Paper 135-2013

Developer Reveals: Extended Data Set Attributes

Diane Olson, SAS Institute Inc., Cary NC

ABSTRACT

Have you ever wanted to save non-data information with your data set? Now, with SAS® 9.4, you can. Extended attributes allow you to store information related to a data set or to a particular variable in a data set. Do you want to store a description of a variable or the formula used to produce the variable value? Do you need to save a URL that specifies information about your data set? Do you want to store the SAS® code that created a particular data set? This presentation by the developer shows you how to do all of that and more with extended attributes.

INTRODUCTION

Every data set and every variable in a data set has attributes. Data set attributes include the name, member type, engine, created and modified date and time, label, compression, encryption, whether the data set is sorted, and the list goes on. For variables, the list is somewhat shorter and includes the variable name, type, length, format, informat, label and whether the variable is a part of an index. SAS automatically defines and stores these attributes with the data set.

Extended attributes are attributes that you define for your data set or variables and have specific meaning to you. You decide on the name of the attribute, its meaning to you and your data, and the value that is assigned to that attribute.

DEFINING AN EXTENDED ATTRIBUTE

The best way to understand extended attributes is with a simple example. Our example data set holds information about a high school class' physics labs. One of the variables in our data set is Volume. The value contained in the Volume variable might change over time depending on what type of volume it holds. Currently it holds the volume of a pyramid. The calculations of the volume come from the formula V = 1/3bh. We could create a separate variable to hold this information, of course, but because the formula is related directly to this variable, it would be good to tie the formula to the Volume variable itself. With extended attributes, we can do that.

There are two varieties of extended attributes – those on the data set itself, and those on variables. There is no limit to the number of extended attributes that you can have on either a data set or a single variable, except for memory and disk limitations. Variable attributes do not have a value for each value of the variable. In this example, we are stating that all of the Volume values are calculated by this one formula.

The DATASETS procedure allows us to create and control extended attributes. Just like when defining an index or an integrity constraint, we use the MODIFY statement in PROC DATASETS. The XATTR statements are the subgroup of statements under the MODIFY statement that control the extended attributes. Here is the syntax to define one extended attribute for a data set and one extended attribute for a variable:

```
XATTR ADD DS attribute_name=attribute_value;
```

or

```
XATTR ADD VAR variable_name (attribute_name=attribute_value);
```

The attribute value can be character or numeric. If it is character, the value must be enclosed in quotation marks. The syntax can be expanded to specify multiple *attribute_name=attribute_value* pairs. You can also specify more than one *variable_name* with subsequent values for attribute names and values. Let us define our character extended attribute, Formula, on the Volume variable for the Mylib.Geometric data set:

```
proc datasets lib=mylib;
   modify geometric;
      xattr add var volume ( formula = "V=1/3bh" );
   run;
quit;
```

If we wanted, we could define the same extended attribute for each of the variables in the data set, using the same extended attribute name. We could also use the same extended attribute name for the data set. However, you cannot use the same name twice for a particular variable or for the data set. Using the XATTR ADD statement, if the Formula extended attribute already existed for the Volume variable, we would see an ERROR returned.

In SAS 9.4, the BASE engine is the only engine that supports extended attributes. The META LIBNAME engine and the REMOTE engine also support extended attributes when the underlying engine is the BASE engine. Let us assume that Mylib is a libref that is assigned with the BASE engine for the purposes of this paper. Now that we have an extended attribute defined, when we submit a PROC CONTENTS statement for the Geometric data set, we will see an output section that describes our variable extended attribute:

```
              Alphabetic List of Extended Attributes on Variables

                                                          Numeric    Character
Extended Attribute              Attribute Variable        Value      Value

formula                         volume                          .    V=1/3bh
```

Output 1. PROC CONTENTS Output for a Variable Extended Attribute

Because our variable extended attribute has a character value, the Numeric Value field is set to missing. If Formula had a numeric value instead, then the value would be listed under Numeric Value and the Character Value would be set to blank, indicating a character missing value. The extended attribute has either the Numeric Value or Character Value populated.

Each engine that supports extended attributes might implement it in a different way. The BASE engine creates an auxiliary file with the name of the data set and the extension of sas7bxat. On disk, we see the presence of the geometric.sas7bxat file. Using PROC DATASETS for the Mylib library, we see that the extended attributes member has a member type of EXTATTR:

```
                          Gen  Member       File
                 #  Name  Num  Type         Size  Last Modified

                 1  GEOMETRIC     DATA       131072  01/25/2013 08:44:57
                    GEOMETRIC     EXTATTR     17408  01/25/2013 08:44:57
```

Output 2. PROC DATASETS Output for a BASE Engine Data Set with Extended Attributes

Previously, we learned that if the Formula extended attribute already exists for the Volume variable, the XATTR ADD statement would return an ERROR. However, if you wish to add or update an extended attribute, whether it already exists or not, you can use the XATTR SET statement. If the extended attribute already exists, its value is replaced by the value that is specified in the SET statement. If it does not exist, it is added. The XATTR SET syntax greatly resembles the XATTR ADD statement. Here we show how multiple definitions and updates are possible in a single statement:

```
XATTR SET DS attribute_name-1=attribute_value-1
            <attribute_name-2=attribute_value-2 ...>;
```

or

```
XATTR SET VAR variable_name-1 (attribute_name-1=attribute_value-1
                               <attribute_name-2=attribute_value-2> ...)
             <variable_name-2 (attribute_name-1=attribute_value-1
                               <attribute_name-2=attribute_value-2> ...)...>;
```

As with all XATTR statements, if the *attribute_value* is character, you must use quotation marks.

Assume we would like to add the website of the physics experiments as the Url data set extended attribute, but only if it does not exist. We use the XATTR ADD rather than the XATTR SET. We also want to set other data set and variable extended attributes at the same time. Here we define or update several extended variable attributes named Units, a character variable extended attribute named ConstraintForce, a numeric data set extended attribute named Experiments, and a character data set extended attribute named Description:

```
proc datasets lib=mylib;
   modify geometric;
```

```
      xattr add ds url="www.highschoollabs.com/physics1";
      xattr set var force ( units = "Newton" );
      xattr set var work ( units = "Joule" constraintForce = "magnetic" );
      xattr set var speed ( units = "m/s" );
      xattr set ds experiments = 5 description = "Physics lab, 2nd quarter " ;
   run;
quit;
proc contents data=mylib.geometric; run;
```

PROC CONTENTS OUTPUT

Note that when we look at the PROC CONTENTS output of Mylib.Geometric created from the code above, we now see two sections for extended attributes:

Alphabetic List of Data Set Extended Attributes

Extended Attribute	Numeric Value	Character Value
description	.	Physics lab, 2nd quarter
experiments	5	
url	.	www.highschoollabs.com/physics

Alphabetic List of Extended Attributes on Variables

Extended Attribute	Attribute Variable	Numeric Value	Character Value
constraintForce	work	.	magnetic
formula	volume	.	V=1/3bh
units	force	.	Newton
units	speed	.	m/s
units	work	.	Joule

Output 3. PROC CONTENTS Output for a Data Set Showing Data Set and Variable Extended Attributes

Data set and variable extended attributes appear in separate tables in the PROC CONTENTS output. To use ODS output to store the results in data sets, use the following code:

```
ods output
          ExtendedAttributesDS=mylib.mydsattr
          ExtendedAttributesVar=mylib.myvarattr;
ods listing close;
proc contents;run;
ods listing;
```

Two data sets are created, Mylib.Mydsattr and Mylib.Myvarattr. A PROC PRINT of the first data set shows the data set extended attributes:

Obs	Member	Extended Attribute	AttributeCharValue	Attribute NumValue
1	MYLIB.GEOMETRIC	description	Physics lab, 2nd quarter	.
2	MYLIB.GEOMETRIC	experiments		5
3	MYLIB.GEOMETRIC	url	www.highschoollabs.com/physics	.

Output 4. PROC PRINT Output of the Mylib.Mydsattr Data Set Showing the ExtendedAttributesDS ODS Output Object

The second data set contains one additional variable, AttributeVariable, to hold the variable name for each variable extended attribute.

Obs	Member	Extended Attribute	Attribute Variable	Attribute CharValue	Attribute NumValue
1	MYLIB.GEOMETRIC	constraintForce	work	magnetic	.
2	MYLIB.GEOMETRIC	formula	volume	V=1/3bh	.
3	MYLIB.GEOMETRIC	units	force	Newton	.
4	MYLIB.GEOMETRIC	units	speed	m/s	.
5	MYLIB.GEOMETRIC	units	work	Joule	.

Output 5. PROC PRINT Output of the Mylib.Myvarattr Data Set Showing the ExtendedAttributesVar ODS Output Object

UPDATING AND REMOVING AN EXTENDED ATTRIBUTE

We have already seen that we can update an existing extended attribute if we use the XATTR SET statement. However, there might be times that we want to update the extended attribute value only if it already exists. In those instances, we use the XATTR UPDATE statement. The statement's syntax is the same as that of XATTR ADD or XATTR SET:

```
XATTR UPDATE DS attribute_name-1=attribute_value-1
              <attribute_name-2=attribute_value-2 ...>;
```

or

```
XATTR UPDATE VAR variable_name-1 (attribute_name-1=attribute_value-1
                                  <attribute_name-2=attribute_value-2> ...)
               <variable_name-2 (attribute_name-1=attribute_value-1
                                  <attribute_name-2=attribute_value-2> ...)>...;
```

If we use XATTR UPDATE and the specified extended attribute does not exist, an ERROR is returned.

We can also remove an extended attribute. Once an extended attribute is removed, you cannot recover its value. To remove an extended attribute, or a group of extended attributes, use the XATTR REMOVE statement. Note that an *attribute_value* is not needed.

```
XATTR REMOVE DS attribute_name-1 <attribute_name-2 ...>;
```

or

```
XATTR REMOVE VAR variable_name-1 (attribute_name-1 <attribute_name-2>...)
               <variable_name-2 (attribute_name-1 <attribute_name-2>...)>...;
```

Note that we can also specify multiple names to remove extended attributes in one statement:

```
proc datasets lib=mylib;
   modify geometric;
      xattr remove var work ( constraintForce units );
      xattr remove ds description;
   run;
quit;
```

One data set extended attribute, Description, is removed, and two extended attributes, ConstraintForce and Units, are removed from the Work variable.

DELETING ALL EXTENDED ATTRIBUTES

You can remove extended attributes one at a time, or if you would like to delete them all, you can use the XATTR DELETE statement:

```
proc datasets lib=mylib;
   modify geometric;
      xattr delete;
   run;
quit;
```

All extended attributes are deleted. The extended attributes cannot be recovered after this statement.

OPTIMIZING EXTENDED ATTRIBUTE STORAGE

In the BASE engine (other engines might vary), the extended attribute character values are stored in segments. If the value is larger than the segment size, more than one segment is used to store the value. Therefore, there is no hard limit on the length of the character value. The length of the segment affects the size of the EXTATTR file on disk. The segment length is controlled by the SEGLEN option when using the XATTR OPTIONS statement:

XATTR OPTIONS <SEGLEN=number_of_bytes>;

SEGLEN= is the segment length that is used to store all character extended attribute values for a particular data set. The default is 256 bytes, and the values between 1 and 32760 can be specified. The specified value might be adjusted by SAS for alignment or other factors. You can see the value by using PROC CONTENTS and looking at the EXTATTR Segment Length property.

```
proc datasets lib=mylib;
   modify geometric;
      xattr options seglen = 80;
      xattr add var volume ( formula = "V=1/3bh" );
   run;
quit;
```

The CONTENTS Procedure

Data Set Name	MYLIB.GEOMETRIC	Observations	1
Member Type	DATA	Variables	1
Engine	V9	Indexes	0
Created	01/25/2013 15:56:49	Observation Length	8
Last Modified	01/25/2013 15:56:49	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label		EXTATTR Segment Length	80
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Output 6. PROC CONTENTS Output Showing the EXTATTR Segment Length Value

Once an extended attribute is created for a data set, the SEGLEN= value cannot be changed unless all of the extended attributes are removed. Setting the optimal SEGLEN= value depends on your data. For example, if you have character values that you know are always less than 80 characters, setting SEGLEN= 80 would save 136 characters per extended attribute. If you have an occasional value that goes over 80, the value can still be stored, but it requires multiple segments. It also requires processing time to put the character segments back together when the value is used. Knowing your data enables you to set the SEGLEN= option to get the right balance between processing time and disk space used.

EXTENDED ATTRIBUTES THROUGHOUT THE SAS SYSTEM

Now that you have your extended attributes set on your data set, what happens when you process that data set with the DATA step and other procedures in SAS?

PROC COPY, PROC CPORT, and PROC CIMPORT always copy the extended attributes to a new data set, provided that the output library's engine supports extended attributes. Of course, engines from releases prior to SAS 9.4 do not support them, so you might see something like this:

```
14   libname v6 v6 '.';
NOTE: Libref V6 was successfully assigned as follows:
      Engine:        V6
      Physical Name: C:\Users\sasddo
15   proc copy in=work out=v6;run;

NOTE: Copying WORK.A to V6.A (memtype=DATA).
WARNING: Engine V6 does not support extended attribute operations.
NOTE: There were 1 observations read from the data set WORK.A.
NOTE: The data set V6.A has 1 observations and 1 variables.
WARNING: Extended attributes could not be added to V6.A.DATA.
```

Output 7. Warning When Copying Extended Attributes Using An Engine That Does Not Support Extended Attributes

The data set itself is copied successfully, but the extended attributes are not, as the warning indicates.

PROC SORT propagates extended attributes to the sorted output data set. However, they are not propagated to any alternative output data sets that you create using the DUPOUT= or the UNIOUT= PROC SORT options.

With the DATA step, variable extended attributes are automatically passed from the input data set to the output data set. Data set extended attributes are not propagated. If two input data sets that contain extended attributes are combined using a SET statement, SAS preserves the variable extended attributes from the first data set that has extended attributes, and applies those attributes to the output data set. Only the variable extended attributes from the first data set with extended attributes are applied to any output data sets in the DATA step. See below for an example.

```
libname mylib v9 'c:\temp';
data mylib.geometric mylib.geometric3 mylib.noattrs;
   volume = 300;force=5; work=4;speed=800;
run;

/* Set up our GEOMETRIC data set with the "real" data. */

proc datasets lib=mylib nolist;
   modify geometric;
      xattr add var volume ( formula = 'V=1/3bh' );
      xattr add ds url="www.highschoollabs.com/physics1";
      xattr add var force ( units = "Newton" );
      xattr add var work ( units = "Joule" constraintForce = "magnetic" );
      xattr add var speed ( units = "m/s" );
      xattr add ds experiments = 5 description = "Physics lab, 2nd quarter " ;
   run;
quit;

/* Set up GEOMETRIC3 with the same extended attributes, plus two SPECIAL attributes.*/

proc datasets lib=mylib nolist;
   modify geometric3;
      xattr add var volume ( formula = 'geo3' );
      xattr add ds url="geo3";
      xattr add var force ( units = "geo3" );
      xattr add var work ( units = "geo3" constraintForce = "geo3" );
      xattr add var speed ( units = "geo3" );
      xattr add ds experiments = 42 description = "geo3 " ;
      xattr add var force ( special = "geo3" );
      xattr add ds special="geo3";
   run;
quit;

/* NEW1 from three data sets - one without extended attributes, two with. */
```

```
data new1;
   set mylib.noattrs mylib.geometric mylib.geometric3;
run;

/* Results are only variable attributes from the GEOMETRIC data set, the first data
set on the list with extended attributes. */

proc contents data=new1;
run;

/* Results when creating multiple data sets are the same for each data set – only
variable extended attributes from GEOMETRIC. */

data new2 new3;
   set mylib.noattrs mylib.geometric mylib.geometric3;
run;

/* Shows variable extended attributes from GEOMETRIC data set. */

proc contents data=new2;
run;

/* Shows variable extended attributes from GEOMETRIC data set. */

proc contents data=new3;
run;
```

In the creation of Work.New1, note that the first data set in the SET statement had no extended attributes. The second data set, Geometric, has both data set and variable extended attributes. The third data set, Geometric3, has all of the same extended attributes that Geometric has, plus two unique extended attributes named Special. Work.New1 has only the variable extended attributes from Geometric, which is the first data set in the SET statement that had extended attributes. No data set extended attributes were propagated.

PROC SQL propagates variable extended attributes for the CREATE TABLE AS and ALTER TABLE statements. Data set extended attributes are not propagated. In the example below, the New1 table inherits the variable extended attributes from Mylib.Geometric:

```
proc sql;
   create table new1 as select * from mylib.geometric;
quit;
proc contents data=new1;
run;
```

In PROC APPEND, if the data set that is specified by BASE= option is nonexistent and the data set that is specified by the DATA= option has extended attributes, the new BASE= option data set inherits the extended attributes from the DATA= option data set (assuming that the engine for the BASE= option data set supports extended attributes). Note that with PROC APPEND, the data set extended attributes as well as the variable extended attributes are propagated. If the BASE= option data set already exists, no extended attributes are propagated.

```
proc append base=noexist data=mylib.geometric;
run;
proc contents data=noexist;
run;
```

USING AN EXTENDED ATTRIBUTE TO KEEP THE SAS PROGRAM WITH THE DATA SET

As promised in the abstract, you can keep the SAS program that created a data set in the data set as an extended attribute. Our SAS program, which resides at c:\temp\geometry.sas, creates a very simple data set, Mylib.Geometric. This program creates the Formula extended attribute for the Volume variable followed by PROC CONTENTS:

```
libname mylib v9 'c:\temp';

data mylib.geometric;
   volume = 300;
run;

proc datasets lib=mylib nolist;
   modify geometric;
      xattr add var volume ( formula = 'V=1/3bh' );
   run;
quit;

proc contents data=mylib.geometric;
run;
```

Next, we read geometry.sas. We retain everything that is in the file, including end-of-line characters.

```
data inputsas;
  length text $32760;
  infile 'c:\temp\geometry.sas' lrecl=32760  recfm=f length=lng truncover ;
  input @;input @1 text $varying32760.lng;
  keep text ;
run;
```

We set the length of the Text variable and the LRECL= option to 32K in order to get as many characters as possible into the Text variable. The default record format is varying, but we need to use a fixed record format in order to include the newline characters. Using the LENGTH= option allows us to obtain and use the actual length, preventing any warnings later on. The TRUNCOVER option allows the character value to be assigned even if it is shorter than expected. The Lng variable holds the actual length, which is used with the $VARYING. format. The Text variable holds all of the characters in the geometry.sas file, if the file contains 32760 characters or less. If it holds more, there is more than one observation in the Inputsas data set that holds the remaining data.

```
proc sql noprint;
   select text into :mytext from inputsas;
quit;
```

Here we use PROC SQL to take the value that is stored in the Text variable and place it into the macro variable Mytext. Note that we did not handle the case when the SAS program is longer than 32760 bytes and the value is held in more than one observation. Now that we have the value that we need (the geometry.sas program) in a macro variable, we can add it to the Geometry data set as a data set extended attribute value:

```
libname mylib v9 'c:\temp';
options noquotelenmax;
proc datasets lib=mylib;
   modify geometric;
      xattr add ds sascode="%STR(&mytext)";
   run;
quit;
```

We need the option NOQUOTELENMAX so that we do not get warnings about exceeding the length of a quoted string. %STR is used to mask special characters should they be present in the input SAS program. When the PROC DATASETS code finishes, there is a new data set extended attribute named Sascode that contains the contents of the geometry.sas file. Of course, the data does not automatically update when the geometry.sas file changes, but should you want to update the value, you can use an XATTR UPDATE substatement to do so.

CONCLUSION

With extended attributes, new to SAS 9.4, it is easy to associate non-data numeric and character information with a data set and with variables within that data set. PROC DATASETS allows you to create and manage the extended attributes, and other procedures in SAS allow the extended attributes to propagate. Use PROC CONTENTS to view the attributes. With the addition of ODS output, you can use those values in your SAS program. Your imagination is the limit for the non-data information that you can now store with your data set.

ACKNOWLEDGMENTS

Thanks, so much, to these wonderful folks who helped with this paper: Brad Richardson, Ed Vlazny, Rick Langston, Kanthi Yedavalli, Lisa Brown, Darrell Massengill, Elizabeth Maldonado and Miguel Bamberger. Apparently it takes a village.

CONTACT INFORMATION

If you have any questions or comments, please contact the author at:

Diane Olson
SAS Campus Drive
SAS Institute Inc.
Cary NC 27513
E-mail: Diane.Olson@sas.com