/programs/initialize.properties
1	CST_SETPROPERTIES	CST0108	Info	No	The properties were processed from the PATH c:/cstGlobalLibrary/standards/cdisc-sdtm-3.1.1/programs/initialize.properties
1	CST_SETPROPERTIES	CST0108	Info	No	The properties were processed from the PATH !sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-3.1.1/sascstdemodata\programs/validation.properties
1	SDTM_VALIDATE	CST0200	Info	No	PROCESS STANDARD: CDISC-SDTM
2	SDTM_VALIDATE	CST0200	Info	No	PROCESS STANDARDVERSION: 3.1.1
3	SDTM_VALIDATE	CST0200	Info	No	PROCESS DRIVER: SDTM_VALIDATE
4	SDTM_VALIDATE	CST0200	Info	No	PROCESS DATE: 2009-06-01T14:22:14
5	SDTM_VALIDATE	CST0200	Info	No	PROCESS TYPE: VALIDATION
6	SDTM_VALIDATE	CST0200	Info	No	PROCESS SASREFERENCES: !sasroot/../SASClinicalStandardsToolkitSDTM311/9.1.3/sample/cdisc-sdtm-3.1.1/sascstdemodata\control/sasreferences.sas7bdat

Figure 6.23 Sample Report 1, Validation Results, by CheckID

SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Results, CheckID: SDTM0202

Description: Identifies a null (empty) value found in a column where (Standard) Core attribute is 'Exp'
Check scope: (Tables) _ALL_, (Columns)
Source: SAS (SAS0015)
Validation check macro: cstcheck_violatesstd, using source metadata

Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
1	1	SRCDATA.AE	SDTM0202	Null value in column	Note	Yes	AEREL=	STUDYID=SASCSTDEMODATA, USUBJID=S001P005,AETERM= ABDOMINAL CRAMP,AESTDTC=2008-02-22
1	2	SRCDATA.AE	SDTM0202	Null value in column	Note	Yes	AEREL=	STUDYID=SASCSTDEMODATA, USUBJID=S001P005,AETERM= PRURITIC RASH, AESTDTC=2008-03-10T21:00
1	3	SRCDATA.AE	SDTM0202	Null value in column	Note	Yes	AEREL=	STUDYID=SASCSTDEMODATA, USUBJID=S002P003,AETERM= HEADACHE, AESTDTC=2008-03-17T00:30
1	4	SRCDATA.AE	SDTM0202	Null value in column	Note	Yes	AEREL=	STUDYID=SASCSTDEMODATA, USUBJID=S002P010,AETERM= DIARRHEA, AESTDTC=2008-03-21T10:00
1	5	SRCDATA.AE	SDTM0202	Null value in column	Note	Yes	AEREL=	STUDYID=SASCSTDEMODATA, USUBJID=S002P013,AETERM= HEARTBURN, AESTDTC=2008-04-06T09:30
1	6	SRCDATA.AE	SDTM0202	Null value in column	Note	Yes	AEREL=	STUDYID=SASCSTDEMODATA, USUBJID=S002P027,AETERM= RASH, AESTDTC=2008-05-12T02:00
1	7	SRCDATA.AE	SDTM0202	Null value in column	Note	Yes	AEREL=	STUDYID=SASCSTDEMODATA, USUBJID=S002P031,AETERM= HEADACHE, AESTDTC=2008-05-21T01:30
1	8	SRCDATA.AE	SDTM0202	Null value in column	Note	Yes	AEREL=	STUDYID=SASCSTDEMODATA, USUBJID=S003P010,AETERM= PRURITIC RASH, AESTDTC=2008-06-01T14:00

1868 records have not been printed because a printing limit of 50 was requested
Report generated 2009-06-05T08:30:51 on process run 2009-06-01T14:22:14

Figure 6.24 Sample Report 2, Validation Results, by Domain

SAS Clinical Standards Toolkit 1.2
CDISC-SDTM 3.1.1 VALIDATION

Process Results, Domain: DM

Check ID	Check Invocation	Seq #	Source Data	Result Identifier	Message	Severity	Problem Detected?	Actual Value	Keys
SDTM0001	1	2	SRCDATA.DM	CST0100	No errors detected in SRCDATA.DM	Info	No		
SDTM0011	1	1	WORK._ CSTSRCCOLUMNMETADATA	CST0100	No errors detected in source data	Info	No		
SDTM0101	1	2	SRCDATA.DM	CST0100	No errors detected in SRCDATA.DM	Info	No		
SDTM0202	1	10	SRCDATA.DM	SDTM0202	Null value in column	Note	Yes	AGE=.	STUDYID= SASCSTDEMODATA, USUBJID=S001P001
SDTM0202	1	11	SRCDATA.DM	SDTM0202	Null value in column	Note	Yes	AGEU=	STUDYID= SASCSTDEMODATA, USUBJID=S001P001
SDTM0202	1	12	SRCDATA.DM	SDTM0202	Null value in column	Note	Yes	RACE=	STUDYID= SASCSTDEMODATA, USUBJID=S001P002
SDTM0202	1	13	SRCDATA.DM	SDTM0202	Null value in column	Note	Yes	RACE=	STUDYID= SASCSTDEMODATA, USUBJID=S001P003
SDTM0202	1	14	SRCDATA.DM	SDTM0202	Null value in column	Note	Yes	RACE=	STUDYID= SASCSTDEMODATA, USUBJID=S001P004
SDTM0601	1	2	SRCDATA.DM	CST0100	No errors detected in SRCDATA.DM	Info	No		
SDTM0602	1	2	SRCDATA.DM	CST0100	No errors detected in SRCDATA.DM	Info	No	keys=STUDYID USUBJID	
SDTM0802	1	1	SRCDATA.DM (SRCDATA.DS)	CST0100	No errors detected in source data	Info	No		

One or more records have not been printed because a printing limit of 5 was requested
Report generated 2009-06-05T08:30:53 on process run 2009-06-01T14:22:14

Special Topic: Performance Considerations

It should not be surprising that the validation checks that take the longest to run within SAS Clinical Standards Toolkit are those that:

- evaluate the source data (as opposed to the smaller source metadata data sets)
- run on all domains (rather than on single, targeted domains)
- require that the code evaluate cell-level values for one or more columns on every source data record

CDISC-SDTM Metrics

As a benchmarking exercise, the following test was performed to assess performance on a set of test data with the characteristics described below.

Benchmark Test Data

Source metadata: 22 domains, 407 columns

Source data: Study with 2000 patients, 14 visits, but with only the following domains reflecting these patient/visit numbers:

- DM (2000 records)
- DS (5888 records)
- SC (2479 records)
- SV (27272 records)
- LB (108864 records)
- VS (54432 records)
- SU (834 records)
- all other domains < 200 records

Validation checks run: 142 unique checks (excluding SDTM0450)

SAS submission details: SAS 9.1.3, Windows XP (3.2 GHz, 2 GB RAM), batch process

Benchmark Results

Table 6.24 CDISC-SDTM 3.1.1 Benchmark Results, Checks with Greatest Elapsed Time

Check	Duration (min:sec)	Macro	Description	Scope
SDTM0202	36:13	cstcheck_violatesstd	Null value in column (Exp)	_ALL_
SDTM0204	12:20	cstcheck_column	Column value contains numeric missing value	_ALL_
SDTM0201	11:15	cstcheck_violatesstd	Null value in column (Req)	_ALL_
SDTM0203	08:22	cstcheck_violatesstd	Character column value not upcased	_ALL_
SDTM0606	07:35	cstcheck_violatesstd	Non-numeric values detected	_ALL_
SDTM0604	04:56	cstcheck_notconsistent	Non-incremental **SEQ values	_ALL_-TS
SDTM0271	04:30	cstcheck_column	Data set key has missing value	_ALL_
SDTM0221	03:42	cstcheck_notincodelist	Value for variable not found in codelist	_ALL_
SDTM0631	02:16	cstcheck_notunique	Inconsistent value for Standard Unit across records	LB
SDTM0401	01:54	cstcheck_notincodelist	Invalid baseline flag value	CLASS:FINDINGS

This table lists the 10 longest running checks (greater than 1 minute elapsed time). Results were consistent with the expectations stated above. Note that SDTM0631 and SDTM0401, though not run on all available domains, were run on the largest findings domains. Also note that results are highly dependent on system load, data quality (that is, the number of errors detected and the overhead to write error records to the results data set), and the number of specific domains and columns that match the validation control scoping fields.

Best Practice Recommendations:

- SAS Clinical Standards Toolkit validation should first be run on a subset of source data to identify general process problems, missing or inconsistent process control metadata, and common (and perhaps correctable) data errors.
- The SAS Clinical Standards Toolkit standard-specific validation master data set should be subsetted to remove essentially duplicate checks. For example, CDISC-SDTM Janus (checksource ? “Janus”) checks are generally duplicates of WebSDM checks with (occasional) different resultseverity values.
- The _cstDebug option should be toggled off except for debugging specific program errors to avoid exceeding the SAS log size limitations or generating large SAS log files.

- A SAS Clinical Standards Toolkit validation process that involves a large number of checks, or is run with the _cstDebug option toggled on, should be run in batch, or using PROC PRINTTO, to avoid exceeding the SAS log size limitations.

CDISC-CRTDDS Metrics

No performance concerns are anticipated for validation of CDISC-CRTDDS, as validation (of the SAS representation of CRTDDS) is limited to metadata comparisons against only the CDISC-CRTDDS reference standard. Note that validation of the generated define.xml is a very efficient comparison of the file structure to the associated XML schema.

CHAPTER 7

Creating CDISC-CRTDDS Define.xml

CDISC-CRTDDS Workflow

The basic CRTDDS workflow is to:

1. Create the SAS representation of the 39 CDISC-CRTDDS 1.0 data sets (or subset if not all of the data sets are required) from the SDTM 3.1.1 domains for a study.
2. Validate the data sets
3. Create the define.xml
4. Validate the syntax of the define.xml

Step 2 is very similar to the process followed in the SDTM validation. SAS supplies reference_tables and reference_columns data sets that contain all the metadata for the SAS representation of the CRTDDS tables. There is also a validation_master that contains all of the checks listed above in Table XX. If all 39 CRTDDS tables are used in the define.xml, then the user can run directly against the reference tables and columns data sets and the "Use source data" flag in the validation master will need to be set to 'N' when creating the validation_control data set. Most users, however, will most likely run the validations against a subset of the 39 data sets. In this case a source_tables data set containing the subset will need to be created from reference_tables along with a corresponding source_columns created from reference_columns. Validation_master can contain all of the checks and leave "Use source data" = Y (Default value). For efficiency it is recommended when creating the validation_control to subset the validation_master based on the checks required for the data sets being validated.

There are four macros shipped with the Toolkit to facilitate CRTDDS workflow and the creation of the DEFINE.XML. These four macros (in order of execution) are listed below:

- crtdds_sdtm311todefine10.sas – populates 9 of the 39 tables in the SAS representation of the CRTDDS files from the SDTM metadata. These are the minimum required data sets needed to create a valid define.xml from SDTM table and column metadata.
- crtdds_validate.sas – the validation macro that submits the validation checks based on what is defined in the validation_control data set.

- crtdds_write.sas – the macro used to create the define.xml from the SAS representation of the CRTDDS files.
- crtdds_xmlvalidate.sas – the macro that validates that the XML file is syntactically correct. This macro is particularly important if the define.xml has been customized outside of the above work flow by the user. For example, if the user uses an XML editor to add links for annotated CRF pages into the define.xml, this macro will validate the syntax of the edited file.

These macros should be called by "driver" programs that create study-specific metadata that is required by the framework macros. In the corresponding CRTDDS sample study that is shipped with the framework, there are three examples of these driver programs. The driver programs will usually be stored at the study or compound level within a clinical trial. The CRTDDS driver program validate_crtdds_data is very similar to the one defined for SDTM in this document. The driver programs are listed below in the order in which they should be executed.

1. create_crtdds10_from_sdtm311.sas – when executed, sets up the required metadata and sasreferences for the sample study and runs the crtdds_sdtm311todefine10.sas macro listed above. Creates the SAS representation of the CRTDDS define data sets from the sample study SDTM data sets.
2. validate_crtdds_data.sas – creates the required metadata for the sample study and submits the crtdds_validate.sas macro listed above. Validates the SAS representation of the CRTDDS define data sets based on the CRTDDS validation checks in the table listed previously. This program could possibly be run many times until the data validation has been reconciled.
3. create_crtdds_define.sas – creates the define.xml. This driver program runs the crtdds_write and the crtdds_xmlvalidate macros listed above. This program creates and then validates the XML syntax for the define.xml.

The three SAS driver programs listed above are examples that are shipped with the SAS Clinical Standards Toolkit. The customer can choose to use these are create their own. The names of these 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 CDISC-CRTDDS Framework Macros

How to Use crtdds_sdtm311todefine10.sas

This macro extracts data from the SDTM 3.1.1 metadata files (refer to "Source Metadata" in Chapter 4 that describes the source_tables and source_columns data sets) and converts the data into a subset (9) of the 39 tables in the SAS representation of the CDISC-CRTDDS 1.0 model. The following CRTDDS tables are created:

- definedocument
- study
- metadataversion
- itemgroupdefs

- itemdefs
- itemgroupdefitemrefs
- codelists
- codelistitems
- clitemdecodetranslatedtext

The inputs are specified via the SASReferences file along with any required global macro variables as defined. In addition, the framework and CRTDDS initialization properties need to be set. The macro variable _cstResultsDS must point to an existing results data set.

A source study data set is required by this macro before execution and requires the following variables:

Table 7.1 Variables Required in the Source Study Data Set

Variable*	Required	Description
StudyName	Yes	Contains the name of the study.
DefineDocumentName	Yes	Name of the define document being created (that is, define).
SASref	Yes	Reference that ties this study name to the corresponding domains associated with this study in the source_tables and source_columns tables.
ProtocolName	Yes	Contains the name of the protocol for the study
StudyDescription	Yes	Contains a description of the study (note, should not use commas or semicolons or quotes in the description).

*All variables are required to be nonblank.

Multiple studies can be referenced in the source_study data set (as well as source_columns and source_tables), by using different SASref values to link them across the 3 tables.

The following parameters must be set by the user before submitting the macro:

Table 7.2 Parameters for crtdds_sdtm311todefine10.sas

Parameter	Required	Description
_cstOutLib	Yes	Identifies the library reference (libname) where the resulting tables will be created.
_cstSourceTables	Yes	A data set containing the SDTM metadata for the domains to be included in the CRTDDS file.
_cstSourceColumns	Yes	A data set containing the SDTM metadata for the domain columns to be included in the CRTDDS file
_cstSourceStudy	Yes	A data set containing the metadata for the studies to be included in the CRTDDS file

Below is an example of a call to the crtdds_sdtm311todefine10.sas macro.

```
%crtdds_sdtm311todefine10(
```

```
 _cstOutLib=work,
  _cstSourceTables=sampdata.source_tables,
  _cstSourceColumns=sampdata.source_columns,
  _cstSourceStudy=sampdata.source_study
 );
```

In the example above, the crtdds_sdtm311todefine10 macro has set the _cstOutLib to work. All of the CRTDDS defined tables will be written out to the SAS work library. In this example the tables are not being stored as permanent data sets and will be deleted when the SAS session is ended. The _cstSourceTables parameter is accessing the source_tables data set that exists in the sampdata library (sampdata.source_tables). The _cstSourceColumns parameter is accessing the source_columns data set that exists in the sampdata library (sampdata.source_columns). The _cstSourceStudy parameter is accessing the source_study data set that exists in the sampdata library (sampdata.source_study).

How to Use crtdds_validate.sas

This is the macro for the CRTDDS data validation. The main data sets that determine how this macro is executed are the table and column metadata files. The reference_tables and reference_columns contain all of the CRTDDS metadata and are used when validating the entire CDISC-CRTDDS model. Initially the source_tables and the source_columns data sets contain the same content as the reference_tables and reference_columns data sets to validate the entire model. If a subset of the model is run the user can filter the required domains from the reference_tables and reference_columns into the source_tables and source_columns respectively. The validation_control data set contains the same information as the validation_master, and can be filtered from validation_master to run a subset of the validation checks.

In addition, the following settings for **CST input/output data or metadata** are needed in the SASReferences data set (see Figure 7.1 below).

control: References the location of the validation_control data set.

fmtsearch: References the location of the CRTDDS format catalog.

messages: References the location of the CRTDDS and SAS Clinical Standards Toolkit framework messages (notice there are two, each with a different libref associated with them).

referencemetadata: References the location of the reference_tables and reference_columns data sets (used if not accessing sourcemetadata).

results: References the location for the validationresults data set and (optionally) results for the validationmetrics data set (notice there are two, each with a different libref associated with them).

sourcedata: References the location of the CRTDDS data sets.

sourcemetadata: References the location of the source_tables and source_columns data sets (used if not accessing referencemetadata).

There are no parameters for this macro.

Figure 7.1 CDISC-CRTDDS SASReferences Data Set Example

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

How to Use crtdds_write.sas

This macro uses the SAS representation of a CRTDDS file as source data and converts it the required XML structure. The inputs and outputs are specified via the SASReferences file along with any required global macro variables as defined. In addition any framework and CRTDDS initialization properties need to be set as required. The macro variable _cstresultsds must point to an existing results data set or override this value using the _cstResultsOverrideDS parameter to this macro.

Table 7.3 Parameters for crtdds_write.sas

Parameter	Required	Description
_cstCreateDisplayStyleSheet	Optional	Identifies whether the macro should create a style sheet in the same directory as the output XML file. If this is set to 1, the macro will look in the supplied SASreferences file for a record with type/subtype of referencexml/stylesheet and will use that file. If set to 0, the macro will not create the XSL, even if one is specified in the SASReferences file.
_cstOutputEncoding	Optional	The XML encoding to use for the CRTDDS file that is created.
_cstHeaderComment	Optional	A short comment will be added to top of the CRTDDS file that is produced. If none is provided, a default will be used.
_cstResultsOverrideDS	Optional	The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS global macro variable will be used.

Below is an example of a call to the crtdds_sdtm311todefine10.sas macro.

```
%crtdds_write(_cstCreateDisplayStyleSheet=1,_cstResultsOverrideDS=
&_cstResultsDS);
```

In this example only 2 of the parameters are being set by the user, _cstResultsOverrideDS and _cstCreateDisplayStyleSheet. A default style sheet is being generated in the same directory as the XML output based on information found in the

sasreferences data set. The global macro variable _cstResultsDS is setting the value for the _cstResultsOverrideDS, which in this case is unnecessary since the default is the value of _cstResultsDS.

How to Use crtdds_xmlvalidate.sas

This macro runs XML syntax checks to verify the XML statements in the define.xml are correct. It should be the last macro run in the define.xml creation process. If the user customizes the define.xml after its generation in the step above, this macro is useful in determining the validity of the user defined changes. In addition, the following information might need to be provided in the sasreferences data set.

- externalxml - points to the location and name of the crtdds xml file. (Required)
- referencexml - points to the style sheet to use with the define file. (Optional)

Table 7.4 Parameters for crtdds_xmlvalidate.sas

Parameter	Required	Description
_cstLogLevel	Yes	Identifies the log level. Current valid values are Info, Warning, Error, Fatal Error, and None
_cstResultsOverrideDS	Yes	The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS global macro variable will be used.

There are 4 logging levels. These log levels refer to the XML generated log and do not refer to the log generated by SAS.

Table 7.5 Log Levels for crtdds_xmlvalidate.sas

Log Level	Description
Info	Informational messages such as the system properties of the current Java environment, as well as progress messages.
Warning	Messages that indicate that there might be an issue with the CRTDDS document or with the execution of the validation routine.
Error	Messages that indicate that something in the define.xml document is invalid with respect to the normative XML Schema for CRTDDS, or that a non-fatal error has occurred during processing.
Fatal Error	Messages that indicate that the XML document could not be processed at all. This can have any number of causes, including, but not limited to, filesystem access errors, incorrect file paths, and malformed XML.
None	No messages of any kind will be produced

Each error message generated during XML validation is associated with one of these levels. The level chosen will cause all messages of that and all more-severe levels to be generated. For example, if "WARNING" is selected, all "WARNING", "ERROR" and "FATAL ERROR" messages will be generated. As another example, if "ERROR" is selected, only "ERROR" and "FATAL ERROR" messages will be generated.

Below is an example of a call to the crtdds_xmlvalidate.sas macro.

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

In this example, the crtdds_xmlvalidate macro is being submitted with a log level of Info, and the results data set is named XMLVALIDATE and resides in the work library.

The CDISC-CRTDDS Enumeration Validation

The CRTDDS enumeration checks listed in Table 6.13 are performed by comparing the data against a set of predetermined values that have been stored in a SAS format catalog. These are case sensitive checks. The format catalog and a SAS data set are created using the CRTDDS createstandarddatasets.sas program that is shipped with the product. The format catalog is named crtddsct.sas7bcat and the data set is crtddsct.sas7bdat. The table below lists the format names and values that are used when submitting the CRTDDS validation programs. This methodology provides a mechanism to ensure case-sensitivity compliance required by the XML schema validation. For example, the ItemRangeChecks data set requires an enumeration edit for values such as 'LT' and 'LE'. If mixed or lower-case values are detected, the validation check (in this case, CRT0114, see Table 6.13) will use the Comp format as described in the table below to report this as an error.

Table 7.6 Enumeration Validation Format Values*

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

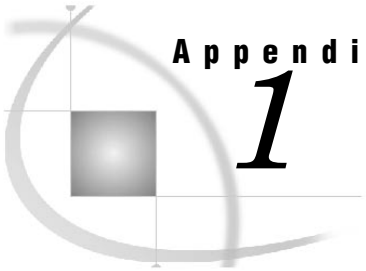
Format	Value 1	Value 2
	NE	NE
	IN	IN
	NOTIN	NOTIN
Soft	Soft	Soft
	Hard	Hard
CLType	integer	integer
	float	float
	text	text

*Value 1 and Value 2 are case sensitive

The sasreferences file will need to contain a row for fmtsearch with SAS libref set to crtfmt and filename refers to crtddsct.sas7bcat. See the example below.

Figure 7.2 Example sasreferences File

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



Appendix 1

Global Macro Variables

Overview

The following is a description of the global macro variables used by SAS Clinical Standards Toolkit (CST). Most CST global macro variables supplied by SAS are defined in property files in the form of name/value pairs, such as:

_cstDebug=

Each registered standard, including the CST-Framework, has an initialize.properties file. This file specifies those global macro variables required by that standard and available for use in any CST processes referencing that standard. Each registered standard might also have an action-related properties file (for example, validation.properties) that specifies global macro variables needed for processes performing that specific action.

Properties files are generally processed in one of two ways:

With a direct call to the CST utility macro %cstutil_setproperties within a code module, such as a driver-program like validate_data.sas

By inclusion in the sasreferences data set (with type=properties), where the %cstutil_allocatesasreferences macro calls %cstutil_setproperties

Global macro variables can be deleted at the end of a process if the CST utility macro %cstutil_cleanupcstsession is called with the _cstDeleteGlobalMacroVars parameter set to 1.

Two commonly used global macro variables are not defined in the property files described above. _cstGRoot defines the location of the _cstGlobalLibrary, and is set with the autocall macro call %cstutil_setcstgroot, which is called in most framework macros. The studyRootPath macro variable defines the location of the study data and metadata, and is often set in user-defined driver programs (for example, validate_data.sas).

Table 1.1 Global Macro Variables

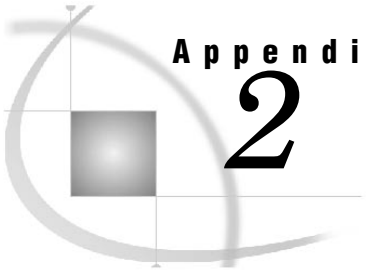
Global Macro Variable	Values	Comments
	CST-Framework initialize.properties	
_cstDebug	0 (off)** 1 (on)	If on, _cstDebugOptions are set, and many files remain in the work library at process conclusion
_cstDebugOptions	mprint mlogic symbolgen mautolocdisplay**	SAS system options set when _cstDebug=1
_cst_rc	0 (no error)** 1 (error)	Set to 1 during processing if an error is encountered that should halt the process
_cst_MsgID	<blank>**	The result or validation check ID used for reporting process results; value set in each code module
_cst_MsgParm1	<blank>**	Any result message parameter (1) used for reporting process results; value set in each code module
_cst_MsgParm2	<blank>**	Any result message parameter (2) used for reporting process results; value set in each code module
_cstResultSeq	0**	Sequence indicator, used most often to signal multiple instances of the same event (such as running the same validation check multiple times). Should be initialized to 0. Used for reporting process results; values incremented in each code module. Used in the joining of results and metrics data sets.
_cstSeqCnt	0**	Sequence indicator to count the number of records output to the results data set within _cstResultSeq. Should be initialized to 0. Used for reporting process results; values incremented in each code module
_cstResultsDS	work._cstresults **	The default data set name used to accumulate results during a process. May be persisted at the end of the process based on the sasreferences (type=results) entry
_cstSrcData	<blank>**	Used for reporting process results; value set in each code module
_cstResultFlag	0** -1 1	Reports the status of any given result. 0 indicates an informational or non-error status. A positive integer indicates an error status. A negative integer indicates the assessment could not be completed, often due to metadata problems or the occurrence of SAS errors.
_cstReallocateSASRefs	0** (no) 1 (yes)	Should CST attempt to reallocate any SAS librefs and filerefs if they are already allocated? If yes, allocation is based on sasreferences content.
_cstFMTLibraries	<blank>** ** work.fmt	Allows users to alter the format search path built from sasreferences (type=fmtsearch) entries with <libref> or <libref.catalog> references. If <libref> only is provided, SAS assumes a catalog name of FORMATS. If value begins with ** (such as ** WORK), CST moves WORK.FORMATS to the

Global Macro Variable	Values	Comments
		end of the format search path.
_cstMessageOrder	APPEND** MERGE	Used in the derivation of _cstMessages. APPEND appends message files based on the order of sasreferences (type=messages) entries. MERGE allows reference to multiple standard-specific message files (including internationalized messages), retaining a single message per message ID, standardversion and checksource.
_cstMessages	work._cstmessages**	The default data set name used to aggregate all standard messages based on sasreferences (type=messages) entries. This file is used during processing to fully resolve the results.message field.
CDISC-SDTM initialize.properties		
_cstSASRefsLoc	&studyrootpath/control**	The path to a directory containing the sasreferences data specified in _cstSASRefsName. The &studyRootPath is assumed to be user-specified within a calling driver program. Use of &studyRootPath is no required.
_cstSASRefsName	sasreferences**	The name of the sasreferences data set (in _cstSASRefsLoc) to be used as the initial source of information about all inputs and outputs defined for a given CST process. The name of the data set that is a SASReferences data set. This allows more than one SASReferences data set to be stored in a directory.
_cstSASRefs	work._cstsasrefs**	Sasreferences data set used during processing that contains fully resolved records (for example, paths) based on look-through to standard-level sasreferences data sets for default values.
_cstSubjectColumns	studyid usubjid**	The standard-specific set of columns that identify a subject, used by standard-specific macros and for metrics calculations. These need not be present in all source tables (for example, non patient-level domains like CDISC trial design domains).
_cstTableMetadata	work._csttablemetadata**	Data set used during processing that contains table-level metadata (derived from either the reference or study table metadata) used by the process.
_cstColumnMetadata	work._cstcolumnmetadata**	Data set used during processing that contains column-level metadata (derived from either the reference or study column metadata) used by the process.
CDISC-SDTM validation.properties		
_cstCheckSortOrder	_DATA_** <keys>	Allows specification of the order checks that are to-be-run. _DATA_ indicates that checks are to be processed in the order as defined in the validation_control data set. Users can optionally specify a set of space-delimited keys from

Global Macro Variable	Values	Comments
		validation_control columns (for example, checksource checkid).
_cstMetrics	0 (off)** 1 (on)	Toggle to enable/disable metrics reporting; attempts to provide a denominator for errors detected. Increased processing time can result.
_cstMetricsDS	work._cstmetrics**	The default data set name used to accumulate results during a process. Typically persisted at the end of the process based on the sasreferences (type=results) entry
_cstMetricsTimer	0 (off) 1 (on)**	Estimate the elapsed time to perform some action. Results are added to the _cstMetricsDS. Value ignored if _cstMetrics=0
_cstMetricsNumSubj	0 (off) 1 (on)**	Enable counts on a subject level. Value ignored if _cstMetrics=0
_cstMetricsNumRecs	0 (off) 1 (on)**	Enable counts on # of records tested. Value ignored if _cstMetrics=0
_cstMetricsNumChecks	0 (off) 1 (on)**	Report # of distinct validation check invocations. Value ignored if _cstMetrics=0
_cstMetricsNumErrors	0 (off) 1 (on)**	Report # of resultseverity=”Error” records in results data set. Value ignored if _cstMetrics=0
_cstMetricsNumWarnings	0 (off) 1 (on)**	Report # of resultseverity=”Warning” records in results data set. Value ignored if _cstMetrics=0
_cstMetricsNumNotes	0 (off) 1 (on)**	Report # of resultseverity=”Note” records in results data set. Value ignored if _cstMetrics=0
_cstMetricsNumStructural	0 (off) 1 (on)**	Report # of structural errors detected. Based on errors reported for checks where checktype= ”Metadata”. Excludes informational records in results data set. Value ignored if _cstMetrics=0
_cstMetricsNumContent	0 (off) 1 (on)**	Report # of content errors detected. Based on errors reported for checks where checktype ^= ”Metadata”. Excludes informational records in results data set. Value ignored if _cstMetrics=0
_cstMetricsCntNumSubj	0**	Actual count of # subjects tested. Value not calculated if _cstMetrics=0
_cstMetricsCntNumRecs	0**	Actual count of # records tested. Value not calculated if _cstMetrics=0
_cstMetricsCntNumChecks	0**	Actual count of # validation checks run. Value not calculated if _cstMetrics=0
_cstMetricsCntNumErrors	0**	Actual count of # Errors reported. Value not calculated if _cstMetrics=0
_cstMetricsCntNumWarnings	0**	Actual count of # Warnings reported. Value not calculated if _cstMetrics=0
_cstMetricsCntNumNotes	0**	Actual count of # Notes reported. Value not calculated if _cstMetrics=0
_cstMetricsCntNumStructural	0**	Actual count of # Structural errors reported. Value not calculated if _cstMetrics=0
_cstMetricsCntNumContent	0**	Actual count of # Content errors reported. Value

Global Macro Variable	Values	Comments
		not calculated if _cstMetrics=0
	General Purpose (not set in properties files)	
_cstGRoot	Example: C:\cstGlobalLibrary	Required. Defines the location of the _cstGlobalLibrary. Set with the autocall macro call %cstutil_setcstgroot, which is called in most framework macros. Used most often in sasreferences paths to enable relative path mobility.
studyRootPath	Example: C:\Study1	Optional. Defines the location of study data and metadata. Often set in user-defined driver programs (for example, validate_data.sas). Used most often in sasreferences paths to limit changes required when changing input data sources.

**default value



Appendix 2

Framework Messages

ResultIDs and Associated Message Text

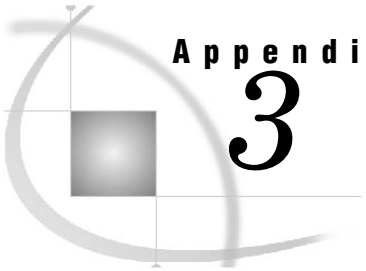
Table 2.1 Result IDs and Associated Message Text

ResultID	Checkseverity	Messagetext
CST0001	Error	Fatal error encountered, process cannot continue
CST0002	Warning: Check not run	No tables evaluated-check validation control data set
CST0003	Warning: Check not run	&_cstparm1 could not be found
CST0004	Warning: Check not run	No columns evaluated - check validation_control specification
CST0005	Error	Input parameters to macro insufficient for &_cstparm1 macro to run
CST0006	Warning: Check not run	Lookup to SASreferences control data set failed
CST0007	Error	SASreferences lookup returned no records
CST0008	Error	&_cstparm1 could not be found
CST0010	Error	SASreferences lookup returned multiple records
CST0012	Error	SASreferences lookup returned inconsistent SASref and memname values
CST0014	Warning: Check not run	Global macro variable &_cstparm1 cannot be null
CST0015	Warning: Check not run	Invalid &_cstparm1 input parameter, &_cstparm2 macro cannot run
CST0016	Warning: Check not run	&_cstparm1 could not be found
CST0020	Info	Check run but non-missing codeLogic is not used
CST0021	Warning: Check not run	Table &_cstparm1 does not contain &_cstparm2 column
CST0022	Warning: Check not run	&_cstparm1 keys could not be found
CST0023	Warning: Check not run	Validation control parsing of &_cstparm1 results in inconsistent sublist lengths
CST0026	Warning: Check not run	One or more check metadata column values is invalid - &_cstparm1
CST0027	Warning: Check not run	Global macro variable &_cstparm1 could not be found or contains an invalid value

ResultID	Checkseverity	Messagetext
CST0028	Warning: Check not run	Format search path has not been set
CST0029	Info	Format catalog &_cstparm1 in fmtsearch could not be found
CST0030	Warning: Check not run	No catalogs in fmtsearch could be found
CST0031	Warning: Check not run	Reference terminology &_cstparm1 not found
CST0032	Info	Reference terminology data set &_cstparm1 was set to &_cstparm2
CST0033	Info	Format search path has been set to &_cstparm1
CST0034	Warning: Check not run	&_cstParm1 has no observations for &_cstParm2
CST0050	Warning: Check not run	Code failed due to SAS error - &_cstparm1
CST0051	Error	Code failed due to SAS error - &_cstparm1
CST0070	Error	Selected target directory, &_cstparm1, does not exist. Please create the Target Directory.
CST0071	Error	The Standards directory, &_cstparm1, is not available.
CST0072	Error	Unable to create directory, &_cstparm1.
CST0073	Info	Study directories created in &_cstparm1.
CST0074	Info	Study reference data created in &_cstparm1.
CST0075	Error	Unable to allocate &_cstparm1 for &_cstparm2.
CST0076	Info	SAS &_cstparm1 from SASref=&_cstparm2 sasreferences record not allocated
CST0080	Info	SASreferences for &_cstParm1 were copied to &_cstParm2
CST0081	Error	A required parameter was not supplied &_cstParm1
CST0082	Error	The standard &_cstParm1 is not registered
CST0083	Error	The version &_cstParm1 does not exist for &_cstParm2
CST0084	Error	The SASReferences type &_cstParm1 is not defined for &_cstParm2
CST0085	Error	No version was supplied and there is no default for the &_cstParm1 standard
CST0086	Error	The SASReferences type &_cstParm1 has more than one subtype and none was specified
CST0087	Error	The type/subtype &_cstParm1 is not defined for &_cstParm2
CST0088	Error	The following columns of &_cstParm1 cannot be empty: &_cstParm2
CST0089	Error	Only libraries are supported for this operation
CST0090	Error	There were problems with the sasreferences data set
CST0099	Warning: Check not run	&_cstparm1 is not supported in the current release of CST
CST0100	Info	No errors detected in &_cstparm1
CST0101	Error	The libref &_cstparm1 must be assigned prior to calling the macro
CST0102	Info	&_cstparm1 was created as requested
CST0103	Error	A SASReferences file must be passed as a parameter or specified using the CST global environment variable
CST0104	Error	Unable to acquire exclusive locks on the global metadata data sets
CST0106	Error	The standard &_cstParm1 does not have a properties file registered for &_cstParm2
CST0107	Error	Invalid location type &_cstParm1
CST0108	Info	The properties were processed from the &_cstParm1 &_cstParm2
CST0109	Info	The default version for &_cstParm1 has been set to &_cstParm2

ResultID	Checkseverity	Messagetext
CST0110	Info	&_cstParm1 is no longer registered as a standard
CST0111	Error	Unable to open data set &_cstParm1
CST0112	Error	Data set &_cstParm1 has no observations
CST0114	Error	No lookup table found in registered standards data set where standard=&_cstParm1 and version=&_cstParm2
CST0115	Error	Null values are not permitted for column &_cstParm2 in dataset &_cstParm1
CST0116	Error	Invalid value for column &_cstParm1 in dataset &_cstParm2
CST0117	Error	No template dataset found for type=&_cstParm1, subtype=&_cstParm2 in the registered data standards
CST0118	Error	The standard &_cstParm1 is not a data standard
CST0119	Error	The standard &_cstParm1 is missing referencemetadata for &_cstParm2
CST0120	Error	Could not continue due to errors encountered in assigning libraries
CST0121	Error	Errors were encountered creating the &_cstParm1 tables - check the log
CST0122	Info	The tables were created for &_cstParm1 in library &_cstParm2
CST0123	Warning	The lookup table has no entries for standard=&_cstParm1 and version=&_cstParm2
CST0124	Error	The default version &_cstParm1 for &_cstParm2 cannot be unregistered while other versions exist
CST0125	Error	Differences found between data set &_cstParm1 and the template dataset &_cstParm2
CST0200	Info	&_cstParm1

Note: Not all fields are displayed.



Appendix 3

CDISC-SDTM SAS Representation

Sample reference_tables Record

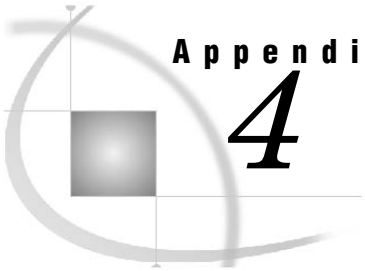
Table 3.1 Sample reference_tables Record

Column Name	Column Value
SASref	REFDATA
Table	AE
Label	Adverse Events
Class	Events
XmlPath	..\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	August 26, 2005
Standard	CDISC-SDTM
StandardVersion	3.1.1
Standardref	SDTM2.2.4
Comment	

Sample reference_columns Record

Table 3.2 Sample reference_columns Record

Column Name	Column Value
SASref	REFDATA
table	SU
column	SUSTRF
label	Start Relative to Reference Period
order	32
type	C
length	20
displayformat	
xmldatatype	text
xmlcodelist	STENRF
core	Perm
origin	Derived
role	Timing
term	** BEFORE, DURING, AFTER
algorithm	
qualifiers	UPPERCASE
standard	CDISC-SDTM
standardversion	3.1.1
standardref	SDTMIG4.1.4.7
comment	Identifies the start of the substance use period with respect to the sponsor-defined reference period. Sponsors should define the reference period in the study metadata. SUSTRF should be populated when a start date is not collected. If information such as "PRIOR", "ONGOING", or "CONTINUING" was collected, this information should be translated into SUSTRF.



Appendix 4

Macro Application Programming Interface

Module CRT-DDS V1.0 (Runtime)

Overview

This is the CDISC-CRTDDS 1.0 runtime macro library.

Table 4.1 Module CRT-DDS V1.0 (Runtime) Macro Summary

Exposure	Macro
	%crtdds_clitemdecodetrans(_cstsourcestudy=, _cstsourcecolumns=, _cstcodelistitemsds=, _cstmdvDS=, _cststudyds=, _cstcodelistsds=, _cstCLlang=, _cstoutclitemdecodetransds=);
	%crtdds_codelistitems(_cstsourcecolumns=, _cstcodelistsds=, _cstoutcodelistitemsds=);
	%crtdds_codelists(_cstsourcecolumns=, _cstmdvds=, _cstmdvname=, _cstoutcodelistsds=);
	%crtdds_definedocument(_cstname=, _cstdescr=, _cstoutdefinedocds=);
	%crtdds_getStatic(_cstName=, _cstVar=);
	%crtdds_itemdefs(_cstsourcecolumns=, _cstsourcestudy=, _cststudyds=, _cstmdvds=,

Exposure	Macro
	_cstcodelistsDS=, _cstoutitemdefsds=, _cstoutitemdefsds2=);
	%crtdds_itemgroupdefitemrefs(_cstsourcecolumns=, _cstsourcetables=, _cstsourcestudy=, _cstitemdefsds2=, _cstmdvds=, _cstitemgroupdefsds=, _cststudyds=, _cstoutitemgroupdefitemrefsds=);
	%crtdds_itemgroupdefs(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupdefsds=);
	%crtdds_metadataversion(_cstname=, _cstdescr=, _cststandard=, _cstversion=, _cstdefineversion=, _cststudyds=, _cststudyname=, _cstoutmdvds=);
External CRT-DDS	%crtdds_sdtm311todefine10(_cstOutLib=, _cstSourceTables=, _cstSourceColumns=, _cstSourceStudy=);
	%crtdds_study(_cstname=, _cstdescr=, _cstprotocol=, _cstdefineds=, _cstdefinename=, _cstoutstudyds=);
External CRTDDS Validation Process	%crtdds_validate /des='CST: Validate CDISC CRTDDS model files';
External CRT-DDS	%crtdds_write(_cstCreateDisplayStyleSheet=1, _cstOutputEncoding=, _cstHeaderComment=, _cstResultsOverrideDS=);
External CRT-DDS	%crtdds_xmlvalidate(_cstLogLevel=info, _cstResultsOverrideDS=);

Macro Detail

%crtdds_clitemdecodetrans

%crtdds_clitemdecodetrans(_cstsourcestudy=, _cstsourcecolumns=, _cstcodelistitemsds=, _cstmdvDS=, _cststudyds=, _cstcodelistsds=, _cstCLlang=, _cstoutclitemdecodetransds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcestudy - Parameter description not provided
- _cstsourcecolumns - Parameter description not provided
- _cstcodelistitemsds - Parameter description not provided
- _cstmdvDS - Parameter description not provided
- _cststudyds - Parameter description not provided
- _cstcodelistsds - Parameter description not provided
- _cstCLlang - Parameter description not provided
- _cstoutclitemdecodetransds - Parameter description not provided

File: crtdds_clitemdecodetrans.sas

%crtdds_codelistitems

%crtdds_codelistitems(_cstsourcecolumns=, _cstcodelistsds=, _cstoutcodelistitemsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns - Parameter description not provided
- _cstcodelistsds - Parameter description not provided
- _cstoutcodelistitemsds - Parameter description not provided

File: crtdds_codelistitems.sas

%crtdds_codelists

%crtdds_codelists(_cstsourcecolumns=, _cstmdvds=, _cstmdvname=, _cstoutcodelistsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns - Parameter description not provided
- _cstmdvds - Parameter description not provided
- _cstmdvname - Parameter description not provided
- _cstoutcodelistsds - Parameter description not provided

File: crtdds_codelists.sas

%crtdds_definedocument

%crtdds_definedocument(_cstname=, _cstdescr=, _cstoutdefinedocds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstname - Parameter description not provided
- _cstdescr - Parameter description not provided
- _cstoutdefinedocds - Parameter description not provided

File: crtdds_definedocument.sas

%crtdds_getStatic(

%crtdds_getStatic(_cstName=, _cstVar=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstName - Parameter description not provided
- _cstVar - Parameter description not provided

File: crtdds_getstatic.sas

%crtdds_itemdefs

%crtdds_itemdefs(_cstsourcecolumns=, _cstsourcestudy=, _cststudyds=, _cstmdvds=, _cstcodelistsDS=, _cstoutitemdefsds=, _cstoutitemdefsds2=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns - Parameter description not provided
- _cstsourcestudy - Parameter description not provided
- _cststudyds - Parameter description not provided
- _cstmdvds - Parameter description not provided
- _cstcodelistsDS - Parameter description not provided
- _cstoutitemdefsds - Parameter description not provided
- _cstoutitemdefsds2 - Parameter description not provided

File: crtdds_itemdefs.sas

%crtdds_itemgroupdefitemrefs

%crtdds_itemgroupdefitemrefs(_cstsourcecolumns=, _cstsourcetables=, _cstsourcestudy=, _cstitemdefsds2=, _cstmdvds=, _cstitemgroupdefsds=, _cststudyds=, _cstoutitemgroupdefitemrefsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcecolumns - Parameter description not provided
- _cstsourcetables - Parameter description not provided
- _cstsourcestudy - Parameter description not provided
- _cstitemdefsds2 - Parameter description not provided
- _cstmdvds - Parameter description not provided
- _cstitemgroupdefsds - Parameter description not provided
- _cststudyds - Parameter description not provided

_cstoutitemgroupdefitemrefsds - Parameter description not provided

File: crtdds_itemgroupdefitemrefs.sas

%crtdds_itemgroupdefs

%crtdds_itemgroupdefs(_cstsourcetables=, _cstsourcestudy=, _cststudyds=, _cstmdvDS=, _cstoutitemgroupdefsds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstsourcetables - Parameter description not provided
- _cstsourcestudy - Parameter description not provided
- _cststudyds - Parameter description not provided

- _cstmdvDS - Parameter description not provided
- _cstoutitemgroupdefsds - Parameter description not provided

File: crtdds_itemgroupdefs.sas

%crtdds_metadataversion

%crtdds_metadataversion(_cstname=, _cstdescr=, _cststandard=, _cstversion=, _cstdefineversion=, _cststudyds=, _cststudyname=, _cstoutmdvds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstname - Parameter description not provided
- _cstdescr - Parameter description not provided
- _cststandard - Parameter description not provided
- _cstversion - Parameter description not provided
- _cstdefineversion - Parameter description not provided
- _cststudyds - Parameter description not provided
- _cststudyname - Parameter description not provided
- _cstoutmdvds - Parameter description not provided

File: crtdds_metadataversion.sas

%crtdds_sdtm311todefine10

%crtdds_sdtm311todefine10(_cstOutLib=, _cstSourceTables=, _cstSourceColumns=, _cstSourceStudy=);

[Exposure: external] [Macro Type: CRT-DDS]

files from SDTM metadata

This macro extract data from the SDTM metadata files and converts of the CRT-DDS model. The following CRT-DDS tables are created:

- definedocument, study
- metadataversion
- itemgroupdefs
- itemdefs
- itemgroupdefitemrefs
- codelists
- codelistitems
- clitemdecodetranslatedtext

The inputs are specified via a SASReferences file.

Required Global Macro Variables:

- The framework initialization properties
- The CRT-DDS 1.0 initialization properties

- _cstresultsds should point to an existing results dataset

Parameters:

- _cstOutLib - Required. Identifies libref where the resulting tables should be written to.
- _cstSourceTables - Required. A dataset containing the SDTM
- _cstSourceColumns - Required. A dataset containing the SDTM
- _cstSourceStudy - Required. A dataset containing the

File: crtdds_sdtm311todefine10.sas

%crtdds_study

%crtdds_study(_cstname=, _cstdescr=, _cstprotocol=, _cstdefineds=, _cstdefinename=, _cstoutstudyds=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstname - Parameter description not provided
- _cstdescr - Parameter description not provided
- _cstprotocol - Parameter description not provided
- _cstdefineds - Parameter description not provided
- _cstdefinename - Parameter description not provided
- _cstoutstudyds - Parameter description not provided

File: crtdds_study.sas

%crtdds_validate

%crtdds_validate /des='CST: Validate CDISC CRTDDS model files';

[Exposure: external] [Macro Type: CRTDDS Validation Process]

crtdds_validate

Validate CDISC CRTDDS model files

The basic function of this code module is to cycle through the validation checks to-be-run, writing validation results to the process results and (optionally) metrics data sets. These are then persisted to any permanent location based on type=results records in sasreferences. Process cleanup is based on the _cstDebug global macro variable.

Required Global Macro Variables (beyond reporting and debugging variables):

(none)

Required File Inputs:

run-time (type=control,subtype=validation in sasreferences) check data set

File: crtdds_validate.sas

%crtdds_write

%crtdds_write(_cstCreateDisplayStyleSheet=1, _cstOutputEncoding=, _cstHeaderComment=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: CRT-DDS]

Writes a CDISC-CRTDDS V1.0 XML file.

This macro uses the SAS representation of a CRT-DDS file as source outputs are specified via a SASReferences file.

Required Global Macro Variables:

- The framework initialization properties
- The CRT-DDS 1.0 initialization properties _cstresultsds should point to an existing results dataset, or overide this value using the _cstResultsOverrideDS parm to this macro

Parameters:

- _cstCreateDisplayStyleSheet - Optional. Identifies whether the macro should create a stylesheet in the same directory as the output XML file. If this is set to 1, the macro will look in the supplied SASreferences file for a record with type/subtype of referencexml/stylesheet and will use that file. If set to 0, the macro will not create the XSL, even if one is specified in the SASReferences file.
- _cstOutputEncoding - Optional. The XML encoding to use for
- _cstHeaderComment - Optional. A short comment will be added
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: crtdds_write.sas

%crtdds_xmlvalidate

%crtdds_xmlvalidate(_cstLogLevel=info, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: CRT-DDS]

Writes a CDISC-CRTDDS V1.0 XML file.

This macro uses the SAS representation of a CRT-DDS file as source outputs are specified via a SASReferences file.

Required Global Macro Variables:

- The framework initialization properties
- The CRT-DDS 1.0 initialization properties _cstSASRefs _cstresultsds should point to an existing results dataset, or overide this value using the _cstResultsOverrideDS parm to this macro

Parameters:

- _cstLogLevel - Parameter description not provided
- _cstResultsOverrideDS - Parameter description not provided

File: crtdds_xmlvalidate.sas

Module Framework

Overview

This is the framework description. It will describe what the framework does and how it fits together.

Since: 1.2

See: CST Framework Global Macro Variables

Table 4.2 Module Framework macro Summary

Exposure	Macro
External framework	%cst_createDS(_cstStandard=, _cstStandardVersion=, _cstType=, _cstSubType=, _cstOutputDS=, _cstResultsOverrideDS=);
External standard_name	%cst_createEmptyTables;
External study creation	%cst_createStudyFromStandard(_cstModel=, _cstVersion=, _cstStudyRootPath=);
External framework	%cst_createTablesForDataStandard(_cstStandard=, _cstStandardVersion=, _cstOutputLibrary=, _cstResultsOverrideDS=);
External framework	%cst_deleteProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);
	%cst_getRegisteredStandards(_cstOutputDS=, _cstResultsDS=);
External standard_name	%cst_getStandardMetadata(_cstSASReferences=, _cstResultsOverrideDS=);
External Framework	%cst_getStandardSASReferences(_cstStandard=, _cstStandardVersion=, _cstOutputDS=, _cstResultsOverrideDS=);
	%cst_getStatic(_cstName=, _cstVar=);
external	%cst_insertStandardSASRefs(_cstSASReferences=, _cstOutputDS=, _cstAddRequiredCSTRefs=0, _cstResultsOverrideDS=);
	%cst_registerStandard(_cstRootPath=, _cstControlSubPath=, _cstStdDSName=, _cstStdSASRefsDSName=, _cstOutputDS=);
External framework	%cst_setProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=); Reads a properties file/data set and sets global macros accordingly. ...
External framework	%cst_setStandardProperties(_cstStandard=, _cstStandardVersion=, _cstSubType=, _cstResultsOverrideDS=);
	%cst_setStandardVersionDefault(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);
	%cst_unregisterStandard(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);
External	%cst_unsetProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);

Exposure	Macro
framework	
External	%cstcheck_column(_cstControl=);
Validation Check	
External	%cstcheck_columncompare(_cstControl=);
Validation Check	
External	%cstcheck_comparedomains(_cstControl=);
Validation Check	
External	%cstcheck_dsmismatch(_cstControl=);
Validation Check	
External	%cstcheck_metamismatch(_cstControl=);
Validation Check	
External	%cstcheck_notconsistent(_cstControl=);
Validation Check	
External	%cstcheck_notincodelist(_cstControl=);
Validation Check	
External	%cstcheck_notsorted(_cstControl=);
Validation check	
External	%cstcheck_notunique(_cstControl=);
Validation Check	
External	%cstcheck_recmismatch(_cstControl=);
Validation Check	
External	%cstcheck_recnotfound(_cstControl=);
Validation Check	
External	%cstcheck_violatesstd(_cstControl=);
Validation Check	
External	%cstcheck_zeroobs(_cstControl=);
Validation Check	
Internal	%cstutil_allocatesasreferences / des='CST: Allocate sasreferences';
Framework utility	
	%cstutil_allocGlobalMetadataLib(_cstLibname=);
Internal	%cstutil_appendresultds(_cstErrorDS=, _cstVersion=&_cstStandardVersion,

Exposure	Macro
Framework utility	_cstSource=&_cstCheckSource, _cstStdRef=);
Internal Framework utility	%cstutil_buildcollist(_cstFormatType=DATASET, _cstColWhere=, _cstDomWhere=, _cstColDSName=&_cstColumnMetadata, _cstDomDSName=&_cstTableMetadata, _cstColSubOverride=N, _cstDomSubOverride=N);
Internal Framework utility	%cstutil_builddomlist(_cstFormatType=DATASET, _cstDomWhere=, _cstDomDSName=&_cstTableMetadata, _cstSubOverride=N);
Internal framework check	%cstutil_checkds(_cstdsname=, _csttype=, _cstsubtype=, _cststandard=*, _cststandardversion=*);
	%chkvals;
Internal Framework utility	%cstutil_cleanupcstsession(_cstClearCompiledMacros=0, _cstClearLibRefs=0, _cstResetSASAutos=0, _cstResetFmtSearch=0, _cstResetSASOptions=1, _cstDeleteFiles=1, _cstDeleteGlobalMacroVars=0);
Internal Framework	%cstutil_createTempMessages(_cstCreationFlag=);
Internal standard_name	%cstutil_deleteDataSet(_cstDataSetName=);
	%cstutil_getRandomNumber(_cstVarname=);
Internal Framework utility	%cstutil_getsasreference(_cstStandard=, _cstStandardVersion=, _cstSASRefType=, _cstSASRefSubtype=, _cstSASRefsasref=, _cstSASRefmember=, _cstConcatenate=0, _cstFullname=0);
Internal Framework utility	%cstutil_getsubjectcount(_cstDS=, _cstsubid=&_cstSubjectColumns);
	%cstutil_internalmanageresults(_cstAction=);
Internal Framework utility	%cstutil_messagesdsattr /des='CST: Messages data set column attributes';
Internal Framework utility	%cstutil_metricsdsattr /des='CST: Metrics data set column attributes';
Internal Framework utility	%cstutil_parsecolumnscope(_cstscopestr=, _cstopsource=, _cstsublistnum=);
Internal Framework utility	%cstutil_parsescopesegment(_cstPart=, _cstVarName=, _cstMessageID=CST0004);
Internal Framework utility	%cstutil_parsetablescope(_cstscopestr=, _cstopsource=, _cstsublistnum=);
Internal Framework utility	%cstutil_readcontrol /des="CST: Create control file macro variables";

Exposure	Macro
Internal Framework utility	%cstutil_resultsdsattr /des='CST: Results data set column attributes';
Internal Framework utility	%cstutil_resultsdskeep /des='CST: Results data set columns';
	%cstutil_setcstgroot;
Internal Framework utility	%cstutil_setmodel /des="Set Which Model Definition to Use";
	%cstutil_writecubexml(_cstXMLOut=, _cstMDPFile=, _cstDebug=);
Internal Framework utility	%cstutil_writemetric(_cstMetricParameter=, _cstResultID=, _cstResultSeqParm=, _cstMetricCnt=, _cstSrcDataParm=, _cstMetricsDSParm=&_cstMetricsDS);
Internal Framework utility	%cstutil_writeresult(_cstResultID=, _cstValCheckID=, _cstResultParm1=, _cstResultParm2=, _cstResultSeqParm=1, _cstSeqNoParm=1, _cstSrcDataParm=, _cstResultFlagParm=0, _cstRCParm=0, _cstActualParm=, _cstKeyValuesParm=, _cstResultDetails=, _cstResultsDSParm=&_cstResultsDS);

Macro Detail

%cst_createDS

%cst_createDS(_cstStandard=, _cstStandardVersion=, _cstType=, _cstSubType=, _cstOutputDS=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Creates a zero observation data set based on those provided by a registerd standard.

Parameters:

- _cstStandard - Required. The name of a registered standard.
- _cstStandardVersion - Optional. The version of the standard that the data set should be created from. If this is ommitted, then the default version for the given standard will be used. If a default version is not defined, then an error will be generated.
- _cstType - Required. The type of data set to be created. This value comes from the TYPE column in the sasreferences for the standard-version combination.
- _cstSubType - Optional. Specifies the subtype for the type. This value comes from the SUBTYPE column in the sasreferences for the standard-version combination. If the type has no subtypes
- _cstOutputDS - Required. The name of the data set to be created.
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_createds.sas

%cst_createEmptyTables

%cst_createEmptyTables;

[Exposure: external] [Macro Type: standard_name]

Create empty table shells using reference metadata.

Full, multi-line explanation

Required Global Macro Variables:

- _cstVar1
- _cstVar2

Deprecated. explanation

File: cst_createemptytables.sas

%cst_createStudyFromStandard

%cst_createStudyFromStandard(_cstModel=, _cstVersion=, _cstStudyRootPath=);

[Exposure: external] [Macro Type: study creation]

cst_createStudyFromStandard

Creates a study from selected Model and Version

Required Global Macro Variables: (none)

Required File Inputs: (none)

Parameters:

- _cstModel - The name of the data model to use for this study
- _cstVersion - The version of the data model to use for this study
- _cstStudyRootPath - Parameter description not provided

File: cst_createStudyFromStandard.sas

See: This document

%cst_createTablesForDataStandard

%cst_createTablesForDataStandard(_cstStandard=, _cstStandardVersion=, _cstOutputLibrary=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Creates tables from registered referencemetadata.

Where a standard is registred and is a data standard (with referenceMetadata for tables and columns), this macro will generate all the table shells defined for that standard in a library specified by the caller.

Required Global Macro Variables: CST-FRAMEWORK standard variables

Parameters:

- _cstStandard - Required. The name of a registered standard.
- _cstStandardVersion - Optional. The version of the standard that the data set should be created from. If this is ommitted, then the default version for the given standard will be used. If a default version is not defined, then an error will be generated.
- _cstOutputLibrary - Required. Specifies the libname that the table shells should be created in.
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_createtablesfordatastandard.sas

%cst_deleteProperties

%cst_deleteProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Reads a properties file/data set and unsets global macros accordingly. Property files should have the format name=value. Property data sets should have a character field for name and value. It may also have a comment field but this will be ignored.

Parameters:

- _cstPropertiesLocation - Required. The location of the property file. The format depends upon the value of _cstLocationType.
- _cstLocationType - Required. Identifies the format for the value of _cstPropertiesLocation. Valid Values are: PATH: The path to a properties file. FILENAME: A valid, assigned SAS filename to the properties file. DATA: A (libname.)membername of a SAS data set containing the properties.
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_deleteproperties.sas

%cst_getRegisteredStandards

%cst_getRegisteredStandards(_cstOutputDS=, _cstResultsDS=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstOutputDS - Parameter description not provided
- _cstResultsDS - Parameter description not provided

File: cst_getregisteredstandards.sas

%cst_getStandardMetadata

%cst_getStandardMetadata(_cstSASReferences=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: standard_name]

Retrieves the standard metadata for standards.

A valid SASReferences data set is passed into the macro. It should *; contain records which point to the metadata for data standard. A row should be present for each metadata table that is to be returned. The row should identify the standard, standardVersion, type, subType which can be mapped to the standards registered information. Additionally, the SASRef and memName columns should identify where the new data set is to be created. The RefType must be set to LIBREF.

For example, to retrieve SDTM 3.1.1 reference metadata about tables, the data set should have the columns standard=CDISC-SDTM, standardVersion=3.1.1. Type should be set to "referencemetadata", subtype to "table". SASRef could be set to "work" and memname to "refTableMD".

Deprecated. explanation

Parameters:

- _cstSASReferences - Required. The (libname.)member that refers to a valid SASReferences file.
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_getstandardmetadata.sas

%cst_getStandardSASReferences

%cst_getStandardSASReferences(_cstStandard=, _cstStandardVersion=, _cstOutputDS=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: Framework]

Retrieves the global SASReference records for the given standard.

- If the macro succeeds the global variable _cst_rc will be set to 0. If it fails for some reason _cst_rc will be set to 1. The results data set will contain more information as to why it failed.
- Parameters:
- _cstStandard - Required. The name of a registered standard.
- _cstStandardVersion - Optional. The version of the standard that the caller wants to retieve the global SASReferences for. This may be omitted if the caller is requesting the default version for the standard.
- _cstOutputDS - Required. The (libname.)member name of the output datat set to be created.
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_getstandardsasreferences.sas

%cst_getStatic

%cst_getStatic(_cstName=, _cstVar=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstName - Parameter description not provided
- _cstVar - Parameter description not provided

File: cst_getstatic.sas

%cst_insertStandardSASRefs

%cst_insertStandardSASRefs(_cstSASReferences=, _cstOutputDS=, _cstAddRequiredCSTRefs=0, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: Not specified]

Inserts missing standards information into a SASReferences file.

Where a SASReferences file uses a standard, it is possible to specify only the standard, standardVersion, type and subType for information that has been registered by the standard. Calling this macro will "fill-in" the missing information. If a standardVersion is not specified, the information for the default version of that standard will be used.

Parameters:

- _cstSASReferences - Optional. The(libname.)member that points to a SASReferences file to be completed. If this is not specified then the global macro variables _cstSASRefsLoc, _cstSASRefsName may be used to specify the SASReferences file information. Finally, the _cstSASRefs macro variable will be used if none of the other mechanisms are provided/available.
- _cstOutputDS - Required. The output data set to create that contains the completed information.
- _cstAddRequiredCSTRefs - Parameter description not provided
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_insertstandardsasrefs.sas

See: CST Framework Global Macro Variables

%cst_registerStandard

%cst_registerStandard(_cstRootPath=, _cstControlSubPath=, _cstStdDSName=, _cstStdSASRefsDSName=, _cstOutputDS=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstRootPath - Parameter description not provided
- _cstControlSubPath - Parameter description not provided
- _cstStdDSName - Parameter description not provided
- _cstStdSASRefsDSName - Parameter description not provided
- _cstOutputDS - Parameter description not provided

File: cst_registerstandard.sas

%cst_setProperties

%cst_setProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Reads a properties file/data set and sets global macros accordingly. Property files should have the format name=value. Property data sets should have a character field for name and value. It may also have a comment field but this will be ignored.

Parameters:

- _cstPropertiesLocation - Required. The location of the property file. The format depends upon the value of _cstLocationType.
- _cstLocationType - Required. Identifies the format for the value of _cstPropertiesLocation. Valid Values are: PATH: The path to a properties file. FILENAME: A valid, assigned SAS filename to the properties file. DATA: A (libname.)membername of a SAS data set containing the properties.
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_setproperties.sas

%cst_setStandardProperties

%cst_setStandardProperties(_cstStandard=, _cstStandardVersion=, _cstSubType=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

standard.

When a standard is registered, it will likely also register values in a SASReferences file. A number of these values may be for properties files that are used by the standard, or provided by the standard to help users. For example, CST_FRAMEWORK provides a property subType of 'required' that points to a property file that has default settings for required properties. A user can call this method using the following code to set these properties:

```
%cst_setStandardProperties(
_cstStandard=CST_FRAMEWORK,
_cstStandardVersion=1.2,
_cstSubType=required);
```

Parameters:

- _cstStandard - Required. The name of a registered.
- _cstStandardVersion - Optional if the standard has a default set, otherwise it is mandatory. This specified the version of the standards.
- _cstSubType - Required. The name of the properties subtype that is to be read and properties set from.
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_setstandardproperties.sas

%cst_setStandardVersionDefault

%cst_setStandardVersionDefault(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstStandard - Parameter description not provided
- _cstStandardVersion - Parameter description not provided
- _cstResultsOverrideDS - Parameter description not provided

File: cst_setstandardversiondefault.sas

%cst_unregisterStandard

%cst_unregisterStandard(_cstStandard=, _cstStandardVersion=, _cstResultsOverrideDS=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstStandard - Parameter description not provided
- _cstStandardVersion - Parameter description not provided
- _cstResultsOverrideDS - Parameter description not provided

File: cst_unregisterstandard.sas

%cst_unsetProperties

%cst_unsetProperties(_cstPropertiesLocation=, _cstLocationType=, _cstResultsOverrideDS=);

[Exposure: external] [Macro Type: framework]

Reads a properties file/data set and unsets global macros accordingly. Property files should have the format name=value. Property data sets should have a character field for name and value. It may also have a comment field but this will be ignored.

Parameters:

- _cstPropertiesLocation - Required. The location of the property file. The format depends upon the value of _cstLocationType.
- _cstLocationType - Required. Identifies the format for the value of _cstPropertiesLocation. Valid Values are: PATH: The path to a properties file. FILENAME: A valid, assigned SAS filename to the properties file. DATA: A (libname.)membername of a SAS data set containing the properties.
- _cstResultsOverrideDS - Optional. The (libname.)member that refers to a results data set to be created. If omitted, the results data set specified by the &_cstResultsDS will be written to.

File: cst_unsetproperties.sas

%cstcheck_column

%cstcheck_column(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_column

Identifies any invalid column value(s) or attribute(s).

NOTE: Macro requires use of _cstCodeLogic at a statement level within a SAS data step context. _cstCodeLogic identifies records in errors by setting _cstError=1.

Example validation checks that use this macro:

- Value of Visit Number is formatted to > 3 decimal places
- A column character value is not left-justified
- Study day of Visit/Collection/Exam (**DY) equals 0
- Length of **TEST > 40

Required Global Macro Variables (beyond reporting and debugging variables): _cstSubjectColumns

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_column.sas

%cstcheck_columncompare

%cstcheck_columncompare(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_columncompare

Supports comparison of column values (much like cstcheck_multicolumn) providing additional functionality in the form of step-level code that allows, for example, optional reference to column metadata.

NOTE: Macro requires use of _cstCodeLogic at a data step level (that is, a full data step or proc sql invocation). _cstCodeLogic creates a work file (_cstproblems) containing records in error.

Example validation checks that use this macro: **DOSE and **DOSU inconsistencies for Expected columns

Required Global Macro Variables:

- _cstSubjectColumns
- _cstMetrics*
- <messaging, error>

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_columncompare.sas

%cstcheck_comparedomains

%cstcheck_comparedomains(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_comparedomains

Generally compares values for 1+ columns in one domain with values for those same columns in another domain. For example, USUBJID value in any domain does not have a matching USUBJID value in the DM domain.

NOTE: Macro requires use of _cstCodeLogic at a statement level within a SAS data step context. _cstCodeLogic identifies records in error by setting _cstError=1.

Example validation checks that use this macro: Unique USUBJID+VISIT+VISITNUM combinations in each domain not found in SV

Required Global Macro Variables: (none)

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_comparedomains.sas

%cstcheck_dsmismatch

%cstcheck_dsmismatch(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_dsmismatch

Identifies any data set mismatches between study and template metadata and the source data library.

NOTE: This macro module currently ignores tableScope and columnScope from the _cstControl input data set.

Required Global Macro Variables: (none)

Required File Inputs: Single-record control data set identified by control input parameter

Parameters:

_cstControl - The single observation data set containing check-specific metadata

File: cstcheck_dsmismatch.sas

%cstcheck_metamismatch

%cstcheck_metamismatch(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_metamismatch

Identifies inconsistencies between study and reference column metadata.

NOTE: Macro requires use of _cstCodeLogic as a full 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.

ASSUMPTIONS:

- No data content is accessed for this check
- Both study and reference metadata must be present to assess compliance
- Current coding approach assumes no reporting on non-errors

Example validation checks that use this macro:

- Required column not found (Error)
- Expected column not found (Warning)
- Permissible column not found (Note) Column found in data set but not in specification
- Supplemental Qualifier data set without USUBJID column
- Column metadata attribute differences (type, length, label, order, CT)

Required Global Macro Variables: (none)

Required File Inputs: Single-record control data set identified by control input parameter

Parameters:

- _cstControl - The single observation data set containing check-specific metadata

File: cstcheck_metamismatch.sas

%cstcheck_notconsistent

%cstcheck_notconsistent(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_notconsistent

Identifies any inconsistent column values across records.

NOTE: Macro requires use of _cstCodeLogic at a data step level (that is, a full data step or proc sql invocation). _cstCodeLogic creates a work file (_cstproblems) containing records in error.

Example validation checks that use this macro: **SEQ not consecutively incremented beginning at 1 Standard units inconsistent within **TESTCD across records

Required Global Macro Variables:

- _cstSubjectColumns
- _cstMetrics*
- <messaging, error>

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_notconsistent.sas

%cstcheck_notincodelist

%cstcheck_notincodelist(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_notincodelist

Identifies any column values inconsistent with controlled terminologies. For example, a **STAT value is found other than 'NOT DONE'.

NOTE: This macro requires reference to the SAS format search path built based on type=FMTSEARCH records in the sasreferences control file.

Processing is conditioned on the value of the check metadata LOOKUPTYPE field. When (FORMAT) the code compares column values against a SAS format in the format search path. CodeLogic is optional (that is,, if the user does not specify any codeLogic, cstcheck_notincodelist uses default logic, which is proc sql step code that creates work._cstproblems if 1+ error is detected). The SAS format is specified in the check metadata LOOKUPSOURCE field. When (DATASET) the code requires the use of CodeLogic to create the data set work._cstproblems. LOOKUPSOURCE must contain the reference data set (for example, MedDRA for AE preferred term lookups) used in CodeLogic. Given that any reference dictionary with any given structure may be used, it is incumbent on the user to code correct joins and lookup logic within CodeLogic. When (CODELIST) functionality is deferred for v1.2 When (METADATA) the code compares column values against a SAS format in the format search path. CodeLogic is optional (that is,, if the user does not specify any codeLogic, cstcheck_notincodelist uses default logic, which is proc sql step code that creates work._cstproblems if 1+ error is detected). The SAS format is specified in the source column metadata XMLCODELIST field.

Required Global Macro Variables: (none)

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_notincodelist.sas

%cstcheck_notsorted

%cstcheck_notsorted(_cstControl=);

[Exposure: external] [Macro Type: Validation check]

cstcheck_notsorted

Identifies any domain that is not sorted by the keys defined in the metadata.

Example validation check that use this macro: Identifies domain table that is not correctly sorted.

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_notsorted.sas

%cstcheck_notunique

%cstcheck_notunique(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_notunique

This is a multi-function macro that assesses the uniqueness of data sets, columns or value-pairs from two columns. Each of these three functions accesses different code sections within the macro.

Function 1: Is data set unique by a set of columns?

Data Sets - Assumed if control column columnScope is blank, code will cycle thru domains specified in control column tableScope. Code will identify any records that are not unique by the domain keys defined in the table-level metadata.

Multiple Columns - This option allows the specification of a single set of columns (in the form var1+var2+...varn). Code will identify any records that are not unique by the specified set of columns within each domain specified in tableScope. For purposes of reporting, the specified columns will be treated as the domain keys. No codeLogic is used (or currently checked).

Function 2: For any given subject, are column values unique? Single Columns - For single columns (for example, **SEQ), the code checks for uniqueness within USUBJID (except TSSEQ, within TSPARMCD). No codeLogic is used (or currently checked).

Function 3: Does a combination of 2 columns have unique values? Column Pairs - For multiple columns (for example, **TEST and **TESTCD), the code checks that there is a unique set of values for the pair of columns. MUST be specified in the form of matching columnScope sublists. Exactly and only two sublists may be specified. No codeLogic is used (or currently checked).

Function 4: Are the values in one column (Column2) consistent within each value of another column (Column1)?

Column Pairs - For multiple columns (for example, **TESTCD and **STRESU), the code checks that there is a unique value in Column2 for each value of Column1. MUST be specified in the form of matching columnScope sublists. Exactly and only two sublists may be specified, with the first list containing Column1 (for example, VSTESTCD) and the second list containing Column2 (for example, VSSTRESU). Codelogic is required and it is the presence of codelogic that distinguishes Function 3 and Function 4 processing.

columnScope sublists should be bounded by brackets in the following style:

[LBTEST+VSTEST][LBTESTCD+VSTESTCD]
The following limitations apply:

- the 2 lists must resolve to the same number of columns
- the columns to-be-compared must be in the same data set
- the first item in list 1 is paired with the first item in list 2, etc.

Example combinations of tableScope and columnScope:

tableScope columnScope How code interprets

- ALL For all domains, is each unique by its keys?
- FINDINGS [**TEST][**TESTCD] For all FINDINGS domains, **TEST and **TESTCD must map 1:1
- ALL **SEQ For all domains, check **SEQ for uniqueness within USUBJID
- DM Is DM unique by its keys (STUDYID+USUBJID)?
- DV [DVTERM][DVDECOD] For DV, DVTERM and DVDECOD must map 1:1
- SUPP** For all SUPP** domains, are records unique by their keys?
- DV USUBJID+DVTERM For DV, are records unique by USUBJID and DVTERM?

Required Global Macro Variables:

- _cstSubjectColumns
- _cstMetrics*
- <messaging, error>

Required File Inputs: Single-record control data set identified by _cstControl input parameter

Parameters:

- _cstControl - The single observation data set containing check-specific metadata

File: cstcheck_notunique.sas

%cstcheck_recmismatch

%cstcheck_recmismatch(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_recmismatch

Identifies any record mismatches across domains

NOTE: Macro requires use of _cstCodeLogic at a data step level (that is, a full data step or proc sql invocation). _cstCodeLogic creates a work file (_cstproblems) containing records in error.

Example CDISC-SDTM validation checks that use this macro: Comments, Relrec or Supplemental Qualifier RDOMAIN references to other domains or domain records that do not exist

Required Global Macro Variables:

- _cstMetrics*
- <messaging, error>

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_recmismatch.sas

%cstcheck_recnotfound

%cstcheck_recnotfound(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_recnotfound

Generally compares the consistency of one or more columns across two tables, or allows comparison of the consistency of one <table>.<column> with another <table>.<column>. For example (CDISC-SDTM), STUDYID in the TA domain does not match STUDYID in the DM domain.

NOTE: Macro requires use of _cstCodeLogic at a statement level within a SAS data setp context. _cstCodeLogic identifies records in error by setting _cstError=1.

NOTE: Macro requires that tableScope syntax specifies two sublists in the form [DM][TA], comparing one or more columnScope fields across the tables in these sublists.

CDISC-SDTM example validation check that uses this macro: DM subjects where no record for the subject is found in the DS table

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_recnotfound.sas

%cstcheck_violatesstd

%cstcheck_violatesstd(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_violatesstd

Identifies any invalid column value(s) defined within a reference standard.

NOTE: Macro requires use of _cstCodeLogic at a statement level within a SAS data step context. _cstCodeLogic identifies records in errors by setting _cstError=1.

Example validation checks that use this macro:

- Identifies a null value found in a column where core attribute is REQ
- Identifies a null value found in a column where core attribute is EXP
- A column character value is not correctly upcased
- A numeric column column containing non-numeric entries

Required Global Macro Variables: _cstSubjectColumns - Currently used only with SDTM model, CRT-DDS does not require this global macro. CRT-DDS will not use _cstMetricsNumSubj para- when running metrics (not subject based).

Parameters:

- _cstControl - The single observation data set containing check-specific metadata.

File: cstcheck_violatesstd.sas

See: This document

%cstcheck_zeroobs

%cstcheck_zeroobs(_cstControl=);

[Exposure: external] [Macro Type: Validation Check]

cstcheck_zeroobs

Identifies any data set with zero observations

Required Global Macro Variables: (none)

Required File Inputs: Single-record control data set identified by control input parameter

Parameters:

- _cstControl - The single observation data set containing check-specific metadata

File: cstcheck_zeroobs.sas

See: This document

%cstutil_allocatesasreferences

%cstutil_allocatesasreferences / des='CST: Allocate sasreferences';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_allocatesasreferences

Method to allocate any librefs and filerefs in the sasreferences data set and set the autocall and format search paths based upon sasreferences settings.

Must be called outside the context of a data step, typically as an initial step in any CST driver module (for example, cst_validate)

Note: A call to a framework macro to validate the structure and content of the sasreferences data set is a required initial step.

Required Global Macro Variables:

- _cstResultsDS
- _cstSASRefsLoc (provides location of sasreferences input file)
- _cstSASRefsName (provides name of sasreferences input file)
- _cstSASRefs (work library version of sasreferences)
- _cstFMTLibraries (include WORK and LIBRARY in fmtsearch?)
- _cstMessageOrder (Append or Merge, where Merge honors order precedence)

Required File Inputs: sasreferences.sas7bdat

File: cstutil_allocatesasreferences.sas

See: This document

%cstutil_allocGlobalMetadataLib

%cstutil_allocGlobalMetadataLib(_cstLibname=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstLibname - Parameter description not provided

File: cstutil_allocglobalmetadatalib.sas

%cstutil_appendresultds

%cstutil_appendresultds(_cstErrorDS=, _cstVersion=&_cstStandardVersion, _cstSource=&_cstCheckSource, _cstStdRef=);

[Exposure: internal] [Macro Type: Framework utility]

Appends a check-level work results data set to the process work results data set. Parameters passed are check-level -- not record-level -- values

Must be called outside the context of a data step.

Required File Inputs: (none)

Required Global Macro Variables: (none)

Parameters:

- _cstErrorDS - - A SAS work data set that contains 1 or more observations documenting, on a source data set record level, results of check-level validation processing.
- _cstVersion - - The specific version of the model, this defaults to the global _cstStandardVersion macro variable value. Used to lookup an associated message from the messages dataset.
- _cstSource - - The source of the check, allowing source-specific messaging. Used to lookup an associated message from the messages dataset.

- _cstStdRef - - Optional. Reference in standard supporting check.

File: cstutil_appendresultds.sas

%cstutil_buildcollist

%cstutil_buildcollist(_cstFormatType=DATASET, _cstColWhere=, _cstDomWhere=, _cstColDSName=&_cstColumnMetadata, _cstDomDSName=&_cstTableMetadata, _cstColSubOverride=N, _cstDomSubOverride=N);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_buildcollist

Builds set of columns (in either list or data set format) based upon the value from the validation check control file validation_control.columnscope.

Note that the expected result is that work._csttablemetadata and work._cstcolumnmetadata data sets are created and are in sync with each other. That means they are consistent with regard to the tables represented based on resolving the tableScope and columnScope check macro fields.

Rules used to interpret columnscope values (using mostly CDISC-SDTM examples):

- Validation_control.columnscope may be null
- Blanks are translated to "+" (for example, LBDTC LBENDTC becomes LBDTC+LBENDTC)
- Value should not begin with a "+" or "-"
- If the blank translation results in multiple "+" characters, all but one of this is removed (for example, AE1 +DM1 becomes AE1++DM1 becomes AE1+DM1)
- No attempt is made to assess the validity of the columnscope value (for example, **TEST-AE1 is allowed, though no change to the resolved set of **TEST columns occurs)
- The derived set of columns is built by parsing columnscope from left to columns)
- If <libref> is included, it must be listed in the sasreferences.SASRef column

Wildcarding conventions:

- must use the string **
- may appear as a suffix (for example, SUPP**, for all columns that start with SUPP)
- may appear as a prefix (for example, **DTC, for all columns that end with DTC)
- may appear alone (for example, **) - equivalent to _ALL_
- <table>.** for all columns in the specified data set
- **.USUBJID for all USUBJID columns across referenced data sets

Sublists are delimited by brackets and:

- resolved lengths (that is, # columns) must be the same unless _cst*SubOverride is set to Y
- must conform to non-sublist rules stated above

A special naming convention of <column>:<value>, such as QUALIFIERS:DATETIME allows specification of a _cstColumnMetadata column and column value to subset

columns. In this example, all _cstColumnMetadata.QUALIFIERS= 'DATETIME' columns are returned.

Sample columnscope values:

- _ALL_ (all columns)
- AESEQ (a single column)
- LBDTC+LBENDTC (multiple columns)
- QUALIFIERS:DATETIME (_cstColumnMetadata.QUALIFIERS='DATETIME')
- **TEST (all columns ending in "TEST")
- DM** (all columns beginning with "DM")
- **TEST+**TESTCD (all columns ending in "TEST" or "TESTCD")
- [AESTDY+CMSTDY+EXSTDY][AEENDY+CMENDY+EXENDY] (two paired sublists)
- SRCDATA1.AE.AESTDY+SRCDATA2.AE.AESTDY (AESTDY column from AE data sets in two different libraries)
- AE.** (all columns in the AE table)
- **.USUBJID (all USUBJID columns from all tables)

Required Global Macro Variables (beyond reporting and debugging variables):

- _cstTableMetadata
- _cstColumnMetadata

Required File Inputs: work._cstcolumnmetadata

Parameters:

- _cstFormatType - LIST, sets macro variables of # tables and space-delimited list of tables DATASET (default), returns data set of tables matching tableScope specification
- _cstColWhere - Where clause to subset returned set of columns Any where clause is applied as the last step.
- _cstDomWhere - Where clause to subset returned set of tables Any where clause is applied as the last step.
- _cstColDSName - Name of data set with column metadata returned when _cstFormatType=DATASET
- _cstDomDSName - Name of data set with table metadata returned when _cstFormatType=DATASET
- _cstColSubOverride - Y or N (default), if Y override sublist processing to allow sublists of different lengths (such as columnScope=[**DTC][RFSTDTC])
- _cstDomSubOverride - Y or N (default), if Y override sublist processing to allow sublists of different lengths (such as tableScope=[_ALL_-DM][DM])

File: cstutil_buildcollist.sas

See: This document

%cstutil_builddomlist

%cstutil_builddomlist(_cstFormatType=DATASET, _cstDomWhere=, _cstDomDSName=&_cstTableMetadata, _cstSubOverride=N);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_builddomlist

Builds set of tables (in either list or data set format) based upon the value from the validation check control file validation_control.tablescope.

Rules used to interpret tablescope values (using mostly CDISC-SDTM examples):

- Validation_control.tablescope may not be null
- Blanks are translated to "+" (for example, AE DM becomes AE+DM)
- Value should not begin with a "+" or "-"
- If the blank translation results in multiple "+" characters, all but one of this is removed (for example, AE +DM becomes AE++DM becomes AE+DM)
- No attempt is made to assess the validity of the tablescope value (for example, CLASS:FINDINGS-AE is allowed, though no change to the resolved set of CLASS:FINDINGS tables occurs)
- The derived set of tables is built by parsing tablescope from left to right (for example, _ALL_-CLASS:RELATES builds a set of all tables removing RELREC and SUPP**)
- If <libref> is included, it must be listed in the sasreferences.SASRef column

Wildcarding conventions:

- must use the string **
- may appear as a suffix (for example, SUPP**, for all tables that start with SUPP)
- may appear as a prefix (for example, **DM, for all tables that end with DM)
- may appear alone (for example, **) - equivalent to _ALL_
- <libref>.** for all tables in the specified library
- **.AE for all AE tables across referenced libraries

Sublists are delimited by brackets and:

- resolved lengths (that is, # tables) must be the same unless _cstSubOverride is set to Y
- must conform to non-sublist rules stated above

A special naming convention of <column>:<value>, such as: CLASS:EVENTS allows specification of a _cstTableMetadata column and column value to subset tables. In this example, all CLASS='EVENTS' tables are returned.

Sample tablescope values:

- _ALL_ (all tables)
- AE (a single table)
- DM+DS (multiple tables)
- CLASS:EVENTS (_cstTableMetadata.CLASS='EVENTS')
- SUPP** (all Supplemental Qualifier tables)
- _ALL_-SUPP** (all tables except Supplemental Qualifier tables)
- [DM][EX] (two sublists comparing DM with EX)
- SRCDATA1.AE+SRCDATA2.AE (AE table from two different libraries)
- SRCDATA.** (all tables from the SRCDATA library)

- **.AE (all AE tables from all sourcedata libraries)

Required Global Macro Variables (beyond reporting and debugging variables): _cstTableMetadata

Required File Inputs: none

Parameters:

- _cstFormatType - LIST, sets macro variables of # tables and space-delimited list of tables DATASET (default), returns data set of tables matching tableScope specification
- _cstDomWhere - Where clause to subset returned set of tables Any where clause is applied as the last step.
- _cstDomDSName - Name of data set returned when _cstFormatType=DATASET
- _cstSubOverride - Y or N (default), if Y override sublist processing to allow sublists of different lengths (such as tableScope=[_ALL_-DM][DM])

File: cstutil_builddomlist.sas

See: This document

%cstutil_checkds

%cstutil_checkds(_cstdsname=, _csttype=, _cstsubtype=, _cststandard=*, _cststandardversion=*);

[Exposure: internal] [Macro Type: framework check]

cstutil_checkDS

validates the structure of the dataset against the template dataset structure shipped with the standard

Required Global Macro Variables: assumes &_cstResultsDS macro set to a valid 2 level name

Required File Inputs:

Parameters:

- _cstdsname - 2-level name of dataset to validate(required)
- _csttype - Required. The type of data set to be created. This value comes from the TYPE column in the sasreferences for the standard-version combination.
- _cstsubtype - Optional. Specifies the subtype for the type. This value comes from the SUBTYPE column in the sasreferences for the standard-version combination. If the type has no subtypes
- _cststandard - The name of the data standard to validate against (optional) By default all standard will be included
- _cststandardversion - The version of the data standard to validate agains (optional) By default all standardVersions will be included

File: cstutil_checkds.sas

See: This document

%chkvals

%chkvals;

[Exposure: Not specified] [Macro Type: Not specified]

**********macro parms not defined **********

dataset exists and has some records

dump contents of template table, compare it to the dataset passed in

only keep those columns which have lookup values shipped with the standard

load the list of columns to check into macro variable col

load the number of columns to check into macro variable numcol

seperate out which lookup columns have a dependancy on the refcolumn in the lookup table

load the list of columns which have a dependancy on a reference column into macro variable refcol

load the number of columns which depend on a refcolumn into macro variable numrefcol

sort the lookuptable

File: cstutil_checkds.sas

%cstutil_cleanupcstsession

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

[Exposure: internal] [Macro Type: Framework utility]

cstutil_cleanupcstsession

Cleans up after a Toolkit session, including such tasks as removing any process-level SAS files and clearing the work.sasmacr catalog.

Use: Most often used at the end of a Toolkit driver program, such as validate_data. Should be called where a data step or proc is
allowed.

Required Global Macro Variables:

- _cstDeBug
- _cstsasrefs
- _cstmessages

Parameters:

- _cstClearCompiledMacros - Remove all compiled macros from the work.sasmacr catalog. Values: 0 (No, default), 1 (Yes)

- _cstClearLibRefs - Deallocate all librefs and filerefs set based upon sasreferences content. Values: 0 (No, default), 1 (Yes)
- _cstResetSASAutos - Reset the autocall search path to its initial state. Values: 0 (No, default), 1 (Yes)
- _cstResetFmtSearch - Reset the format search path to its initial state. Values: 0 (No, default), 1 (Yes)
- _cstResetSASOptions - Reset SAS options to their initial state. Values: 0 (No), 1 (Yes, default)
- _cstDeleteFiles - Delete all CST work files and catalogs. Values: 0 (No), 1 (Yes, default) Note that is _cstDebug=1, files are NOT deleted even if _cstDeleteFiles=1.
- _cstDeleteGlobalMacroVars - Delete all CST global macro variables set based upon property file name/value pairs. Values: 0 (No, default), 1 (Yes)

File: cstutil_cleanupcstsession.sas

%cstutil_createTempMessages

%cstutil_createTempMessages(_cstCreationFlag=);

[Exposure: internal] [Macro Type: Framework]

Creates a temporary messages data set using the CST-FRAMEWORK messages.If the messages data set specified by the macro variable &_cstMessages does not exist,this macro will create a temporary version. It will look for the default version ofthe toolkit framework. It will then copy the messages data set specified in thedefault SASReferences file to the name specified in the &_cstMessages macro variable.If the caller supplies the name of a macro variable via the _cstCreationFlag, thenthis will be set if the data set was created in this macro.

Parameters:

- _cstCreationFlag - Optional. The name of a macro variable that is set in the macro. It will be set to 0 of the macro did not create the messages table (because it existed. It will be set to 1 if this macro created the data set. It is strongly suggested that the caller use this variable ro ensure that they clean up the temporary data set afterwards.

File: cstutil_createtempmessages.sas

%cstutil_deleteDataSet

%cstutil_deleteDataSet(_cstDataSetName=);

[Exposure: internal] [Macro Type: standard_name]

Deletes a data set if it exists. _cst_rc is set to 0 if it issuccessful and 1 otherwise. If the library is not assigned, orthe data set does not exist then this still returns 0.

Parameters:

- _cstDataSetName - Required. The (libname.)memname of the data set to be deleted.

File: cstutil_deletedataset.sas

%cstutil_getRandomNumber

%cstutil_getRandomNumber(_cstVarname=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstVarname - Parameter description not provided

File: cstutil_getrandomnumber.sas

%cstutil_getsasreference

%cstutil_getsasreference(_cstStandard=, _cstStandardVersion=, _cstSASRefType=, _cstSASRefSubtype=, _cstSASRefsasref=, _cstSASRefmember=, _cstConcatenate=0, _cstFullname=0);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_getsasreference

Gets the row-level metadata from the sasreferences data set given the type and optionally subtype

ASSUMPTIONS: sasreferences exists and has interpretable content

Required Global Macro Variables (beyond reporting and debugging variables):

- _cstTableMetadata
- _cstColumnMetadata
- _cstSASRefs

Required File Inputs: sasreferences data set (as defined by &_cstSASRefs)

Parameters:

- _cstStandard - - Identifies the name of a registered standard. If blank, no subsetting by standard is attempted.
- _cstStandardVersion - - Identifies the version of a registered standard. If blank, no subsetting by version is attempted.
- _cstSASRefType - - File or data type from sasreferences.type (required) Representative values: autocall control fmtsearch messages properties referencecontrol referencemetadata results sourcedata sourcemetadata
- _cstSASRefSubtype - - File or data subtype from sasreferences.subtype Values are specific to type. Optional - some types do not have subtypes Representative values: column data log lookup metrics package reference results table validation
- _cstSASRefsasref - - Identifies the calling macro variable name to populate with the value of sasreferences.sasref
- _cstSASRefmember - - Identifies the calling macro variable name to populate with the value of sasreferences.memname, based upon the value of the _cstFullname parameter.
- _cstConcatenate - - If 1, return multiple row values, space delimited, for each macro variable requested (sasref, member).
- _cstFullname - - If 1, return full name from sasreferences.memname

File: cstutil_getsasreference.sas

See: This document

%cstutil_getsubjectcount

%cstutil_getsubjectcount(_cstDS=, _cstsubid=&_cstSubjectColumns);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_getsubjectcount

Part of metrics processing, populates the metrics global macro variable _cstMetricsCntNumSubj with the count of the number of subjects

Called by any macro code module for which a count of the number of subjects is desired.

Required Global Macro Variables (beyond reporting and debugging variables): _cstSubjectColumns (used by default for a null _cstsubid input parameter)

Required File Inputs: source data set to-be-processed (as parameter _cstDS)

Parameters:

- _cstDS - The source data set containing subject data of interest
- _cstsubid - The set of subject identifiers appropriate for the _cstDS

File: cstutil_getsubjectcount.sas

%cstutil_internalmanageresults

%cstutil_internalmanageresults(_cstAction=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstAction - Parameter description not provided

File: cstutil_internalmanageresults.sas

%cstutil_messagesdsattr

%cstutil_messagesdsattr /des='CST: Messages data set column attributes';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_messagesdsattr

Defines messages data set column attributes

Use: Statement level within a SAS data step, where a SAS attrib statement may be used.

Required Global Macro Variables: (none)

Required File Inputs: (none)

File: cstutil_messagesdsattr.sas

%cstutil_metricsdsattr

%cstutil_metricsdsattr /des='CST: Metrics data set column attributes';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_metricsdsattr

Defines metrics data set column attributes

Use: Statement level within a SAS data step, where a SAS attrib statement may be used.

Required Global Macro Variables: (none)

Required File Inputs: (none)

File: cstutil_metricsdsattr.sas

%cstutil_parsecolumnscope

%cstutil_parsecolumnscope(_cstscopestr=, _cstopsource=, _cstsublistnum=);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_parsecolumnscope

Parses input parameter strings to add or remove columns from the work data set _cstColumnMetadata.

Called only by cstutil_buildcollist

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs:

- work._csttempcolumnmetadata
- work._cstcolumnmetadata

Parameters:

- _cstscopestr - The string value being parsed. Generally, this will be the entire columnScope value if there are no sublists, or a specific sublist.
- _cstopsource - A modified string value used to populate the _cstRefValue macro value.
- _cstsublistnum - The sublist number within columnScope. If there is no sublist, this is set to 1.

File: cstutil_parsecolumnscope.sas

%cstutil_parsescopesegment

%cstutil_parsescopesegment(_cstPart=, _cstVarName=, _cstMessageID=CST0004);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_parsescopesegment

Parses validation check metadata columns tableScope and columnScope tohandle extended values such as <libref>.<table>.<column> and wildcardingto build a logical SAS code string to subset _cstTableMetadata and_cstColumnMetadata.

Called only by cstutil_parsecolumnscope and cstutil_parsetablescope

Required Global Macro Variables (beyond reporting and debugging variables):(none)

Required File Inputs:(none)

Parameters:

- _cstPart - Which part of the tableScope or columnScope string to be interpreted. Expected values: _cstLibPart, _cstTabPart or _cstColPart.
- _cstVarName - The column name in either _csttablemetadata or _cstcolumnmetadata. Typical values: sasref, table or column
- _cstMessageID - Parameter description not provided

File: cstutil_parsescopesegment.sas

%cstutil_parsetablescope

%cstutil_parsetablescope(_cstscopestr=, _cstopsource=, _cstsublistnum=);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_parsetablescope

Parses input parameter strings to add or remove tables from the workdata set _cstTableMetadata.

Called only by cstutil_builddomlist

Required Global Macro Variables (beyond reporting and debugging variables):(none)

Required File Inputs:

- work._csttablemetadata
- work._csttemptablemetadata

Parameters:

- _cstscopestr - The string value being parsed. Generally, this will be the entire tableScope value if there are no sublists, or a specific sublist.
- _cstopsource - A modified string value used to populate the _cstRefValue macro value.
- _cstsublistnum - The sublist number within tableScope. If there is no sublist, this is set to 1.

File: cstutil_parsetablescope.sas

%cstutil_readcontrol

%cstutil_readcontrol /des="CST: Create control file macro variables";

[Exposure: internal] [Macro Type: Framework utility]

cstutil_readcontrol

Purpose: To read a single validation_control record, as passed in via thedata set referenced by the _cstThisCheckDS global macro variable,and to create local macro variables for each column in the controlfile. These macro variables are then available within the contextof each specific check macro.

Called by each check macro

Required Global Macro Variables:_cstThisCheckDS

Required File Inputs:control file as stored in _cstThisCheckDS

File: cstutil_readcontrol.sas

%cstutil_resultsdsattr

%cstutil_resultsdsattr /des='CST: Results data set column attributes';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_resultsdsattr

Defines results data set column attributes

Use: Statement level within a SAS data step, where a SAS attrib statementmay be used.

Required Global Macro Variables:(none)

Required File Inputs:(none)

File: cstutil_resultsdsattr.sas

%cstutil_resultsdskeep

%cstutil_resultsdskeep /des='CST: Results data set columns';

[Exposure: internal] [Macro Type: Framework utility]

cstutil_resultsdskeep

Specifies results data set columns to keep in a data step

Use: Statement level within a SAS data step, where a SAS KEEP statementmay be used.

Required Global Macro Variables:(none)

Required File Inputs:(none)

File: cstutil_resultsdskeep.sas

%cstutil_setcstgroot

%cstutil_setcstgroot;

[Exposure: Not specified] [Macro Type: Not specified]

File: cstutil_setcstgroot.sas

%cstutil_setmodel

%cstutil_setmodel /des="Set Which Model Definition to Use";

[Exposure: internal] [Macro Type: Framework utility]

cstutil_setmodel

Purpose: To establish the comparison reference metadata for a given check.This is based on the validation_control.usesourcemetadata flag.If this flag is "Y", sourcemetadata.* serves as the comparisonmetadata, otherwise referencemetadata.* does

Called for each check, but ONLY by builddomlist and buildcollist macros

Required Global Macro Variables (beyond reporting and debugging variables):

- _cstTableMetadata
- _cstColumnMetadata

Required File Inputs:

- Source tables and column metadata (derived from sasreferences)
- Reference tables and column metadata (derived from sasreferences)

Assumptions:

- While there should generally only be a single source ofreferencemetadata.* from the sasreferences control data set, the codeallows multiple sources. These are concatenated here in the derivationof the work._cstTableMetadata and work._cstColumnMetadata data sets
- There may be multiple sources of sourcemetadata.* from the sasreferencescontrol data set. These are concatenated here in the derivation ofthe work._cstTableMetadata and work._cstColumnMetadata data sets

File: cstutil_setmodel.sas

%cstutil_writecubexml

%cstutil_writecubexml(_cstXMLOut=, _cstMDPFile=, _cstDebug=);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- _cstXMLOut - Parameter description not provided
- _cstMDPFile - Parameter description not provided
- _cstDebug - Parameter description not provided

File: cstutil_writecubexml.sas

%cstutil_writemetric

%cstutil_writemetric(_cstMetricParameter=, _cstResultID=, _cstResultSeqParm=, _cstMetricCnt=, _cstSrcDataParm=, _cstMetricsDSParm=&_cstMetricsDS);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_writemetric

Adds a single record to the metrics data set based solely on parameter values.

Must be called outside the context of a data step.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs: &_cstMetricsDS (as parameter _cstMetricsDSParm)

Parameters:

- _cstMetricParameter - - Metric parameter. Extensible set of metrics. Examples include: # of subjects # of records tested # of distinct check invocations Errors (severity=High) reported Warnings (severity=Medium) reported Notes (severity=Low) reported # of structural errors # of content errors METRICS value.
- _cstResultID - Parameter description not provided
- _cstResultSeqParm - - Generally 1, unless duplicate values of resultid need to be distinguished, such as multiple invocations of the same validation checkid.
- _cstMetricCnt - - Record counter for _cstMetricParameter
- _cstSrcDataParm - - Information to link metric back to source (for example, SDTM domain name or calling validation code module).
- _cstMetricsDSParm - - The base (cross-check) metrics data set to which this record is to be appended. By default, this is the data set referenced by the _cstMetricsDS global macrovar.

File: cstutil_writemetric.sas

%cstutil_writeresult

%cstutil_writeresult(_cstResultID=, _cstValCheckID=, _cstResultParm1=, _cstResultParm2=, _cstResultSeqParm=1, _cstSeqNoParm=1, _cstSrcDataParm=, _cstResultFlagParm=0, _cstRCParm=0, _cstActualParm=, _cstKeyValuesParm=, _cstResultDetails=, _cstResultsDSParm=&_cstResultsDS);

[Exposure: internal] [Macro Type: Framework utility]

cstutil_writeresult

Adds a single record to the results data set based solely on parameter values.

Must be called outside the context of a data step.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs:

- &_cstMessages (created by cstutil_allocatesasreferences)
- &_cstResultsDS (as parameter _cstResultsDSParm)

Parameters:

- _cstResultID - - Set to validation process id (for example, CST0017). Should have matching entry in messages data set.
- _cstValCheckID - - Validation check identifier from validation_control data set
- _cstResultParm1 - - An optional parameter to appear in first substitution field of the associated message with the same resultid.
- _cstResultParm2 - - An optional parameter to appear in second substitutio field of the associated message with the same resultid.
- _cstResultSeqParm - - Generally 1, unless duplicate values of resultid need to be distinguished, such as multiple invocations of the same validation checkid.
- _cstSeqNoParm - - Sequence number within _cstResultSeqParm, beginning with 1 and incremented by 1 for each observation written to data set.
- _cstSrcDataParm - - Information to link result back to source (for example, SDTM domain name or calling validation code module).
- _cstResultFlagParm - - Problem detected? Set to 0 if this is an informational rather than error record. A positive value indicates an error was detected. A negative value indicates the check failed to run for some reason.
- _cstRCParm - - Value of _cst_rc at the point the result is written to data set.
- _cstActualParm - - Source data value(s) causing result to be written to data set.
- _cstKeyValuesParm - - Information to link result back to a specific source data record (for example, data set key or xml row/column values).
- _cstResultDetails - - Provides the ability to specify run-time details about the result. These take precedence over metadata result details.
- _cstResultsDSParm - - The base (cross-check) result data set to which this record is to be appended. By default, this is the data set referenced by the _cstResultsDS global macrovar.

File: cstutil_writeresult.sas

Module SDTM V3.1.1 (Runtime)

Overview

This is the SDTM 3.1.1 runtime macro library.

Since: V1.2

Macro Summary

Table 4.3 Module SDTM V3.1.1 (Runtime) Macro Summary

Exposure	Macro
external	SDTM Validation Process %sdtm_validate /des='CST: Validate CDISC SDTM model files';
internal	SDTM Validation Check Utility %sdtmcheckutil_recordlookup(_cstSourceDS=&_cstDSName, _cstSourceLib=&_cstRefOnly);
	%sdtmutil_getchecks(_cstControl=, _cstMeta=, _cstMsg=, _cstDomain="*", _cstOutDS=, _cstIncludeDraft=true);
internal	SDTM utility %sdtmutil_iso8601(_cstString=);
	%sdtmutil_listsettings(group=_ALL_);

Macro Detail

%sdtm_validate

%sdtm_validate /des='CST: Validate CDISC SDTM model files';

[Exposure: external] [Macro Type: SDTM Validation Process]

sdtm_validate

Validate CDISC SDTM model files

The basic function of this code module is to cycle through the validation checks to-be-run, writing validation results to the process results and (optionally) metrics data sets. These are then persisted to any permanent location based on type=results records in sasreferences. Process cleanup is based on the _cstDebug global macro variable.

Required Global Macro Variables (beyond reporting and debugging variables): (none)

Required File Inputs: run-time (type=control,subtype=validation in sasreferences) check data set

File: sdtm_validate.sas

%sdtmcheckutil_recordlookup

%sdtmcheckutil_recordlookup(_cstSourceDS=&_cstDSName, _cstSourceLib=&_cstRefOnly);

[Exposure: internal] [Macro Type: SDTM Validation Check Utility]

sdtmcheckutil_recordlookup

Creates work._cstproblems containing any records included in the _cstSourceDS data set that cannot be found in the referenced lookup data set. For example, SUPPAE includes a record that points to record in the AE domain that does not exist by the key values specified.

NOTE: Macro is called within _cstCodeLogic at a data step level (that is, a full data step or proc sql invocation).

Required Global Macro Variables:

(none)

Parameters:

- _cstSourceDS - The source data set evaluated by the validation check
- _cstSourceLib - The source libref for the lookup domain

File: sdtmcheckutil_recordlookup.sas

%sdtmutil_buildsasreferences

%sdtmutil_buildsasreferences;

[Exposure: Not specified] [Macro Type: Not specified]

File: sdtmutil_buildsasreferences.sas

%sdtmutil_getchecks

%sdtmutil_getchecks(_cstControl=, _cstMeta=, _cstMsg=, _cstDomain="*", _cstOutDS=, _cstIncludeDraft=true);

[Exposure: Not specified] [Macro Type: Not specified]

- _cstControl - Parameter description not provided
- _cstMeta - Parameter description not provided
- _cstMsg - Parameter description not provided
- _cstDomain - Parameter description not provided
- _cstOutDS - Parameter description not provided
- _cstIncludeDraft - Parameter description not provided

File: sdtmutil_getchecks.sas

%sdtmutil_iso8601

%sdtmutil_iso8601(_cstString=);

[Exposure: internal] [Macro Type: SDTM utility]

sdtmutil_iso8601

Verifies if a string in a valid ISO-8601 format. The verification includes tests specific to SDTM and clinical trials data in general Must be called from within a data step. It may be called more than once within a single data step. Required Global Macro Variables: (none)

Required File Inputs: (none)

Parameters:

- _cstString - string that designates the name of the SAS dataset variable that is being checked. Returns the following as SAS dataset variables. The programmer is expected to copy any values they wish to keep. All variables are automatically dropped at the end of the current dataset.
- _cstISOisValid – numeric Binary flag that denotes whether or not the ISO string is a valid ISO 8601 string. 0=String is invalid 1=String is valid.
- _cstISOrc – numeric Return Code. A value of 0 indicates that no problems were found. Any other value is a coding error number.
- _cstISOmsg – string Message describing the validity of the input string.
- _cstISOinfo – string An informational message providing additional details about string.
- _cstISOtype - string

File: sdtmutil_iso8601.sas

%sdtmutil_listsettings

%sdtmutil_listsettings(group=_ALL_);

[Exposure: Not specified] [Macro Type: Not specified]

Parameters:

- group - Parameter description not provided

File: sdtmutil_listsettings.sas

Appendix 5

CDISC-SDTM 3.1.1 Validation Checks

Validation Checks

The following table provides a complete list of all CDISC-SDTM 3.1.1 validation checks.

Table 5.1 Validation Checks

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0001	Janus	IR4000	Identifies domain table that has zero rows and therefore contains no data	_ALL_	
SDTM0001	WebSDM	IR4000	Identifies domain table that has zero rows and therefore contains no data	_ALL_	
SDTM0002	JanusFR	SAS0017	A load of data into JANUS requires that the DM, DS and EX domains be submitted for each study to be loaded.	DM+DS+EX	
SDTM0003	WebSDM	SAS0018	WebSDM and the SDTM model require only the DM domain be present.	DM	
SDTM0004	SAS	SAS0033	Source metadata includes domain data set not found in reference metadata	_ALL_	
SDTM0005	SAS	SAS0034	Custom domain data set does not adhere to specification naming guidelines	_ALL_	
SDTM0006	SAS	SAS0035	Source data library contains domain data not found in study metadata	_ALL_	

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0011	Janus	IR4250	Identifies a column that was described in the domain description but not included in the SAS dataset for that domain	_ALL_	
SDTM0011	WebSDM	IR4250	Identifies a column that was described in the domain description but not included in the SAS dataset for that domain	_ALL_	
SDTM0012	JanusFR	IR4252	Identifies a column listed in the domain description as Required (‘Req’) but not included in the SAS dataset for that domain	_ALL_	
SDTM0012	WebSDM	IR4252	Identifies a column listed in the domain description as Required (‘Req’) but not included in the SAS dataset for that domain	_ALL_	
SDTM0013	Janus	IR4253	Identifies a column listed in the domain description as Expected (‘Exp’) but not included in the SAS dataset for that domain	_ALL_	
SDTM0013	WebSDM	IR4253	Identifies a column listed in the domain description as Expected (‘Exp’) but not included in the SAS dataset for that domain	_ALL_	
SDTM0014	SAS	SAS0008	Identifies a column listed in the domain description as Permissible (‘Perm’) but not included in the SAS dataset for that domain	_ALL_	
SDTM0015	Janus	IR4254	Identifies a column that appears in the SAS dataset but is not listed in the domain description	_ALL_	
SDTM0015	WebSDM	IR4254	Identifies a column that appears in the SAS dataset but is not listed in the domain description	_ALL_	
SDTM0017	Janus	IR4258	Identifies a domain that appears to contain supplemental qualifier data but does not contain the Unique Subject Identifier (USUBJID)	SUPP**	

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0017	WebSDM	IR4258	Identifies a domain that appears to contain supplemental qualifier data but does not contain the Unique Subject Identifier (USUBJID)	SUPP**	
SDTM0019	JanusFR	IR4259	Identifies a variable where datatype in (study specific) description is not consistent with datatype implicit in SAS dataset	_ALL_	
SDTM0019	WebSDM	IR4259	Identifies a variable where datatype in (study specific) description is not consistent with datatype implicit in SAS dataset	_ALL_	
SDTM0020	SAS	SAS0006	Column order does not match standard	_ALL_	
SDTM0022	SAS	SAS0001	Column length < length defined in standard	_ALL_	
SDTM0023	SAS	SAS0002	Column length > length defined in standard	_ALL_	
SDTM0030	SAS	SAS0003	Column label inconsistent with label defined in standard	_ALL_	
SDTM0031	SAS	SAS0004	Column format found but column not subject to controlled terminology	_ALL_	
SDTM0032	SAS	SAS0005	Column format found but format name mismatch with standard controlled terminology name	_ALL_	
SDTM0101	JanusFR	IR4002	Identifies values that do not conform to the ISO 8601 standard for datetimes	_ALL_	**DTC+**STDTC+**ENDTC+BRTHDTC+RFSTDTC+RFENDTC
SDTM0101	WebSDM	IR4002	Identifies values that do not conform to the ISO 8601 standard for datetimes	_ALL_	**DTC+**STDTC+**ENDTC+BRTHDTC+RFSTDTC+RFENDTC
SDTM0102	JanusFR	IR4002	Identifies values that do not conform to the ISO 8601 standard for durations	_ALL_	**DUR
SDTM0102	WebSDM	IR4002	Identifies values that do not conform to the ISO 8601 standard for durations	_ALL_	**DUR

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0124	Janus	IR4113	Identifies records that violate the condition [LENGTH(Name of Measurement, Test, or Examination (**TEST)) less than or equal to 40 characters]	CLASS:FINDINGS	**TEST
SDTM0124	WebSDM	IR4113	Identifies records that violate the condition [LENGTH(Name of Measurement, Test, or Examination (**TEST)) less than or equal to 40 characters]	CLASS:FINDINGS	**TEST
SDTM0125	Janus	IR4114	Identifies records that violate the condition [LENGTH(Sort Name of Measurement, Test, or Examination (**TESTCD)) less than or equal to 8 characters, cannot start with a number or contain special characters]	CLASS:FINDINGS	**TESTCD
SDTM0125	WebSDM	IR4114	Identifies records that violate the condition [LENGTH(Sort Name of Measurement, Test, or Examination (**TESTCD)) less than or equal to 8 characters, cannot start with a number or contain special characters]	CLASS:FINDINGS	**TESTCD
SDTM0126	SAS	SAS0017	Qualifier Variable Label (QLABEL) length > 40	SUPP**	QLABEL
SDTM0127	SAS	SAS0018	Qualifier Variable Name (QNAM) length > 8, starts with a number or contains special characters	SUPP**	QNAM
SDTM0128	Janus	IR4115	Identifies records that violate the condition [LENGTH(Trial Summary Parameter (**PARM)) less than or equal to 40 characters]	TS	TSPARM
SDTM0128	WebSDM	IR4115	Identifies records that violate the condition [LENGTH(Trial Summary Parameter (**PARM)) less than or equal to 40 characters]	TS	TSPARM

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0129	Janus	IR4116	Identifies records that violate the condition [LENGTH(Trial Summary Parameter Sort Name (**PARMCD)) less than or equal to 8 characters, cannot start with a number or contain special characters]	TS	TSPARMCD
SDTM0129	WebSDM	IR4116	Identifies records that violate the condition [LENGTH(Trial Summary Parameter Sort Name (**PARMCD)) less than or equal to 8 characters, cannot start with a number or contain special characters]	TS	TSPARMCD
SDTM0201	Janus	IR4001	Identifies a null (empty) value found in a column where (Standard) Core attribute is 'Req'	_ALL_	
SDTM0201	WebSDM	IR4001	Identifies a null (empty) value found in a column where (Standard) Core attribute is 'Req'	_ALL_	
SDTM0202	SAS	SAS0015	Identifies a null (empty) value found in a column where (Standard) Core attribute is 'Exp'	_ALL_	
SDTM0203	SAS	SAS0010	Character column value is not correctly upcased per spec	_ALL_	
SDTM0204	SAS	SAS0011	Character column value contains the numeric missing '.' or any special missing value like '.N'	_ALL_	
SDTM0205	SAS	SAS0012	Column value is not left-justified	_ALL_	
SDTM0206	Janus	IR4003	Identifies records where the value in the Domain Abbreviation column (DOMAIN) doesn't match the name of Domain	_ALL_-SUPP**-RELREC	DOMAIN
SDTM0206	WebSDM	IR4003	Identifies records where the value in the Domain Abbreviation column (DOMAIN) doesn't match the name of Domain	_ALL_-SUPP**-RELREC	DOMAIN

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0207	Janus	IR4010	Identifies records where the value for Visit Number (VISITNUM) is formatted to more than 3 decimal places	_ALL_	VISITNUM
SDTM0207	WebSDM	IR4010	Identifies records where the value for Visit Number (VISITNUM) is formatted to more than 3 decimal places	_ALL_	VISITNUM
SDTM0208	Janus	IR4009	Identifies records where Result or Finding in Original Units (**ORRES) and Status (**STAT) both have a value, or where both are null and Derived Flag (**DRVFL) is not equal to 'Y'	CLASS:FINDINGS-IE	[**ORRES][**STAT]
SDTM0208	WebSDM	IR4009	Identifies records where Result or Finding in Original Units (**ORRES) and Status (**STAT) both have a value, or where both are null and Derived Flag (**DRVFL) is not equal to 'Y'	CLASS:FINDINGS-IE	[**ORRES][**STAT]
SDTM0209	JanusFR	IR4100	Identifies records that violate the condition [Study Day of Start of Observation (**STDY) less than or equal to Study Day of End of Observation (**ENDY)], limited to records where **STDY is not null and **ENDY is not null	_ALL_-DS	[**STDY][**ENDY]
SDTM0209	WebSDM	IR4100	Identifies records that violate the condition [Study Day of Start of Observation (**STDY) less than or equal to Study Day of End of Observation (**ENDY)], limited to records where **STDY is not null and **ENDY is not null	_ALL_-DS	[**STDY][**ENDY]
SDTM0210	JanusFR	IR4101	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **STDTC is not null and **ENDTC is not null	_ALL_-DS-LB	[**STDTC][**ENDTC]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0210	WebSDM	IR4101	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **STDTC is not null and **ENDTC is not null	_ALL_-DS-LB	[**STDTC][**ENDTC]
SDTM0211	WebSDM	IR4130	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) or Start Relative to Reference Period (**STRF) is not null], limited to records where [End Date/Time of Observation (**ENDTC) or End Relative to Reference Period (**ENRF) is not null]	CM+SU	[**STRF][**ENRF]
SDTM0211	WebSDM	IR4130	Identifies records that violate the condition [Start Date/Time of Observation (**STDTC) or Start Relative to Reference Period (**STRF) is not null], limited to records where [End Date/Time of Observation (**ENDTC) or End Relative to Reference Period (**ENRF) is not null]	_ALL_-DS-LB	[**STDTC][**ENDTC]
SDTM0212	Janus	IR4131	Identifies records that violate the condition [Planned Time Point Name (**TPT) is not null], limited to records where [Planned Time Point Number (**TPTNUM) is not null]	_ALL_	[**TPT][**TPTNUM]
SDTM0212	WebSDM	IR4131	Identifies records that violate the condition [Planned Time Point Name (**TPT) is not null], limited to records where [Planned Time Point Number (**TPTNUM) is not null]	_ALL_	[**TPT][**TPTNUM]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0213	Janus	IR4132	Identifies records that violate the condition [Planned Time Point Number (**TPTNUM) is not null], limited to records where [Planned Time Point Name (**TPT) is not null]	_ALL_	[**TPT][**TPTNUM]
SDTM0213	WebSDM	IR4132	Identifies records that violate the condition [Planned Time Point Number (**TPTNUM) is not null], limited to records where [Planned Time Point Name (**TPT) is not null]	_ALL_	[**TPT][**TPTNUM]
SDTM0214	Janus	IR4133	Identifies records that violate the condition [Time Point Reference (**TPTREF) is not null], limited to records where [Elapsed Time from Reference Point (**ELTM) is not null]	_ALL_	[**TPTREF][**ELTM]
SDTM0214	WebSDM	IR4133	Identifies records that violate the condition [Time Point Reference (**TPTREF) is not null], limited to records where [Elapsed Time from Reference Point (**ELTM) is not null]	_ALL_	[**TPTREF][**ELTM]
SDTM0215	WebSDM	IR4117	Identifies records that violate the condition [End Relative to Reference Period (**ENRF) is not null], limited to records where [End Date/Time of Observation (**ENDTC) is null] and [Occurrence (**OCCUR) doesn't equal 'N']	AE+CM+MH+SU	[**ENRF][**ENDTC]
SDTM0216	WebSDM	IR4118	Identifies records that violate the condition [Start Relative to Reference Period (**STRF) is not null], limited to records where [Start Date/Time of Observation (**STDTC) is null] and [Occurrence (**OCCUR) doesn't equal 'N']	CM+SU	[**STRF][**STDTC]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0217	JanusFR	IR4120	Identifies records that violate the condition [Evaluation Interval (**EVLINT) greater than or equal to 0], limited to records where **EVLINT is not null	_ALL_	**EVLINT
SDTM0217	WebSDM	IR4120	Identifies records that violate the condition [Evaluation Interval (**EVLINT) greater than or equal to 0], limited to records where **EVLINT is not null	_ALL_	**EVLINT
SDTM0218	Janus	IR4107	Identifies records that violate the condition [Status (**STAT) equals 'NOT DONE'], limited to records where **STAT is not null	_ALL_	**STAT
SDTM0218	WebSDM	IR4107	Identifies records that violate the condition [Status (**STAT) equals 'NOT DONE'], limited to records where **STAT is not null	_ALL_	**STAT
SDTM0219	Janus	IR4122	Identifies records that violate the condition [Reason Not Done (**REASND) is null], limited to records where [Status (**STAT) is null]	CM+EG+LB+ MH+PE+QS+ SC+SU+VS	[**REASND][**S TAT]
SDTM0219	WebSDM	IR4122	Identifies records that violate the condition [Reason Not Done (**REASND) is null], limited to records where [Status (**STAT) is null]	CM+EG+LB+ MH+PE+QS+ SC+SU+VS	[**REASND][**S TAT]
SDTM0220	Janus	IR4110	Identifies records that violate the condition [Duration (**DUR) greater than or equal to 0], limited to records where **DUR is not null	_ALL_	**DUR
SDTM0220	WebSDM	IR4110	Identifies records that violate the condition [Duration (**DUR) greater than or equal to 0], limited to records where **DUR is not null	_ALL_	**DUR

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0221	Janus	IR4136	Identifies records where values are not found in the study-specific codelist attached to a variable	_ALL_	
SDTM0221	WebSDM	IR4136	Identifies records where values are not found in the study-specific codelist attached to a variable	_ALL_	
SDTM0222	Janus	IR4137	Identifies records that violate the condition [Study Day of Visit/Collection/Exam (**DY) does not equal 0]	_ALL_	**DY+**STDY+**ENDY+VISITDY
SDTM0222	WebSDM	IR4137	Identifies records that violate the condition [Study Day of Visit/Collection/Exam (**DY) does not equal 0]	_ALL_	**DY+**STDY+**ENDY+VISITDY
SDTM0223	SAS	SAS0030	Identifies records with the condition [Subcategory (**SCAT) is not null when Category of related records (**CAT) is null]	AE+CM+DS+EG+EX+IE+LB+MH+QS+SC+SU+VS	[**SCAT][**CAT]
SDTM0251	Janus	IR4121	Identifies records that violate the condition [Toxicity Grade (**TOXGR) is a valid number], limited to records where **TOXGR is not null	CLASS:EVENTS	**TOXGR
SDTM0251	SAS	IR4121	Identifies records that violate the condition [Toxicity Grade (**TOXGR) is a valid number], limited to records where **TOXGR is not null	_ALL_	**TOXGR
SDTM0251	WebSDM	IR4121	Identifies records that violate the condition [Toxicity Grade (**TOXGR) is a valid number], limited to records where **TOXGR is not null	CLASS:EVENTS	**TOXGR
SDTM0271	SAS	SAS0036	Value for column defined as a data set key is null	_ALL_	
SDTM0301	JanusFR	IR4104	Identifies records that violate the condition [End Relative to Reference Period (**ENRF) in ('BEFORE','DURING','AFTER','DURING/AFTER','U')], limited to records where **ENRF is not null	CLASS:EVENTS+CLASS:INTERVENTIONS	**ENRF

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0301	WebSDM	IR4104	Identifies records that violate the condition [End Relative to Reference Period (**ENRF) in ('BEFORE','DURING','AFTER','DURING/AFTER','U')], limited to records where **ENRF is not null	CLASS:EVENTS+CLASS:INTERVENTIONS	**ENRF
SDTM0302	JanusFR	IR4106	Identifies records that violate the condition [Occurrence (**OCCUR) in ('Y','N')], limited to records where **OCCUR is not null	CLASS:EVENTS+CLASS:INTERVENTIONS	**OCCUR
SDTM0302	WebSDM	IR4106	Identifies records that violate the condition [Occurrence (**OCCUR) in ('Y','N')], limited to records where **OCCUR is not null	CLASS:EVENTS+CLASS:INTERVENTIONS	**OCCUR
SDTM0303	JanusFR	IR4108	Identifies records that violate the condition [Start Relative to Reference Period (**STRF) in ('BEFORE','DURING','AFTER','U')], limited to records where **STRF is not null	CLASS:EVENTS+CLASS:INTERVENTIONS	**STRF
SDTM0303	WebSDM	IR4108	Identifies records that violate the condition [Start Relative to Reference Period (**STRF) in ('BEFORE','DURING','AFTER','U')], limited to records where **STRF is not null	CLASS:EVENTS+CLASS:INTERVENTIONS	**STRF
SDTM0351	JanusFR	IR4134	Identifies records that violate the condition [Dose units (**DOSU) is not null], limited to records where [Dose (**DOSE) is not null] and (Standard) Core attribute is 'Perm'	CLASS:INTERVENTIONS	[**DOSE][**DOSU]
SDTM0351	WebSDM	IR4134	Identifies records that violate the condition [Dose units (**DOSU) is not null], limited to records where [Dose (**DOSE) is not null] and (Standard) Core attribute is 'Perm'	CLASS:INTERVENTIONS	[**DOSE][**DOSU]
SDTM0352	JanusFR	IR4109	Identifies records that violate the condition [Dose (**DOSE) greater than or equal to 0], limited to records where **DOSE is not null	CLASS:INTERVENTIONS	**DOSE

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0352	WebSDM	IR4109	Identifies records that violate the condition [Dose (**DOSE) greater than or equal to 0], limited to records where **DOSE is not null	CLASS:INTERVENTIONS	**DOSE
SDTM0353	JanusFR	IR4138	Identifies records that violate the condition [Dose units (**DOSU) is not null], limited to records where [Dose (**DOSE) is not null] and (Standard) Core attribute is 'Exp'	CLASS:INTERVENTIONS	[**DOSE][**DOSU]
SDTM0353	WebSDM	IR4138	Identifies records that violate the condition [Dose units (**DOSU) is not null], limited to records where [Dose (**DOSE) is not null] and (Standard) Core attribute is 'Exp'	CLASS:INTERVENTIONS	[**DOSE][**DOSU]
SDTM0354	WebSDM	IR4139	Identifies records that violate the condition [Related Domain (RDOMAIN) is not null]	SUPP**	RDOMAIN
SDTM0355	SAS	SAS0040	Value for Related Domain (RDOMAIN) is inconsistent with data set name	SUPP**-SUPPQUAL	RDOMAIN
SDTM0401	Janus	IR4102	Identifies records that violate the condition [Baseline Flag (**BLFL) either 'Y' or null]	CLASS:FINDINGS	**BLFL
SDTM0401	WebSDM	IR4102	Identifies records that violate the condition [Baseline Flag (**BLFL) either 'Y' or null]	CLASS:FINDINGS	**BLFL
SDTM0402	JanusFR	IR4103	Identifies records that violate the condition [Derived Flag (**DRVFL) either 'Y' or null]	CLASS:FINDINGS	**DRVFL
SDTM0402	WebSDM	IR4103	Identifies records that violate the condition [Derived Flag (**DRVFL) either 'Y' or null]	CLASS:FINDINGS	**DRVFL
SDTM0403	JanusFR	IR4105	Identifies records that violate the condition [Fasting Status (**FAST) in ('Y','N','U')], limited to records where **FAST is not null	CLASS:FINDINGS	**FAST

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0403	WebSDM	IR4105	Identifies records that violate the condition [Fasting Status (**FAST) in ('Y','N','U')], limited to records where **FAST is not null	CLASS:FINDINGS	**FAST
SDTM0405	Janus	IR4112	Identifies records that violate the condition [Result or Finding in Standard Format (**STRESC) is not null], limited to records where [Derived Flag (**DRVFL) equals 'Y']	CLASS:FINDINGS-IE-PE-SC	[**STRESC][**DRVFL]
SDTM0405	WebSDM	IR4112	Identifies records that violate the condition [Result or Finding in Standard Format (**STRESC) is not null], limited to records where [Derived Flag (**DRVFL) equals 'Y']	CLASS:FINDINGS-IE-PE-SC	[**STRESC][**DRVFL]
SDTM0406	Janus	IR4123	Identifies records that violate the condition [Date/Time of Collection (**DTC) is not null], limited to records where [End Date/Time of Observation (**ENDTC) is not null]	LB	[LBDTC][LBENDTC]
SDTM0406	WebSDM	IR4123	Identifies records that violate the condition [Date/Time of Collection (**DTC) is not null], limited to records where [End Date/Time of Observation (**ENDTC) is not null]	LB	[LBDTC][LBENDTC]
SDTM0407	JanusFR	IR4124	Identifies records that violate the condition [Date/Time of Collection (**DTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **DTC is not null and **ENDTC exists	LB	[LBDTC][LBENDTC]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0407	WebSDM	IR4124	Identifies records that violate the condition [Date/Time of Collection (**DTC) less than or equal to End Date/Time of Observation (**ENDTC)], limited to records where **DTC is not null and **ENDTC exists	LB	[LBDTC][LBENDTC]
SDTM0408	Janus	IR4125	Identifies records that violate the condition [Original units (**ORRESU) is not null], limited to records where [Result or Finding in Original Units (**ORRES) is not null]	CLASS:FINDINGS-IE	[**ORRES][**ORRESU]
SDTM0408	WebSDM	IR4125	Identifies records that violate the condition [Original units (**ORRESU) is not null], limited to records where [Result or Finding in Original Units (**ORRES) is not null]	CLASS:FINDINGS-IE	[**ORRES][**ORRESU]
SDTM0409	Janus	IR4126	Identifies records that violate the condition [Original units (**ORRESU) is null], limited to records where [Result or Finding in Original Units (**ORRES) is null]	CLASS:FINDINGS-IE	[**ORRES][**ORRESU]
SDTM0409	WebSDM	IR4126	Identifies records that violate the condition [Original units (**ORRESU) is null], limited to records where [Result or Finding in Original Units (**ORRES) is null]	CLASS:FINDINGS-IE	[**ORRES][**ORRESU]
SDTM0410	JanusFR	IR4127	Identifies records that violate the condition [Normal Range Upper Limit-Standard Units (**STNRHI) greater than or equal to Normal Range Lower Limit-Standard Units (**STNRLO)], limited to records where **STNRHI is not null and **STNRLO is not null	CLASS:FINDINGS	[**STNRHI][**STNRLO]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0410	WebSDM	IR4127	Identifies records that violate the condition [Normal Range Upper Limit-Standard Units (**STNRHI) greater than or equal to Normal Range Lower Limit-Standard Units (**STNRLO)], limited to records where **STNRHI is not null and **STNRLO is not null	CLASS:FINDINGS	[**STNRHI][**STNRLO]
SDTM0411	SAS	SAS0029	Identifies records that violate the condition [Normal Range Upper Limit-Standard Units (**STNRHI) is null and Normal Range Lower Limit-Standard Units (**STNRLO) is null], or the condition [**STNRHI is not null and **STNRLO is not null]	CLASS:FINDINGS	[**STNRHI][**STNRLO]
SDTM0412	Janus	IR4128	Identifies records that violate the condition [Standard Units (**STRESU) is not null], limited to records where [Result or Finding in Standard Format (**STRESC) is not null]	CLASS:FINDINGS-IE	[**STRESC][**STRESU]
SDTM0412	WebSDM	IR4128	Identifies records that violate the condition [Standard Units (**STRESU) is not null], limited to records where [Result or Finding in Standard Format (**STRESC) is not null]	CLASS:FINDINGS-IE	[**STRESC][**STRESU]
SDTM0413	Janus	IR4129	Identifies records that violate the condition [Standard Units (**STRESU) is null], limited to records where [Result or Finding in Standard Format (**STRESC) is null]	CLASS:FINDINGS-IE	[**STRESC][**STRESU]
SDTM0413	WebSDM	IR4129	Identifies records that violate the condition [Standard Units (**STRESU) is null], limited to records where [Result or Finding in Standard Format (**STRESC) is null]	CLASS:FINDINGS-IE	[**STRESC][**STRESU]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0414	JanusFR	IR4135	Identifies records that violate the condition [Result or Finding in Standard Format (**STRESC) is not null], limited to records where [Result or Finding in Original Units (**ORRES) is not null]	CLASS:FINDINGS	[**ORRES][**STRESC]
SDTM0414	WebSDM	IR4135	Identifies records that violate the condition [Result or Finding in Standard Format (**STRESC) is not null], limited to records where [Result or Finding in Original Units (**ORRES) is not null]	CLASS:FINDINGS	[**ORRES][**STRESC]
SDTM0450	SAS	SAS0037	Identifies records where the lookup value for a coded field (such as **DECOD, **BODSYS or **LOINC) could not be found in the associated dictionary	_ALL_	**DECOD
SDTM0451	JanusFR	IR4007	Identifies records where the value for the Preferred Term could not be found in the MedDRA dictionary	AE	AEDECOD
SDTM0451	WebSDM	IR4007	Identifies records where the value for the Preferred Term could not be found in the MedDRA dictionary	AE	AEDECOD
SDTM0452	Janus	IR4008	Identifies records where Serious Event (AESER)='Y' but none of Involves Cancer (AESCAN), Congenital Anomaly or Birth Defect (AESCONG), Persist or Signif Disability/Incapacity (AESDISAB), Results in Death (AESDTH), Requires or Prolongs Hospitalization (AESHOSP), Is Life Threatening (AESLIFE), Other Medically Important Serious Event (AESMIE), or Occurred with Overdose (AESOD) equals 'Y'	AE	AESER

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0452	WebSDM	IR4008	Identifies records where Serious Event (AESER)='Y' but none of Involves Cancer (AESCAN), Congenital Anomaly or Birth Defect (AESCONG), Persist or Signif Disability/Incapacity (AESDISAB), Results in Death (AESDTH), Requires or Prolongs Hospitalization (AESHOSP), Is Life Threatening (AESLIFE), Other Medically Important Serious Event (AESMIE), or Occurred with Overdose (AESOD) equals 'Y'	AE	AESER
SDTM0453	JanusFR	R4019	Identifies records where value for [Serious Event (AESER)] is not found in Codelist [YESNO]	AE	AESER
SDTM0453	WebSDM	R4019	Identifies records where value for [Serious Event (AESER)] is not found in Codelist [YESNO]	AE	AESER
SDTM0454	JanusFR	R4023	Identifies records where value for [Congenital Anomaly or Birth Defect (AESCONG)] is not found in Codelist [YESNO], limited to records where AESCONG is not null	AE	AESCONG
SDTM0454	WebSDM	R4023	Identifies records where value for [Congenital Anomaly or Birth Defect (AESCONG)] is not found in Codelist [YESNO], limited to records where AESCONG is not null	AE	AESCONG
SDTM0455	JanusFR	R4024	Identifies records where value for [Persist or Signif Disability/Incapacity (AESDISAB)] is not found in Codelist [YESNO], limited to records where AESDISAB is not null	AE	AESDISAB
SDTM0455	WebSDM	R4024	Identifies records where value for [Persist or Signif Disability/Incapacity (AESDISAB)] is not found in Codelist [YESNO], limited to records where AESDISAB is not null	AE	AESDISAB

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0456	JanusFR	R4025	Identifies records where value for [Results in Death (AESDTH)] is not found in Codelist [YESNO], limited to records where AESDTH is not null	AE	AESDTH
SDTM0456	WebSDM	R4025	Identifies records where value for [Results in Death (AESDTH)] is not found in Codelist [YESNO], limited to records where AESDTH is not null	AE	AESDTH
SDTM0457	JanusFR	R4026	Identifies records where value for [Requires or Prolongs Hospitalization (AESHOSP)] is not found in Codelist [YESNO], limited to records where AESHOSP is not null	AE	AESHOSP
SDTM0457	WebSDM	R4026	Identifies records where value for [Requires or Prolongs Hospitalization (AESHOSP)] is not found in Codelist [YESNO], limited to records where AESHOSP is not null	AE	AESHOSP
SDTM0458	JanusFR	R4027	Identifies records where value for [Is Life Threatening (AESLIFE)] is not found in Codelist [YESNO], limited to records where AESLIFE is not null	AE	AESLIFE
SDTM0458	WebSDM	R4027	Identifies records where value for [Is Life Threatening (AESLIFE)] is not found in Codelist [YESNO], limited to records where AESLIFE is not null	AE	AESLIFE
SDTM0459	JanusFR	R4045	Identifies records where value for [Involves Cancer (AESCAN)] is not found in Codelist [YESNO], limited to records where AESCAN is not null	AE	AESCAN
SDTM0459	WebSDM	R4045	Identifies records where value for [Involves Cancer (AESCAN)] is not found in Codelist [YESNO], limited to records where AESCAN is not null	AE	AESCAN

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0460	JanusFR	R4046	Identifies records where value for [Other Medically Important Serious Event (AESMIE)] is not found in Codelist [YESNO], limited to records where AESMIE is not null	AE	AESMIE
SDTM0460	WebSDM	R4046	Identifies records where value for [Other Medically Important Serious Event (AESMIE)] is not found in Codelist [YESNO], limited to records where AESMIE is not null	AE	AESMIE
SDTM0461	JanusFR	R4047	Identifies records where value for [Occurred with Overdose (AESOD)] is not found in Codelist [YESNO], limited to records where AESOD is not null	AE	AESOD
SDTM0461	WebSDM	R4047	Identifies records where value for [Occurred with Overdose (AESOD)] is not found in Codelist [YESNO], limited to records where AESOD is not null	AE	AESOD
SDTM0462	Janus	R4102	Identifies records that violate the condition [Results in Death (AESDTH)='Y'], limited to records where [Outcome of Adverse Event (AEOUT)='FATAL']	AE	[AESDTH][AEOUT]
SDTM0462	WebSDM	R4102	Identifies records that violate the condition [Results in Death (AESDTH)='Y'], limited to records where [Outcome of Adverse Event (AEOUT)='FATAL']	AE	[AESDTH][AEOUT]
SDTM0463	Janus	R4103	Identifies records that violate the condition [Outcome of Adverse Event (AEOUT)='FATAL'], limited to records where [Results in Death (AESDTH)='Y']	AE	[AESDTH][AEOUT]
SDTM0463	WebSDM	R4103	Identifies records that violate the condition [Outcome of Adverse Event (AEOUT)='FATAL'], limited to records where [Results in Death (AESDTH)='Y']	AE	[AESDTH][AEOUT]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0464	JanusFR	R4043	Identifies records where value for [Concomitant or Additional Trtmnt Given (AECONTRT)] is not found in Codelist [YESNO], limited to records where AECONTRT is not null	AE	AECONTRT
SDTM0464	WebSDM	R4043	Identifies records where value for [Concomitant or Additional Trtmnt Given (AECONTRT)] is not found in Codelist [YESNO], limited to records where AECONTRT is not null	AE	AECONTRT
SDTM0501	Janus	IR4011	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFAIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa]	DM	[ARM][ARMCD]
SDTM0501	WebSDM	IR4011	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFAIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa]	DM	[ARM][ARMCD]
SDTM0502	JanusFR	R4096	Identifies records that violate the condition [Subject Reference Start Date/Time (RFSTDTC) is not null], limited to records where upper(Arm Code (ARMCD)) doesn't equal 'SCRNFAIL'	DM	[RFSTDTC][ARMCD]
SDTM0502	WebSDM	R4096	Identifies records that violate the condition [Subject Reference Start Date/Time (RFSTDTC) is not null], limited to records where upper(Arm Code (ARMCD)) doesn't equal 'SCRNFAIL'	DM	[RFSTDTC][ARMCD]
SDTM0503	JanusFR	R4097	Identifies records that violate the condition [Subject Reference End Date/Time (RFENDTC) is not null], limited to records where upper(Arm Code (ARMCD)) doesn't equal 'SCRNFAIL'	DM	[RFENDTC][ARMCD]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0503	WebSDM	R4097	Identifies records that violate the condition [Subject Reference End Date/Time (RFENDTC) is not null], limited to records where upper(Arm Code (ARMCD)) doesn't equal 'SCRNFAIL'	DM	[RFENDTC][ARMCD]
SDTM0504	JanusFR	R4007	Identifies records where value for [SEX] is not found in Codelist [SEX]	DM	SEX
SDTM0504	WebSDM	R4007	Identifies records where value for [SEX] is not found in Codelist [SEX]	DM	SEX
SDTM0505	Janus	R4008	Identifies records where value for [COUNTRY] is not found in Codelist [COUNTRY]	DM	COUNTRY
SDTM0505	WebSDM	R4008	Identifies records where value for [COUNTRY] is not found in Codelist [COUNTRY]	DM	COUNTRY
SDTM0506	JanusFR	R4006	Identifies records that violate the condition [Age (AGE) greater than or equal to 0], limited to records where AGE is not null	DM	AGE
SDTM0506	WebSDM	R4006	Identifies records that violate the condition [Age (AGE) greater than or equal to 0], limited to records where AGE is not null	DM	AGE
SDTM0507	Janus	R4106	Identifies records that violate the condition [Age Units (AGEU) is not null], limited to records where AGE is not null	DM	[AGE][AGEU]
SDTM0507	WebSDM	R4106	Identifies records that violate the condition [Age Units (AGEU) is not null], limited to records where AGE is not null	DM	[AGE][AGEU]
SDTM0508	JanusFR	R4062	Identifies records where value for [Age Units (AGEU)] is not found in Codelist [AGEUNITS2], limited to records where AGEU is not null	DM	AGEU

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0508	WebSDM	R4062	Identifies records where value for [Age Units (AGEU)] is not found in Codelist [AGEUNITS2], limited to records where AGEU is not null	DM	AGEU
SDTM0521	Janus	IR4119	Identifies records that violate the condition [Planned Elapsed Time from Reference Pt (**ELTM) greater than or equal to 0], limited to records where **ELTM is not null	EX	EXELTM
SDTM0521	WebSDM	IR4119	Identifies records that violate the condition [Planned Elapsed Time from Reference Pt (**ELTM) greater than or equal to 0], limited to records where **ELTM is not null	EX	EXELTM
SDTM0531	JanusFR	R4031	Identifies records where value for [Inclusion/Exclusion Category (IECAT)] is not found in Codelist [INCEX], limited to records where IECAT is not null	IE	IECAT
SDTM0531	WebSDM	R4031	Identifies records where value for [Inclusion/Exclusion Category (IECAT)] is not found in Codelist [INCEX], limited to records where IECAT is not null	IE	IECAT
SDTM0532	JanusFR	R4071	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES)] is not found in Codelist[YESNO], limited to records where IEORRES is not null	IE	IEORRES
SDTM0532	WebSDM	R4071	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES)] is not found in Codelist[YESNO], limited to records where IEORRES is not null	IE	IEORRES

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0533	JanusFR	R4072	Identifies records that violate the condition [I/E Criterion Original Result in Std Format (IESTRESC)] is not found in Codelist[YESNO], limited to records where IESTRESC is not null	IE	IESTRESC
SDTM0533	WebSDM	R4072	Identifies records that violate the condition [I/E Criterion Original Result in Std Format (IESTRESC)] is not found in Codelist[YESNO], limited to records where IESTRESC is not null	IE	IESTRESC
SDTM0534	Janus	R4073	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES) = I/E Criterion Original Result in Std Format (IESTRESC)]	IE	[IEORRES][IESTRESC]
SDTM0534	WebSDM	R4073	Identifies records that violate the condition [I/E Criterion Original Result (IEORRES) = I/E Criterion Original Result in Std Format (IESTRESC)]	IE	[IEORRES][IESTRESC]
SDTM0541	Janus	R4105	Identifies records that violate the condition [Description of Unplanned Element (SEUPDES) is not null], limited to records where Subject Element Code (ETCD) ='UNPLAN'	SE	[SEUPDES][ETCD]
SDTM0541	WebSDM	R4105	Identifies records that violate the condition [Description of Unplanned Element (SEUPDES) is not null], limited to records where Subject Element Code (ETCD) ='UNPLAN'	SE	[SEUPDES][ETCD]
SDTM0551	JanusFR	IR4012	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFAIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa]	TA	[ARM][ARMCD]

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0551	WebSDM	IR4012	Identifies records that violate the condition [If Arm Code (ARMCD)='SCRNFAIL' then Description of Arm (ARM) must equal 'Screen Failure', and vice versa]	TA	[ARM][ARMCD]
SDTM0561	Janus	R4101	Identifies records that violate the condition [Rule for End of Element (TEENRL) is not null or Planned Duration of Element (TEDUR) is not null]	TE	[TEENRL][TEDUR]
SDTM0561	WebSDM	R4101	Identifies records that violate the condition [Rule for End of Element (TEENRL) is not null or Planned Duration of Element (TEDUR) is not null]	TE	[TEENRL][TEDUR]
SDTM0601	SAS	SAS0013	Domain not sorted by keys as defined in standard	_ALL_	
SDTM0602	SAS	SAS0007	Records are not unique by the expected keys	_ALL_	
SDTM0603	JanusFR	IR4004	Identifies records where non-unique values for Sequence Number variable (**SEQ) exist within a subject	_ALL_	**SEQ
SDTM0603	WebSDM	IR4004	Identifies records where non-unique values for Sequence Number variable (**SEQ) exist within a subject	_ALL_	**SEQ
SDTM0604	SAS	SAS0009	Sequence Number (**SEQ) values are not consecutively incremented beginning at 1 for each USUBJID	TS	TSSEQ
SDTM0604	SAS	SAS0009	Sequence Number (**SEQ) values are not consecutively incremented beginning at 1 for each USUBJID	_ALL_-TS	**SEQ
SDTM0605	SAS	SAS0014	Report any variable for the domain which contains all missing/null values	_ALL_	_ALL_
SDTM0606	JanusFR	SAS0022	Identify any column defined as numeric in the standard that contains non-numeric values	_ALL_	

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0607	JanusFR	SAS0038	Site Study Identifier (SITEID) is null for all records	DM	SITEID
SDTM0621	WebSDM	IR4005	Identifies subjects where there are no records with a value of 'Y' in the baseline flag variable (**BLFL), excluding Arm Code (ARMCD)='SCRNFAIL'	CLASS:FINDINGS	**BLFL
SDTM0622	SAS	SAS0025	Inconsistency between Test (**TEST) and Test Code (**TESTCD)	CLASS:FINDINGS	[**TEST][**TESTCD]
SDTM0623	SAS	SAS0027	Identifies Test Code (**TESTCD) values where Standard Units (**STRESU) value is not consistent across all records	CLASS:FINDINGS-LB-IE	[**TESTCD][**STRESU]
SDTM0631	JanusFR	IR4006	Identifies Short Name of Measurement, Test or Examination (**TESTCD) values where Standard Units (**STRESU) value is not consistent across all records	LB	[**TESTCD][**STRESU]
SDTM0631	WebSDM	IR4006	Identifies Short Name of Measurement, Test or Examination (**TESTCD) values where Standard Units (**STRESU) value is not consistent across all records	LB	[**TESTCD][**STRESU]
SDTM0641	JanusFR	R4005	Identifies records where values for Unique Subject ID (USUBJID) are not unique, limited to records where USUBJID is not null	DM	
SDTM0641	WebSDM	R4005	Identifies records where values for Unique Subject ID (USUBJID) are not unique, limited to records where USUBJID is not null	DM	
SDTM0642	SAS	SAS0028	Inconsistency between Description of Arm (ARM) and Arm Code (ARMCD) values across all records	DM	[ARM][ARMCD]
SDTM0643	SAS	SAS0016	AGE precision inconsistent across records	DM	AGE
SDTM0644	JanusFR	SAS0019	The current version of JANUS requires that the STUDYID column have the same value for all records within a study	_ALL_-DM	STUDYID

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0644	JanusFR	SAS0019	The current version of JANUS requires that the STUDYID column have the same value for all records within a study	DM	STUDYID
SDTM0651	JanusFR	R4140	Identifies cases where Protocol Deviation Term (DVTERM) and Protocol Deviation Coded Term (DVDECOD)] do not map 1:1	DV	[DVTERM][DVDECOD]
SDTM0651	WebSDM	R4140	Identifies cases where Protocol Deviation Term (DVTERM) and Protocol Deviation Coded Term (DVDECOD)] do not map 1:1	DV	[DVTERM][DVDECOD]
SDTM0661	JanusFR	R4083	Identifies records where values for [Study Identifier (STUDYID), Unique Subject Identifier (USUBJID), Identifying Variable (IDVAR), Identifying Variable Value (IDVARVAL), Qualifier Variable Name (QNAM)] variable(s) are not unique	SUPP**	
SDTM0661	WebSDM	R4083	Identifies records where values for [Study Identifier (STUDYID), Unique Subject Identifier (USUBJID), Identifying Variable (IDVAR), Identifying Variable Value (IDVARVAL), Qualifier Variable Name (QNAM)] variable(s) are not unique	SUPP**	
SDTM0671	SAS	SAS0032	Inconsistency between Trial Summary Parameter (TSPARM) and Trial Summary Parameter Short Name (TSPARMCD)	TS	[TSPARM][TSPARMCD]
SDTM0801	JanusFR	IR4500	Identifies non-Demographics domain subjects (USUBJID) not found in the Demographics domain	[_ALL_-DM][DM]	STUDYID+USUBJID
SDTM0801	WebSDM	IR4500	Identifies non-Demographics domain subjects (USUBJID) not found in the Demographics domain	[_ALL_-DM][DM]	STUDYID+USUBJID

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0802	Janus	IR4505	Identifies Demographics subjects where no record for the subject is found in the Disposition domain	[DM][DS]	STUDYID+USUBJID
SDTM0802	WebSDM	IR4505	Identifies Demographics subjects where no record for the subject is found in the Disposition domain	[DM][DS]	STUDYID+USUBJID
SDTM0803	Janus	IR4506	Identifies Demographics subjects where no record for the subject is found in the Exposure domain	[DM][EX]	STUDYID+USUBJID
SDTM0803	WebSDM	IR4506	Identifies Demographics subjects where no record for the subject is found in the Exposure domain	[DM][EX]	STUDYID+USUBJID
SDTM0804	Janus	IR4501	Identifies Unique Subject Identifier (USUBJID) + Visit Name (VISIT) + Visit Number (VISITNUM) combinations not found in the SV domain	[_ALL_-SV][SV]	USUBJID+VISITNUM+VISIT
SDTM0804	WebSDM	IR4501	Identifies Unique Subject Identifier (USUBJID) + Visit Name (VISIT) + Visit Number (VISITNUM) combinations not found in the SV domain	[_ALL_-SV][SV]	USUBJID+VISITNUM+VISIT
SDTM0805	Janus	IR4502	Identifies records where the value for ARM Code (ARMCD) is not found in the TA domain, excluding 'SCRNFAIL'	[DM][TA]	ARMCD
SDTM0805	WebSDM	IR4502	Identifies records where the value for ARM Code (ARMCD) is not found in the TA domain, excluding 'SCRNFAIL'	[DM][TA]	ARMCD
SDTM0806	JanusFR	IR4507	Identifies Demographics treatment arms (Description of Arm (ARM) + Arm Code (ARMCD) combination) not found in the TA domain, excluding 'Screen Failure', 'SCRNFAIL'	[DM][TA]	ARM+ARMCD

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0806	WebSDM	IR4507	Identifies Demographics treatment arms (Description of Arm (ARM) + Arm Code (ARMCD) combination) not found in the TA domain, excluding 'Screen Failure', 'SCRNFAIL'	[DM][TA]	ARM+ARMCD
SDTM0807	JanusFR	SAS0039	TA domain is not provided and Planned Arm Code (ARMCD) is null for all rows in the Demographics domain	DM	ARMCD
SDTM0811	Janus	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain	[TA][TE]	ETCD
SDTM0811	Janus	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain	[SE][TE]	ETCD
SDTM0811	WebSDM	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain	[TA][TE]	ETCD
SDTM0811	WebSDM	IR4503	Identifies records where the value for Subject Element Code (ETCD) is not found in the TE domain	[SE][TE]	ETCD
SDTM0821	JanusFR	IR4504	Identifies records where the value for Inclusion/Exclusion Criterion Short Name (IETESTCD) is not found in the TI domain	[IE][TI]	IETESTCD
SDTM0821	WebSDM	IR4504	Identifies records where the value for Inclusion/Exclusion Criterion Short Name (IETESTCD) is not found in the TI domain	[IE][TI]	IETESTCD
SDTM0822	Janus	SAS0023	Identifies records where the value for Inclusion/Exclusion Category (IECAT) in the IE domain does not exist in the TI domain if the TI domain was supplied	[IE][TI]	IECAT
SDTM0831	JanusFR	SAS0020	The Study Identifier (STUDYID) in the TA domain does not match STUDYID in the DM domain	[TA][DM]	STUDYID

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0836	JanusFR	SAS0021	The Study Identifier (STUDYID) in the TV domain does not match STUDYID in the DM domain	[TV][DM]	STUDYID
SDTM0841	Janus	SAS0026	Identifies records where a value for VISITNUM in the SV domain is not found in the TV domain, limited to records where both the SV and TV domains exist and the Description of Unplanned Visit (SVUPDES) is null	[SV][TV]	VISITNUM
SDTM0851	JanusFR	IR4508	Identifies Comments (CO) domain reference to an unknown related domain	CO	RDOMAIN
SDTM0851	WebSDM	IR4508	Identifies Comments (CO) domain reference to an unknown related domain	CO	RDOMAIN
SDTM0861	Janus	IR4509	Identifies Related Records (RELREC) domain reference to an unknown related domain	RELREC	RDOMAIN
SDTM0861	WebSDM	IR4509	Identifies Related Records (RELREC) domain reference to an unknown related domain	RELREC	RDOMAIN
SDTM0862	JanusFR	IR4510	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to an unknown related domain	SUPP**	RDOMAIN
SDTM0862	WebSDM	IR4510	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to an unknown related domain	SUPP**	RDOMAIN
SDTM0863	Janus	IR4511	Identifies Related Records (RELREC) domain reference to a key variable that isn't defined in the target domain	RELREC	IDVAR
SDTM0863	WebSDM	IR4511	Identifies Related Records (RELREC) domain reference to a key variable that isn't defined in the target domain	RELREC	IDVAR
SDTM0864	JanusFR	IR4512	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a key variable that isn't defined in the target domain	SUPP**	IDVAR

checkid	checksource	sourceid	sourcedescription	tablescope	columnscope
SDTM0864	WebSDM	IR4512	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a key variable that isn't defined in the target domain	SUPP**	IDVAR
SDTM0865	Janus	IR4513	Identifies Related Records (RELREC) domain reference to a record that doesn't exist in the target domain	RELREC	IDVAR
SDTM0865	WebSDM	IR4513	Identifies Related Records (RELREC) domain reference to a record that doesn't exist in the target domain	RELREC	IDVAR
SDTM0866	JanusFR	IR4514	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a record that doesn't exist in the target domain	SUPP**	IDVAR
SDTM0866	WebSDM	IR4514	Identifies Supplemental Qualifiers (SUPPQUAL) domain reference to a record that doesn't exist in the target domain	SUPP**	IDVAR
SDTM0871	Janus	SAS0024	Identifies Comments (CO) domain reference to a record that doesn't exist in the target domain	CO	IDVAR

Appendix 6

CDISC-CRTDDS SAS Representation

Sample reference_tables Record

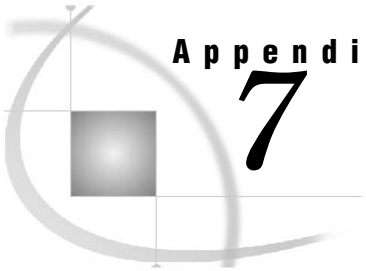
Table 6.1 Sample reference_tables Record

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

Sample reference_columns Record

Table 6.2 Sample reference_columns Record

Column Name	Column Value
SASref	SRCDATA
table	DefineDocument
column	FileType
label	
order	5
type	C
length	13
displayformat	$13.
standard	CDISC-CRTDDS
standardversion	1.0
standardref	
comment	
core	Req
xmlcodelist	FILETYPE
qualifier	



Appendix 7 CDISC-CRTDDS 1.0 Validation Checks

Validation Checks

The following table provides a complete list of all CDISC-CRTDDS 1.0 validation checks.

Table 7.1 Validation Checks

checkid	checktype	tablescope	columnscope	messagetext
CRT0100	Structural	CodeListItems		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	CodeLists		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	ComputationMethods		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	DefineDocument		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	FormDefArchLayouts		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	FormDefs		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	ImputationMethods		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	ItemDefs		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	ItemGroupDefs		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	ItemGroupLeaf		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	ItemRangeChecks		No two values for the source column can

checkid	checktype	tablescope	columnscope	messagetext
				be equivalent within the same source data set
CRT0100	Structural	MDVLeaf		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	MeasurementUnits		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	MetadataVersion		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	Presentation		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	Study		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	StudyEventDefs		No two values for the source column can be equivalent within the same source data set
CRT0100	Structural	ValueLists		No two values for the source column can be equivalent within the same source data set
CRT0101	Content	AnnotatedCRFs		Data is required for this field
CRT0101	Content	CLItemDecodeTranslatedText		Data is required for this field
CRT0101	Content	CodeListItems		Data is required for this field
CRT0101	Content	CodeLists		Data is required for this field
CRT0101	Content	ComputationMethods		Data is required for this field
CRT0101	Content	DefineDocument		Data is required for this field
CRT0101	Content	ExternalCodeLists		Data is required for this field
CRT0101	Content	FormDefArchLayouts		Data is required for this field
CRT0101	Content	FormDefItemGroupRefs		Data is required for this field
CRT0101	Content	FormDefs		Data is required for this field
CRT0101	Content	ImputationMethods		Data is required for this field
CRT0101	Content	ItemAliases		Data is required for this field
CRT0101	Content	ItemDefs		Data is required for this field
CRT0101	Content	ItemGroupAliases		Data is required for this field
CRT0101	Content	ItemGroupDefItemRefs		Data is required for this field
CRT0101	Content	ItemGroupDefs		Data is required for this field
CRT0101	Content	ItemGroupLeaf		Data is required for this field
CRT0101	Content	ItemGroupLeafTitles		Data is required for this field
CRT0101	Content	ItemMURefs		Data is required for this field
CRT0101	Content	ItemQuestionExternal		Data is required for this field

checkid	checktype	tablescope	columnscope	messagetext
CRT0101	Content	ItemQuestionTranslatedText		Data is required for this field
CRT0101	Content	ItemRangeCheckValues		Data is required for this field
CRT0101	Content	ItemRangeChecks		Data is required for this field
CRT0101	Content	ItemRole		Data is required for this field
CRT0101	Content	ItemValueListRefs		Data is required for this field
CRT0101	Content	MDVLeaf		Data is required for this field
CRT0101	Content	MDVLeafTitles		Data is required for this field
CRT0101	Content	MUTranslatedText		Data is required for this field
CRT0101	Content	MeasurementUnits		Data is required for this field
CRT0101	Content	MetaDataVersion		Data is required for this field
CRT0101	Content	Presentation		Data is required for this field
CRT0101	Content	ProtocolEventRefs		Data is required for this field
CRT0101	Content	RCErrorTranslatedText		Data is required for this field
CRT0101	Content	Study		Data is required for this field
CRT0101	Content	StudyEventDefs		Data is required for this field
CRT0101	Content	StudyEventFormRefs		Data is required for this field
CRT0101	Content	SupplementalDocs		Data is required for this field
CRT0101	Content	ValueListItemRefs		Data is required for this field
CRT0101	Content	ValueLists		Data is required for this field
CRT0106	Content	CLItemDecodeTranslatedText	lang	The data in the &_cstparm1 field is improperly constructed . Must be in the form [A-Za-z-0-9]
CRT0106	Content	ItemQuestionTranslatedText	lang	The data in the &_cstparm1 field is improperly constructed . Must be in the form [A-Za-z-0-9]
CRT0106	Content	MUTtranslatedText	lang	The data in the &_cstparm1 field is improperly constructed . Must be in the form [A-Za-z-0-9]
CRT0106	Content	Presentation	lang	The data in the &_cstparm1 field is improperly constructed . Must be in the form [A-Za-z-0-9]
CRT0106	Content	RCErrorTranslatedText	lang	The data in the &_cstparm1 field is improperly constructed . Must be in the form [A-Za-z-0-9]
CRT0107	Content	FormDefArchLayouts	PdfFileName	The data in the &_cstparm1 field is an improperly constructed filename. Must be in the form [A-Za-z0-9_.]
CRT0108	Content	ItemDefs	SASFieldName SDSVarName	The data in the &_cstparm1 field is an improperly constructed SAS name. Must be in the form [A-Za-z0-9_]
CRT0108	Content	ItemGroupDefs	SASDatasetName	The data in the &_cstparm1 field is an improperly constructed SAS name. Must be in the form [A-Za-z0-9_]

checkid	checktype	tablescope	columnscope	messagetext
CRT0109	Content	CodeLists	SASFormatName	The data in the &_cstparm1 field is an improperly constructed SAS format name. Must be in the form [($)A-Za-z0-9_]
CRT0110	Content	[AnnotatedCRFs] [MDVLeaf]	[AnnotatedCRFs.leafID] [MDVLeaf.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[AnnotatedCRFs] [MetaDataVersion]	[AnnotatedCRFs.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[CLItemDecodeTranslatedText] [CodeListItems]	[CLItemDecodeTranslatedText.FK_CodeListItems] [CodeListItems.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[CodeListItems] [CodeLists]	[CodeListItems.FK_CodeLists] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[CodeLists] [MetaDataVersion]	[CodeLists.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ComputationMethods] [MetaDataVersion]	[ComputationMethods.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ExternalCodeLists] [CodeLists]	[ExternalCodeLists.FK_CodeLists] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[FormDefArchLayouts] [FormDefs]	[FormDefArchLayouts.FK_FormDefs] [FormDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[FormDefArchLayouts] [Presentation]	[FormDefArchLayouts.PresentationOID] [Presentation.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[FormDefItemGroupRefs] [FormDefs]	[FormDefItemGroupRefs.FK_FormDefs] [FormDefs.OI	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
			D]	
CRT0110	Content	[FormDefItemGroupRefs] [ItemGroupDefs]	[FormDefItemGroupRefs.ItemGroupOID] [ItemGroupDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[FormDefs] [MetaDataVersion]	[FormDefs.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ImputationMethods] [MetaDataVersion]	[ImputationMethods.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemAliases] [ItemDefs]	[ItemAliases.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemDefs] [CodeLists]	[ItemDefs.CodeListRef] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemDefs] [ComputationMethods]	[ItemDefs.ComputationMethodOID] [ComputationMethods.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemDefs] [MetaDataVersion]	[ItemDefs.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemGroupAliases] [ItemGroupDefs]	[ItemGroupAliases.FK_ItemGroupDefs] [ItemGroupDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemGroupDefItemRefs] [CodeLists]	[ItemGroupDefItemRefs.RoleCodeListOID] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemGroupDefItemRefs] [ImputationMethods]	[ItemGroupDefItemRefs.ImputationMethodOID] [ImputationMethods.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[ItemGroupDefItemRefs] [ItemDefs]	[ItemGroupDefItemRefs.ItemOID] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemGroupDefItemRefs] [ItemGroupDefs]	[ItemGroupDefItemRefs.FK_ItemGroupDefs] [ItemGroupDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemGroupDefs] [MetaDataVersion]	[ItemGroupDefs.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemGroupLeafTitles] [ItemGroupLeaf]	[ItemGroupLeafTitles.FK_ItemGroupLeaf] [ItemGroupLeaf.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemGroupLeaf] [ItemGroupDefs]	[ItemGroupLeaf.FK_ItemGroupDefs] [ItemGroupDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemMURefs] [ItemDefs]	[ItemMURefs.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemMURefs] [MeasurementUnits]	[ItemMURefs.MeasurementUnitOID] [MeasurementUnits.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemQuestionExternal] [ItemDefs]	[ItemQuestionExternal.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemQuestionTranslatedText] [ItemDefs]	[ItemQuestionTranslatedText.FK_ItemDefs] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemRangeCheckValues] [ItemRangeChecks]	[ItemRangeCheckValues.FK_ItemRangeChecks] [ItemRangeChecks.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemRangeChecks]	[ItemRangeCh	The foreign key &_cstparm1 does not

checkid	checktype	tablescope	columnscope	messagetext
		[ItemDefs]	ecks.FK_Item Defs] [ItemDefs.OI D]	have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemRangeChecks] [MeasurementUnits]	[ItemRangeCh ecks.MURefOI D] [Measurement Units.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemRole] [ItemDefs]	[ItemRole.FK_ ItemDefs] [ItemDefs.OI D]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemValueListRefs] [ItemDefs]	[ItemValueLis tRefs.FK_Ite mDefs] [ItemDefs.OI D]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ItemValueListRefs] [ValueLists]	[ItemValueLis tRefs.ValueLi stOID] [ValueLists.OI D]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[MDVLeafTitles] [MDVLeaf]	[MDVLeafTitl es.FK_MDVLe af] [MDVLeaf.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[MDVLeaf] [MetaDataVersion]	[MDVLeaf.FK _MetaDataVe rsion] [MetaDataVer sion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[MUTranslatedText] [MeasurementUnits]	[MUTranslate dText.FK_Me asurementUni ts] [Measurement Units.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[MeasurementUnits] [Study]	[Measurement Units.FK_Stu dy] [Study.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[MetaDataVersion] [Study]	[MetaDataVer sion.FK_Stud y] [Study.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[Presentation] [MetaDataVersion]	[Presentation. FK_MetaData Version] [MetaDataVer sion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ProtocolEventRefs] [MetaDataVersion]	[ProtocolEven tRefs.FK_Met aDataVersion]	The foreign key &_cstparm1 does not have a corresponding value in the

checkid	checktype	tablescope	columnscope	messagetext
			[MetaDataVersion.OID]	target data set &_cstparm2
CRT0110	Content	[ProtocolEventRefs] [StudyEventDefs]	[ProtocolEventRefs.StudyEventOID] [StudyEventDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[RCErrorTranslatedText] [ItemRangeChecks]	[RCErrorTranslatedText.FK_ItemRangeChecks] [ItemRangeChecks.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[StudyEventDefs] [MetaDataVersion]	[StudyEventDefs.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[StudyEventFormRefs] [FormDefs]	[StudyEventFormRefs.FormOID] [FormDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[StudyEventFormRefs] [StudyEventDefs]	[StudyEventFormRefs.FK_StudyEventDefs] [StudyEventDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[Study] [DefineDocument]	[Study.FK_DefineDocument] [DefineDocument.FileOID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[SupplementalDocs] [MDVLeaf]	[SupplementalDocs.leafID] [MDVLeaf.ID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[SupplementalDocs] [MetaDataVersion]	[SupplementalDocs.FK_MetaDataVersion] [MetaDataVersion.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ValueListItemRefs] [CodeLists]	[ValueListItemRefs.RoleCodeListOID] [CodeLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ValueListItemRefs] [ImputationMethods]	[ValueListItemRefs.ImputationMethodOID] [ImputationMethods.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
CRT0110	Content	[ValueListItemRefs] [ItemDefs]	[ValueListItemRefs.ItemOID] [ItemDefs.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ValueListItemRefs] [ValueLists]	[ValueListItemRefs.FK_ValueLists] [ValueLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0110	Content	[ValueLists] [ValueLists]	[ValueLists.FK_MetaDataVersion] [ValueLists.OID]	The foreign key &_cstparm1 does not have a corresponding value in the target data set &_cstparm2
CRT0111	Content	[ItemGroupDefs] [ItemGroupDefItemRefs]	[ItemGroupDefs.OID] [ItemGroupDefItemRefs.FK_ItemGroupDefs]	Each distinct value of &_cstparm1 must have a corresponding value in the target data set &_cstparm2
CRT0111	Content	[ItemRangeChecks] [ItemRangeCheckValues]	[ItemRangeChecks.OID] [ItemRangeCheckValues.FK_ItemRangeChecks]	Each distinct value of &_cstparm1 must have a corresponding value in the target data set &_cstparm2
CRT0111	Content	[MDVLeaf] [MDVLeafTitles]	[MDVLeaf.ID] [MDVLeafTitles.FK_MDVLeaf]	Each distinct value of &_cstparm1 must have a corresponding value in the target data set &_cstparm2
CRT0112	Content	[CodeListItems] [ExternalCodeLists]	[CodeListItems.FK_CodeLists] [ExternalCodeLists.FK_CodeLists]	No value in &_cstparm1 can be equal to any value in &_cstparm2
CRT0112	Content	[DefineDocument] [ItemGroupLeaf]	[DefineDocument.ID] [ItemGroupLeaf.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2
CRT0112	Content	[DefineDocument] [MDVLeaf]	[DefineDocument.ID] [MDVLeaf.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2
CRT0112	Content	[ExternalCodeLists] [CodeListItems]	[ExternalCodeLists.FK_CodeLists] [CodeListItems.FK_CodeLists]	No value in &_cstparm1 can be equal to any value in &_cstparm2
CRT0112	Content	[ItemGroupLeaf] [DefineDocument]	[ItemGroupLeaf.ID] [DefineDocum	No value in &_cstparm1 can be equal to any value in &_cstparm2

checkid	checktype	tablescope	columnscope	messagetext
			ent.ID]	
CRT0112	Content	[ItemGroupLeaf] [MDVLeaf]	[ItemGroupLeaf.ID] [MDVLeaf.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2
CRT0112	Content	[MDVLeaf] [DefineDocument]	[MDVLeaf.ID] [DefineDocument.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2
CRT0112	Content	[MDVLeaf] [ItemGroupLeaf]	[MDVLeaf.ID] [ItemGroupLeaf.ID]	No value in &_cstparm1 can be equal to any value in &_cstparm2
CRT0113	Content	CodeListItems	[CodedValue] [FK_CodeLists]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0113	Content	FormDefItemGroupRefs	[ItemGroupOID] [FK_FormDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0113	Content	FormDefItemGroupRefs	[OrderNumber] [FK_FormDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0113	Content	ItemGroupDefItemRefs	[ItemOID] [FK_ItemGroupDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0113	Content	ItemGroupDefItemRefs	[OrderNumber] [FK_ItemGroupDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0113	Content	ProtocolEventRefs	[OrderNumber] [FK_MetaDataVersion]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0113	Content	ProtocolEventRefs	[StudyEventOID] [FK_MetaDataVersion]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0113	Content	StudyEventFormRefs	[FormOID] [FK_StudyEventDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0113	Content	StudyEventFormRefs	[OrderNumber] [FK_StudyEventDefs]	Foreign key variables cannot have multiple values in &_cstparm2. They must be unique
CRT0114	Content	_ALL_		Coded value is either incorrect, missing, or wrong case.

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **`suggest@sas.com`**.