

Dear Readers,

Although I am not in Technical Support, I have some news from that team to share with you. SAS Technical Support has begun using a new release of its internal problem-tracking system for U.S. customers. (The system will be rolled out for international use at a later date.) While most of the updates to the tracking system will be transparent to you, the format for tracking-entry numbers has changed.

Problems submitted before Aug. 27 will maintain the old 9-digit format (a prefix of two alphabetic characters followed by seven numeric digits). **Example:** us1234567

Problems submitted on Aug. 27 and beyond are being tracked with a new numbering format (10 numeric digits that always begin with the prefix '76'). **Example:** 7623456789

Read this [Web article](#) for more details. And here's a bit of trivia for the curious mind -- the prefix '76' represents the year that SAS was incorporated!



Shelley Sessoms

Editor, *SAS Tech Report*

Sample 1772: Creating Subfolders Using Hash Tables

This sample will create subdirectories "on the fly" by walking down the hierarchy and creating the subdirectory if it doesn't exist. It also demonstrates iterating through a hash table. It is easily adaptable as a macro function.

```

/***** BEGIN CODE *****/

/* Set a couple of macro variables: Parent directory and fiscal year
*/
%LET mreports = C:\temp\reports; * parent directory;
%LET mfy = 2008; * fiscal year;

DATA _null_;

/* Set the length of character variables */
LENGTH d dir $ 80 fsep $ 1;

/* fsep is character to separate directories */
RETAIN fsep "\";

/* Create the full path name (uses v.9 catx() function)
something like: c:\temp\reports\09jan2007\FY_2008 */
d = Catx(fsep, "&mreports", Put(Today(), DATE9.),
Catt('FY_', "&mfy"));

/* _n_ not necessary here, but... */
IF _n_ EQ 1 THEN DO; * define path hash table ;

/* Declare the hash table, Declare the iterator */
DECLARE hash hdir(hashexp: 3, ordered:"yes");
DECLARE hiter idir('hdir');

/* Define the key, p variable*/
rc = hdir.defineKey('p');

/* Define the data, dir variable and Define Done */
rc = hdir.defineData('dir');
rc = hdir.defineDone();

/* Avoid annoying log messages */
CALL missing(p, dir);
END;

/* Initialize the end of folder pointer, l */
len = Length(d);

/* Now break each folder (dir) into the hash table */
DO UNTIL(len LE 0); * parse the path, put in hash table;

/* Find next deepest folder (-1 starts at end and
searches forward) */
p = Find(d, fsep, "it", -len) ;

```

```

        /* Set dir to what's between the two fsep ("\"") */
        dir = Substr(d, IfN(p=0, 1,p+1), len-p);

        /* If the pointers at the start set the parent, */
        IF p = 0 THEN par = dir;

        /* Otherwise add the folder to the hash table */
        ELSE rc = hdir.add(key: p, data: dir);

        /* And decrement for next folder */
        len = p-1;
END;

/* Fird the first folder */
rc = idir.first();

DO WHILE(rc=0);

        /* Try to create it using DCreate() */
        newdir = Trimn(dcreate(dir, par));

        /* If it already exists (newdir is blank), set it
yourself */
        IF newdir = " " THEN newDir = Catx('\',par,dir);

        /* Set parent to the new directory */
        par = newdir;

        /* And go again */
        rc = idir.next();

        /* And done. */
END;

RUN;

/***** END CODE *****/

```

Reference Documentation:

- <http://support.sas.com/onlinedoc/913/getDoc/en/lrdict.hlp/a002588142.htm>
- <http://support.sas.com/onlinedoc/913/getDoc/en/lrcon.hlp/a002586295.htm>
- <http://www2.sas.com/proceedings/sugi31/241-31.pdf>

Links for the new functions:

- FIND Function - support.sas.com/onlinedoc/913/getDoc/en/lrdict.hlp/a002267763.htm
- IFN Function - support.sas.com/onlinedoc/913/getDoc/en/lrdict.hlp/a002604573.htm
- CATT Function - support.sas.com/onlinedoc/913/getDoc/en/lrdict.hlp/a002257057.htm
- TRIMN Function - support.sas.com/onlinedoc/913/getDoc/en/lrdict.hlp/a000212475.htm

- CATX Function - support.sas.com/onlinedoc/913/getDoc/en/lrdict.hlp/a002257076.htm

About the Author

Richard Wright began writing SAS code over 25 years ago keying code and data on punch cards as a student at University of Oregon. Since then he has worked at various agencies for the state of Texas. He is certified in Base and Advanced SAS.

Save up to 36 percent on SAS® training before Dec. 31

Now's the time to save up to 36 percent on 2008 training fees and plan for your future training needs when you purchase blocks of EPTO training units. Exercise the flexibility of using your EPTO units as needed for training services over 18 months. This special savings only lasts until Dec. 31, so enroll today!

Read more: <http://support.sas.com/training/us/epto.html>

What SAS® Administrators Should Know about Libraries, Metadata and SAS® Enterprise Guide® 4.1

Background

SAS Enterprise Guide 4.1 and SAS 9.1.3 Service Pack 4 were released in the spring of 2006. Among the enhancements offered by SAS Enterprise Guide 4.1 is better integration with SAS libraries that are defined in metadata. At the same time, SAS 9.1.3 Service Pack 4 introduced some behavior changes in the metadata library engine (META engine). This document provides details about those changes and behaviors, and provides guidance to SAS administrators who would like to see the libraries behave in a certain way.

About SAS Metadata Libraries

SAS libraries defined in metadata have two main components:

1. The library definition, which describes how to assign the library. This information includes the path, engine, and any library-specific options that are needed. These parts of the definition correspond to the syntax of the traditional LIBNAME statement that most SAS programmers are familiar with.

An administrator can attach authorization details to a library definition, controlling which users can view/access the contents of the library and change its contents.

2. The table definitions, which describe the tables and columns available within the library. Table definitions are added to metadata in three main ways:
 - Using SAS® Management Console, you can select the library and use the Import Tables feature, which allows you to connect to the library, view the physical tables that exist, and import the desired table definitions into metadata.
 - Using the METALIB procedure from within a SAS session, which is a programmatic method of importing table metadata, synchronizing metadata with the physical contents of the library.
 - Using SAS® Data Integration Studio to build a data warehouse. The process of building a data warehouse updates metadata about libraries, tables, and columns.

Administrators can also attach authorization details to table definitions, controlling which users can access specific tables, and even specific columns, within those tables.

Library and table metadata is a prerequisite for many data-related features in SAS clients and solutions. For example, if you want to provide a business view of data by defining information maps in SAS® Information Map Studio, you must first add metadata about the tables and columns that feed into those information maps.

About the Metadata Library Engine (META Engine)

The META engine is a special SAS library engine that enforces the metadata-centric view of a SAS library. The engine provides a level of indirection between your SAS programs and the physical location of your data, which allows you to reference a library definition that exists in metadata.

For example, this LIBNAME statement:

```
libname hr meta library="Human Resources"  
rename="Foundation";
```

looks up the library definition named "Human Resources" within metadata and provides access to its contents via the HR libref. The physical location of the tables is not evident in this statement. For example, the library definition might refer to SAS data sets in a file path using the BASE engine, or it might refer to a DBMS source using the ORACLE engine.

The tables and columns surfaced in the resulting HR libref reflect the tables and columns defined in metadata and the permissions that you, as an authenticated user, have to view the data.

META Engine Behavior – Read Only by Default

The META engine, by default, prevents your SAS programs from modifying the library contents in a way that would cause the data to become out of sync with the registered metadata. Attempts to do so result in an error message similar to the following:

```
ERROR: You cannot create or delete datasets, views or  
indexes in this  
mode. Try the option METAOUT=DATA. Use Proc Metalib  
to create  
metadata for datasets.
```

For example, using the example HR library definition from earlier, this SAS program would fail:

```
data hr.newdata;  
set hr.empinfo (where=(gender="F" ) );  
run;
```

Adding or Modifying Tables via the META Engine

The default behavior of the META engine is to provide a purely metadata-centric view of your library. However, you can specify an additional option on the LIBNAME statement to produce a sort of hybrid view of the library. The METAOUT=DATA option allows the META engine to show any additional tables that don't exist in metadata, but do exist in the physical library location. Here is an example statement:

```
libname hr meta library="Human Resources"  
rename="Foundation" metaout=data;
```

With the METAOUT=DATA option in place, you can add and modify tables in the library. While library engine, in this mode, allows access to "non-metadata" tables, the engine still enforces the permissions defined on those tables and columns that do exist in metadata. That is, the METAOUT=DATA option does not open access to tables and columns where READ permission has been set to Deny in metadata.

For example, with the proper permissions in place, this SAS program adds a table to the physical library, but not update the metadata:

```
libname hr meta library="Human Resources" metaout=data;
data hr.newdata;
set hr.empinfo(where=(gender="F"));
run;
```

Here is an example SAS log:

```
NOTE: Libref HR was successfully assigned as follows:
      Engine:          META
      Physical Name:
14      libname hr meta library="Human Resources"
metaout=data;
15      data hr.newdata;
16      set hr.empinfo(where=(gender="F"));
NOTE: HR.EMPINFO was found in the SAS Metadata
Repository.
17      run;

NOTE: There were 104 observations read from the data set
HR.EMPINFO.
      WHERE gender='F';
NOTE: The data set HR.NEWDATA has 104 observations and 17
variables.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time           0.03 seconds
```

Synchronizing Physical Data and Table Metadata

If your SAS program makes changes to the contents of the library in METAOUT=DATA mode, those changes are **not** reflected in the metadata. In order to synchronize the metadata with the physical changes made during your program, you must:

- Use SAS Management Console to import the new or changed tables into the library definition
- Use the METALIB procedure to update metadata definitions with the physical table attributes.
- Use the “Update Library Metadata” task for SAS Enterprise Guide, available for download from:
www.sas.com/apps/demosdownloads/setupcat.jsp?cat=SAS+Enterprise+Guide

This task is a simple user interface that helps generate METALIB syntax for you. For example, this SAS program synchronizes the metadata to add the additional table defined earlier:

```

/* sync up metadata with physical library */
/* requires Create permissions for the library */

libname _temp meta repname="Foundation" library="Human
Resources" metaout=data;

proc metalib;
  omr (library="Human Resources"
      metarepository="Foundation");
  update_rule=(delete);
  report;
run;

libname _temp clear;

```

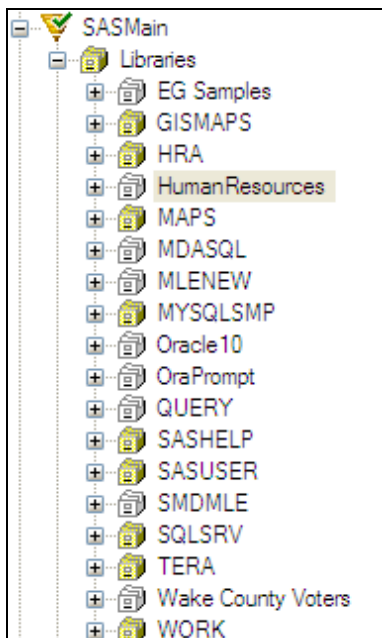
SAS Enterprise Guide 4.1 and Library Assignments

Note: All library behaviors described for SAS Enterprise Guide 4.1 also apply to SAS® Add-In for Microsoft Office 2.1.

SAS Enterprise Guide 4.1 offers improved integration with libraries defined in metadata, including new options for controlling how libraries are assigned.

When you view a SAS server in SAS Enterprise Guide in the server view, you see a list of libraries. This list includes assigned libraries – those that are built-in, pre-assigned, or assigned in an autoexec program. The list also includes *potential* libraries – those defined in metadata that haven't actually been assigned or accessed yet, but that you are authorized to view.

This example view shows a SAS server with a mix of assigned and potential libraries:



The library icons that are shaded yellow represent the **assigned** libraries. Those that are white represent the **potential** or **unassigned** libraries. (Note: it's also possible for a "potential" library to be displayed with a **red X** indicator, indicating that an error occurred when trying to assign it. Such errors can happen as a result of incorrect library paths or invalid library options.)

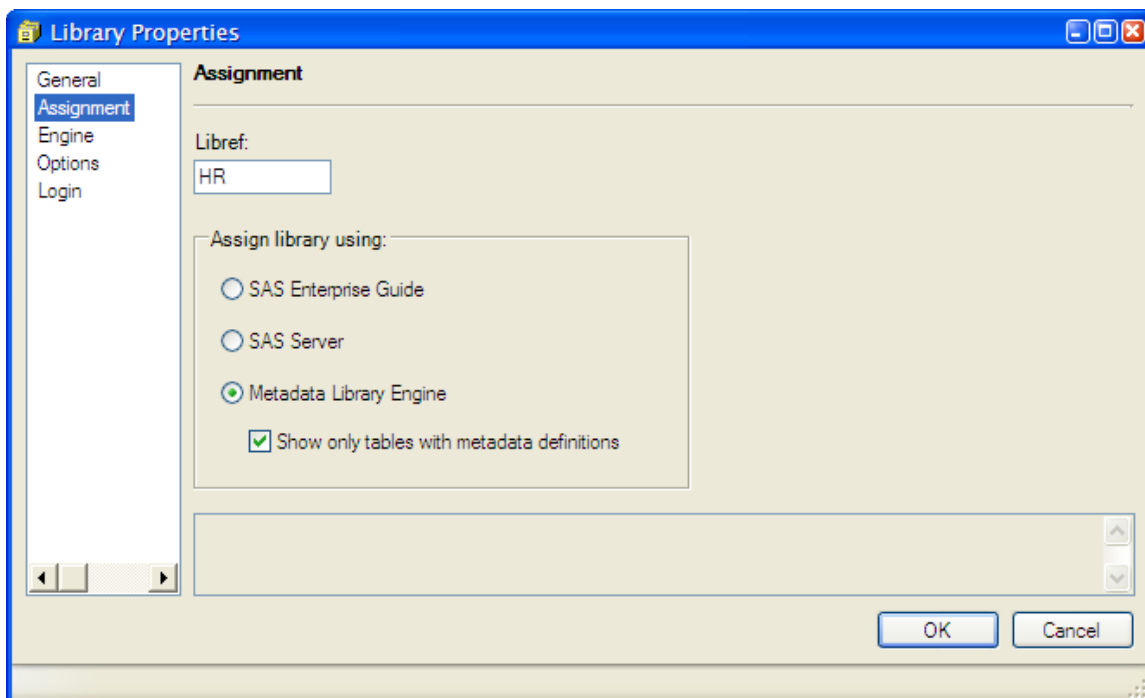
By default, when you expand an unassigned library by clicking on it, or assign it by right-clicking on it and selecting **Assign**, SAS Enterprise Guide assigns the library using the META engine in its default mode. That means that the library contents match the tables and columns registered in metadata, and the views that you see match the permissions assigned to you within metadata. It also means that the assigned library is effectively read-only, because the META engine does not allow you to change add/modify tables.

Controlling Library Assignment Behavior

It is possible for you to control how SAS Enterprise Guide assigns metadata-defined libraries. You can specify the behavior with Tools->SAS Enterprise Guide Explorer, which offers an administrative view of SAS servers and libraries similar to SAS Management Console.

Note: For more information about managing libraries using SAS Enterprise Guide Explorer, see *Administering SAS Enterprise Guide 4.1*, available from support.sas.com/eguide.

This screen shows the default library properties for an example metadata library:



Here is an explanation of the "Assign library using" modes.

SAS Enterprise Guide

SAS Enterprise Guide reads the library definition from metadata and assigns the library using the native engine. No table/column level metadata is read/used in this library – it's as if you submitted a libname statement that used the underlying native engine, bypassing the META engine and any permissions specified in metadata.

SAS Server

SAS Enterprise Guide does not assign the library, but treats it as pre-assigned. This means that the library, even though its definition exists in metadata, is actually assigned in an autoexec or via the METAUTOINIT mechanism.

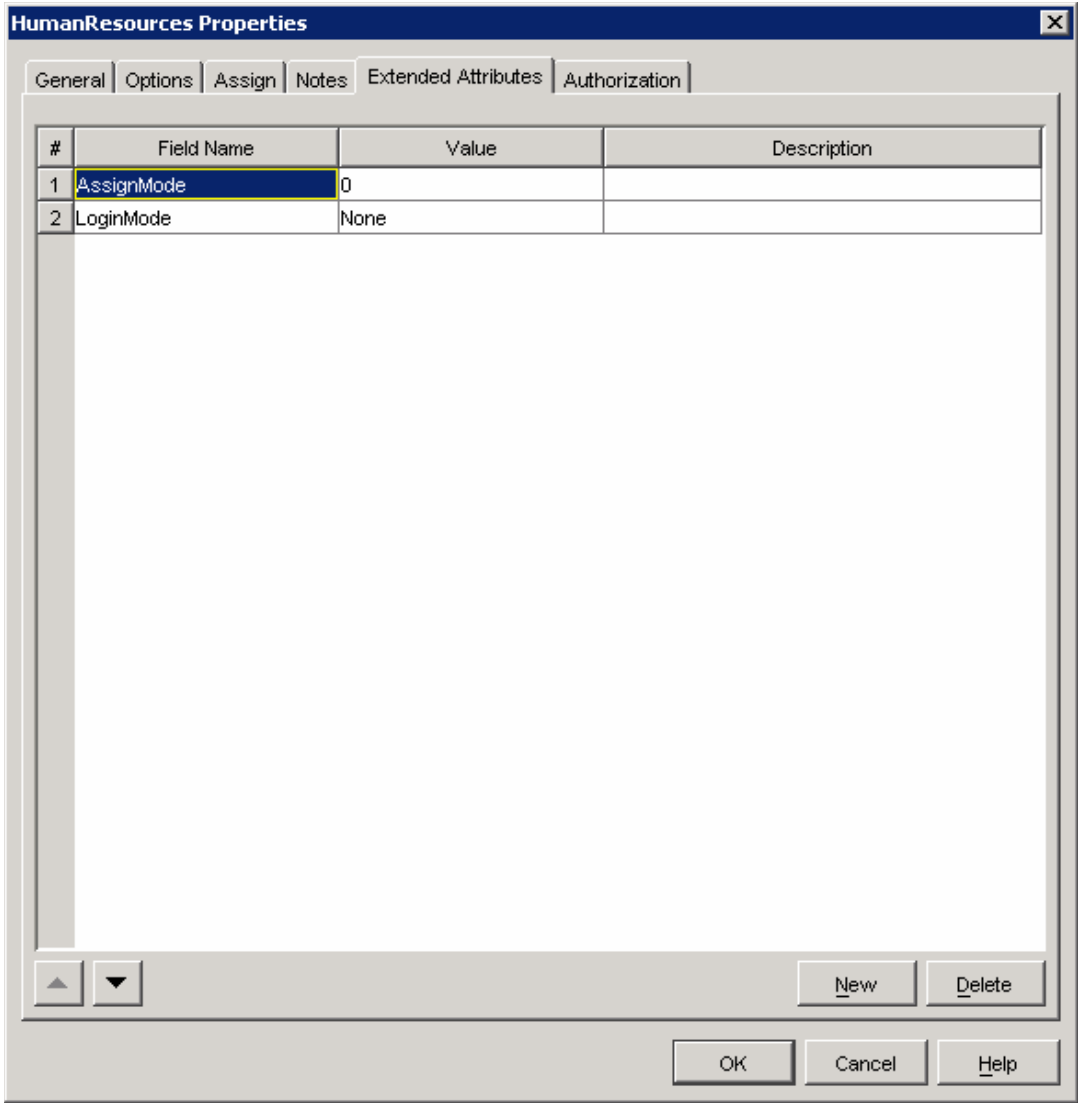
Metadata Library Engine

SAS Enterprise Guide uses the META engine to assign the library. This option offers two modes:

- Show only tables with metadata definitions (checkbox selected) – only tables that have metadata defined actually appear in the library. In this mode, the library is always Read Only.
- Show all physical tables (checkbox not selected) – this mode shows all physical tables in the library. Metadata READ permissions are still enforced when you try to open data. This uses the METAOUT=DATA option on the META engine. In this mode, it is possible to add/modify/delete tables within the library. That is, the library is not Read Only.

When you make changes to the Assignment page of the library definition in SAS Enterprise Guide Explorer, the changes are reflected in metadata as Extended Attributes on the library. You can view/modify these extended attributes using SAS Management Console, if desired.

To view the extended attributes in SAS Management Console, right-click on the library name and select **Properties**, then click on the Extended Attributes tab. This is an example:



The **AssignMode** name is the attribute that controls the assignment behavior. Valid values are:

0	Assign using SAS Enterprise Guide
1	Assign using the META engine, METAOUT=ALL (default META engine behavior)
2	Assign using the META engine, METAOUT=DATA
3	Assigned by the SAS server (pre-assigned)

Frequently Asked Questions about Libraries in SAS Enterprise Guide 4.1

I am trying to save output tables to a library, but I get an error message that says “you cannot create tables in this mode.” What does that mean and how can I fix it?

When accessing a library defined in metadata, SAS Enterprise Guide uses the META engine in “METAOUT=ALL” mode, by default. This means that the engine enforces restrictions to prevent additions and modifications that would cause library metadata to become out of sync with the physical tables.

There are a couple of options to allow output tables to be added to the library.

1. Use SAS Enterprise Guide Explorer to change the library assignment behavior, unchecking the “Show only tables with metadata definitions” checkbox. This tells SAS Enterprise Guide to use the METAOUT=DATA mode, which allows you to add/modify tables.
2. Use SAS Enterprise Guide Explorer to change the library assignment behavior to “Assign library using SAS Enterprise Guide.” This tells SAS Enterprise Guide to bypass use of the META engine and assign the library using the native engine and options that are defined in metadata. This mode bypasses the metadata view of the library altogether, ignoring permissions that might have been set on individual tables and columns in metadata. Instead, all permissions are dependent on the security set at the system level (for folder/file-based libraries) or DBMS level (for SAS/ACCESS libraries).

Why do I get an “access denied” error message when I try to save output tables to a library assigned with METAOUT=DATA?

It's possible that the authenticated user does not have write permissions to the physical location that the library maps to. If the library is defined to a file path, make sure that your user account has write permissions to the server folder. Without proper system-level permissions, you might see an error similar to this:

```
ERROR: User does not have appropriate authorization level  
for library <library>.
```

Note that granting or denying read/write permissions in SAS metadata affects only the view of tables accessed with the META engine. The metadata permissions do not override other permissions that have been set at the system level (for example, permissions on a system folder or in a database system). You can use metadata permissions to restrict a user's view of the library contents. Use system or DBMS permissions to prevent unauthorized access to sensitive information.

My users can see tables that they shouldn't have access to in a certain library. I changed the metadata permissions to deny them access; why aren't the permissions working?

There are a couple of things to check here:

1. First, ensure that the library is assigned using the META engine, which enforces the metadata-defined view of the library contents. This is the default behavior in SAS Enterprise Guide, but it can be affected by library settings controlled in SAS Enterprise Guide Explorer. The SAS workspace server can also assign the library using the native engine (circumventing the META engine) when the library is marked as “Pre-assigned”

and the METAUTOINIT object spawner option is in effect on the server definition.

2. If the META engine is used in METAOUT=DATA mode, ensure that the tables that you want to prevent access to are tagged with Deny for Read, but not Deny for ReadMetadata. If you select Deny for ReadMetadata, the META engine is unable to determine the intended Read permissions for the table. It might seem counterintuitive, but the META engine requires the ability to ReadMetadata in order to determine the proper Read/Write access control for the table.

WARNING: If you use the object server parameter METAUTOINIT in combination with the SAS Library advanced option, "Library is pre-assigned," the SAS workspace assigns the library automatically when it initializes, and it uses the native engine (for example, BASE or ORACLE) for the assignment (not the META engine). This overrides any settings you specified in SAS Enterprise Guide Explorer for the library Assign Mode.

I have denied ReadMetadata and Read access to a table for my user, but when he views the library in SAS Enterprise Guide, he can not only see the table, but also the records. Why aren't the permissions working?

If you have assigned the library with the METAOUT=DATA option, remember that SAS Enterprise Guide displays all data sets in the physical location and not just the ones that have been registered in the metadata. Therefore, even though you have denied ReadMetadata access, the table still appears. Also, because you have denied ReadMetadata access, SAS Enterprise Guide is not able to view the metadata for that table and, therefore, never sees that the Read permission has been denied. You must grant ReadMetadata in order for the denied Read to be enforced.

I used SAS Enterprise Guide to create some output tables for use in information maps. When I view the library in SAS Information Map Studio, the new tables don't appear there. Why not?

Before tables can be used in metadata-based objects such as information maps, you must first add the table definitions to metadata. You can achieve this by using the Import Tables feature in SAS Management Console, or within a SAS program using the METALIB procedure to synchronize metadata definitions with the physical contents of the library. An easy way to achieve this is to use the "Update Library Metadata" task, available for download from:

www.sas.com/apps/demosdownloads/setupcat.jsp?cat=SAS+Enterprise+Guide

How can I make sure that my users see only the metadata-defined view of a data library?

To guarantee that the library is assigned using the META engine, you can add a LIBNAME statement to your SAS workspace autoexec file.

For example, a LIBNAME statement like this one:

```
libname hr meta library="Human Resources"  
rename="Foundation";
```

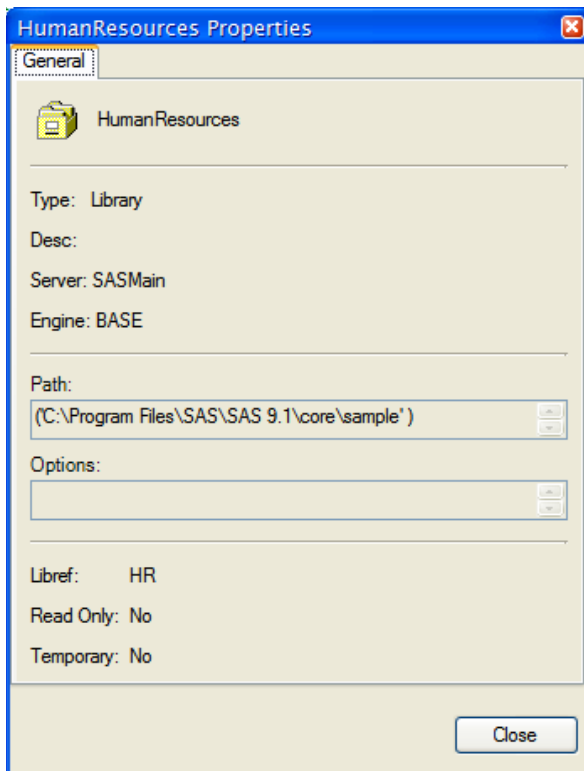
ensures that all users of that SAS workspace server see only the metadata-defined view of the "Human Resources" library, with their particular permissions applied. This library is immediately assigned within the SAS session (meaning it appears as a yellow icon in the server view, not white).

Important: Make sure that the libref used in the LIBNAME statement ("HR" in the above example) matches the libref defined in the metadata for the library. If the librefs don't match, your

users see two views of the library: one assigned with the libref in your autoexec, and the other as a potential library, not yet assigned, using the library name defined in metadata.

When I view the properties of a library in SAS Enterprise Guide, I see the native engine and path rather than META. Why? How can I determine for certain that the META engine is used?

When you right-click on a library in the server view and select Properties, SAS Enterprise Guide shows you details about the library as defined in metadata. Here is an example view:



Even though you see the engine/path details here, the default behavior is to use the META engine to assign the library when you access it.

To verify that the META engine is used, right-click on the library and select Assign. Then, from a SAS code window, submit the following statement:

```
libname _all_ list;
```

The log shows the details about how the library is defined. For example:

```
NOTE: Libref=    HR
      Scope=    IOM ROOT COMP ENV
      Engine=    META
      Physical Name=
      File Name= C:\Program Files\SAS\SAS
9.1\core\sample
```

From this log you can see that the META engine, not the BASE engine, is controlling access to the library. Therefore, you can expect the metadata-defined view of the library to be in place.

I created a stored process that references a data library that I accessed within SAS Enterprise Guide. When I run the stored process, it fails to write data to the library. However, when I run the code within SAS Enterprise Guide, it works. Why?

If you changed the library definition within Enterprise Guide to use METAOUT=DATA (AssignMode=2), then SAS Enterprise Guide tasks is able to write tables to the library. However, when running in an environment outside of SAS Enterprise Guide's control, such as within a stored process, the default behavior of the META engine (METAOUT=ALL) might apply. This difference can occur even when you are running the stored process from within SAS Enterprise Guide.

When you create a stored process that must access or write data tables in a metadata-controlled library, you cannot rely on the "Assign mode" behavior specified within SAS Enterprise Guide Explorer. The Assignment options affect the behavior within SAS Enterprise Guide and SAS Add-In for Microsoft Office, but they don't affect the behavior within the SAS® Stored Process server or other SAS products.

How can I prevent an end user (not an administrator) from adding or updating SAS library definitions in metadata?

Use SAS Management Console to deny Write Metadata (WM) permission on the SAS application server (for example "SASMain") for the end users you want to restrict. You can apply this Deny WM permission to individual users or groups. While this alone does not prevent users from creating new library definitions, it does prevent them from being able to associate library definitions with the SAS application server. In practice, this prevents them from creating or updating libraries that can be used from within SAS Enterprise Guide.

You can also use SAS Management Console to deny WM permission to individual SAS library definitions, which prevents end users from updating the table metadata within a library.

Other Resources

SAS Institute Inc. 2006. *Administering SAS® Enterprise Guide 4.1*. Available support.sas.com/documentation/onlinedoc/guide/admin4.pdf.

SAS Institute Inc. 2006. *SAS® 9.1.3 Intelligence Platform: Data Administration Guide*. Available support.sas.com/documentation/configuration/bidsag.pdf.

SAS Institute Inc. 2006. *SAS® 9.1.3 Intelligence Platform: Desktop Application Administration Guide*. Available support.sas.com/documentation/configuration/bidaag.pdf.

User Profile: Alexey Goncharov

Russian-born Alexey Goncharov worked as a physician, operating a heart-lung machine during surgeries in a cardiothoracic hospital in Novosibirsk, Siberia. Between 2001 and 2004, he worked in cardiovascular medicine at a Beijing hospital. He imagined he would stay in the public health field forever. Little did he know that a scholarship from the U.S. government, awarded to him in 2004, would change his future.

"I came to the State University of New York at Albany to study Public Health. Out of many subfields, I chose Environmental Health because it related to the biological and medical activity I was involved in before," Goncharov says. "But one of the optional courses the university offered was *Introduction to Base SAS Programming*. When I took that course, I discovered the new and amazing world called SAS!"

[Mike Zdeb](#) taught the course at the time. As Goncharov tells it, the professor's advice was indispensable: "He planted and cultivated my passion for SAS."

Learning (and teaching) the ropes

Goncharov says he never took a formal course to prepare for the [SAS® certification exam](#). Instead, he relied on Professor Zdeb's generosity in lending him SAS books for preparation – and on notes that he'd taken in the Base SAS class. "That helped me a lot."

Practical application of SAS also expanded his proficiency. A project undertaken in 2005 with David Carpenter, whom Goncharov calls "a brilliant environmentalist, scientist and person," stretched the student's SAS knowledge to its limits.

"Being interested in statistics, I began to analyze health data for David's projects," Goncharov explains. "This particular project related to the affect of polychlorinated biphenyls (environmental organic pollutants) on human biochemical parameters like lipids, glucose and sex hormones. It required me to study and apply detailed SAS procedures, which increased my SAS abilities enormously."

As the statistical difficulties involved in his projects increased, Goncharov began to master more and more complex aspects of SAS programming. Lately, he has focused on SAS Macro language –which he says is his favorite SAS tool for its speed and usability – and SQL. "In addition to that, I am sophisticating my graphical abilities in SAS."

Having learned much from his experiences, Goncharov can now help other students deepen their background in SAS. "Many of my friends and classmates ask me to explain one thing or another about programming. Some aspects of SAS are quite challenging and require detailed explanation to understand. But I readily help everybody who asks me about it."

A bright future with SAS®

Goncharov announces proudly that he has used SAS every day since 2005. But does he plan to use it for years to come? His answer is yes. He says the technology is vital to performing robust statistical analysis. It has helped him in every situation requiring analytics. "SAS gives full output for all our requests."

In fact, Goncharov likes SAS so much that he hopes to find a programming job. "I want to do what I like, what I know and what I hope to polish all through my life. Even if I thought about possible work in a lab before, now I changed my future perspective. I will direct all my energy to working with SAS."

How Agile Are Your Analytic Skills?

SAS Training provides the most [comprehensive program of analytics training](#) that you can find. What's more, this training is available in a variety of delivery methods to allow you the greatest flexibility in meeting your training needs. Attend instructor-led training or learn with [SAS Self-Paced e-Learning](#) and the [SAS Analytic Lecture Series](#) at your desktop.

What Happened to My Data?

Understanding how results are affected when formatting is applied to classification variables.

Introduction

Classification variables are used to group observations that have the same values. In the REPORT procedure, ORDER, GROUP, and ACROSS variables are classification variables. In the TABULATE, SUMMARY, and MEANS procedures, a CLASS variable is a classification variable. Because the classification process groups individual values, when a formatted value is used for classification the results might be unexpected.

This paper explains how classification variable values are handled when a FORMAT statement or FORMAT option is applied. Topics include grouping observations with one classification variable and with multiple classification variables. The supporting examples focus on Base SAS[®] reporting procedures, PROC REPORT, PROC TABULATE, PROC SUMMARY, and PROC MEANS.

Grouping observations with one classification variable

The following simple PROC REPORT example shows how the order of the output changes when a format is applied--the grouping occurs around the formatted value. Notice how the order of the values for 'f' and 'g' changes. The output is shown in Display 1.

```
proc format;
value $myfmt 'g' = 'one'
             'f' = 'two';
run;

data test;
input myval $ mynum;
cards;
f 1
f 3
f 5
g 2
g 4
g 6
;
run;

proc report data=test nowd;
column myval mynum;
define myval / group format=$myfmt3.;
/* order=formatted is the default ordering of proc report */
run;
```

Here is a comparable PROC TABULATE example. Note that the ORDER=FORMATTED option must be added because the default order is INTERNAL for all reporting procedures other than PROC REPORT.

```
proc tabulate data=test order=formatted;
class myval;
var mynum;
format myval $myfmt3.;
table myval, mynum;
run;
```

Because the variable MYVAL is a classification variable, all equivalent values are grouped together. In this case, a format is applied so the result is two distinct formatted values for MYVAL, "one" and "two". The output is shown in Display 1.

myval	mynum
one	12
two	9

Display 1. PROC REPORT and PROC TABULATE Output Using a FORMAT Statement

Internally, a separate mapping table is formed for each class variable. For each unique formatted value that is encountered during the input data processing, a new row is added to the mapping table. This includes the row number, the formatted value, and the internal value that is associated with the formatted value and is in the order of the data. The output in Display 2 is an example of the class mapping table for *myval*.

R_num	Fmt_val	Int_val
1	'two'	'f'
2	'one'	'g'

Display 2. Output of the Class Mapping Table *myval*

Notice that "two" is the first formatted entry because 'f' occurs before 'g' in the input data set.

In addition to associating the formatted value with each mapping table row, the procedure also keeps or retains the unformatted (internal) value.

If the ORDER=INTERNAL option is used, as shown in the following PROC REPORT example, then the ordering of the variable MYVAL is based on the unformatted value.

```
proc report data=test nowd;
column myval mynum;
define myval / group format=$myfmt3. order=internal;
run;
```

Here is a comparable PROC TABULATE code.

```
proc tabulate data=test;
class myval;
var mynum;
format myval $myfmt3.;
```

```
table myval, mynum;
run;
```

In this case, because the internal value of 'f' comes before 'g', the output would look like Display 3:

myval	mynum
two	9
one	12

Display 3. PROC REPORT Output Using the ORDER=INTERNAL Option

Now let's make the example a little more complicated by having two different data values that use the same formatted value.

```
proc format;
value $myfmt 'd' = 'one'
              'e' = 'two'
              'f' = 'two'
              'g' =
'one';
run;

data test;
input myval $ mynum;
cards;
f 1
f 3
f 5
g 2
g 4
g 6
e 2
e 4
e 6
d 7
d 9
d 11
;
run;

proc report data=test nowd;
column myval mynum;
define myval / group format=$myfmt3.;
run;
```

Again, there are only two distinct formatted values, 'one' and 'two'. The observations are consolidated into those two distinct mapping table rows as shown in the output in Display 4.

myval	mynum
one	39
two	21

Display 4. PROC REPORT Output for Two Distinct Formatted Values

Expected results

In the first example, we stated that the unformatted value is stored with each mapping table row. What value is stored when there are two or more unformatted values for a mapping table row? The procedure cannot keep all of the unformatted values; instead, it keeps only one unformatted value for each row in the mapping table. The rule, therefore, is that the *lowest* unformatted value is kept for that mapping table row, where *lowest* is determined by the ORDER= option.

For the preceding example, 'd' is kept for the 'one' mapping table row, and 'e' is kept for the 'two' mapping table row.

In PROC REPORT, if the ORDER=INTERNAL option is requested, the 'one' mapping table row would again print first because the unformatted value 'd' comes before 'e'. The output is shown in Display 5.

```
proc report data=test nowd;
column myval mynum;
  define myval / group format=$myfmt3. order=internal;
run;
```

myval	mynum
one	39
two	21

Display 5. PROC REPORT Output Using the ORDER=INTERNAL Option

Let's change the data to verify that.

```
proc format;
value $myfmt 'd' = 'one'
              'c' = 'two'
              'f' = 'two'
              'g' = 'one' ;
run;

data test;
input myval $ mynum;
cards;
f 1
f 3
f 5
g 2
g 4
g 6
c 2
c 4
c 6
d 7
d 9
d 11
;
run;

proc report data=test nowd;
column myval mynum;
  define myval / group format=$myfmt3.;
```

```
run;

proc report data=test nowd;
column myval mynum;
  define myval / group format=$myfmt3. order=internal;
run;
```

Here is a comparable PROC TABULATE example.

```
proc tabulate data=test order=formatted;
class myval;
var mynum;
format myval $myfmt3.;
table myval, mynum;
run;

proc tabulate data=test;
class myval;
var mynum;
format myval $myfmt3.;
table myval, mynum;
run;
```

The output using the ORDER=FORMATTED option is shown in Display 6. The output using the ORDER=INTERNAL option is shown in Display 7.

myval	mynum
one	39
two	21

Display 6. PROC REPORT and PROC TABULATE Output Using the ORDER=FORMATTED Option

myval	mynum
two	21
one	39

Display 7. PROC REPORT and PROC TABULATE Output Using the ORDER=INTERNAL Option

There are still only two mapping table rows—'one' and 'two'. However, the lowest value for 'one' is 'd' now, and the lowest value for 'two' is 'c'. If the formatted value (the default for PROC REPORT) determines the order of your output, then 'one' prints first. If ORDER=INTERNAL is specified, then 'two' would print first.

Now let's consider another example.

Grouping observations with multiple classification variables

In this example, there are two classification variables, YR and MON_YR. There are two distinct levels for YR, 2005 and 2006, and five distinct levels for MON_YR. Formatted as MONNAME3, the values are Aug, Sep, Oct, Nov, and Dec.

```

data test;
input yr mon_yr;
cards;
2005      16771
2006      17014
2006      17045
2006      17075
2006      17106
2006      17136
;
run;

proc report nowd data=test out=myoutput;
column yr mon_yr;
  define yr / group;
  define mon_yr / group format=monname3. order=internal;
run;

```

Here is a comparable PROC TABULATE example.

```

proc tabulate data=test;
class yr mon_yr;
format mon_yr monname3.;
table yr*mon_yr, n;
run;

```

The output is shown in Display 8.

yr	mon_yr
2005	Dec
2006	Dec
	Aug
	Sep
	Oct
	Nov

Display 8. PROC REPORT and PROC TABULATE Output Using the ORDER=INTERNAL Option

There are two internal values for the mapping table row Dec: 16771 and 17136. Because the lowest unformatted value is kept when there are two or more unformatted values in the same mapping table row, the value 16771 is kept for 'Dec'.

The associated internal values for MON_YR are:

```

"Aug"      17014
"Sep"      17045
"Oct"      17075
"Nov"      17106
"Dec"      16771

```

The associated internal values for YR are:

```

2005      2005
2006      2006

```

Now that the mapping table rows have been created, the procedure orders the columns and rows. Because ORDER=INTERNAL was specified, values are ordered using the internal values that are associated with each mapping table row.

Unexpected results

When YR=2005, there is only one MON_YR value, so no ordering needs to take place. However, when YR=2006, the MON_YR column has the following ordered values: 16771, 17014, 17045, 17075, and 17106. When the report is printed, the formatted value is printed in this order with the formatted value shown. As a result, the months do not appear in chronological order.

When you specify an output data set with PROC REPORT, the unformatted value is shown. Display 9 shows a PROC PRINT of the output data set MYOUTPUT from PROC REPORT:

yr	mon_yr	_BREAK_
2005	16771	
2006	16771	
2006	17014	
2006	17045	
2006	17075	
2006	17106	

Display 9. PROC REPORT Output Showing an Unformatted Value

Unexpected output can also occur when the classification variable is numeric and it is grouped using a format with less precision than the internal values.

In the following code, look at the value for CFB in the input data set. There are three different values.

```
data temp_ds;
input cfb;
cards;
.
0.6594
.
0.6629
;
run;

proc report data=temp_ds nowd missing out=test2;
column cfb;
define cfb / order format=6.3;
run;

proc print;
run;
```

Because CFB is a classification variable and has a format, there is a mapping table row for the formatted value of 0.066 and the missing value. The lowest unformatted value is 0.06594 and is kept with the formatted value 0.066 mapping table row. This output is shown in Display 10.

cfb
.
0.066

Display 10. PROC PRINT of PROC REPORT Output Showing an Unformatted Value

The output data set, TEST2, contains the lowest value of CFB. Display 11 shows a PROC PRINT output for TEST2.

```
cfb      _BREAK_
.
.
0.06594
0.06594
```

Display 11. PROC PRINT of PROC REPORT Output

By default, PROC TABULATE and PROC SUMMARY display the formatted value in the output data set. The internal values are the same as those displayed in PROC REPORT.

Conclusion

In summary, classification variables are used to group data values that are the same. When a FORMAT statement or FORMAT option is applied and causes more than one value of a variable to be in a mapping table row, then only the lowest unformatted value is kept with the associated bucket. Understanding this behavior will clarify unexpected results for SAS[®] reporting procedures.

Resources

Basic information on the REPORT, TABULATE, SUMMARY, and MEANS procedures is available in the following resource:

SAS Institute Inc. 2006. *SAS OnlineDoc*® 9.1.3. Cary, NC: SAS Institute Inc. Available at support.sas.com/onlinedoc/913/docMainpage.jsp.

Questions

If you have questions about the content in this paper send e-mail to support@sas.com.

Webcasts and Events

[Business Intelligence Virtual Symposium](#)

Sept. 18, 9:00 a.m. ET

Introducing "From Insight to Foresight: Secure Knowledge Now!" – a complimentary Business Intelligence Virtual Conference and Exposition.

[SAS' 10th Annual Data Mining Conference, M2007](#)

Oct. 1 - 2

Caesars Palace, Las Vegas

You'll learn best practices, new techniques and the latest trends through case studies, keynote presentations, hands-on training and workshops.

[WUSS](#)

Oct. 17 - 19

San Francisco

Join us at the Grand Hyatt at Union Square for this year's event.

[MWSUG](#)

Oct. 28 - 30

Des Moines, Iowa

This year's conference is being held at the Des Moines Marriott Downtown.

[ITIL Manager's Certificate in IT Service Management](#)

This program debuts October 22, 2007, and is delivered in three learning sessions.