PharmaSUG 2012 - Paper DS20

An Innovative ADaM Programming Tool for FDA Submission

Xiangchen (Bob) Cui, Vertex Pharmaceuticals, Cambridge, MA
Min Chen, Vertex Pharmaceuticals, Cambridge, MA
Tathabbai Pakalapati, Vertex Pharmaceuticals, Cambridge, MA

ABSTRACT

It is a good practice to include data definition tables (define.xml) and a reviewer's guide along with ADaM datasets to minimize the time to familiarize with submitted clinical data and expedite the approval process by FDA reviewers. It is important to ensure consistency in metadata among data definition tables, reviewer's guide and ADaM datasets. This paper describes automated ADaM Programming Tool, consisting of six SAS macros, to streamline the process of creating programming specification, compliance checking of specifications with FDA and CDISC requirements, deriving ADaM datasets and generating define.xml and a reviewer's guide. The tool also automates the processes of version control of specifications, consistency checking of controlled terminology and value level metadata between ADaM and define files, detection of empty variables in ADaM datasets, preparation of batch files, and addition of core variables to both all ADaM datasets and define.xml at final run thereby achieving accuracy and efficiency.

INTRODUCTION

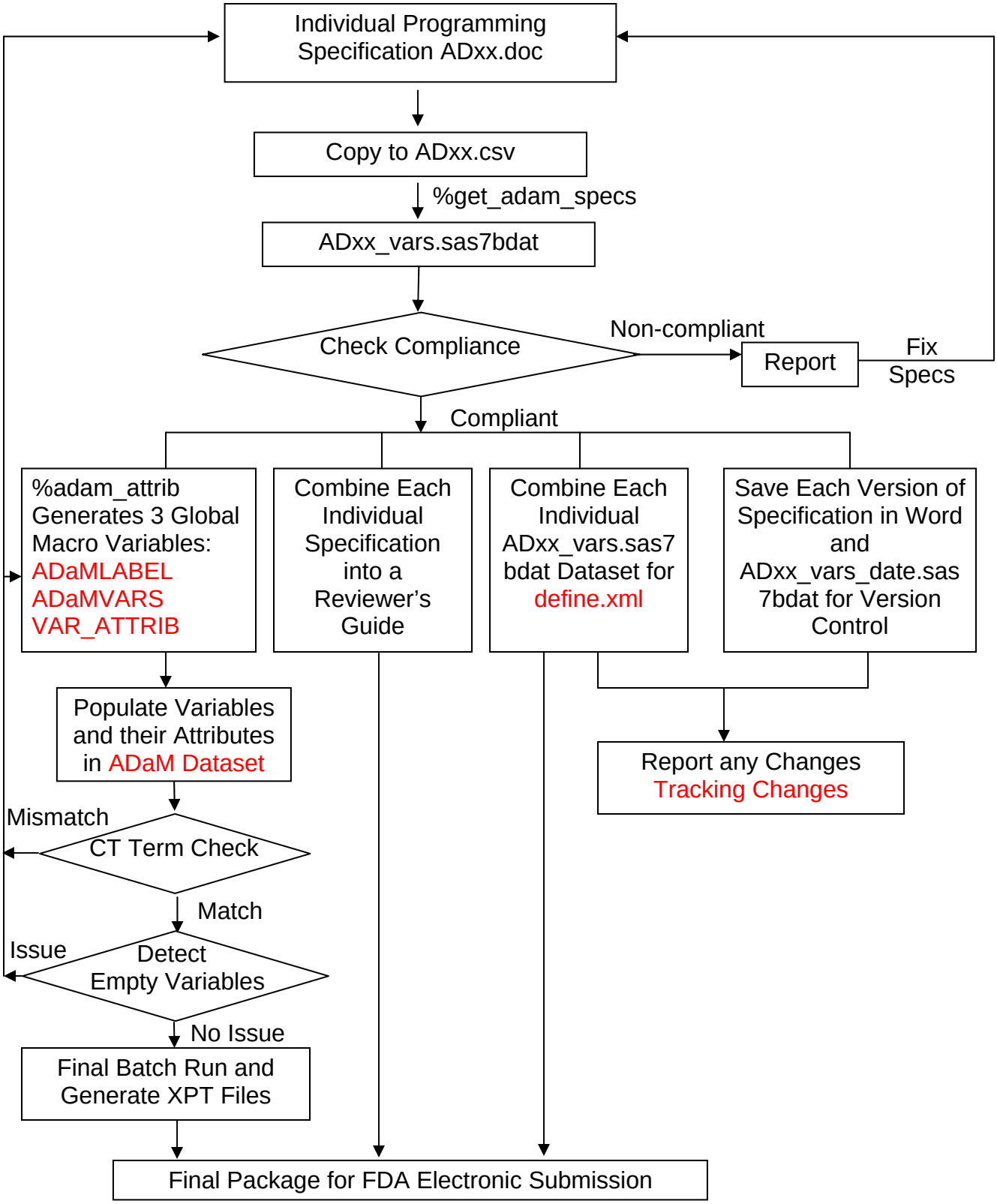
ADaM programming is an important and challenging part in biometrics deliverable life cycle. In addition to ADaM datasets, sponsor needs to submit analysis reviewer guide and data definition tables (define.xml) to FDA reviewers. Considering stringent timelines and frequent changes in statistical analysis plan it is always essential for a sponsor/vendor to have an efficient ADaM processing mechanism to deliver data and supplemental documents with high quality and accuracy. This paper presents an innovative ADaM Programming Tool to streamline the whole process of creation of programming specification, compliance checking of metadata against FDA and CDISC requirements, ADaM derivation, version control, tracking changes in specifications and generation of reviewer guide and data definition tables. The tool consists of 6 macros: %**get_adam_specs**, %**adam_attrib**, %**ctlist_checking**, %**empty_var_checking**, %**get_adam_specs_final_call**, %**get_batch _file,** and there are 10 step processes in the programming tool to generate a complete ADaM package for FDA electronic submission.

Ten automations provided by the ADaM Programming Tool are summarized below and illustrated in the form of flowchart in Display1.

1. **Automatic compliance checking of metadata against CDISC standards**
2. **Automatic version control**
3. **Automatic track changes in analysis programming specification**
4. **Automatic define.xml generation**
5. **Automatic generation of ADaM dataset label and variable attributes in the form of macro variables**
6. **Automatic addition of core variables to both define.xml and ADaM datasets in the final run**
7. **Automatic consistency checking of controlled terminology and value level metadata between ADaM datasets and programming specification**
8. **Automatic detection of empty variables in ADaM datasets**
9. **Automatic preparation of SAS scripts for final run of all ADaM specifications and combination of all ADaM specifications into one Word document in a specified order**
10. **Automatic batch file preparation for the final run of ADaM datasets**

Each step above is explained in detail in a separate section with real examples used in our FDA submission process. Lastly, the paper tries to showcase the advantage of using the suggested ADaM Programming Tool to achieve high operational efficiency.

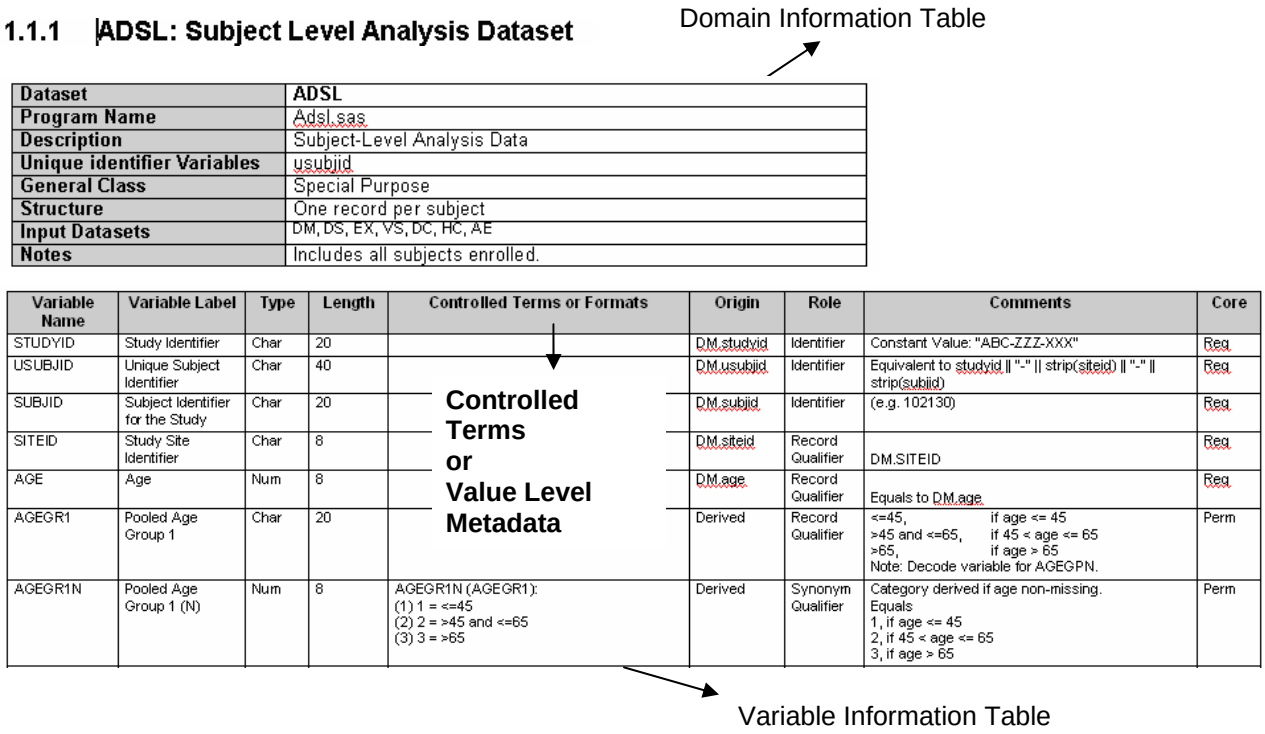
Display 1 shows the process flow.



Display 1. Overall of Process Flow

AN INTRODUCTION OF MODULARIZED WORD® SPECIFICATION FOR ADAM

An individual programming specification for ADaM in MS Word® format facilitates programmers and statisticians to review and communicate derivation rules among them, as well as track the change. Display 2 shows the snapshot of an ADaM programming specification. The specification for each domain is composed of three modules: domain information table, variable information table, and an optional appendix or notes for a complex algorithm or derivation rules. Information in the first two modules is the core for this ADaM programming tool and will be used for 10 automations listed in Introduction section.



1.1.1 ADSL: Subject Level Analysis Dataset

Domain Information Table

Dataset	**ADSL**
Program Name	Adsl.sas
Description	Subject-Level Analysis Data
Unique identifier Variables	usubjid
General Class	Special Purpose
Structure	One record per subject
Input Datasets	DM, DS, EX, VS, DC, HC, AE
Notes	Includes all subjects enrolled.

Variable Name	Variable Label	Type	Length	Controlled Terms or Formats	Origin	Role	Comments	Core
STUDYID	Study Identifier	Char	20		DM.studyid	Identifier	Constant Value: "ABC-ZZZ-XXX"	Req
USUBJID	Unique Subject Identifier	Char	40		DM.usubjid	Identifier	Equivalent to studyid || "-" || strip(siteid) || "-" || strip(subjid)	Req
SUBJID	Subject Identifier for the Study	Char	20		DM.subjid	Identifier	(e.g. 102130)	Req
SITEID	Study Site Identifier	Char	8		DM.siteid	Record Qualifier	DM.SITEID	Req
AGE	Age	Num	8		DM.age	Record Qualifier	Equals to DM.age	Req
AGEGR1	Pooled Age Group 1	Char	20		Derived	Record Qualifier	<=45, if age <= 45 >45 and <=65, if 45 < age <= 65 >65, if age > 65 Note: Decode variable for AGEGPN.	Perm
AGEGR1N	Pooled Age Group 1 (N)	Num	8	AGEGR1N (AGEGR1): (1) 1 = <=45 (2) 2 = >45 and <=65 (3) 3 = >65	Derived	Synonym Qualifier	Category derived if age non-missing. Equals 1, if age <= 45 2, if 45 < age <= 65 3, if age > 65	Perm

Controlled Terms or Value Level Metadata

Variable Information Table

Display 2. Individual Programming Specifications in Word® Format

In the domain information table, description of the domain will serve as the label of ADaM dataset; in the variable information table, the variable name, label, type, and the length will define the variable attributes of ADaM dataset. 'Controlled Terms or Formats' Column specifies controlled terminologies for necessary variables and defines formats for date/time variables which will also be presented in define.xml.

The contents of the Word programming specification are copied to a comma delimited document, as shown in Display 3, to be imported to a SAS dataset.

	A	B	C	D	E	F	G	H	I
1	1.1.1	ADSL: Subject Level Analysis Dataset							
2									
3	Dataset	ADSL							
4	Program Name	Adsl.sas							
5	Description	Subject-Level Analysis Data							
6	Unique identifier	usubjid							
7	General Class	Special Purpose							
8	Structure	One record per subject							
9	Input Datasets	DM, DS, EX, VS, DC, HC, AE							
10	Notes	Includes all subjects enrolled.							
11									
12	Variable Name	Variable Label	Type	Length	Controlled Terms or Formats	Origin	Role	Comments	Core
13	STUDYID	Study Identifier	Char	20		DM.studyid	Identifier	Constant Value: "ABC-ZZZ-xxx"	Req
14	USUBJID	Unique Subject Identifier	Char	40		DM.usubjid	Identifier	Equivalent to studyid || "-" || strip(siteid)	Req
15	SUBJID	Subject Identifier for the Study	Char	20		DM.subjid	Identifier	(e.g. 102130)	Req
16	SITEID	Study Site Identifier	Char	8		DM.siteid	Record Qualifier	DM.SITEID	Req
17	AGE	Age	Num	8		DM.age	Record Qualifier	Equals to DM.age	Req
18	AGEGR1	Pooled Age Group 1	Char	20		Derived	Record Qualifier	<=45, if age <= 45	Perm
19								>45 and <=65, if 45 < age <= 65	
20								>65, if age > 65	
21								Note: Decode variable for AGEGPN.	
22	AGEGR1N	Pooled Age Group 1 (N)	Num	8	AGEGR1N (AGEGR1):	Derived	Synonym Qualifie	Category derived if age non-missing.	Perm
23					(1) 1 = <=45			Equals	
24					(2) 2 = >45 and <=65			1, if age <= 45	
25					(3) 3 = >65			2, if 45 < age <= 65	
26								3, if age > 65	

Display 3. Individual Programming Specification in Comma-Delimited CSV Format

AUTOMATION 1: COMPLIANCE CHECKING WITH FDA SUBMISSION REQUIREMENTS AND CDSIC ADAM PROGRAMMING REQUIREMENTS FOR MODULARIZED ADAM SPECIFICATIONS

GUIDELINE FOR WRITING ADAM SPECIFICATION AND COMPLIANCE CHECKING RULES

As shown in Display 2, each domain specification is modularized to facilitate the programming. CDISC ADaM Implementation Guideline clearly defines ADSL and Basic Data Structure (BDS) data. CDISC ADaM validation checks define associated validation checks to ensure high quality in submitted analysis datasets. Our programming specifications are checked against these validation rules so that the submitted analysis dataset metadata will be compliant to CDISC ADaM Guideline. Our macro based approach also checks the compliance of domain information, the compliance between domain and variable information, the requirements or key words for each column in programming specifications, and the existence of decoded variables defined in 'Controlled Terms or Formats' Column. The guideline for writing ADaM programming specification and the compliance checking rules, which are objective and programmable, for ADaM metadata are listed in the Appendix 1.

A MACRO TO RETRIEVE INFORMATION FROM SPECIFICATION AND COMPLIANCE CHECKING

A macro %**get_adam_specs** is used to read the information from the individual domain programming specification in CSV format, retrieve the useful domain information and variable information based on the standard structure of the given specification, performs ADaM compliance checking with CDISC requirements and FDA submission requirements, and outputs non-compliance reports if any. SAS datasets with ADaM specification information will be generated only when the specifications are compliant with the rules predefined in Appendix 1. Other functions of this macro will be introduced in subsequent sections.

The macro call is shown as follows.

```
%macro get_adam_specs(indir       =,    /* path of input ADaM specs        */
                      Specsnm     =,    /* specs name, e.g. adsl.csv       */
                      Outdir      =,    /* path for output data/reports    */
                      Newdtnm     =,    /* dataset name for new specs      */
                      Runorder    =999, /* Run order for a specific domain */
                      track_specs =N,   /* Y: activate tracking change     */
                      olddir      =,    /* path of old specs               */
                      predtnm     =,    /* dataset name for old specs      */
                      generate_xml=N,   /* Y: generate define.xml          */
                      xmldir      =,    /* path for define.xml             */
                      final_run   =N    /* Y: add core vars to define.xml  */
                     );
```

Where,

INDIR: Full Path for ADaM programming specification.
SPECSNM: Name of ADaM programming specification.
OUTDIR: Full Path for output reports or SAS dataset which contains ADaM specs information.
NEWDTNM: A valid SAS dataset name for SAS dataset containing current specs information.
RUNORDER: A valid numeral, defining the order for a specific domain to run (in the final run).
TRACK_SPECS: Flag for audit trail. If TRACK_SPECS is assigned to Y, the macro will compare dataset for the new specs (NEWDTNM) at working folder (OUTDIR) with dataset for the old specs (PREDTNM) in the history folder (OLDDIR).
OLDDIR: Full Path for history folder of SAS dataset containing old specs information.
PREDTNM: A valid SAS dataset name for SAS dataset containing old specs information.
GENERATE_XML: Flag for define.xml generation. If GENERATE_XML is assigned to Y, the macro will generate define.xml for all the existing SAS datasets under the working (OUTDIR) folder.
XMLDIR: Full Path for define.xml.
FINAL_RUN: Flag for Final Run. If FINAL_RUN is assigned to Y, the macro will add core variables to both the final ADaM datasets and define.xml

If one of the compliance checking rules is not satisfied, non-compliance reports in RTF format will be reported. Displays 4-8 show the compliance checking reports for unfulfilled requirements.

The Following Domain Information Should be Provided!

Domain	Error Type
ADSL	Missing Domain General Class
	General Class key word: Special Purpose/Interventions/Events/Findings

Display 4. Non-Compliance Report for Domain Information Table

The Following Variables with Non-compliance Specs.!

Variable	Error Type
RACE	ROLE key word: Identifier/Topic/Timing/Grouping Qualifier/Result Qualifier/Synonym Qualifier/Record Qualifier/Variable Qualifier/Selection/Analysis
VTXGTYPE	VAR Label > 40 Chars
	ROLE key word: Identifier/Topic/Timing/Grouping Qualifier/Result Qualifier/Synonym Qualifier/Record Qualifier/Variable Qualifier/Selection/Analysis
AGE	ROLE key word: Identifier/Topic/Timing/Grouping Qualifier/Result Qualifier/Synonym Qualifier/Record Qualifier/Variable Qualifier/Selection/Analysis
SITEID	ROLE key word: Identifier/Topic/Timing/Grouping Qualifier/Result Qualifier/Synonym Qualifier/Record Qualifier/Variable Qualifier/Selection/Analysis
O_TIMDIAG	VAR Name > 8 Chars
DISCFN	DISCFN variable is present but DISCFL variable is not present
AGEU	AGEU variable is not in the dataset ADSL
ARM	ARM variable is not in the dataset ADSL
*FL	No variable that ends in FL is in the dataset ADSL

Display 5. Non-Compliance Report for Variable Information Table

The Following Variables with Duplicates Specs.!

Variable	Variable Number
SEX	9
	92

Display 6. Non-Compliance Report for Duplicate Variables

The Following Key Variables are Defined in Domain Information Table, but Not Defined in Variable Information Table!

Variable	Error Type
TRTCD	Unique Identified Variable TRTCD is Defined in Domain Information Table, but Not Defined in Variable Information Table

Display 7. Non-Compliance Report for Inconsistently Defined Key Variables

The Following Variables Do Not Have Decoded Variables Defined in Specs.!

Variable	Error Type
AVISITN	Variable AVISITN dose not have a decoded variable AVISIT defined in the specification ADLB.DOC

Display 8. Non-Compliance Report for Decoded Variables

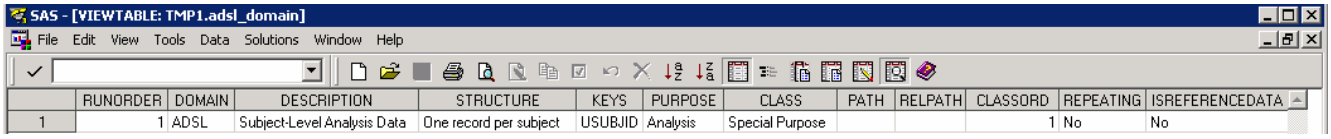
COMPARISON WITH OPENCDISC VALIDATION

Compared with OpenCDISC Validation which normally occurs at the very end of ADaM programming activities, the compliance checking by the tool focuses on the ADaM metadata now, including variable presence. The ADaM programming specification is the ONLY source needed in the macro, which makes it possible to perform compliance checking and finalize the ADaM metadata at the very early stage even before actual ADaM datasets are generated. Specifically, the checking rules such as compliance checking of domain information, compliance checking between domain and variable information, key words checking for each column in programming specifications, and existence checking of decoded variables in 'Controlled Terms or Formats' Column, are not defined in CDISC ADaM Validation Checks V1.1, but uniquely defined and applied in our approach instead. The tool will be further developed to check ADaM datasets for compliance with CDISC models, in addition to ADaM metadata. More checking rules will be incorporated into the tool as the tool is being fully developed regarding compliance checking of metadata against CDISC standards.

OUTPUT OF SAS DATASETS CONTAINING ADAM SPECIFICATION INFORMATION

Not until all issues about the above-mentioned compliance checking are resolved, will the SAS datasets containing ADaM specs information be output. Compared with OpenCDISC Validation which occurs at the very end of ADaM programming activities, the method in this paper will ensure the resolution of non-compliance with CDISC and FDA submission requirements, finalize the ADaM specification at the very early stage in the programming cycle even before actual ADaM datasets are generated, and avoid the repetitive work to revise the ADaM data structure after ADaM Derivation.

Display 9 shows a dataset named **ADXX_DOMAIN** containing the domain information for the individual ADaM dataset after all compliance checking are passed, and Display 10 shows a dataset named **ADXX_VARS** containing the attributes of the variables. They define individual ADaM metadata and will be used in the subsequent ADaM programming activities.



	RUNORDER	DOMAIN	DESCRIPTION	STRUCTURE	KEYS	PURPOSE	CLASS	PATH	RELPATH	CLASSORD	REPEATING	ISREFERENCEDATA
1	1	ADSL	Subject-Level Analysis Data	One record per subject	USUBJID	Analysis	Special Purpose			1	No	No

Display 9. Dataset Containing the Domain Information of an Individual ADaM Domain

SAS - [VIEWTABLE: TMP1.adsl_vars]

File Edit View Tools Data Solutions Window Help

	DOMAIN	VARNUM	VARIABLE	LABEL	TYPE	DATATYPE	LENGTH	ORIGIN	TERM	CODELIST	CONTROLLED_TERMINOLOGY	ROLE	COMMENT	CORE	MANDATOR
99	ADSL	9	SEX	Sex	Char	text	2	DM.sex				Record Qualifier	DM.SEX	Req	Yes
100	ADSL	10	SEXN	Sex (N)	Num	float	8	Derived	SEXN	(1) 1 = M (2) 2 = F	SEXN (SEX): (1) 1 = M (2) 2 = F	Synonym Qualifier	Equals, 1, if sex = M; 2, if sex = F	Req	Yes
101	ADSL	4	SITEID	Study Site Identifier	Char	text	8	DM.siteid				Record Qualifier	DM.SITEID	Req	Yes
102	ADSL	58	STDYEDT	Study End Date	Num	float	8	Derived	YYMMDD10.		YYMMDD10.	Timing	Max (DS.DSSTDTC where DSSCAT='STUDY TERMINATION', SUPPDS.QVAL where QNAM='LASTVDT')	Perm	No

Display 10. Dataset Containing Variables Information of an Individual ADaM Domain

Display 11 shows a SAS dataset containing the domain information for all existing ADaM domains named **ALL_DOMAINS**, and Display 12 shows a SAS dataset containing the attributes of the variables for all existing ADaM domains named **ALL_VARS**. They are generated cumulatively each time when individual ADaM specification programs are run, and output for preparing define.xml and batch file. If a particular ADaM specification program is run several times, the information retrieved from the most recent call will replace the one in the previous run.

SAS - [VIEWTABLE: Tmp1.All_domains]

File Edit View Tools Data Solutions Window Help

	RUNORDER	DOMAIN	DESCRIPTION	STRUCTURE	KEYS	PURPOSE	CLASS	PATH	RELPATH	CLASSORD	REPEATING	ISREFERENCED
1	1	ADSL	Subject-Level Analysis Data	One record per subject	USUBJID	Analysis	Special Purpose			1	No	No
2	2	ADAE	Analysis Dataset for Adverse Events	One record per subject per each AE recorded in SDTM AE domain	STUDYID, USUBJID, AEDECOD, AESDT	Analysis	Events			1	Yes	No
3	3	ADCD	CD4 Data Set	One record per test per time point per subject	USUBJID, APHASEN, AVISITN, CDTESTCD, CDDTC, CDORRES	Analysis	Findings			1	Yes	No
4	4	ADCM	Concomitant Medication Data	One record per medication intervention episode per subject	STUDYID, USUBJID, CMTRT, CMSDT	Analysis	Interventions			1	Yes	No
5	5	ADEG	Electrocardiogram Analysis Data Set	One record per eg test per time point per subject. In case of multiple measurements in a single visit window, we will keep the multiple measurements but 'AVERAGE' will be also created for the multiple	USUBJID, PARCAT1, PARAMCD, APHASEN, AVISITN, EGDTC	Analysis	Findings			1	Yes	No

Display 11. Dataset Containing the Domain Information of All ADaM Domains

SAS - [VIEWTABLE: Tmp1.All_vars]

File Edit View Tools Data Solutions Window Help

	DOMAIN	VARNUM	VARIABLE	LABEL	TYPE	DATATYPE	LENGTH	ORIGIN	TERM	CODELIST	CONTROLLED_TERMINOLOGY	ROLE	COMMENT	CORE	MANDATOR
142	ADSL	68	VTRTEDT	Treatment Phase End Date	Num	float	8	Derived	YYMMDD10.		YYMMDD10.	Timing	Equals TRTEDT + 14.	Perm	No
143	ADSL	22	WEIGHTBL	Baseline Weight (kg)	Num	float	8	VS.vsstres				Record Qualifier	Equals vs.vsstresn for vs.vstestcd = 'WEIGHT'. Baseline value is the assessment at the Day 1 visit. If none, the latest available measurement from the screening period will be used, including unscheduled visits before Day 1.	Req	Yes
144	ADAE	106	ADURN	Numeric AE Duration (days)	Num	float	8	Derived				Timing	If both AESDT and AEEDT are not missing then; ADURN = AEEDT-AESDT+1; Else if either AESDT or AEEDT missing then ADURN is missing.	Perm	No
145	ADAE	26	AEACN	Action Taken with Study Treatment	Char	text	40	AE.aeacn				Record Qualifier	Equals AE.aeacn	Cond	No
146	ADAE	30	AEACNHAA	Action Taken with HAART	Char	text	200	SUPPAE.	AEACNF	(1) DOSE NOT CHANGED (2) DOSE REDUCED (3) DRUG INTERRUPTED (4) DRUG WITHDRAWN (5) NOT APPLICABLE	AEACNF: (1) DOSE NOT CHANGED (2) DOSE REDUCED (3) DRUG INTERRUPTED (4) DRUG WITHDRAWN (5) NOT APPLICABLE	Record Qualifier	Derived from MedDRA dictionary; Equals SUPPAE.qval when SUPPAE.QNAM = 'AEACNHAA'	Perm	No

Display 12. Dataset Containing Variable Information of All ADaM Domains

AUTOMATION 2: VERSION CONTROL

Version control of ADaM programming specifications can be achieved by the tool. In addition to outputting SAS datasets containing ADaM specs information to a study folder &OUTDIR, the macro %**get_adam_specs** can also store both word version and SAS dataset of the programming specification with time stamp in a study subfolder, named as \history. This function is performed after passing the compliance checking.

The word files of the specifications with different time stamps are stored for version control purpose. The SAS dataset of specification with a time stamp can serve as an input to automatically capture the changes of the programming specifications**.** The traceability can be achieved with the storage of the previous version of specifications.

An example of version control is shown in Display 13. If needed, time can be added into the time stamp of word documents in addition to date.

Name	Date modified	Type	Size
ADSL_20110913	9/12/2011 9:08 AM	Microsoft Word Document	341 KB
ADSL_20110915	9/15/2011 9:51 AM	Microsoft Word Document	340 KB
ADSL_20110919	9/19/2011 4:11 PM	Microsoft Word Document	346 KB
ADSL_20111007	9/22/2011 4:35 PM	Microsoft Word Document	342 KB
ADSL_20111018	10/14/2011 9:41 AM	Microsoft Word Document	337 KB
adsl_vars_20110913	9/13/2011 11:15 AM	SAS Data Set	1,921 KB
adsl_vars_20110915	9/15/2011 10:18 AM	SAS Data Set	1,921 KB
adsl_vars_20110919	9/19/2011 4:12 PM	SAS Data Set	1,921 KB
adsl_vars_20111007	10/7/2011 10:31 AM	SAS Data Set	1,889 KB
adsl_vars_20111018	10/18/2011 12:50 PM	SAS Data Set	1,873 KB

Display 13. An Example of Version Control for ADSL Specification Document

AUTOMATION 3: TRACK CHANGES

Since derivation rules may be complex and subject to constant change during the whole ADaM programming activities, it is desirable to automatically keep track of different versions of ADaM programming specifications in order to help statisticians and programmers to review the new specifications and facilitate the decision making for the revision. It is more beneficiary for sponsors to keep track of different versions when ADaM programming is outsourced to external vendors.

A MACRO FOR TRACKING CHANGES

Tracking changes function will be activated when macro variable &**track_specs** is set to Y at the invoking of macro %**get_adam_specs**. The macro users can assign any SAS dataset in \history folder as an old version of specifications, to be compared with the current version of specifications. The reports on specifications revisions will be automatically output in RTF formats.

Tracking changes function makes it possible to capture any changes in current specifications with respect to any previous version of specifications as per user request, including variable added, variable deleted, variable attributes revised, variable comments revised, variable origin and/or controlled terminology revised, and variable number of order revised in the new specifications, and thereby facilitates reviewing the new ADaM specification. The Display 14 - 19 show the typical reports of specification changes when tracking change function is triggered.

The following Variables Were Added in the New Version of Specs.!

Variable	Variable Attribute	Comment
ARMCD	ARMCD label='Arm' length=$8	DM.ARMCD

Display 14. An Example Report for Tracking Changes: Adding Variable(s)

The following Variables Were Deleted in the New Version of Specs.!

Variable	Variable Attribute	Comment
AGEC	AGEC label='Enrollment Age Category' length=$40	if . < age <= 45 then; agec = "AGE <= 45 years"; else if 45 < age <= 65 then; agec = "45 < AGE <= 65 years"; else if 65 < age then; agec = "AGE > 65 years";

Display 15. An Example Report for Tracking Changes: Deleting Variable(s)

The following Variables with Attributes Changed!

Variable	Variable Number	Source	Variable Attribute
COHORTN	29	New Specs	COHORTN label='Cohort (N)' length=8
		Old Specs	COHORTN label='Cohort Code' length=8

Display 16. An Example Report for Tracking Changes: Change of Variable Attributes

The following Variables with Comment Changed!

Variable	Variable Number	Source	Comment
RVRFN	104	New Specs	Equals to 1 if rvrfl="Y"; Equals to 0 if rvrfl="N"
		Old Specs	Equals to 1 if rvrfl="Y"; Equals to 0 if rvrfl="N"; Equals to 9 if rvrfl="U"; Equals to 99 if rvrfl="NA"

Display 17. An Example Report for Tracking Changes: Change of Comments

The following Variables with Origin and/or Controlled Terminology/Format Changed!

Variable	Variable Number	Source	Origin	Controlled Term or Format	Codelist
UNDW24FN	108	New Specs	Derived	YESNOFN	(1) 1 = Y (2) 0 = N
		Old Specs	Derived	NYNULLFN	(1) 1 = Y (2) 0 = N (3) 9 = U (4) 99 = NA

Display 18. An Example Report for Tracking Changes: Change of Origin or Controlled Terminology

The following Variables with Variable Number Changed!

Variable	Source	Variable Number
AGE	New Specs	5
	Old Specs	25
AGEU	New Specs	8
	Old Specs	28
ARM	New Specs	34
	Old Specs	15

Display 19. An Example Report for Tracking Changes: Change of Variable Order in the ADaM Dataset

POST-DELIVERY CHANGES AND TRACKING CHANGES

Tracking changes function is indispensable for programming and/or documentation after a Clinical Study Report (CSR) delivery. If there are any changes in the programming specifications post-delivery, the tracking change function will be triggered when invoking macro **%get_adam_specs,** and reports will be generated to capture changes from the last version of specifications. The reports will serve as documentation for audit. For example, if comments column is updated after the delivery due to editorial change, only one report is generated: the report of change of comments, as shown in Display 17.

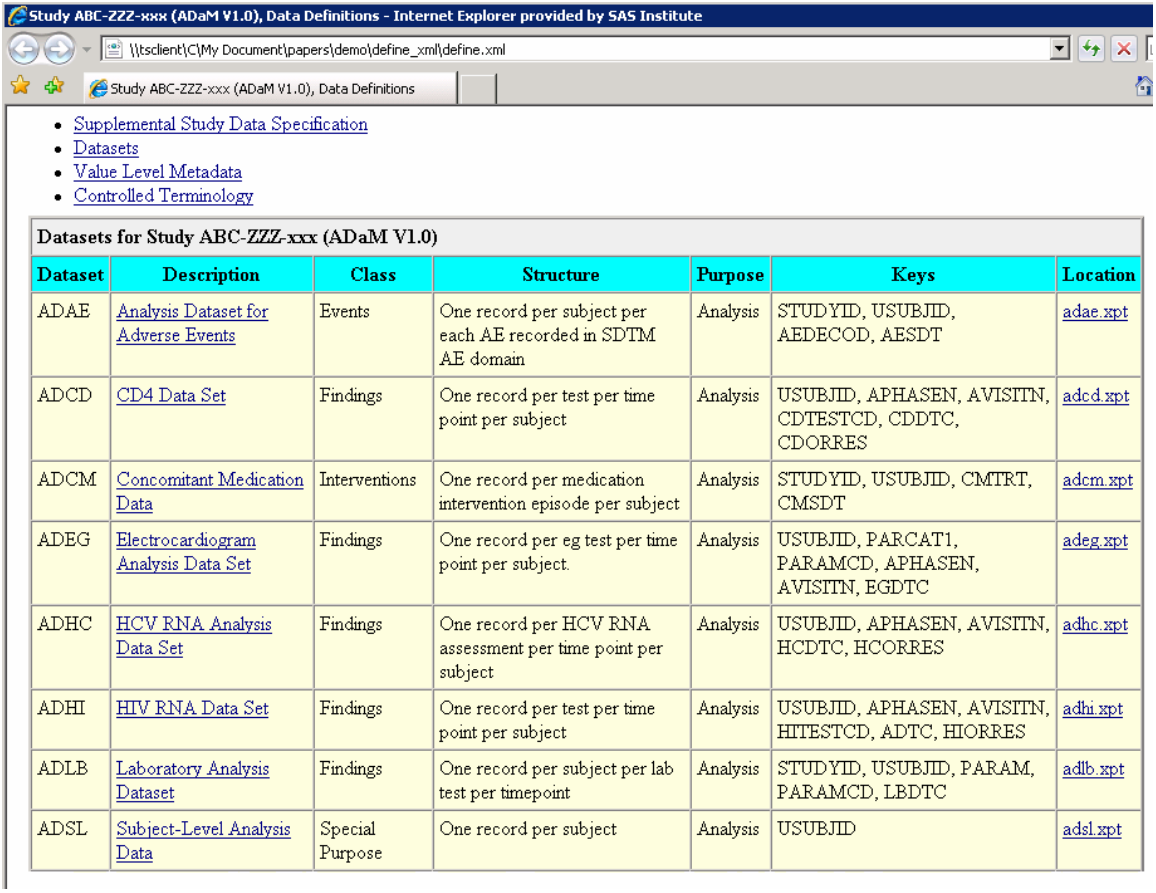
AUTOMATION 4: CREATE DEFINE.XML

Define.xml is used to be generated after the CSR stage for submission purpose only, which is provided for FDA reviewers to familiarize the data and speed up the overall review process. It is desirable if the study statisticians can review metadata earlier in the programming cycle as well. Our ADaM Tool automatically creates define.xml at the same time as CSR stage and thereby makes it possible for the study statisticians to validate the ADaM metadata and provide feedbacks at early stage of ADaM programming activities.

A MACRO FOR CREATING DEFINE.XML

Define.xml generation function will be activated when macro variable &**generate_xml** is set to Y at the invoking of macro %**get_adam_specs**. The generated SAS datasets ALL_DOMAINS and ALL_VARS, which contain domain information and variable information for all domains as shown in Display 11 and 12, respectively, will be output to the domain information spreadsheet, variable information spreadsheet, value level spreadsheet, and controlled terminology spreadsheet for define.xml generation. These spreadsheets will be combined with manually generated study level spreadsheet to create define.xml.

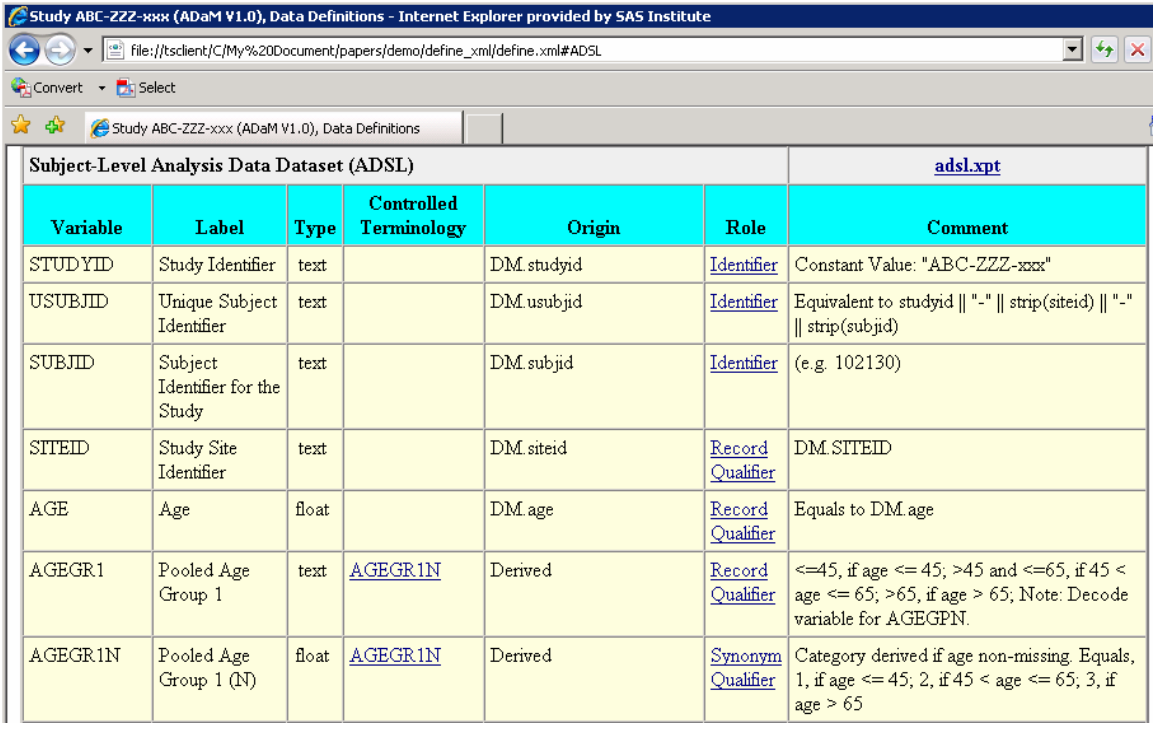
Display 20 shows an example of Table of Contents for define.xml, which describes the domain information. The detailed derivation rules and origin information in the programming specification, in addition to variable attributes, are shown in Display 21. The links to the individual ADaM datasets will not work until ADaM datasets are finalized, QCed, and converted to transport files.



Datasets for Study ABC-ZZZ-xxx (ADaM V1.0)

Dataset	Description	Class	Structure	Purpose	Keys	Location
ADAE	Analysis Dataset for Adverse Events	Events	One record per subject per each AE recorded in SDTM AE domain	Analysis	STUDYID, USUBJID, AEDECOD, AESDT	adae.xpt
ADCD	CD4 Data Set	Findings	One record per test per time point per subject	Analysis	USUBJID, APHASEN, AVISITN, CDTESTCD, CDDTC, CDORRES	adcd.xpt
ADCM	Concomitant Medication Data	Interventions	One record per medication intervention episode per subject	Analysis	STUDYID, USUBJID, CMTRT, CMSDT	adcm.xpt
ADEG	Electrocardiogram Analysis Data Set	Findings	One record per eg test per time point per subject.	Analysis	USUBJID, PARCAT1, PARAMCD, APHASEN, AVISITN, EGDTC	adeg.xpt
ADHC	HCV RNA Analysis Data Set	Findings	One record per HCV RNA assessment per time point per subject	Analysis	USUBJID, APHASEN, AVISITN, HCDTC, HCORRES	adhc.xpt
ADHI	HIV RNA Data Set	Findings	One record per test per time point per subject	Analysis	USUBJID, APHASEN, AVISITN, HITESTCD, ADTC, HIORRES	adhi.xpt
ADLB	Laboratory Analysis Dataset	Findings	One record per subject per lab test per timepoint	Analysis	STUDYID, USUBJID, PARAM, PARAMCD, LBDTC	adlb.xpt
ADSL	Subject-Level Analysis Data	Special Purpose	One record per subject	Analysis	USUBJID	adsl.xpt

Display 20. Table of Contents (TOC) of Sample define.xml - Domain Information



Subject-Level Analysis Data Dataset (ADSL)						adsl.xpt
Variable	**Label**	**Type**	**Controlled Terminology**	**Origin**	**Role**	**Comment**
STUDYID	Study Identifier	text		DM.studyid	Identifier	Constant Value: "ABC-ZZZ-xxx"
USUBJID	Unique Subject Identifier	text		DM.usubjid	Identifier	Equivalent to studyid || "-" || strip(siteid) || "-" || strip(subjid)
SUBJID	Subject Identifier for the Study	text		DM.subjid	Identifier	(e.g. 102130)
SITEID	Study Site Identifier	text		DM.siteid	Record Qualifier	DM.SITEID
AGE	Age	float		DM.age	Record Qualifier	Equals to DM.age
AGEGR1	Pooled Age Group 1	text	AGEGR1N	Derived	Record Qualifier	<=45, if age <= 45; >45 and <=65, if 45 < age <= 65; >65, if age > 65; Note: Decode variable for AGEGPN.
AGEGR1N	Pooled Age Group 1 (N)	float	AGEGR1N	Derived	Synonym Qualifier	Category derived if age non-missing. Equals, 1, if age <= 45; 2, if 45 < age <= 65; 3, if age > 65

Display 21. Data Definition Table of Sample define.xml - Variable Information

Since the ADaM programming specifications are the unique source for generating both ADaM metadata and define.xml, the consistency between ADaM metadata and programming specification, and further between ADaM metadata and the define files can be automatically guaranteed.

AUTOMATION 5: GENERATION OF GLOBAL MACRO VARIABLES OF METADATA FOR ADAM DERIVATION PROGRAMMING

Every analysis dataset should be associated with a dataset label, and all variables in the analysis dataset are defined by attributes label, type (Numeric or Character) and length. Dataset label and variable attributes of an ADaM dataset are retrieved from ADXX_DOMAIN and ADXX_VARS datasets as shown in Display 9 and Display 10, respectively, which are generated by macro %**get_adam_specs** from programming specification document. Macro %**adam_attrib** is called in each ADaM derivation program to generate global macro variables &**ADAMLABEL**, &**ADAMVARS**, and &**VAR_ATTRIB** which upon resolution provide ADaM dataset label, variables in the ADaM dataset and their attributes, respectively.

The macro call of %**adam_attrib** is as follows:

```
%macro adam_attrib(libin=adamspec, dsin=, dmin=);
```

Where,

LIBIN: the libref associated with a SAS data library that has ADaM domain information dataset (**ADXX_DOMAIN**) and variable information dataset (**ADXX_VARS**)
DSIN: Name of the SAS dataset storing ADaM variable information, default value ADXX_VARS
DMIN: Name of the SAS dataset storing ADaM domain information, default value ADXX_DOMAIN

Global macro variables &**ADAMLABEL**, &**ADAMVARS**, and &**VAR_ATTRIB** will be used in the DATA step of ADaM conversion program to populate dataset label and variable attributes in the final ADaM dataset. Sample SAS code from ADLB (Laboratory Analysis Dataset) derivation program is shown below. This methodology avoids defining variable attributes for all the variables in the dataset in the conversion program thereby significantly reducing programming work load and occurrence of human errors. It guarantees the consistency between the ADaM datasets and the specifications. Validation programmer can review specification document for variable metadata and use the same mechanism in the QC program. This methodology is especially feasible in handling any changes of variables and their attributes. Cost effectiveness is achieved since the only update needed is the programming specifications.

```
*** Output ADaM dataset ***;
data ad.adlb(keep=&adamvars. label=&adamlabel.);
    attrib &var_attrib.;
    set adlb;
run;
```

The resolution of global macro variables &**ADAMLABEL**, &**ADAMVARS**, and &**VAR_ATTRIB** in log file of ADHC conversion programs are as follows.

```
%put &adamlabel.;
Laboratory Analysis Dataset

%put &adamvars.;
STUDYID USUBJID LBSEQ LBREFID PARAMCD PARAM PARAMN PARCAT1 APHASE APHASEN DTYPE
AVISIT AVISITN TVRFL ONTRTFL LBORRES LBORRESU LBORNRLO LBORNRHI LBSTRESC LBSTRESN
LBSTRESU LBSTNRLO LBSTNRHI ANRLO ANRHI ANRIND LBSTAT LBREASND LBNAM LBSPEC LBMETHOD
LBFAST BASE BASEC CHG BNRIND ADTM ADT ATM ADY LOCALFL VISIT VISITNUM LBDTC LBRPTLBL
LBPREC ATOXGR ATOXGRN ATOX ATOXGRH ATOXGRHN ATOXH BTOXGR BTOXGRN BTOX BTOXGRH
BTOXGRHN BTOXH MXGR_T MXGR_A HMXGR_T HMXGR_A MXNR_T MXNR_A MNNR_T MNNR_A HGB10FL
HGB8FL UACIDFL HGB10_T HGB10_A HGB10N_T HGB10N_A HGB8_T HGB8_A HGB8N_T HGB8N_A
UACID_T UACID_A UACIDN_T UACIDN_A TABLEFL TABLESFL LISTNGFL ABLFL ANL01FL ANL02FL
AVAL AVALC PCHG MINFL MAXFL MINFL_TW

%put &var_attrib.;
STUDYID label='Study Identifier' length=$20 USUBJID label='Unique Subject
Identifier' length=$40 LBSEQ label='Sequence Number' length= 8 LBREFID
label='Specimen ID' length=$20 PARAMCD label='Parameter Code' length=$8 PARAM
label='Parameter' length=$80 PARAMN label='Parameter (N)' length= 8 PARCAT1
label='Parameter Category 1' length=$40 APHASE label='Phase' length=$40 APHASEN
label='Phase Number' length= 8 DTYPE label='Derivation Type' length=$40 AVISIT
label='Analysis Timepoint Description' length=$40 AVISITN label='Analysis Timepoint
Description Number' length= 8 TVRFL label='TVR/Pbo Treatment Phase Event (+1 day)'
length=$1 ONTRTFL label='On Treatment Record Flag' length=$1 LBORRES label='Result
or Finding in Original Units' length=$120 LBORRESU label='Original Units'
length=$40 LBORNRLO label='Reference Range Lower Limit in Orig Unit' length=$40
LBORNRHI label='Reference Range Upper Limit in Orig Unit' length=$40 LBSTRESC
label='Character Result/Finding in Std Format' length=$120 LBSTRESN label='Numeric
Result/Finding in Standard Units' length= 8 LBSTRESU label='Standard Units'
length=$40 LBSTNRLO label='Reference Range Lower Limit-Std Units' length= 8
LBSTNRHI label='Reference Range Upper Limit-Std Units' length= 8 ANRLO
label='Analysis Normal Range Lower Limit' length=$40 ANRHI label='Analysis Normal
Range Upper Limit' length=$40 ANRIND label='Reference Range Indicator' length=$8
LBSTAT label='Lab Status' length=$8 LBREASND label='Reason Test Not Done'
length=$200 LBNAM label='Vendor Name' length=$200 LBSPEC label='Specimen Type'
length=$40 LBMETHOD label='Method of Test or Examination' length=$100 LBFAST
label='Fasting Status' length=$2 BASE label='Baseline Value' length= 8 BASEC
label='Character Baseline Value' length=$40 CHG label='Change from Baseline'
length= 8 BNRIND label='Baseline Reference Range Indicator' length=$8 ADTM
label='Analysis Date/Time' length= 8 format=DATETIME20. ADT label='Analysis Date'
length= 8 format=YYMMDD10. ATM label='Analysis Time' length= 8 format=TIME5. ADY
label='Analysis Relative Day' length= 8 LOCALFL label='Local Lab Result Flag'
length=$2 VISIT label='Visit Name' length=$80 VISITNUM label='Visit Number' length=
8 LBDTC label='Date/Time of Specimen Collection' length=$20 LBRPTLBL
label='Laboratory Test label for reports' length=$80 LBPREC label='Decimal
precision for reports' length= 8 ATOXGR label='Analysis Toxicity Grade' length=$2
ATOXGRN label='Analysis Toxicity Grade (N)' length= 8 ATOX label='Toxicity'
length=$80 ATOXGRH label='Analysis Toxicity Grade for High Value' length=$2
ATOXGRHN label='Analysis Tox Grade for High Value (N)' length= 8 ATOXH
label='Toxicity, for High Value' length=$80 BTOXGR label='Baseline Toxicity Grade'
length=$2 BTOXGRN label='Baseline Toxicity Grade (N)' length= 8 BTOX
label='Baseline Toxicity' length=$80 BTOXGRH label='Baseline Toxicity Grade, for
High Values' length=$2 BTOXGRHN label='Baseline Tox Grade, for High Values (N)'
length= 8 BTOXH label='Baseline Toxicity, for High Values' length=$80 MXGR_T
label='Max toxicity tru TVR Treatment Phase' length= 8 MXGR_A label='Max toxicity
```

```
tru Overall Treatment Phase' length= 8 HMXGR_T label='Max toxicity tru TVR Phase,
for High' length= 8 HMXGR_A label='Max toxicity tru Overall Phase, for High'
length= 8 MXNR_T label='Normal Range (Max) tru TVR Treat Phase' length=$8 MXNR_A
label='Normal Range (Max) tru Overall Phase' length=$8 MNNR_T label='Normal Range
(Min) tru TVR Treat Phase' length=$8 MNNR_A label='Normal Range (Min) tru Overall
Phase' length=$8 HGB10FL label='HGB flg (Male <105 g/L, Female <100 g/L)' length=$2
HGB8FL label='HGB flag (Male <85 g/L, Female <80 g/L)' length=$2 UACIDFL
label='Elevated Uric Acid flag (>=446 umol/L)' length=$2 HGB10_T label='Days to 1st
HGB10 Flg tru TVR Trt Phase' length= 8 HGB10_A label='Days to 1st HGB10 Flag tru
Overall Phase' length= 8 HGB10N_T label='Days, 1st HGB10 Flg - Norm tru TVR Phase'
length= 8 HGB10N_A label='Days, 1st HGB10 Flg - Norm tru Overall' length= 8 HGB8_T
label='Days to 1st HGB8 Flag tru TVR Trt Phase' length= 8 HGB8_A label='Days to 1st
HGB8 Flag tru Overall Phase' length= 8 HGB8N_T label='Days, 1st HGB8 Flag - Norm
tru TVR Phase' length= 8 HGB8N_A label='Days, 1st HGB8 Flag - Norm tru Overall'
length= 8 UACID_T label='Days to 1st Elevated UAcid tru TVR Phase' length= 8
UACID_A label='Days to 1st Elevated UAcid tru Overall' length= 8 UACIDN_T
label='Days, 1st Elevated UAcid - Norm tru TVR' length= 8 UACIDN_A label='Days,1st
Elevated UAcid-Norm tru Overall' length= 8 TABLEFL label='Selected Analysis Flag
for Summary Table' length=$2 TABLESFL label='Selected Analysis Flag for Shift
Tables' length=$2 LISTNGFL label='Selected Analysis Flag for Listings' length=$2
ABLFL label='Baseline Record Flag' length=$2 ANL01FL label='Analysis Record Flag
01' length=$2 ANL02FL label='Analysis Record Flag 02' length=$2 AVAL
label='Analysis Value' length= 8 AVALC label='Analysis Value (C)' length=$20 PCHG
label='Percent Change from Baseline' length= 8 MINFL label='Minimum on Treatment
Measurement Flag' length=$2 MAXFL label='Maximum on Treatment Measurement Flag'
length=$2 MINFL_TW label='Lowest Measures During Each Trt WD Flag' length=$2
```

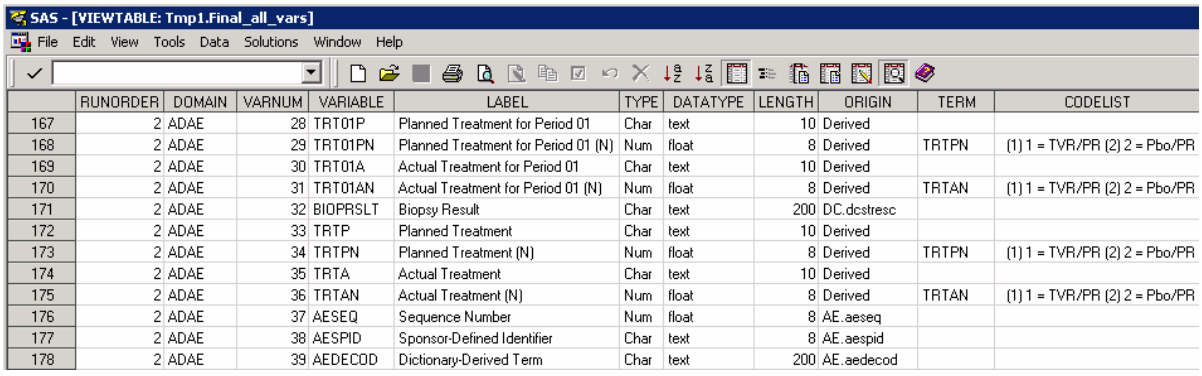
AUTOMATION 6: ADD CORE VARIABLES TO DEFINE.XML AND ADAM DATASETS AT FINAL RUN

FDA advises to populate a set of basic subject level variables to all analysis datasets. These variables are called core variables. Core variables include study/protocol, site, country, treatment assignment, sex, age, race, ethnicity, analysis population flags (e.g. full analysis set flag, per protocol flag etc.) and other important baseline demographic variables. They will be identified from ADSL and populated in all analysis datasets, which avoids the additional step of merging analysis datasets with ADSL to get basic subject level information while generating TFLs.

ADD CORE VARIABLES TO DEFINE.XML AT FINAL RUN

Adding core variable function will be activated when macro variable &**final_run** is set to Y at the invoking of macro %**get_adam_specs** in final stage of ADaM programming activities. The core variables are stored in a global macro variable &**core_vars** which is defined in the study set up file. The attributes of these core variables are retrieved from the specification of subject level analysis dataset ADSL. The core variables will be added to ALL_VARS for all analysis datasets, and further populated into define.xml.

The Display 22 shows the final SAS data named **FINAL_ALL_VARS**, which contains variable information including added core variables for all ADaM datasets. The variables in each anlaysis dataset are re-ordered so that core variables are added after the key variables. If adding core variable function is activated, FINAL_ALL_VARS will be used to create define.xml, which includes core variables in all analysis datasets.



	RUNORDER	DOMAIN	VARNUM	VARIABLE	LABEL	TYPE	DATATYPE	LENGTH	ORIGIN	TERM	CODELIST
167	2	ADAE	28	TRT01P	Planned Treatment for Period 01	Char	text	10	Derived		
168	2	ADAE	29	TRT01PN	Planned Treatment for Period 01 (N)	Num	float	8	Derived	TRTPN	(1) 1 = TVR/PR (2) 2 = Pbo/PR
169	2	ADAE	30	TRT01A	Actual Treatment for Period 01	Char	text	10	Derived		
170	2	ADAE	31	TRT01AN	Actual Treatment for Period 01 (N)	Num	float	8	Derived	TRTAN	(1) 1 = TVR/PR (2) 2 = Pbo/PR
171	2	ADAE	32	BIOPRSLT	Biopsy Result	Char	text	200	DC.dcstresc		
172	2	ADAE	33	TRTP	Planned Treatment	Char	text	10	Derived		
173	2	ADAE	34	TRTPN	Planned Treatment (N)	Num	float	8	Derived	TRTPN	(1) 1 = TVR/PR (2) 2 = Pbo/PR
174	2	ADAE	35	TRTA	Actual Treatment	Char	text	10	Derived		
175	2	ADAE	36	TRTAN	Actual Treatment (N)	Num	float	8	Derived	TRTAN	(1) 1 = TVR/PR (2) 2 = Pbo/PR
176	2	ADAE	37	AESEQ	Sequence Number	Num	float	8	AE.aeseq		
177	2	ADAE	38	AESPID	Sponsor-Defined Identifier	Char	text	8	AE.aespid		
178	2	ADAE	39	AEDECOD	Dictionary-Derived Term	Char	text	200	AE.aedecod		

Display 22. Final Dataset Containing Variable Information of All ADaM Domains with Core Variables Added

ADD CORE VARIABLES TO ADAM DATASETS AT FINAL RUN

Similar to adding core variables to define.xml, core variables are automatically retrieved from ADSL and added to all ADaM datasets, which avoids the redundant and error-prone process to develop the same variables in different ADaM derivation programs. In our ADaM tool core variables are not even included in the individual derivation programs when developing Individual ADaM datasets. Instead, a separate SAS script named **add_corevars.sas** is created to add core variables to all analysis datasets defined in ALL_DOMAINS in the final run. **Add_corevars.sas** will later be used in the batch file _**runADaM.bat** for batch submitting ADaM derivation programs. This process introduces the flexibility of developing individual ADaM datasets before ADSL is ready for use.

AUTOMATION 7: CONSISTENCY CHECKING OF CONTROLLED TERMINOLOGY AND VALUE LEVEL METADATA BETWEEN ADAM DATASETS AND PROGRAMMING SPECIFICATION

It is very critical for FDA submission to ensure consistency in controlled terminology and value level metadata between programming specifications and ADaM datasets. ADaM programming tool uses SAS macro %**ctlist_checking** to automate the process of checking consistency in controlled terminology and value level metadata between ADaM datasets and programming specifications. This macro can be called at any stage of ADaM programming cycle and helps in finalizing the programming specifications at an earlier stage. The controlled terminology in ADaM datasets can be categorized as value level metadata originating from source SDTM datasets to ADaM BDS Datasets, sponsor-defined terminology for the code-decode variable pair, controlled terminology inherited from SDTM domains, and therapeutic-specific terminology defined by FDA. While writing ADaM programming specifications these controlled terminology and value level metadata follow a particular style for proper function of macro %**ctlist_checking** as shown in Display 23 – 26.

Variable Name	Variable Label	Type	Length	Controlled Terms or Formats	Origin	Role	Comments	Core
PARAMCD	Parameter Code	Char	8	(1) BMI = BODY MASS INDEX (KG/M2) (2) DIABP = DIASTOLIC BLOOD PRESSURE (MMHG) (3) HEIGHT = HEIGHT (CM) (4) PULSE = PULSE RATE (BEATS/MIN) (5) RESP = RESPIRATORY RATE (BREATHS/MIN) (6) SYSBP = SYSTOLIC BLOOD PRESSURE (MMHG) (7) TEMP = TEMPERATURE (C) (8) WEIGHT = WEIGHT (KG)	VS.vstestcd	Topic	Equals upcase(strip(VS.vstestcd)).	Req
PARAM	Parameter Description	Char	40		VS.vstest	Synonym Qualifier	If paramcd="BMI" then param=strip(VS.vstest)||" (KG/M2)"; Else if paramcd="DIABP" then param=strip(VS.vstest)||" (MMHG)"; Else if paramcd="HEIGHT" then param=strip(VS.vstest)||" (CM)"; Else if paramcd="PULSE" then param=strip(VS.vstest)||" (BEATS/MIN)"; Else if paramcd="RESP" then param=strip(VS.vstest)||" (BREATHS/MIN)"; Else if paramcd="SYSBP" then param=strip(VS.vstest)||" (MMHG)"; Else if paramcd="TEMP" then param=strip(VS.vstest)||" (C)"; Else if paramcd="WEIGHT" then param=strip(VS.vstest)||" (KG)";	Req

Display 23. Illustration of PARAMCD Value Level Metadata in an ADaM Specification

Variable Name	Variable Label	Type	Length	Controlled Terms or Formats	Origin	Role	Comments	Core
ATOXGR	Analysis Toxicity Grade	Char	20		Derived	Record Qualifier	Get the decoded value of ATOXGRN. Equals 'MILD' if ATOXGRN = 1 'MODERATE' if ATOXGRN = 2 'SEVERE' if ATOXGRN = 3 LIFE-THREATENING if ATOXGRN = 4	Perm
ATOXGRN	Analysis Toxicity Grade Number	Num	8	ATOXGRN (ATOXGR): (1) 1 = MILD (2) 2 = MODERATE (3) 3 = SEVERE (4) 4 = LIFE-THREATENING	AE.AETOXGR	Synonym Qualifier	Equals AE.AETOXGR	Perm

Display 24. Illustration of Sponsor-Defined Controlled Terminology in an ADaM Specification

Variable Name	Variable Label	Type	Length	Controlled Terms or Formats	Origin	Role	Comments	Core
LBSPEC	Specimen Type	Char	40	LBSPEC: (1) BLOOD (2) SERUM (3) URINE	LB.lbspec	Record Qualifier	Equals LB.lbspec	Perm

Display 25. Illustration of Controlled Terminology Inherited from CDISC SDTM Domain

Variable Name	Variable Label	Type	Length	Controlled Terms or Formats	Origin	Role	Comments	Core
OUTCOME	Virologic Outcome	Char	40	OUTCOME: (1) SVR (2) Relapse (3) On-treatment Virologic Failure (4) Other	Derived	Result Qualifier	Outcome equals to "SVR" if a subject has HCV RNA below level of quantification at last assessment in Antiviral Follow-up Week 24. Outcome equals to "Relapse" if undetectable at planned EOT and any detectable during follow-up Outcome equals to "On-treatment Virologic Failure" if subject met a stopping rule or (had a viral breakthrough and detectable at planned EOT) else equals to "Other"	Perm

Display 26. Illustration of FDA Defined Therapeutic Specific Controlled Terminology

Macro %**ctlist_checking** compares the controlled terminology and value level metadata defined in the ADaM programming specifications with that in the ADaM datasets, detects any mismatches, and generates inconsistency report in RTF format if any exists.

The macro call of %**ctlist_checking** is as follows:

```
%ctlist_checking(specdir = &sty_ad_spec.,
                 datadir = &sty_data_ad.,
                 domain  = _ALL_
                );
```

Where,

SPECDIR: Full Path for ADaM Programming Specifications. Default value as study folder for ADaM specifications.
DATADIR: Full Path for ADaM datasets. Default value as study folder for ADaM datasets.
DOMAIN: An ADaM domain to be checked with controlled terminology and value level metadata. If the macro variable &DOMAIN is not assigned a value, all ADaM domains will be checked for consistency of the controlled terminology and value level metadata.

Display 27 – 30 show typical reports of non-consistency between ADaM datasets and specifications. Decision will be made by programmers to update either the programming specifications or the ADaM derivation program to handle these mismatches. The general decision-making rules for mismatches are listed in Appendix 2.

The following PARCAT1 Variables with Different Value Level Metadata between Programming Specs. and Datasets

Domain	Variable	Value	Value Label in Dataset	Value Label in Specs.	Terms In Specs. NOT in Dataset	Terms In Dataset NOT In Specs.
ADLB	PARCAT1	VIROLOGY	VIROLOGY			Yes

Display 27. Non-Consistency Report of Value List Metadata for PARAMCAT Between ADaM Datasets and Specifications

The following PARAM Variables with Different Value Level Metadata between Programming Specs. and Datasets

Domain	Variable	Variable Label	Value	Value Label in Dataset	Value Label in Specs.	Terms In Specs. NOT in Dataset	Terms In Dataset NOT In Specs.	Different Controlled Terminology
ADVS	PARAMCD	Parameter Code	RESP	RESPIRATORY RATE (BREATHS/MIN)			Yes	
			BMI	BODY MASS INDEX (KG/M2)	BODY MASS INDEX			Yes
			RESPR		RESPIRATORY RATE (BREATHS/MIN)	Yes		

Display 28. Non-Consistency Report of Value List Metadata for PARAMCD between ADaM Dataset and Specification

The following Coded Variables with Different Decoded Terminology between Programming Specs. and Datasets

Domain	Variable	Variable Label	Coded Controlled Term	Decoded Controlled Term in Dataset	Decoded Controlled Term in Specs.	Codelist in Specs.	Terms In Specs. NOT in Dataset	Terms In Dataset NOT In Specs.	Different Decoded Terminology
ADAE	AEACNN	Action with Study Treatment Number	5	NOT APPLICABLE				Yes	
	AEOUTN	Outcome of Adverse Event Number	3	RECOVERED/ RESOLVED WITH SEQUELAE	RECOVERED/RESOLVED WITH SEQUELAE	AEOUTN			Yes
			4		FATAL	AEOUTN	Yes		
			5		UNKNOWN	AEOUTN	Yes		
ADEG	BQTGR1N	Baseline Pooled QT Group 1 (N)	4	> 500	> 500 msec	BQTGR1N			Yes
ADSL	OUTCOMEN	Virologic Outcome (N)	3		Rebound at EOT+12	OUTCOMEN	Yes		
			4		Rebound at EOT+24	OUTCOMEN	Yes		

Display 29. Non-Consistency Report of Sponsor Defined Controlled Terminology between ADaM Datasets and Specifications

The following Variables with Different Controlled Terminology between Programming Specs. and Datasets

Domain	Variable	Variable Label	Controlled Term in Dataset	Controlled Term in Specs.	Codelist in Specs.	Terms In Specs. NOT in Dataset	Terms In Dataset NOT In Specs.
ADAE	AEACNTP	Action Taken with Telaprevir		DOSE REDUCED	AEACNF	Yes	
				DRUG INTERUPTED	AEACNF	Yes	
	AEACNHAA	Action Taken with HAART		DRUG WITHDRAWN	AEACNF	Yes	
			DRUG INTERRUPTED				Yes

Display 30. Non-Consistency Report of CDISC or FDA defined Controlled Terminology Between ADaM Datasets and Specifications

AUTOMATION 8: DETECTING EMPTY VARIABLES

When submitting clinical study data in electronic format to the FDA, it is preferable to submit as few as possible unnecessary variables which have all missing values. These variables are called empty variables. CDISC introduced a concept of core variable in an ADaM dataset and categorized a variable as **Required**, **Conditionally Required**, and **Permissible** in an ADaM dataset. Applying the information of core variable categories to these empty variables provides a better decision to handle these empty variables in an FDA submission. ADaM programming tool uses macro %**empty_var_checking** to automatically detect and identify empty variables in ADaM datasets and thereby ensures technical accuracy and submission quality. It can be performed at any stage of the programming cycle.

Macro %**empty_var_checking** calculates the number of observations with missing value for each variable. If the count is equal to the number of the observations in the dataset, then the variable will be flagged as an empty variable. A report will be generated for all empty variables which include error messages for specially-defined ADaM required variables such as USUBJID, STUDYID, SEX, COUNTRY, and etc., warning messages for other ADaM required or

conditionally required variables, and warning messages for ADaM permissible variables. The general decision-making rules for handle empty variables for FDA submission are listed in Appendix 3.

The macro call of %**empty_var_checking** is as follows.

```
%macro empty_var_checking(cdisc=, specdir=,datadir=, domain=_ALL_);
```

Where,

CDISC: Specifies the data model as ADaM.
SPECDIR: Full Path of ADaM Programming Specifications.
DATADIR: Full Path of ADaM datasets.
DOMAIN: An ADaM domain. If assigned _ALL_ or blank all ADaM domains will be checked.

Display 31 shows a typical report of empty variables in ADaM datasets. Decision will be made by programmer whether to drop, retain or update ADaM conversion programs for these variables.

The Following ADaM Variables in Study xxx Have All Missing Values

Domain	Variable Order	Variable	Variable Label	Total Number of Observations	Core	Comment
ADAE	73	DCRASHFL	Discontinuation due to Rash SSC	561	Perm	Warning: Permissible Variable is Empty. Delete it?
	74	DCPRURFL	Discontinuation due to Pruritus SSC	561	Perm	Warning: Permissible Variable is Empty. Delete it?
	76	DCANORFL	Disc. due to Anorectal Disorder	561	Perm	Warning: Permissible Variable is Empty. Delete it?
	77	DCINJSFL	Disc. due to Injection Site Reaction	561	Perm	Warning: Permissible Variable is Empty. Delete it?
ADSL	15	COUNTRY	Country	62	Req	Error: Required Variable is Empty. Correct/Check SAS Program!
	36	TRT01P	Planned Treatment for Period 01	62	Req	Warning: Required Variable is Empty. Check the SAS Program!
	46	TRTSDT	Date of First Exposure to Treatment	62	Cond	Warning: Conditionally Required Variable is Empty. Check the SAS Program!

Display 31. A Report of Empty Variables with Different CORE Attribute Categories in ADaM Datasets

If the final ADaM Datasets still contains empty variables, the rationale to keep these empty variables in the ADaM datasets will be given in the reviewer guide for FDA reviewers. An example of rationale to keep empty variables in ADAE dataset is shown in Display 32.

Domain	Variable Order	Variable	Variable Label	Core	Comment
ADAE	73	DCRASHFL	Discontinuation due to Rash SSC	Perm	The event did not occur in the study, but the variable is needed for analysis.
	74	DCPRURFL	Discontinuation due to Pruritus SSC	Perm	The event did not occur in the study, but the variable is needed for analysis.
	76	DCANORFL	Disc. due to Anorectal Disorder	Perm	The event did not occur in the study, but the variable is needed for analysis.
	77	DCINJSFL	Disc. due to Injection Site Reation	Perm	The event did not occur in the study, but the variable is needed for analysis.

Display 32. The Rationale to Keep Empty Variables in ADaM Datasets – in Reviewer Guide

AUTOMATION 9: PREPARATION OF SAS SCRIPTS FOR FINAL RUN OF ALL ADAM SPECIFICATIONS AND CREATION OF ADAM SPECIFICATIONS FOR ALL DOMAINS FROM INDIVIDUAL ONES

At the final stage of ADaM programming we need two SAS programs for most updated specifications and metadata, one for rerun of all ADaM specifications to update the metadata, and another for combining all individual ADaM specifications into one Word file as ADaM programming specifications. Manually preparing the SAS programs for this function is labor intensive and error prone. ADaM Programming Tool calls macro %**get_adam_specs_final_calls** to automatically generate these SAS scripts. Successful execution of macro %**get_adam_specs_final_calls** generates SAS code **adam_specs_final_calls.sas** which contains macro calls %**get_adam_specs** to convert individual domain programming specification files to SAS datasets, and SAS code **get_all_adam_specs.sas** which combines all individual ADaM specifications into one Word file. SAS code **adam_specs_final_calls.sas** and **get_all_adam_specs.sas** will later be written into the batch file **_runADaM.bat** for batch submitting ADaM derivation programs.

The macro call of %**get_adam_specs_final_calls** is as follows:

```
%macro get_adam_specs_final_calls(indir=, dom_del=, xmldir=);
```

Where,

INDIR: Full Path of SAS dataset ALL_DOMAINS which contains all the ADaM domain information.
DOM_DEL: Name of ADaM domains to be excluded in the final run.
XMLDIR: Full Path of define.xml.

The SAS code **adam_specs_final_calls.sas** generated by macro **%get_adam_specs_final_calls** is shown below:

```
%include " E:\final\standard.sas";
**** Initiation: Set all_domains and all_vars datasets empty;
libname __in "E:\final\convert\analysis\specification\";
Data __in.all_domains; if 0; run;
Data __in.all_vars; if 0; run;

**** macro call for ADaM specs for ADSL;
%get_adam_specs(indir     = %str(E:\final\convert\analysis\specification\),
                specsnm   = ADSL.csv,
                outdir    = %str(E:\final\convert\analysis\specification\),
                runorder  = 1);

**** macro call for ADaM specs for ADAE;
%get_adam_specs(indir     = %str(E:\final\convert\analysis\specification\),
                specsnm   = ADAE.csv,
                outdir    = %str(E:\final\convert\analysis\specification\),
                runorder  = 2);
...

**** macro call for ADaM specs for ADVS;
%get_adam_specs(indir     = %str(E:\final\convert\analysis\specification\),
                specsnm   = ADVS.csv,
                outdir    = %str(E:\final\convert\analysis\specification\),
                runorder  = 9,
                generate_xml = Y,
                xmldir    = %str(E:\final\define_xml\analysis\),
                final_run = Y);
```

The order of each individual ADaM specification macro call in the final run is decided by the variable RUNORDER in ALL_DOMAINS dataset.

SAS code **get_all_adam_specs.sas** generated by macro **%get_adam_specs_final_calls** is shown below:

```
%include "E:\final\standard.sas";
**** macro call of WORDNTO1 to combine all ADaM specs into ONE;
%wordnto1(inputfn  = %str(E:\final\convert\analysis\specification\ADSL.doc,
                          E:\final\convert\analysis\specification\ADAE.doc,
                          E:\final\convert\analysis\specification\ADCD.doc,
                          E:\final\convert\analysis\specification\ADCM.doc,
                          E:\final\convert\analysis\specification\ADEG.doc,
                          E:\final\convert\analysis\specification\ADHC.doc,
                          E:\final\convert\analysis\specification\ADHI.doc,
                          E:\final\convert\analysis\specification\ADLB.doc,
                          E:\final\convert\analysis\specification\ADVS.doc),
          Outputfn = E:\final\convert\analysis\specification\all_ADaM_specs.doc,
          deletein = No);
```

in which macro **%wordnto1** is a macro developed in-house to combine multiple input word files into one Word file.

The order of each individual ADaM specification in the final ADaM specifications is decided by the variable RUNORDER in ALL_DOMAINS dataset.

The combined ADaM specifications will be sent to statisticians for review. Once approved, it is considered to be the final version and will serve as part of the reviewer guide to facilitate FDA reviewers to familiarize the submitted data.

AUTOMATION 10: CREATION OF BATCH FILE FOR FINAL RUN

A batch file for batch submitting both ADaM specification programs and ADaM derivation programs is needed for the final run. Creation of the batch file manually is time consuming and error prone. The Tool calls macro %get_batch_file to automatically generate the batch file with a specified order defined by the variable RUNORDER in ALL_DOMAINS dataset.

Successful execution of macro %**get_batch_file** generates batch file **_runADaM.bat** containing batch commands to run SAS codes **adam_specs_final_calls.sas** to update the variable attributes and define.xml, **get_all_adam_specs.sas** to combine all the individual specifications into one word document for reviewer guide, ADaM derivation programs to update ADaM datasets, **add_corevars.sas** to add core variables to all ADaM datasets, **mk_xpt.sas** to generate SAS transport files for define.xml, **ctlist_checking_call.sas** for final consistency checking of controlled terminology and value level metadata, and **empty_var_checking.sas** for identifying and detecting variables with all values missing in final ADaM datasets. The execution order of each ADaM derivation programs is decided by the variable RUNORDER in ALL_DOMAINS dataset.

The macro call of %**get_batch_file** is as follows:

```
%macro get_batch_file(indir=, saslocat=);
```

Where,

INDIR: Full Path of SAS dataset ALL_DOMAINS and ADaM specification programs.
SASLOCAT: Full Path of ADaM conversion programs and output BAT file.

The batch file **_runADaM.bat** generated by macro %**get_batch_file** is shown as follows:

```
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\specification\adam_specs_final_calls.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\specification\get_all_adam_specs.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADSL.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADAE.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADCD.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADCM.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADEG.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADHC.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADHI.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADLB.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ADVS.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\add_corevars.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\mk_xpt.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\ctlist_checking_call.sas"
"C:\program files\sas\sas.exe" -sysin
"E:\final\convert\analysis\conversion\empty_var_checking.sas"
```

SUMMARY

Comparison of conventional methodology for ADaM programming and the innovative one introduced in this paper is shown in Table 1.

Comparison	Conventional Methodology	New Methodology
1. Automation of Compliance Checking CDSIC ADaM Programing Specifications	No	(1) DETECT ANY NONCOMPLIANCE with FDA submission requirements, CDISC ADaM GUIDELINE, and Vertex Guideline for writing specifications (2) REPORT any findings (3) ENSURE the resolution (4) FINALIZE specifications earlier in programming cycle
2. Version Control of Programming Specifications	No	Keep both word version and SAS dataset of programming specifications in a study subfolder '\History'
3. Track Changes of Programming Specification	No	(1) Report changes from any previous version of specifications (2) Help developer and reviewers to trace back changes (3) Serve as a tool for audit from post delivery changes
4. define.xml Generation	(1) Timing: 2 stages (First CSR, second define.xml) (2) Trainings needed (3) Additional resources/times to prepare spreadsheets	(1) Same time as CSR (2) A tool for statisticians to review metadata earlier (3) Minimal training needed (4) No extra resources/times (5) Automatic generation per the macro user request
5. Generation of Variable Attributes or Any Changes of Variable & Attributes and Adding/Deleting Variables	(1) Write an additional SAS program to generate a template insert it into data step in SAS (2) Update both SAS template program and specifications (3) More time needed (4) Error-prone, inconsistent	(1) Call a SAS macro to automatically generate a dataset by extracting from the programming specifications. (2) Call a SAS macro to automatically generate SAS macro variables to be used in final data step of SAS program (3) Update programming specifications ONLY (4) Less time needed, cost-effective (5) Ensure quality and consistency between data and specifications
6. Adding Core Variables into ADaM Data and define.xml for Final Run	(1) Develop additional SAS programs to populate core variables into ADaM data and define.xml (2) Error-prone and labor-intensive!	(1) Automatically generate define.xml with core variables. Automatically generate a SAS program for populating core variables into ADaM datasets (2) Automation saves time and energy!
7. Automatic Consistency Checking of Controlled Terms between ADaM Data and Specifications	No	(1) DETECT ANY MISMATCHES between ADaM datasets and specifications for Controlled Terminology and Value Level Metadata at any stage of programming cycle (2) REPORT any findings (3) FINALIZE specifications earlier in programming cycle
8. Automatic Detection of Empty Variables in ADaM Datasets	No	(1) DETECT ANY EMPTY VARIABLES (2) REPORT any findings with the information of core attributes (3) DECISION MAKING AND FINALIZE specifications at early stage
9. Combination of All ADaM Specifications into One	Manually copy and paste	Automatically combine all individual ADaM specifications into ONE Word document
10. Batch File Preparation for Final Run of ADaM Datasets	Error-prone type-in or copy/paste	Automatic generation with high quality

Table 1. Comparison of Conventional Methodology for ADaM Programming and the Innovative One

CONCLUSION

In summary, all by automation the new methodology streamlines the process of ADaM programming activity: from compliance checking with CDISC and FDA submission requirements, version control, tracking the changes of the specification, define.xml generation at any time point, combination of all ADaM specifications into one Word document for reviewer guide, generation/update dataset label and variable attributes, adding or deleting variables in ADaM programming, adding core variables into both all ADaM datasets and define.xml, consistency checking of controlled terminology and value level metadata between ADaM specification and datasets, detection and identification of empty variables in ADaM datasets, to batch file preparation for final run of ADaM datasets.

Since the ADaM dataset structure, define.xml, reviewer guide, and the batch files for final run are all generated from the Word® specification documents, the methodology ensures the consistency in the entire study from ADaM

Derivation to FDA Electronic Submission, and achieves the high quality of submission, the cost-effectiveness and the efficiency. Moreover, consistency checking of controlled terminology and value level metadata, and empty variables detection and handling further ensure the submission quality.

The ADaM Programming tool is easy to use and only needs minimal trainings. We hope the methodology can assist you in saving your time and resources for clinical study reporting, especially for FDA submission.

REFERENCES

CDISC Analysis Data Model (ADaM) Team. “CDISC ADaM Validation Checks”, January 2011. http://www.cdisc.org/adam

CDISC Analysis Data Model Team. “Analysis Data Model (ADaM) Implementation Guide”. December 2009. http://www.cdisc.org/adam

Xiangchen (BoB) Cui, Min Chen. “Automatic Version Control and Track Changes of CDISC ADaM Specifications for FDA Submission”, PharmaSUG, May 2012.

Xiangchen (BoB) Cui, Min Chen. “Automatic Consistency Checking of Controlled Terminology and Value Level Metadata between ADaM Datasets and Define.xml”, SAS Global Forum, April 2012.

Xiangchen (BoB) Cui, Min Chen. “Automatic Detection and Identification of Variables with All Missing Values in SDTM/ADaM Datasets for FDA Submission”, PharmaSUG, May 2012.

ACKNOWLEDGEMENTS

Appreciation goes to Kelly Blackburn, Stacy Surensky, Abdul Sankoh, Hang Pang, Hongyu Liu, and Tuanyu Wang for their review and comments.

CONTACT INFORMATION

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

Name: Xiangchen (Bob) Cui**,** Ph.D.
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-6069
Fax: 617-460-8060
E-mail: xiangchen_cui@vrtx.com

Name: Min Chen**,** Ph.D.
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-7134
Fax: 617-460-8060
E-mail: min_chen@vrtx.com

Name: Tathabbai Pakalapati
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-7404
Fax: 617-460-8060
E-mail: Tathabbai_Pakalapati@vrtx.com

Appendix 1

Guideline for ADaM Programming Specifications and Compliance Checking Rules

1. The compliance rules for domain information are described in Table 2. They are not defined in CDISC ADaM Validation Checks V1.1.

Checking Number	Checking	Requirement	Key Words
1	Description	1. Non-Missing 2. length <= 40	
2	Unique Identifier Variables	Non-Missing	
3	Structure	Non-Missing	
4	General Class	Non-Missing	Special Purpose, Interventions, Events, and Findings

Table 2. Domain Information Compliance Checking

2. Table 3 defines variable compliance rules. Some of these rules are also defined in OPENCDISC or CDISC ADaM Validation Checks V1.1 in which case the Rule ID/Checking Number are mentioned in last two columns of the tables.

Checking Number	Checking	Requirement	Key Words	Source	Corresponding Rule ID in OpenCDISC	Checking Number in ADaM Validation Checks
5	Variable Name	1. Length <= 8, 2. Start with a letter, comprised of letters (A-Z), underscore (_), and numerals (0-9).		CDISC Request	AD1006	13 14 15
6	Variable Label	Length <= 40		CDISC Request	AD0016	16
7	Type	Non-Missing	Char Num	CDISC Request		
8	Length	Non-Missing for character variables Length <=200 for character variables		CDISC Request		17
9	Controlled Terms or Formats	If the Controlled Terms are given: 1. Provide Controlled Term Name with Colon (:), followed by code lists preceeded by '(#)'. e.g., YESNOF: (1)Y (2)N Or AVISITN (AVISIT): (1) 950 = Baseline (2) 1001 = Day 1 (3) 1029 = Week 4 for a pair of code-decode variables 2. If no Controlled Term Name is provided, then assume it the same as the Variable Name. 3. If the Formats are given, the following condition must		1.Vertex Request 2.Vertex Request 3.CDISC Request	For Datetime Variables: AD0041 AD0042 AD0043	For Datetime Variables: 41 42 43

		be satisfied: a. if the Variable Name ends with DT, then Variable Label must contain 'Date', Type = 'Num', Role = 'Timing', and format = SAS date format or ISO8601 Format (Date9. in Vertex Guideline) b. if the Variable Name ends with TM, then Variable Label must contain 'Time', Type = 'Num', Role = 'Timing', and format = SAS date format or ISO8601 Format (IS8601dt. in Vertex Guideline) c. if the Variable Name ends with DTM, then Variable Label must contain 'Date/Time', Type = 'Num', Role = 'Timing', and format = SAS date format or ISO8601 Format (time5. in Vertex Guideline)				
10	Origin	Non-Missing		CDISC Request		
11	Role	Non Missing	Identifier Topic Timing Grouping Qualifier Result Qualifier Synonym Qualifier Record Qualifier Variable Qualifier Selection Analysis	Vertex Request		
12	Comments	Non-Missing for Origin = Derived (at the FINAL run)		Vertex Request		
13	Core	Non-Missing	Req Cond Perm	CDISC Request		

Table 3. Variable Information Compliance Checking

3. Table 4 defines general compliance rules. Among them, CDISC requested rules are also defined in OPENCDISC or CDISC ADaM Validation Checks V1.1 in which case the Rule ID/Checking Number are mentioned in last two columns of the tables.

Checking Number	Rule	Source	Corresponding Rule ID in OpenCDISC	Checking Number in ADaM Validation Checks
14	All ADaM datasets must contain SDTM STUDYID and USUBJID variables.	CDISC Request		88, 89
15	ADSL dataset must have the variable SUBJID, SITEID, AGE, AGEU, SEX, RACE, ARM	CDISC Request		47, 49, 50, 51, 52, 55, 71

16	ADSL must have at least one variable that ends in FL as a population flag.	CDISC Request	AD0048	48
17	All *DT, *TM, *DTM, and PARAMN variables must be numeric	CDISC Request	AD0058 AD0059 AD0060 AD0148	58 59 60
18	All variable name are defined in uppercase	Vertex Request		
19	All Unique Identifier Variables should be defined in Variable Information Table	Vertex Request		
20	If the numeric flag (*FN) is used, the character version (*FL) is required	CDISC Request	AD0007	7
21	The decoded variables defined in the 'Controlled Terms and Formats' Column must exist in the specification.	Vertex Request		

Table 4. General Rules for Compliance Checking

Appendix 2

Decision Making for Mismatches of Controlled Terminology and Value Level Metadata between ADaM Specifications and Datasets

#	Scenario	Condition	Action Taken
1	Controlled Terms or Value Lists are not in the Datasets but in the Specifications	Code lists are correctly defined in specifications	No Action Needed
2	Controlled Terms or Value Lists are in the Datasets but not in the Specifications	Specification does not list all the possible values for the controlled terms or value lists	Add Missing Controlled Terms or Value Lists to Specifications
3	Code Value for Sponsor-Defined Controlled Terminology or Value for Value Level Metadata PARAMCD are Differently Defined in the Datasets from that in the Specifications	Code Value or Value in datasets is not consistent with Standard Controlled Terms	Revise ADaM Datasets
		Code Value or Value in specifications is not consistent with Standard Controlled Terms	Revise ADaM Specifications
4	Decoded Value for Sponsor-Defined Controlled Terminology or Value Label for Value Level Metadata PARAMCD are Differently Defined in the Datasets from that in the Specifications	Decode Value or Value Label in datasets is not consistent with Standard Controlled Terms	Revise ADaM Datasets
		Decode Value or Value Label in specifications is not consistent with Standard Controlled Terms	Revise ADaM Specifications
5	Typo Occurs Either in ADaM Specifications or in ADaM Derivation Programs		Correct the typo

Table 5. Summary of 5 Scenarios of Mismatches between ADaM Datasets and Specifications

Appendix 3

Decision Making on the Empty Variables

#	Scenario	Condition	Action Taken
1	Empty Specially-defined ADaM Required Variables	Any Program Errors	Correct ADaM SAS Programs
		No Program Errors	Describe Rationale for Data Oddities in Reviewer Guide
2	Empty ADaM Required Variables Other Than USUBJID, SITEID, SEX, COUNTRY, and etc.	Any Program Errors	Correct ADaM SAS Programs
		No Program Errors	Keep the Variable for Submission, and Document in Reviewer Guide
3	Empty ADaM Conditionally Required Variables	Any Program Errors	Correct ADaM SAS Programs
		No Program Errors, Needed in Analysis	Keep the Variable for Submission, and Document in Reviewer Guide
		No Program Errors, Not Needed in Analysis	Drop the Variable
4	Empty ADaM Permissible Variables	A Variable Derived or Not Specified in CRF	Drop the Variable
		A Variable Collected or Needed in Analysis	Keep the Variable for Submission, and Document in Reviewer Guide

Table 6. Summary of 4 Scenarios of Empty ADaM Variables