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):

```plaintext
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”:

```plaintext
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):

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

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

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

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 SASHHELP.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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>area</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>density</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>distance</td>
<td>Num</td>
<td>6</td>
</tr>
<tr>
<td>mass</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>name</td>
<td>Char</td>
<td>15</td>
</tr>
<tr>
<td>radius</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>volume</td>
<td>Num</td>
<td>8</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>name</th>
<th>distance</th>
<th>radius</th>
<th>area</th>
<th>volume</th>
<th>mass</th>
<th>density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercury</td>
<td>57909175</td>
<td>2439.64</td>
<td>750000000</td>
<td>0.06</td>
<td>0.055</td>
<td>5.430</td>
</tr>
<tr>
<td>Venus</td>
<td>108208930</td>
<td>6051.59</td>
<td>4600000000</td>
<td>0.87</td>
<td>0.815</td>
<td>5.240</td>
</tr>
<tr>
<td>Earth</td>
<td>149507990</td>
<td>6378.10</td>
<td>5100000000</td>
<td>1.00</td>
<td>1.000</td>
<td>5.515</td>
</tr>
<tr>
<td>Mars</td>
<td>227536040</td>
<td>3397.00</td>
<td>1400000000</td>
<td>0.15</td>
<td>0.107</td>
<td>3.940</td>
</tr>
<tr>
<td>Jupiter</td>
<td>778412010</td>
<td>7142.68</td>
<td>64000000000</td>
<td>1321.30</td>
<td>318.000</td>
<td>1.330</td>
</tr>
<tr>
<td>Saturn</td>
<td>1426725400</td>
<td>60267.14</td>
<td>440000000000</td>
<td>763.59</td>
<td>95.000</td>
<td>0.700</td>
</tr>
<tr>
<td>Uranus</td>
<td>2870072200</td>
<td>25557.25</td>
<td>810000000000</td>
<td>63.00</td>
<td>14.000</td>
<td>1.300</td>
</tr>
<tr>
<td>Neptune</td>
<td>4498252900</td>
<td>24766.36</td>
<td>770000000000</td>
<td>57.74</td>
<td>17.000</td>
<td>1.760</td>
</tr>
<tr>
<td>Pluto</td>
<td>5906380000</td>
<td>1184.00</td>
<td>170000000000</td>
<td>0.01</td>
<td>0.002</td>
<td>2.000</td>
</tr>
</tbody>
</table>

Figure 4 List Data for EXATTR планет
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;
```

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 its 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:

```sas
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

```sas
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

```sas
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

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

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.