PharmaSUG 2015 - Paper QT05

EXTENDED ATTRIBUTES: A New Metadata Creation Feature in SAS® 9.4 for Data Sets and Variables

Joseph Hinson, inVentiv Health, Princeton, NJ, USA

ABSTRACT

Wouldn't it be nice if one could run the CONTENTS procedure and obtain such system attributes as "Core", "Source", "DataType", "Derivation", "URL", " SAScode", "Programmer", et cetera? With SAS® 9.4 Extended Attributes, one is no longer limited to just the familiar predefined system attributes "Name", "Label", "Type", "Length", and "Format" for variables. In fact, with Extended Attributes, one can have an entire SAS® code as a data set attribute! These custom data set and variable attributes can be created with the SAS® 9.4 DATASETS procedure through its new XATTR statements. The user-defined attributes get embedded in the data set and can be read from Proc CONTENTS, DICTIONARY.XATTRS table, or from the SASHELP.VXATTR view of that table. In the Dictionary Table, extended attributes have the member type "EXTATTR". The resulting data set gets the extension ".sas7bxdat". For clinical programming, the real usefulness of this new feature is the ability to embed CDISC metadata into data sets such that the creation of define.xml is greatly facilitated.

INTRODUCTION

Often when clinical programmers are creating a Define.xml document, they would first import the already-available SAS® metadata, such as Label, Type, and Length.

SAS® automatically defines eight system attributes for variables in a data set:

SAS® System Attribute	**Notes**
Name	
Type	
Length	2 – 8 bytes for NUMERIC; 1-32767 bytes for CHARACTER
Format	
Informat	
Label	
Position (in observation)	
Index (type)	NONE, SIMPLE, COMPOSITE, or BOTH

Table 1. Predefined Variable System Attributes.

For data sets, SAS® defines as many as twenty-three attributes (prior to version 9.4), consisting of both general attributes and engine/host attributes:

Name	
Created	
Last Modified	
Protection	(if password required)
Type	(eg. CORR, COV, FACTOR)
Encryption	(eg. YES, AES
Generations	(related to the data set option GENNUM=)
Label	
Audit Trail	
Observations	(number of observations)

Variables	(number of variables)
Indexes	(number of indexes)
Integrity Constraints	(data set validation rules)
Observation Length	
Deleted Observations	(observations marked for deletions)
Member	(holds the libname and member name)
Compressed	(CHAR (or YES) or BINARY)
Reuse	(YES/NO)
Point to Observations	(related to the POINTOBS option (YES/NO))
Sorted	(YES/NO)
Data Representation	(identifies the OS that created the data set, eg WINDOWS_64)
Encoding	(based on the OS, eg. EBCDIC, US-ASCII, ANSI-ASCII, wlatin1 Western
Information Maps	

Table 2. Predefined Data Set System Attributes.

Obviously the SAS® system designers did not have CDISC in mind. Even though the predefined variable attributes *Name, Label, and Length* help to create some of the define.xml ItemDef attributes, the "DataType" values in define.xml are quite different from those assigned to the predefined SAS® attribute *Type*:

SAS® System *Type*	**CDISC define.xml *DataType***
character	Text
numeric	Float
	Integer
	Datetime
	Date
	Time
	partialDate
	partialTime
	partialDatetime
	incompleteDatetime
	durationDatetime

Table 3. SAS® system attributes vs CDISC attributes

The CDISC data set metadata also tend to be quite different from the predefined SAS® system attributes for data sets:

Dataset	Description	Class	Structure	Purpose	Keys

Location (.xpt files)	Documentation	SAScode	Language	Owner (Creator)

Table 4. CDISC metadata for data sets.

Thus, the ability to create one's own attributes afforded by the new Extended Attributes feature of SAS® 9.4, comes in very handy when dealing with CDISC metadata.

EXTENDED ATTRIBUTES

Beginning with SAS® version 9.4, the extended attributes feature became available and can be created using the new XATTR statement of Proc DATASETS. The user-defined attributes get embedded in the data set and can be read from Proc CONTENTS, DICTIONARY.XATTRS table, or from the SASHELP.VXATTR view of that table. In the Dictionary Table, the attributes are represented by the member type "EXTATTR", and a data set with extended attributes gets the extension ".sas7bxdat".

a. Creating Extended Attributes for Data Sets

Proc DATASETS is used to define custom attributes. Also required are the MODIFY statement and the new statement XATTR, which has the keyword DS for dataset. The statement can also have ADD, SET, UPDATE, DELETE, or REMOVE keywords.

For example, to create a new extended attribute, you write:

```
Proc datasets lib= LibraryName;
    Modify DatasetName;
           XATTR SET DS  AttributeName1=AttributeValue1
                         AttributeName2=AttributeValue2
                         AttributeName3=AttributeValue3;
    Run;
Quit;
```

"XATTR **SET** DS" is used when one is not sure if the attribute already exist. If that is the case, the attribute is simply updated. "XATTR **ADD** DS" can also be used, but in this case, if the attribute already exists, an error would be issued. "XATTR **UPDATE** DS" would also generate an error if the attribute does NOT already exist.

An extended attribute can have either numeric or character values. If value is character type, there is no maximum length to the value. By default, each value is stored in 256-byte segments and the length of the segment can be changed with the SEGLEN= option of the XATTR OPTIONS statement. If the segment size is not large enough for some particular character attribute value, another segment is allocated. This is particularly important if one is embedding an entire SAS® code as an attribute.

b. Creating Extended Attributes for Variables

Again, Proc Datasets, with its MODIFY and XATTR statements are used, but XATTR gets the keyword "VAR" for variable extended attributes. Also, the variable name is stated and the Name-Value pairs get enclosed in parentheses:

```
Proc datasets lib= LibraryName;
       Modify DatasetName;
              XATTR SET VAR
                     VariableName1=(AttributeName1=AttributeValue1
                                    AttributeName2=AttributeValue2
                                    AttributeName3=AttributeValue3)
                     VariableName2=(AttributeName1=AttributeValue1
                                    AttributeName2=AttributeValue2
                                    AttributeName3=AttributeValue3)
                     VariableName3=(AttributeName1=AttributeValue1
                                    AttributeName2=AttributeValue2
                                    AttributeName3=AttributeValue3)
                     VariableName4=(AttributeName1=AttributeValue1
                                    AttributeName2=AttributeValue2
                                    AttributeName3=AttributeValue3);
       Run;
Quit;
```

APPLICATION TO CDISC METADATA

a. Data Set Attributes:

(i) The Vital Signs ADaM data set, ADVS, is used as an example:

```
data advs;
infile datalines;
input studyid $ usubjid $ aseq trtp $ avisit $ ady adtm :e8601dt19. param $ aval;
datalines;
ABC123 1001 1 CoolDrug Baseline -4 2009-05-15T21:27:05 sysbp 150
ABC123 1001 2 CoolDrug Week1 3 2009-05-22T21:30:00  sysbp 128
ABC123 1001 3 CoolDrug Week2 9 2009-05-28T21:30:00  sysbp 142
ABC123 1001 4 CoolDrug Week3 20 2009-06-08T21:30:00  sysbp 138
ABC123 1002 5 Placebo Baseline -1 2009-05-15T21:00:00  sysbp 127
ABC123 1002 6 Placebo Week1 7 2009-05-23T21:00:00  sysbp 132
ABC123 1002 7 Placebo Week2 15 2009-05-31T21:00:00  sysbp 121
ABC123 1002 8 Placebo Week3 22 2009-06-07T21:00:00  sysbp 128
ABC123 1003 9 CoolDrug Baseline 1 2009-05-15T21:00:00  sysbp 137
ABC123 1003 10 CoolDrug Week1 8 2009-05-22T21:15:00  sysbp 127
ABC123 1003 11 CoolDrug Week2 20 2009-06-03T21:15:00  sysbp 120
ABC123 1003 12 CoolDrug Week3 31 2009-06-14T21:15:00  sysbp 134
;
run;
```

(ii) Metadata consistent with ADaM specifications are then created as extended attributes:

```
options noquotelenmax;
proc datasets lib=work nolist;
       modify ADVS;
              xattr options seglen = 10000;
              xattr add ds
                     ADaMdataset="ADVS"
                     Description="Vital Signs Analysis"
                     Class="BDS"
                     Structure="One record per vital sign measurement per analysis
visit per subject"
                     Purpose="Analysis"
                     Keys="USUBJID PARAMCD AVISIT"
                     Location="ADVS.XPT"
                     ProductionProgrammer="Jane Doe"
                     SAScode="
                            data vsdata;
                            format adt date9.;
                            set folder.vs(where = (subjid ne '' and vsgrpid='VS' and aval ne .)
                            rename =( vsdy=ady vsstresn=aval vstestcd=paramcd)
                            keep=studyid vsdtc vsdy vsstresn vstestcd usubjid
                            subjid vsgrpid epoch taetord visitnum visit vsstresu);
                            paramcat = 'VITALS';
                            adt=input(vsdtc, ??yymmdd10.);
                            run;
                            proc sort data=vsdata;
                            by usubjid subjid;
                            run;
                            proc sort data=sl(where=(subjid ne "")) out=adsl;
                            by usubjid subjid;
                            run;
                            data vsdataall noinadsl;
                            merge vsdata(in=invsdata) adsl(in=inadsl keep=studyid usubjid
                            subjid siteid invid trta trtan trtp trtpn rfstdt age ageu race
                            sex);by usubjid subjid; if invsdata and inadsl then output
                            vsdataall;if not inadsl then output noinadsl;run;data
                            unit_missing;set vsdataall;if vsstresu eq '';run; data ADVS;
```

```
                        set vsdataall;
                        if paramcat='VITAL' then do ;
                        select (paramcd);
                        when('BMI')     paramn=801;
                        when('DIABP')   paramn=802;
                        when('PULSE')   paramn=803;
                        when('SYSBP')   paramn=804;
                        when('WEIGHT')  paramn=805;
                        when('RESP')    paramn=806;
                        when('TEMP')    paramn=807;
                        when('HEIGHT')  paramn=808;
                        when('WAIST')   paramn=809;
                        otherwise paramn=.;
                        end;
                        end;
                        unit=vsstresu;
                        if paramcd ='WEIGHT' and vsstresu='lb' then
                        do;
                        aval1=aval;
                        aval=aval/2.20462262 ;
                        run;”;
            run;
quit;
```

(Please Note: The above SAS® code defined as an attribute is an incomplete segment of a working code and is meant for illustration only).

b. Variable Attributes:

Having created the data set extended attributes, we next create five extended attributes for each variable in ADVS. Those five extra attributes are: *Name*, *Core*, *Source*, *DataType*, and *Derivation*.

Hard-coding the numerous name-value pair individually can be rather unwieldy, so in order to automate the process, the code below employs reading the values from a data set (XPARMS) and feeding into variables in Proc DATASET within a DOSUBL function:

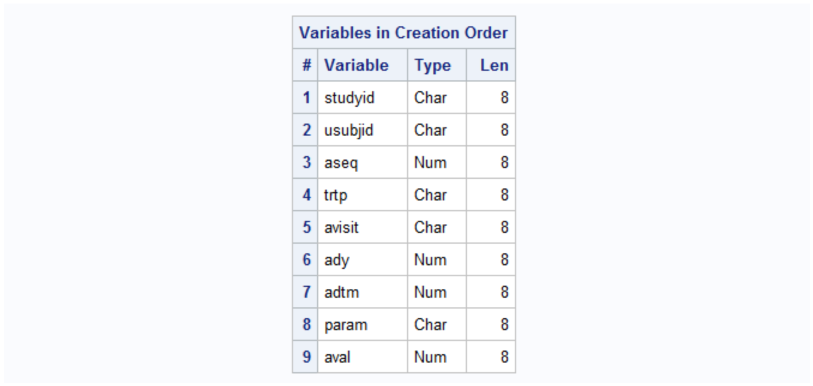
```
data xparms;
infile datalines dlm=" ";
length name $8 core $4 source $4 datatype $10 derivation $20;
input (name core source datatype derivation) ($);
datalines;
studyid Req ADSL text adsl.studyid
usubjid Req ADSL text adsl.usubjid
aseq Perm VS integer  vs.vsseq
trtp Req ADSL text adsl.trt01p
avisit Perm VS text vs.visit
ady Perm VS integer vs.vsdy
adtm Perm VS datetime vs.vsstdtc
param Req VS text vs.vstest
aval Req VS float vs.stresn
;
run;
```

```
data _null_;
set xparms;
rc=dosubl("proc datasets lib=work nolist;modify ADVS;xattr add var " ||name||"
("||"core='"||core||"' source='"||source||"' Datatype='"||Datatype||"'
Derivation='"||Derivation||"');run;quit;");
run;
```

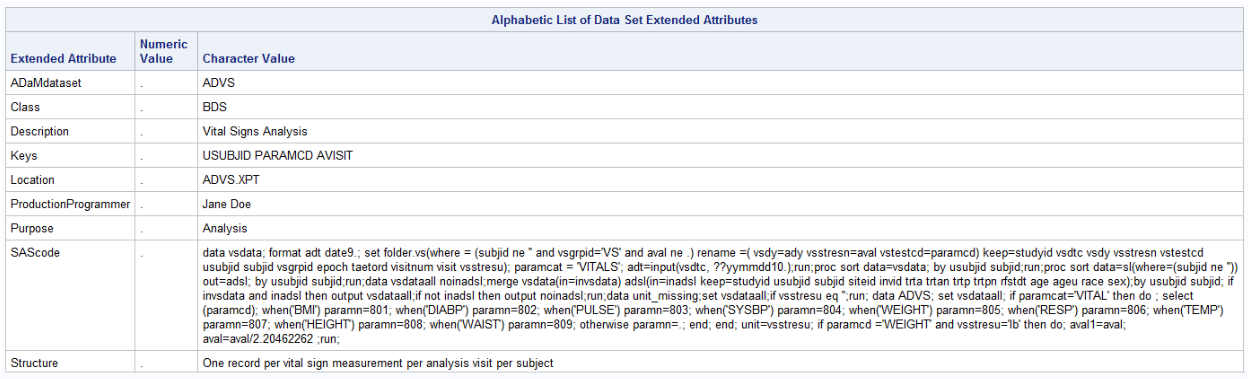
OUTPUT

The newly-embedded custom attributes for both data set and variables can be displayed with Proc CONTENTS as shown below:

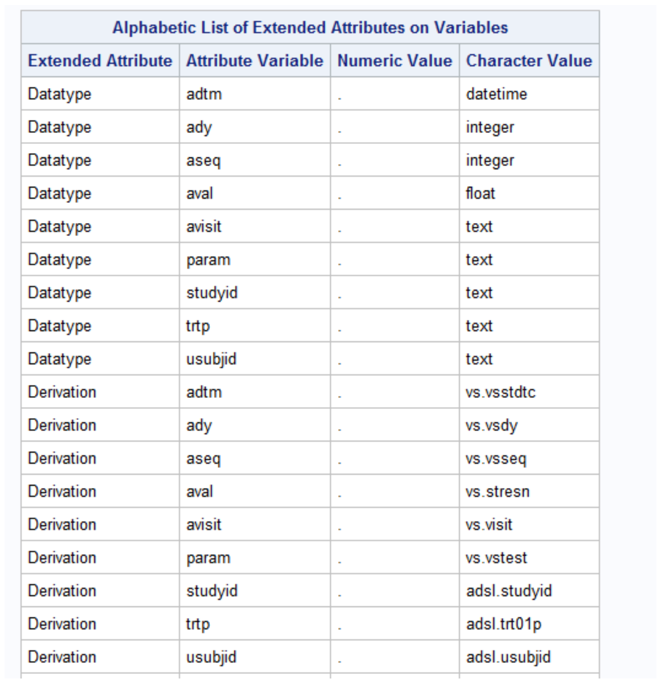
```
        proc contents data=advs varnum;
        run;
```



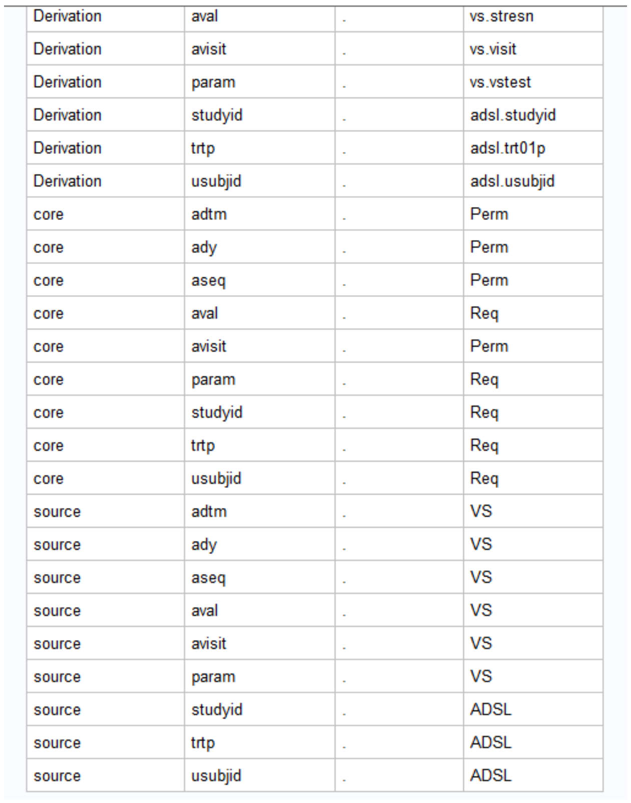
Variables in Creation Order			
#	Variable	Type	Len
1	studyid	Char	8
2	usubjid	Char	8
3	aseq	Num	8
4	trtp	Char	8
5	avisit	Char	8
6	ady	Num	8
7	adtm	Num	8
8	param	Char	8
9	aval	Num	8



Alphabetic List of Data Set Extended Attributes		
Extended Attribute	Numeric Value	Character Value
ADaMdataset	.	ADVS
Class	.	BDS
Description	.	Vital Signs Analysis
Keys	.	USUBJID PARAMCD AVISIT
Location	.	ADVS.XPT
ProductionProgrammer	.	Jane Doe
Purpose	.	Analysis
SAScode	.	data vsdata; format adt date9.; set folder.vs(where = (subjid ne '' and vsgrpid='VS' and aval ne .) rename =(vsdy=ady vsstresn=aval vstestcd=paramcd) keep=studyid vsdtc vsdy vsstresn vstestcd usubjid subjid vsgrpid epoch taetord visitnum visit vsstresu); paramcat = 'VITALS'; adt=input(vsdtc, ??yymmdd10.);run;proc sort data=vsdata; by usubjid subjid;run;proc sort data=sl(where=(subjid ne '')) out=adsl; by usubjid subjid;run;data vsdataall noinadsl;merge vsdata(in=invsdata) adsl(in=inadsl keep=studyid usubjid subjid siteid invid trta trtan trtp trtpn rfstdt age ageu race sex);by usubjid subjid; if invsdata and inadsl then output vsdataall;if not inadsl then output noinadsl;run;data unit_missing;set vsdataall;if vsstresu eq '';run; data ADVS; set vsdataall; if paramcat='VITAL' then do ; select (paramcd); when('BMI') paramn=801; when('DIABP') paramn=802; when('PULSE') paramn=803; when('SYSBP') paramn=804; when('WEIGHT') paramn=805; when('RESP') paramn=806; when('TEMP') paramn=807; when('HEIGHT') paramn=808; when('WAIST') paramn=809; otherwise paramn=.; end; end; unit=vsstresu; if paramcd ='WEIGHT' and vsstresu='lb' then do; aval1=aval; aval=aval/2.20462262 ;run;
Structure	.	One record per vital sign measurement per analysis visit per subject



Alphabetic List of Extended Attributes on Variables			
Extended Attribute	**Attribute Variable**	**Numeric Value**	**Character Value**
Datatype	adtm	.	datetime
Datatype	ady	.	integer
Datatype	aseq	.	integer
Datatype	aval	.	float
Datatype	avisit	.	text
Datatype	param	.	text
Datatype	studyid	.	text
Datatype	trtp	.	text
Datatype	usubjid	.	text
Derivation	adtm	.	vs.vsstdtc
Derivation	ady	.	vs.vsdy
Derivation	aseq	.	vs.vsseq
Derivation	aval	.	vs.stresn
Derivation	avisit	.	vs.visit
Derivation	param	.	vs.vstest
Derivation	studyid	.	adsl.studyid
Derivation	trtp	.	adsl.trt01p
Derivation	usubjid	.	adsl.usubjid



Derivation	aval	.	vs.stresn
Derivation	avisit	.	vs.visit
Derivation	param	.	vs.vstest
Derivation	studyid	.	adsl.studyid
Derivation	trtp	.	adsl.trt01p
Derivation	usubjid	.	adsl.usubjid
core	adtm	.	Perm
core	ady	.	Perm
core	aseq	.	Perm
core	aval	.	Req
core	avisit	.	Perm
core	param	.	Req
core	studyid	.	Req
core	trtp	.	Req
core	usubjid	.	Req
source	adtm	.	VS
source	ady	.	VS
source	aseq	.	VS
source	aval	.	VS
source	avisit	.	VS
source	param	.	VS
source	studyid	.	ADSL
source	trtp	.	ADSL
source	usubjid	.	ADSL

CONCLUSION

By embedding CDISC metadata into SAS® data sets as system attributes, the metadata can be retrieved from Dictionary Tables or Proc CONTENTS output, making the creation of Define.XML much more straightforward. SDTM programmers can utilize this tool for embedding CDISC metadata in SDTM data sets such that ADaM programmers using the data sets would already have these metadata as they design and program define.xml for ADaM datasets.

REFERENCE

Olson, Diane. “Developer Reveals: Extended Data Set Attributes”.
Proceedings of SAS® Global Forum 2013 Conference, Paper 135-2013.

http://support.sas.com/resources/papers/proceedings13/135-2013.pdf

CONTACT INFORMATION

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

Joseph W. Hinson, PhD
inVentiv Health
504 Carnegie Center
Princeton, NJ, 08540
1-609-951-6596
joehinson@outlook.com