Paper 1980-2015

A Practical Guide to SAS® Extended Attributes

Chris Brooks, Melrose Analytics Ltd

ABSTRACT

All SAS® data sets and variables have standard attributes. These include items such as creation date, engine, compression and sort information for data sets, and format and length information for variables. However, for the first time in SAS 9.4, the developer can add their own customized attributes to both data sets and variables. This paper shows how these extended attributes can be created, modified, and maintained. It suggests the sort of items that might be candidates for use as extended attributes and explains in what circumstances they can be used. It also provides a worked example of how they can be used to inform and aid the SAS programmer in creating SAS applications.

INTRODUCTION

When data sets and variables are created they automatically have certain attributes which are stored in the data set and which can be accessed by using the CONTENTS procedure. This metadata can often be used by SAS programmers to assist them in evaluating their data and in determining program flow. We will explore how the developer can add their own extended attributes to this metadata, what sort of data items should be considered for adding as extended attributes, how to manage them and how they can be effectively used to improve your SAS applications.

THE BASICS – CREATE, UPDATE AND DELETE EXTENDED ATTRIBUTES

Extended attributes for both data sets and variables are created, updated and removed by using the MODIFY and XATTR Statements in the DATASETS procedure. Extended Attribute types can be either text or numeric and (subject to memory and disk space limitations) there is no limit to the number of them you can create for either a data set or variable.

CREATING ATTRIBUTES

To create an extended attribute for a data set you use the following syntax in PROC DATASETS (if the attribute already exists an error will be generated):

```
XATTR ADD DS attribute_name=attribute_value <
attribute_name=attribute_value...>;
```

Example – The following code creates a single Extended Attribute on the data set mydata with the name “creator” and the value “Chris Brooks”:

```
Proc Datasets lib=mylib;
  modify mydata;
  xattr add ds creator=”Chris Brooks”;
Quit;
```

To create an extended attribute for a variable on a data set use the following syntax in PROC DATASETS (again if the attribute already exists an error will be generated):

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

```
    1=attribute_value-1 < attribute_name-2=attribute_value-2...>)  >;
```

Example – The following code creates a single Extended Attribute on the variable called “distance” on the data set mydata with the name “unit” and the value “Miles”

```
Proc Datasets lib=mylib;
  modify mydata;
  xattr add var distance (unit=”Miles”);
Quit;
```

DELETING ATTRIBUTES

To delete an extended attribute for a data set you use the following syntax in PROC DATASETS (if the attribute does not exist an error will be generated):

```
    XATTR REMOVE DS attribute_name-1 <attribute_name_2…>;
```

Example – The following code deletes the previously created Extended Attribute “creator” on the data set mydata:

```
Proc Datasets lib=mylib;
    modify mydata;
    xattr remove ds creator;
Quit;
```

To delete an extended attribute for a variable on a data set use the following syntax in PROC DATASETS (again if the attribute does not already exist an error will be generated):

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

Example – The following code deletes the previously created Extended Attribute “unit” on the variable “distance” on the data set mydata:

```
Proc Datasets lib=mylib;
    modify mydata;
    xattr remove distance (unit);
Quit;
```

It is also possibly to delete all the extended attributes on a data set and it’s variables with a single DELETE Statement:

```
Proc Datasets lib=mylib;
    modify mydata;
    xattr delete;
Quit;
```

UPDATING ATTRIBUTES

There are two ways to update the value of an extended attribute for both a data set attribute and a variable attribute. You can either use the UPDATE or SET Statements. There is a significant difference between the two. If you use the UPDATE Statement and the attribute doesn't exist an error will be generated. However if you use the SET Statement and the attribute doesn't exist it will be created. The syntax for updating and setting a data set extended attribute is:

```
XATTR SET DS attribute_name=attribute_value <
attribute_name=attribute_value...>;
```

```
XATTR UPDATE DS attribute_name=attribute_value <
attribute_name=attribute_value...>;
```

The following code updates the previously created extended attribute "creator" on the data set mydata using firstly the SET Statement and then the UPDATE Statement:

```
Proc Datasets lib=mylib;
   modify mydata;
   set DS creator="John Smith";
Quit;

Proc Datasets lib=mylib;
   modify mydata;
   update DS creator="David Jones";
Quit;
```

READING EXTENDED ATTRIBUTE VALUES

There are two ways we can read existing extended attribute values. Firstly we can use PROC CONTENTS to display them or write their values to a data set which we can subsequently access (Figure 1 below shows the partial output from a PROC CONTENTS statement detailing the name and value of extended attributes for both the data set and individual variables).

Figure 1

Alphabetic List of Data Set Extended Attributes

Extended Attribute	Numeric Value	Character Value
create_date	2004	
creator	.	C Brooks

Alphabetic List of Extended Attributes on Variables

Extended Attribute	Attribute Variable	Numeric Value	Character Value
measure	radius	.	equatorial
units	area	.	km2
units	density	.	g/cm3
units	distance	.	km
units	mass	.	Earth Mass
units	radius	.	km
units	volume	.	Earth Volume

Figure 1 Partial PROC CONTENTS Listing Showing Extended Attributes

The other way is to access one of the SASHELP views (SASHELP.VXATTR) or dictionary tables (DICTIONARY.XATTRS). In this paper we will use the SASHELP view. Figure 2 shows an extract from the view.

Figure 2

Partial Listing of SASHELP.VXATTR

Library Name	Member Name	Column Name	Extended Attribute Name	Extended Attribute Type	Extended Attribute Value
EXATTR	PLANETS		create_date	num	2004
EXATTR	PLANETS		creator	char	C Brooks
EXATTR	PLANETS	distance	units	char	km
EXATTR	PLANETS	radius	measure	char	equatorial
EXATTR	PLANETS	radius	units	char	km
EXATTR	PLANETS	area	units	char	km2
EXATTR	PLANETS	volume	units	char	Earth Volume
EXATTR	PLANETS	mass	units	char	Earth Mass
EXATTR	PLANETS	density	units	char	g/cm3

Figure 2 Partial Listing of SASHELP.VXATTR

For programming purposes it is easy to retrieve a single attributes value and store it in a SAS macro variable as below:

```
Proc Sql;
   select xvalue into :area_units
   from sashelp.vxattr
   where libname="EXATTR"
   and memname="PLANETS"
   and name="area"
   and xattr="units";
quit;
```

PRACTICAL USES OF EXTENDED ATTRIBUTES

Now that we have an understanding of how to create and maintain extended attributes we must ask ourselves how we can use them in a practical way. Firstly we need to decide what sort of data we can and should store there. It is important to remember that data set extended attributes apply to the whole data set and variable extended attributes apply to every record in the data set. Therefore they should not be used to store data itself but rather metadata (i.e. data about data). There should also be consideration given to using controlled vocabularies when both the names and values of extended attributes are defined. The following table shows some suggested attribute names and their allowable values.

Table 1

Attribute Type	Extended Attribute Name	Description	Allowable Value
Data Set	orig_creator	Original Creator	Initial + surname
Data Set	orig_create_date	Original Creation Date	Date
Data Set	Last_updater	Last person to Update File	Initial + surname
Data Set	update_date	Date Last Updated	Date
Data Set	version_no	Version Number	Integer > 0
Data Set	source_file	Source of Data	File name
Data Set	source_url	URL Source of Data	URL
Data Set	access_level	Level of Access	Public/Confidential
Data Set	status	Provisional or Final	Provisional/Final
Variable	units	Unit of Measurement	SI Unit
Variable	agg_method	Aggregation Method	sum, avg, count
Variable	formats_list	List of allowable formats	Valid SAS Formats
Variable	cont_disc	Continuous or Discrete	Cont/Disc

Table 1. Suggested Attributes

WAYS THESE ATTRIBUTES COULD BE USED

1. The attributes orig_creator, orig_create_date, last_updater, update_date and version_no could be used to create an audit trail if a program was written to take regular extracts from SASHELP.VXATTR holding the values of these against each data set.
2. Source_file and source_url could be used in data validation checks
3. Access_level and status could be used in a program to determine which files can be published at any particular point in time
4. Units can be used to ensure consistency when two files containing the same variable names are merged or compared
5. Agg_method could be used to distinguish between variables with values which can sensibly be summed (e.g.turnover) and which should be averaged (return on investment percentages)

6. Formats_list should be used to determine which SAS formats can be applied to variable values (e.g. date/time formats or currency)

A PRACTICAL EXAMPLE

For our example we shall use a scenario where we have two data sets holding information about planets and dwarf planets in our solar system. We wish to produce a combined file showing data gathered from both data sets. This could then be the source for further reports or analyses.

The two files both contain the same variables which are listed below

Figure 3

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
4	area	Num	8
7	density	Num	8
2	distance	Num	8
6	mass	Num	8
1	name	Char	15
3	radius	Num	8
5	volume	Num	8

Figure 3 Variable Listing

None of the variables have any labels or formats attached to them which can aid us in our analysis.

At first glance because both files contain the same variables with the same data types you would naturally think that they could be simply combined together to produce any desired result. However if we look at extracts from the files we will see that we have some issues with that approach.

Figure 4

List Data for EXATTR.PLANETS

name	distance	radius	area	volume	mass	density
Mercury	57909175	2439.64	75000000	0.06	0.055	5.430
Venus	108208930	6051.59	460000000	0.87	0.815	5.240
Earth	149597890	6378.10	510000000	1.00	1.000	5.515
Mars	227936640	3397.00	140000000	0.15	0.107	3.940
Jupiter	778412010	71492.68	64000000000	1321.30	318.000	1.330
Saturn	1426725400	60267.14	44000000000	763.59	95.000	0.700
Uranus	2870972200	25557.25	8100000000	63.09	14.000	1.300
Neptune	4498252900	24766.36	7700000000	57.74	17.000	1.760
Pluto	5906380000	1184.00	17000000	0.01	0.002	2.000

Figure 4 List Data for EXATTR.PLANETS

Figure 5

List Data for EXATTR.DWARF_PLANETS

name	distance	radius	area	volume	mass	density
Ceres	2.766	0.0738	2800000	.0005	.00016	2.08
Pluto	39.482	0.1800	17000000	.0070	.00220	2.00
Haumes	43.335	0.1000	6800000	.0010	.00070	2.60
Makemake	45.792	0.1100	6400000	.0010	.00030	1.40
Eris	67.668	0.1900	17000000	.0080	.00280	2.25

Figure 5 List Data for EXATTR.DWARF_PLANETS

The two most obvious issues are:

1. Pluto appears in both files; and
2. Some of the variables (distance and radius), although sharing the same name, are of vastly different orders of magnitude implying some issue which needs to be investigated

In order to try to resolve these problems we should look at the extended attributes for both files for clues. These (along with the code to retrieve them) are as follows:

```
Proc SQL;
   select *
   from sashelp.vxattr
   where libname="EXATTR";
quit;
```

Figure 6

Extended Attributes for Files in EXATTR Library

Library Name	Member Name	Column Name	Extended Attribute Name	Extended Attribute Type	Offset Into Extended Attribute Value	Extended Attribute Value
EXATTR	DWARF_PLANETS		create_date	num	0	2015
EXATTR	DWARF_PLANETS		source_url	char	0	http://en.wikipedia.org/wiki/List_of_gravitationally_rounded_objects_of_the_Solar_System
EXATTR	DWARF_PLANETS		creator	char	0	C Brooks
EXATTR	DWARF_PLANETS	distance	units	char	0	AU
EXATTR	DWARF_PLANETS	radius	measure	char	0	mean
EXATTR	DWARF_PLANETS	radius	units	char	0	Earth Radius
EXATTR	DWARF_PLANETS	area	units	char	0	km2
EXATTR	DWARF_PLANETS	volume	units	char	0	Earth Volume
EXATTR	DWARF_PLANETS	mass	units	char	0	Earth Mass
EXATTR	DWARF_PLANETS	density	units	char	0	g/cm3
EXATTR	PLANETS		create_date	num	0	2004
EXATTR	PLANETS		creator	char	0	C Brooks
EXATTR	PLANETS	distance	units	char	0	km
EXATTR	PLANETS	radius	measure	char	0	equatorial
EXATTR	PLANETS	radius	units	char	0	km
EXATTR	PLANETS	area	units	char	0	km2
EXATTR	PLANETS	volume	units	char	0	Earth Volume
EXATTR	PLANETS	mass	units	char	0	Earth Mass
EXATTR	PLANETS	density	units	char	0	g/cm3

Figure 6 Extended Attributes for Files in EXATTR Library

If we consider the first problem i.e. that of Pluto being present in both files, we will see that the create_date attribute has a different value in each of the two files. The Planets data set was created in 2004 and the Dwarf_Planets data set was created in 2015. This could imply that either one of the files was incorrect or something changed between the two dates. We can then look to see where the data

originated by referring to the source_url attribute of the Dwarf Planets data set (there is no attribute of this type for the Planets file). We can see that the data was gathered from a Wikipedia entry which, when referred to, tells us that before 2006 the Dwarf Planets category did not exist and it was only on it's creation that Pluto was moved from the "main list".

Regarding the second issue (that of some of the variables holding values of vastly different orders of magnitude) the extended attributes again provide the answer. Each of the numeric variables has an extended attribute called "units". This holds the unit of measurement for the data held in the variable and we can see that there are a number of differences in the units of measurement used by the two files for the same variable. For example the radius variable in the Planets data set holds its data in kilometers whereas in the Dwarf_Planets data set the unit is "earth radius" i.e. relative to the radius of earth. Therefore if we are to combine the data from these files we will need to convert the data into the desired common units.

We can use the following code to remove Pluto from the Planets file, make the necessary unit conversions and combine the two files:

```
data d_planets_conv;
   set exattr.dwarf_planets;
   distance=distance*149597871;
   radius=radius*6378.1;
run;

Proc SQL;
   create table planets
   as select *
   from exattr.planets
   where name not in
      (select name from exattr.dwarf_planets);
quit;

Proc Append base=planets data=d_planets_conv;
run;
```

This gives us our required output file which we can use for further processing.

Next we should amend the creator and create_date attributes of the new file

```
Proc Datasets lib=exattr;
   modify planets;
   xattr set ds creator="C Brooks"
   create_date=2015;
quit;
```

Finally we might also consider adding the formula used for converting the distance and radius values to consistent units to the dwarf planets file for future reference.

We should consider whether there are any other new attributes we should add to the file to help document it. For example we could add a source_program attribute containing the path and name of the SAS program we used to create the file

```
Proc Datasets lib=work;
   modify planets;
   xattr set ds source_file=
      "/folders/myfolders/SASGF2015ExtAttr/MakeData.sas";
quit;
```

CONCLUSION

Extended attributes can be an extremely powerful tool in the hands of the SAS programmer. We have seen how they can be used to understand and document data held in a file, how they can record the history of the data and which program or data source was used to create the file. There are many more items you can hold as extended attributes and uses that they can be put to. It is hoped that this paper stimulates the increased usage of extended attributes within the SAS community by providing practical ideas for adding extended attributes and scenarios in which they can be used.

REFERENCES

Olson, Diane. 2013. Developer Reveals: Extended Data Set Attributes. Proceedings of the SAS Global 2013 Conference, San Francisco, California.

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

CONTACT INFORMATION

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

Chris Brooks
Melrose Analytics Ltd
chrisbrooks@melroseanalytics.co.uk
www.melroseanalytics.co.uk