Paper 2977-2019

The Mother Of All Metadata Files: Using SAS® Provided Macros To Manage Metadata

Hugh G. McCabe
Excellus Blue Cross Blue Shield

Abstract

Excellus Blue Cross Blue Shield is a regional health plan in Central and Western New York. We run SAS® 9.4 in a grid environment. Our enterprise provisioning and identity management processes are not optimally integrated, so acquiring and managing data on users' access can be challenging. Given our highly sensitive health care data, regulatory and compliance requirements, and the current security climate, effective management and reporting on user access is critical to our success.
This paper outlines how we used SAS® Institute provided macros, SAS® Metadata, SAS® Schedule Manager, Unix scripting, human resources information system data, LDAP data, and some crafty programming to manage accounts and generate consistent, comprehensive, and timely reporting on user access to corporate data resources.

This paper is for SAS® Administrators and managers who require quick access to comprehensive and accurate information on who in their enterprise has access to a sensitive data stored in a complex environment.

Introduction

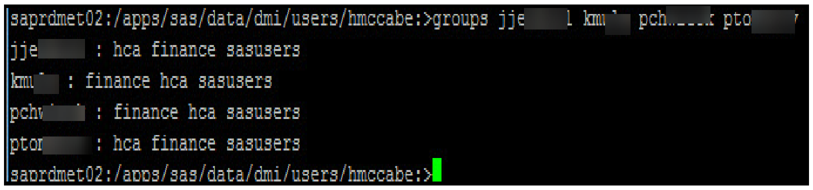
Excellus BlueCross BlueShield, headquartered in Rochester, NY, is part of a $6 billion family of companies that finances and delivers health care services across upstate New York and long-term care insurance nationwide. Collectively, the enterprise provides health insurance to nearly 1.5 million members and employs about 5,000 New Yorkers. We have approximately 200 SAS users.   We have been running the SAS® 9.4 grid environment on an AIX platform since 2016.   Our grid has two metadata servers and four compute servers. In addition to SAS datasets, our data sources include Oracle, DB2, and SQL server databases. Excellus SAS users do their development in SAS® Enterprise Guide and SAS® Studio. Scheduled jobs are run in SAS® Management Console Schedule Manager.

Our challenge was that we had no centralized repository of comprehensive metadata for our users. By comprehensive, I mean unix account information, human resources data, SAS Metadata, and inactivated user account data. In addition, referencing a users' unix account information or SAS metadata in SAS® Management Console (SMC) is labor intensive and only provides a limited view of a users' access to all the resources available in our complex SAS Grid Environment.   Over the course of a about a year were able to extract and stage the data components needed for a single unified repository of SAS user metadata that I call the Mother of All Metadata files (MOAM for short).

Harvesting Data

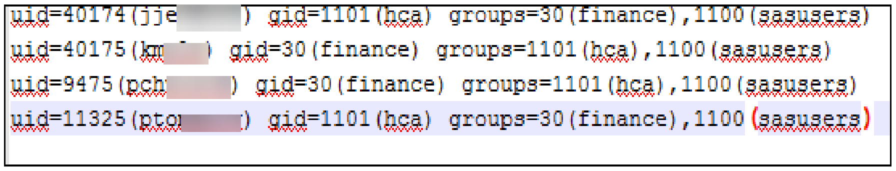
Unix Account Information

Typically, we when a person is hired or transfers into a new team we model account set up to match their new team. This journey began when we tired of logging into our metadata server to run the unix "groups" command to check the unix groups of a team mate. Figure 1. shows a unix session where unix accounts are checked. The initial step in the journey to create the MOAM file was to ask our Unix Administration team to write a script to query our LDAP table and create a text file containing each user's network identifier and unix groups and have that extract run each morning. We then read and manipulate the LDAP extract data and create a SAS data set. We manipulate the LDAP data so that users having the same unix groups that are ordered differently are resorted to appear in the same order. Figure 2. shows the output from the unix script that extracts users' unix groups. Note that the all four users have the same unix groups but the first and fourth users' unix group are in different order than the second and third users. Figure 3. shows the users' unix group in a SAS data set following manipulation. This might seem like a small matter, but it simplifies the data and makes subsequent use and analysis easier.



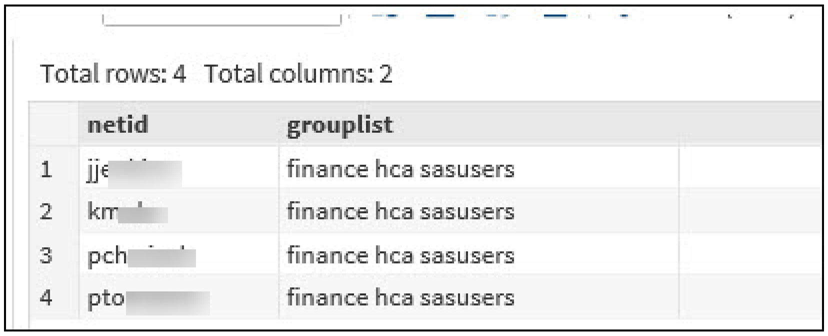
```
saprdmet02:/apps/sas/data/dmi/users/hmccabe:>groups jje[redacted]l km[redacted] pch[redacted] pto[redacted]
jje[redacted] : hca finance sasusers
km[redacted] : finance hca sasusers
pch[redacted] : finance hca sasusers
pto[redacted] : hca finance sasusers
saprdmet02:/apps/sas/data/dmi/users/hmccabe:>
```

Figure 1. Unix Command To Check User Accounts



```
uid=40174(jje[redacted]) gid=1101(hca) groups=30(finance),1100(sasusers)
uid=40175(km[redacted]) gid=30(finance) groups=1101(hca),1100(sasusers)
uid=9475(pch[redacted]) gid=30(finance) groups=1101(hca),1100(sasusers)
uid=11325(pto[redacted]) gid=1101(hca) groups=30(finance),1100(sasusers)
```

Figure 2 Unix Script LDAP Extract Output



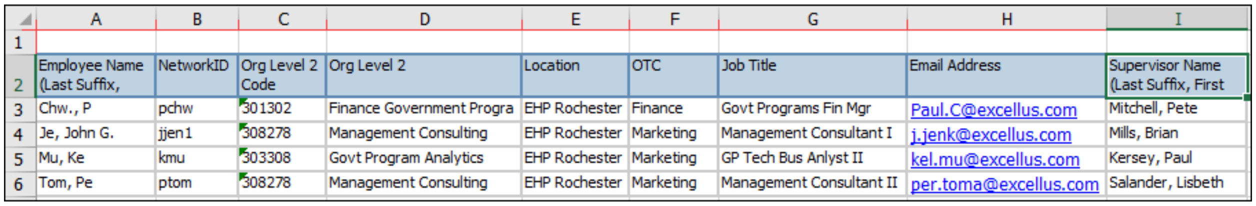
Total rows: 4 Total columns: 2

	netid	grouplist
1	jje[redacted]	finance hca sasusers
2	km[redacted]	finance hca sasusers
3	pch[redacted]	finance hca sasusers
4	pto[redacted]	finance hca sasusers

Figure 3. Manipulated Unix Users' Groups

Human Resources Information Systems Data

As I mentioned earlier, knowing a user's business unit is important. We have minimum access configurations for business units and often need check peer access for users. Working with our Information Technology and Services Division we were able to procure a download containing key information from our Human Resources Information System (HRIS). In the past we would manually reference our corporate organization chart for business unit information. The HRIS download allows us to integrate the HRIS data with the SMC metadata and unix group data. As shown in Figure 4., the HRIS data includes fields like user's full name, division name, team name, location, and manager name.



	A	B	C	D	E	F	G	H	I
1									
2	Employee Name (Last Suffix,	NetworkID	Org Level 2 Code	Org Level 2	Location	OTC	Job Title	Email Address	Supervisor Name (Last Suffix, First
3	Chw., P	pchw	301302	Finance Government Progra	EHP Rochester	Finance	Govt Programs Fin Mgr	Paul.C@excellus.com	Mitchell, Pete
4	Je, John G.	jjen1	308278	Management Consulting	EHP Rochester	Marketing	Management Consultant I	j.jenk@excellus.com	Mills, Brian
5	Mu, Ke	kmu	303308	Govt Program Analytics	EHP Rochester	Marketing	GP Tech Bus Anlyst II	kel.mu@excellus.com	Kersey, Paul
6	Tom, Pe	ptom	308278	Management Consulting	EHP Rochester	Marketing	Management Consultant II	per.toma@excellus.com	Salander, Lisbeth

Figure 4. HRIS Data

SAS Metadata

Referencing SAS Metadata needs to be done through SMC or SAS® Environment Manager (we seldom use Environment Manager). This is a manual process and is limited to an individual user by user look up exercise. When looking at a user in SMC one needs to check multiple locations to see what the resources (library, file, dataset, database) they can access. Figure 5. displays the SMC panel for a user's SMC Groups.

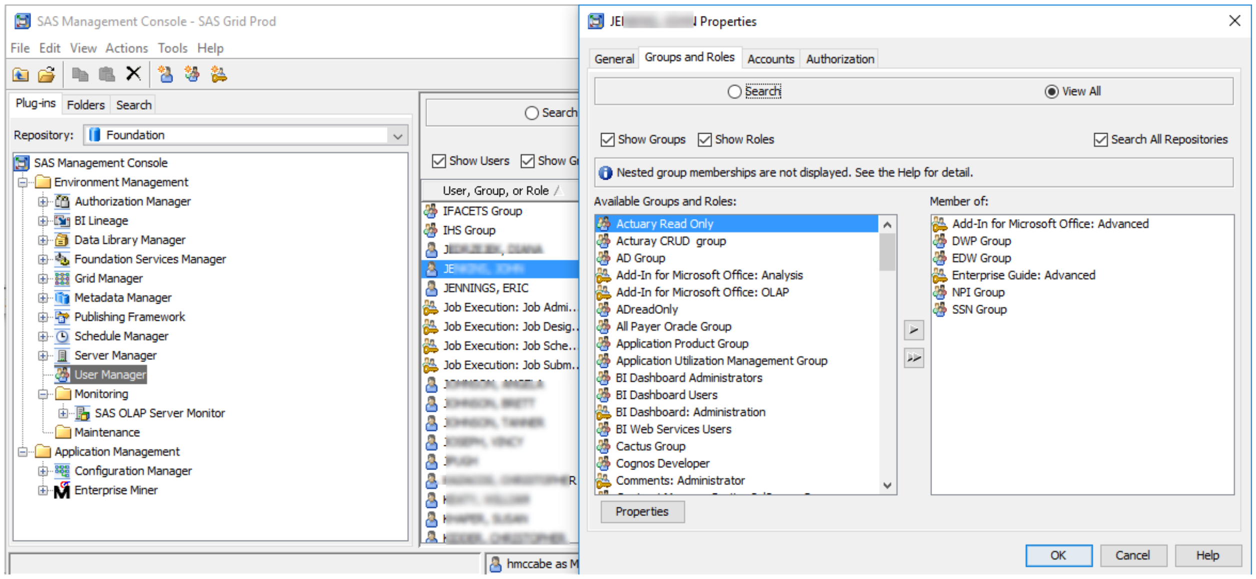


Figure 5. SAS Management Console User Account

Figure 6-A. shows the properties for a SAS Metadata library. The library properties window shows the permission for all metadata groups with any authorization to that library.

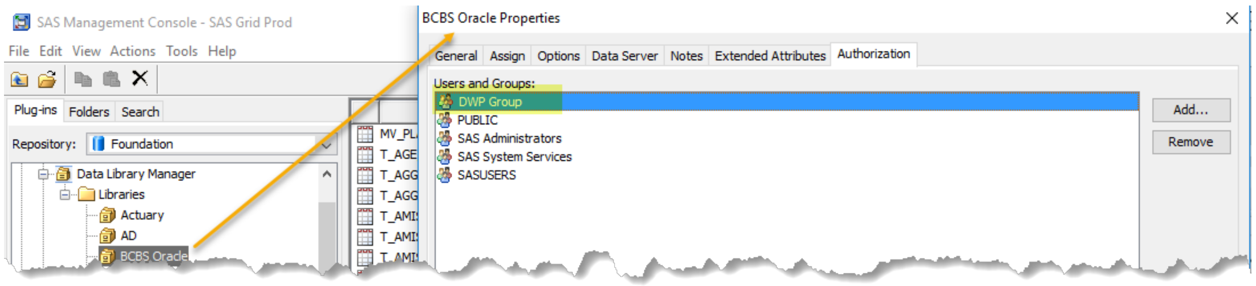
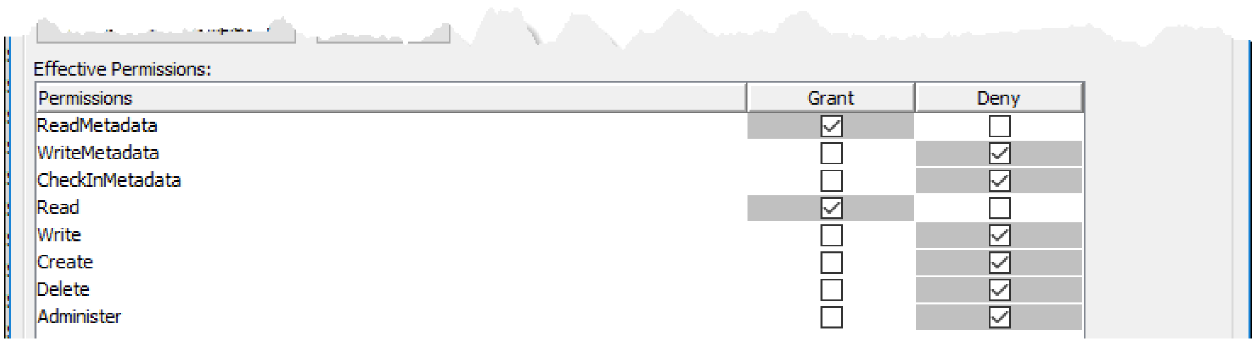


Figure 6A. SAS Management Console Library Properties

Figure 6-B. shows the granular level permissions for each metadata groups with any Authorization to that library.



Effective Permissions:

Permissions	Grant	Deny
ReadMetadata	✓	
WriteMetadata		✓
CheckInMetadata		✓
Read	✓	
Write		✓
Create		✓
Delete		✓
Administer		✓

Figure 6B. Library Permissions/Authorization

Conversely, trying to identify all users having access to a resource requires checking the Authorizations tab for the library of interest, noting the groups with access to that library and finally checking the membership of the groups that have access to that library. Figure 7. shows the properties for the group that has access to the library shown in Figure 6A. The right panel show users in that group.

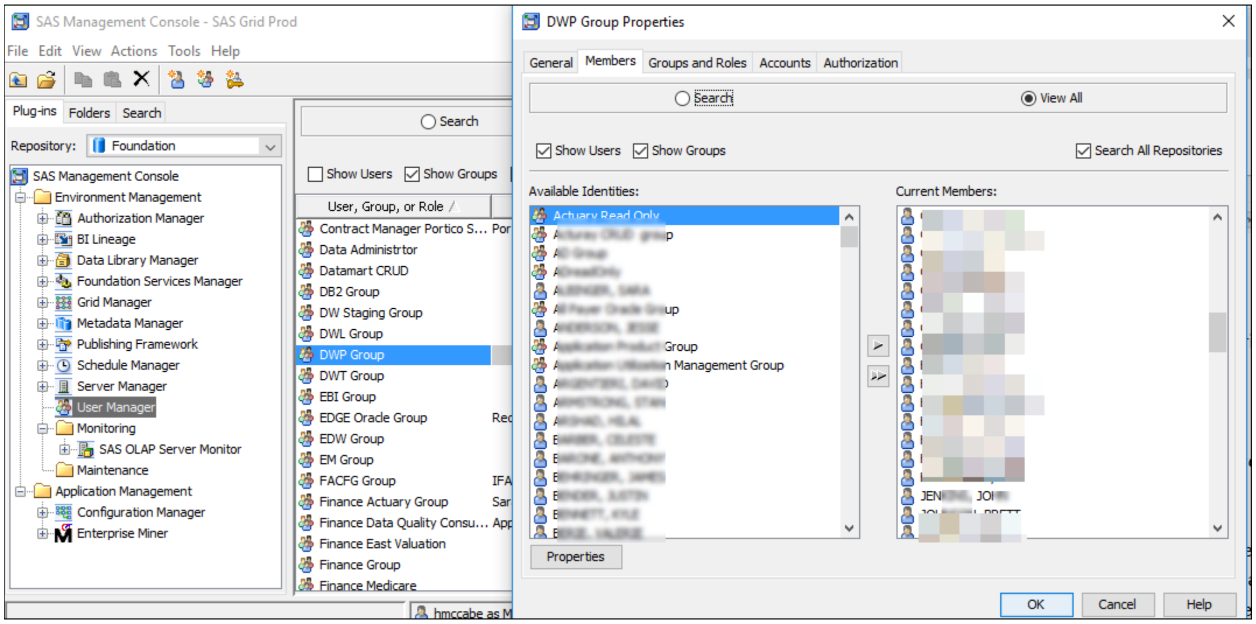
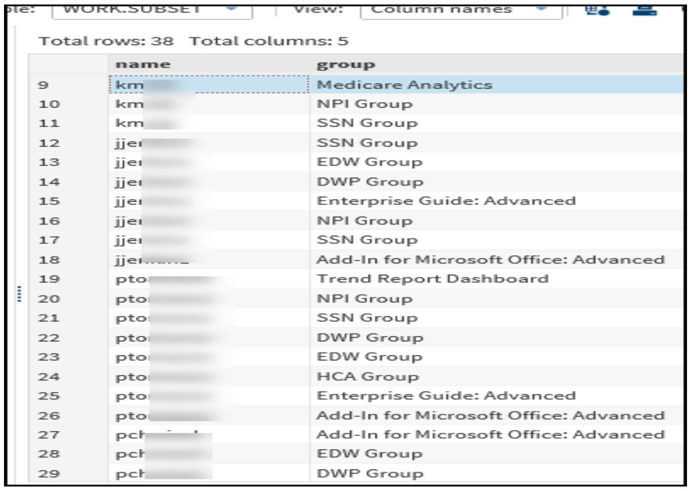


Figure 7. Metadata Group Properties and Membership

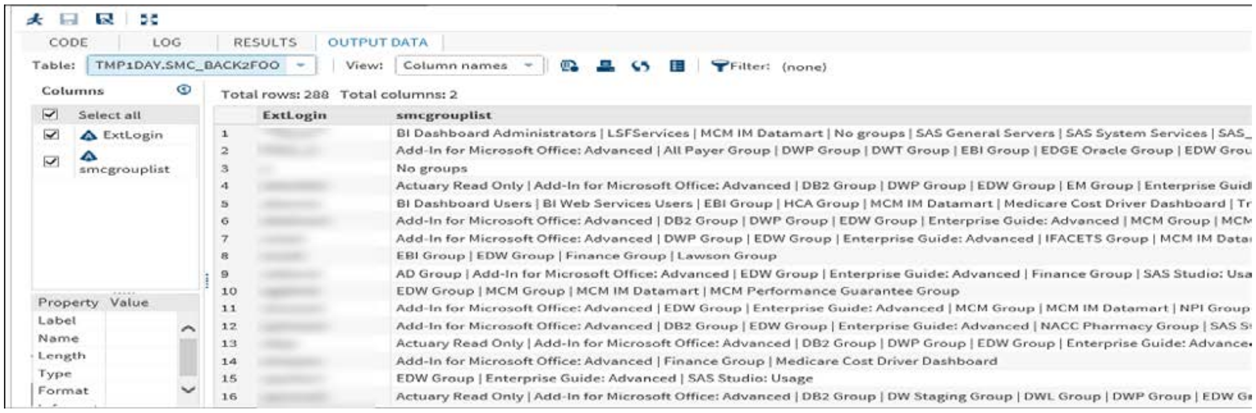
SAS® Institute stores its identity data in an XML format that it calls Canonical tables. SAS provides one set of macro code to extract metadata for users and another set of macro code to extract metadata attributes and permissions for objects (libraries and directories).   The output datasets from these two metadata extracts share a field for metadata group which we use to combine these data. The output from the user metadata extract includes one record per user per metadata group. Figure 8. shows the multiple rows per user. We take the output data from the extract code, sort the data by user and group and create a data set with one row per user where all the metadata groups are concatenated in one field. Figure 9. shows the same data for these users collapsed into one record per user with all the groups in a single field with pipe delimited values.



Total rows: 38 Total columns: 5

	name	group
9	km[redacted]	Medicare Analytics
10	km[redacted]	NPI Group
11	km[redacted]	SSN Group
12	jje[redacted]	SSN Group
13	jje[redacted]	EDW Group
14	jje[redacted]	DWP Group
15	jje[redacted]	Enterprise Guide: Advanced
16	jje[redacted]	NPI Group
17	jje[redacted]	SSN Group
18	jje[redacted]	Add-In for Microsoft Office: Advanced
19	pto[redacted]	Trend Report Dashboard
20	pto[redacted]	NPI Group
21	pto[redacted]	SSN Group
22	pto[redacted]	DWP Group
23	pto[redacted]	EDW Group
24	pto[redacted]	HCA Group
25	pto[redacted]	Enterprise Guide: Advanced
26	pto[redacted]	Add-In for Microsoft Office: Advanced
27	pch[redacted]	Add-In for Microsoft Office: Advanced
28	pch[redacted]	EDW Group
29	pch[redacted]	DWP Group

Figure 8. SAS Metadata User Output Dataset



CODE | LOG | RESULTS | OUTPUT DATA

Table: TMP1DAY.SMC_BACK2FOO View: Column names Filter: (none)

Columns: Select all, ExtLogin, smcgrouplist

Property | Value: Label, Name, Length, Type, Format

Total rows: 288 Total columns: 2

	ExtLogin	smcgrouplist
1	[redacted]	BI Dashboard Administrators | LSFServices | MCM IM Datamart | No groups | SAS General Servers | SAS System Services | SAS_
2	[redacted]	Add-In for Microsoft Office: Advanced | All Payer Group | DWP Group | DWT Group | EBI Group | EDGE Oracle Group | EDW Grou
3	[redacted]	No groups
4	[redacted]	Actuary Read Only | Add-In for Microsoft Office: Advanced | DB2 Group | DWP Group | EDW Group | EM Group | Enterprise Guid
5	[redacted]	BI Dashboard Users | BI Web Services Users | EBI Group | HCA Group | MCM IM Datamart | Medicare Cost Driver Dashboard | Tr
6	[redacted]	Add-In for Microsoft Office: Advanced | DB2 Group | DWP Group | EDW Group | Enterprise Guide: Advanced | MCM Group | MCM
7	[redacted]	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Enterprise Guide: Advanced | IFACETS Group | MCM IM Data
8	[redacted]	EBI Group | EDW Group | Finance Group | Lawson Group
9	[redacted]	AD Group | Add-In for Microsoft Office: Advanced | EDW Group | Enterprise Guide: Advanced | Finance Group | SAS Studio: Usa
10	[redacted]	EDW Group | MCM Group | MCM IM Datamart | MCM Performance Guarantee Group
11	[redacted]	Add-In for Microsoft Office: Advanced | EDW Group | Enterprise Guide: Advanced | MCM Group | MCM IM Datamart | NPI Group
12	[redacted]	Add-In for Microsoft Office: Advanced | DB2 Group | EDW Group | Enterprise Guide: Advanced | NACC Pharmacy Group | SAS S
13	[redacted]	Actuary Read Only | Add-In for Microsoft Office: Advanced | DB2 Group | DWP Group | EDW Group | Enterprise Guide: Advance
14	[redacted]	Add-In for Microsoft Office: Advanced | Finance Group | Medicare Cost Driver Dashboard
15	[redacted]	EDW Group | Enterprise Guide: Advanced | SAS Studio: Usage
16	[redacted]	Actuary Read Only | Add-In for Microsoft Office: Advanced | DB2 Group | DW Staging Group | DWL Group | DWP Group | EDW G

Figure 9. Collapsed SAS Metadata User Output Dataset

This is similar to what we do for the unix group data. The final MOAM file has all the unix groups in one field and all the SAS metadata groups in another field.

We extract metadata for three object types; Databases, Datamarts and Division Folders. These objects are highlighted in Figure 10. The code to extract metadata attributes and permissions for these objects (libraries, files, directories) produces a data set with one record per library object and metadata group with all permission grants to the objects. You should note that when querying relational data bases the metadata is returned at the database field level and in our environment the data warehouse returned 30 million rows. We only run this section of the code periodically and retain table level data. Figure 11. shows the extracted metadata for Databases, Datamarts and Division Folders. As noted earlier there are multiple rows per library because multiple groups have access to these resources.

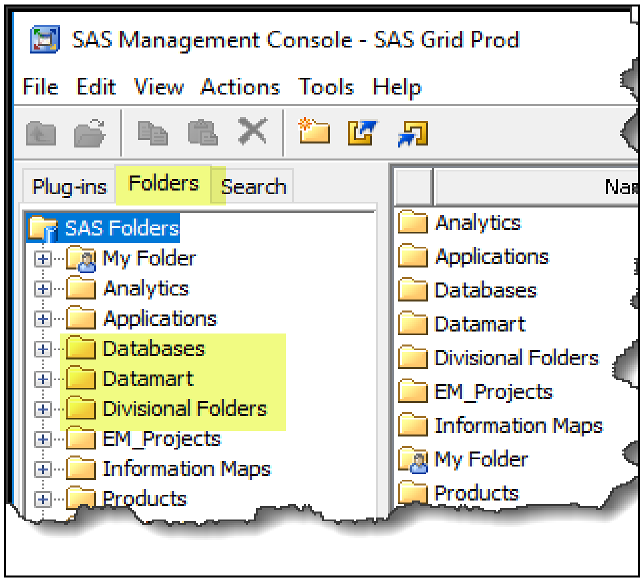
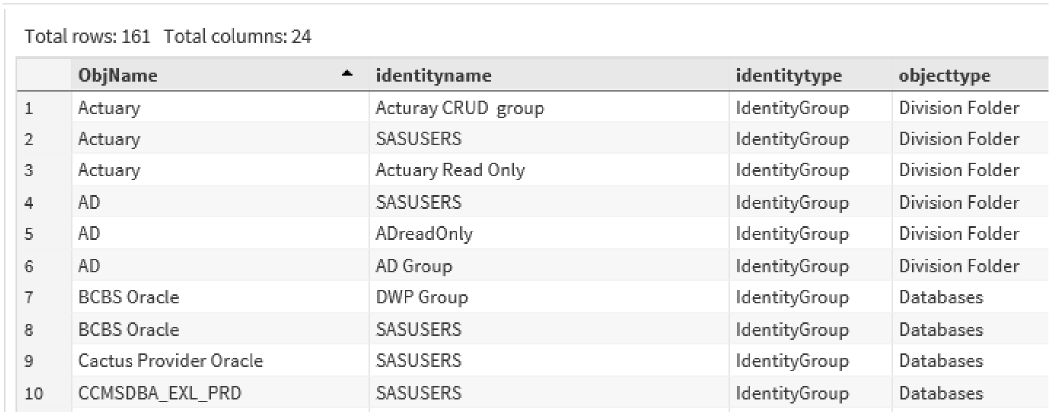


Figure 10. SAS Metadata Objects of Interest

The final piece of data that we procure in the SAS Metadata section of our process is the physical storage location for the SAS libraries. This is done with another set of code that queries the xml stored metadata. The key for joining the library physical location to the library authorizations data is the libref.

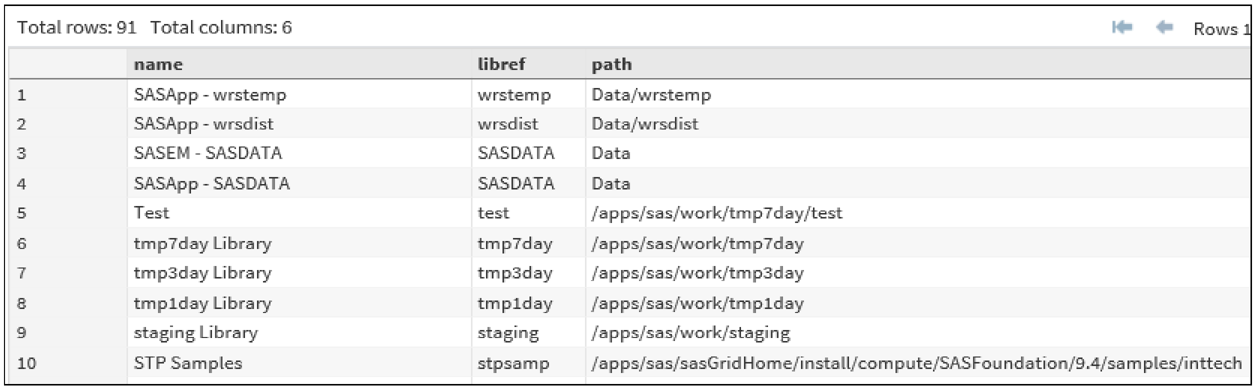


Total rows: 161 Total columns: 24

	ObjName	identityname	identitytype	objecttype
1	Actuary	Acturay CRUD group	IdentityGroup	Division Folder
2	Actuary	SASUSERS	IdentityGroup	Division Folder
3	Actuary	Actuary Read Only	IdentityGroup	Division Folder
4	AD	SASUSERS	IdentityGroup	Division Folder
5	AD	ADreadOnly	IdentityGroup	Division Folder
6	AD	AD Group	IdentityGroup	Division Folder
7	BCBS Oracle	DWP Group	IdentityGroup	Databases
8	BCBS Oracle	SASUSERS	IdentityGroup	Databases
9	Cactus Provider Oracle	SASUSERS	IdentityGroup	Databases
10	CCMSDBA_EXL_PRD	SASUSERS	IdentityGroup	Databases

Figure 11. SAS Metadata for Databases, Datamarts and Division Folders

Figure 12. shows the dataset for the physical paths for libraries.



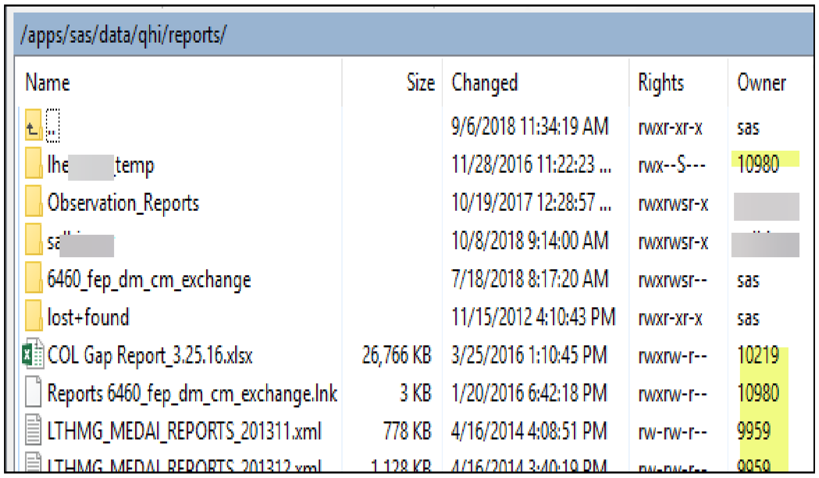
Total rows: 91 Total columns: 6 Rows 1

	name	libref	path
1	SASApp - wrstemp	wrstemp	Data/wrstemp
2	SASApp - wrsdist	wrsdist	Data/wrsdist
3	SASEM - SASDATA	SASDATA	Data
4	SASApp - SASDATA	SASDATA	Data
5	Test	test	/apps/sas/work/tmp7day/test
6	tmp7day Library	tmp7day	/apps/sas/work/tmp7day
7	tmp3day Library	tmp3day	/apps/sas/work/tmp3day
8	tmp1day Library	tmp1day	/apps/sas/work/tmp1day
9	staging Library	staging	/apps/sas/work/staging
10	STP Samples	stpsamp	/apps/sas/sasGridHome/install/compute/SASFoundation/9.4/samples/inttech

Figure 12. SAS Metadata Library Physical Locations

Legacy Unix Account Information

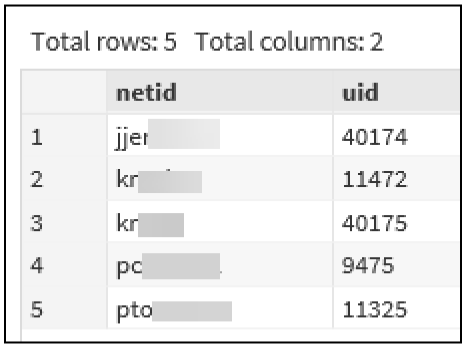
In the process of analyzing our environment's data storage utilization we noticed that there were files and directories that did not have a recognizable owner identifier. Typically, an object's owner is an alphanumeric string that is recognizable as network identifier (user's first initial and last name). We were seeing files and directories where the owner was just a numeric value (highlighted in Figure 13.). After some research we learned in that in our system, when an account is inactivated, the alphanumeric network identifier is removed from objects owned by the inactivated user and the object owner is left as a numeric user identification number (UID). Leaving only a numeric UID effectively renders objects unidentifiable. Further research indicated that there wasn't any systematic or institutional process for mapping recognizable network identifiers (typically first name initial concatenated with last name) to UIDs for inactivated accounts. We were able to obtain a onetime run of network identifiers to UIDs mappings from our Unix Administration Team which provided recognizable network identifiers for all but a handful of inactivated users and their respective stored files and directories. Knowing that that there wasn't any systematic or institutional process for mapping recognizable network identifiers to UIDs we decided to begin building our own compendium (Figure 14.) of these mappings with the onetime run as the starting point. When we build the MOAM file we also check the base compendium dataset and add a record for users containing network identifier (useful alphanumeric value) and UID (useless numeric value). The compendium dataset serves as a reference tool for inactivated accounts whose owner is only identified with the numeric UID.



/apps/sas/data/qhi/reports/

Name	Size	Changed	Rights	Owner
..		9/6/2018 11:34:19 AM	rwxr-xr-x	sas
lhe[redacted]temp		11/28/2016 11:22:23 ...	rwx--S---	10980
Observation_Reports		10/19/2017 12:28:57 ...	rwxrwsr-x	[redacted]
sa[redacted]		10/8/2018 9:14:00 AM	rwxrwsr-x	[redacted]
6460_fep_dm_cm_exchange		7/18/2018 8:17:20 AM	rwxrwsr--	sas
lost+found		11/15/2012 4:10:43 PM	rwxr-xr-x	sas
COL Gap Report_3.25.16.xlsx	26,766 KB	3/25/2016 1:10:45 PM	rwxrw-r--	10219
Reports 6460_fep_dm_cm_exchange.lnk	3 KB	1/20/2016 6:42:18 PM	rwxrw-r--	10980
LTHMG_MEDAI_REPORTS_201311.xml	778 KB	4/16/2014 4:08:51 PM	rw-rw-r--	9959
LTHMG_MEDAI_REPORTS_201312.xml	1,128 KB	4/16/2014 3:40:19 PM	rw-rw-r--	9959

Figure 13. Unix Directory and Object Ownership



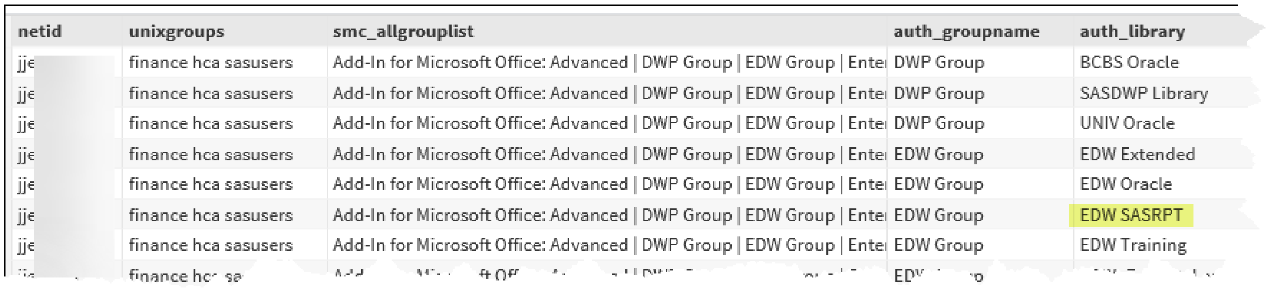
Total rows: 5 Total columns: 2

	netid	uid
1	jje[redacted]	40174
2	kr[redacted]	11472
3	kr[redacted]	40175
4	pc[redacted]	9475
5	pto[redacted]	11325

Figure 14. Collected Network IDs and UIDs

Assembling the Data

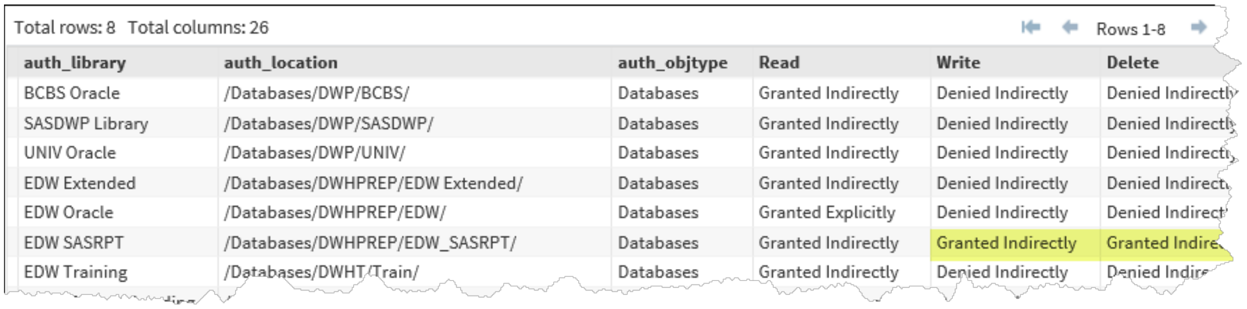
After harvesting the input data, we assemble the data by running joins using the various keys. It is important to understand the cardinality amongst the various input data sets. The resulting MOAM file has multiple rows per user. Each user has a set of unix groups attached to their unix identity. I refer to this as their operating system security. Users also have a set of metadata groups which I refer to as application level security. As discussed, we placed all the unix groups into a single field and all SAS Metadata groups in to a single field in the MOAM file. Up to this point we still maintain one row per user account (one to one relationships). Unfortunately, bringing in the library metadata results in multiple rows per user. Because each user can have multiple metadata groups allowing access to multiple metadata resources (files and libraries), there is a one to many relationship and the join of user data to resource data creates multiple rows per user. Figure 15-A. shows the repeating rows of data for a single user in the MOAM file. Note that the repeating rows have distinct "auth_library" values.



netid	unixgroups	smc_allgrouplist	auth_groupname	auth_library
jje[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Ente	DWP Group	BCBS Oracle
jje[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Ente	DWP Group	SASDWP Library
jje[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Ente	DWP Group	UNIV Oracle
jje[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Ente	EDW Group	EDW Extended
jje[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Ente	EDW Group	EDW Oracle
jje[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Ente	EDW Group	EDW SASRPT
jje[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Ente	EDW Group	EDW Training
[illegible]	finance hc sa	[illegible]	ED	[illegible]

Figure 15A. Library Metadata

Figure 15-B. displays the permission grants for each library. There is not much variation here but if you note, the user has full Read, Write and Delete grants for the EDW SASRPT library (highlighted).

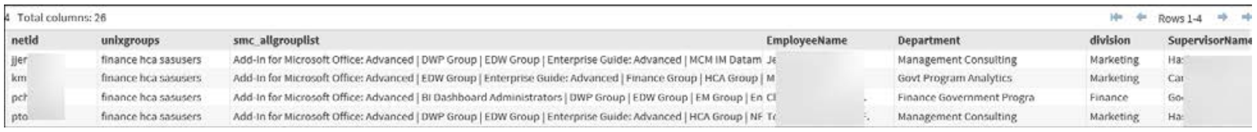


Total rows: 8 Total columns: 26 Rows 1-8

auth_library	auth_location	auth_objtype	Read	Write	Delete
BCBS Oracle	/Databases/DWP/BCBS/	Databases	Granted Indirectly	Denied Indirectly	Denied Indirectly
SASDWP Library	/Databases/DWP/SASDWP/	Databases	Granted Indirectly	Denied Indirectly	Denied Indirectly
UNIV Oracle	/Databases/DWP/UNIV/	Databases	Granted Indirectly	Denied Indirectly	Denied Indirectly
EDW Extended	/Databases/DWHPREP/EDW Extended/	Databases	Granted Indirectly	Denied Indirectly	Denied Indirect
EDW Oracle	/Databases/DWHPREP/EDW/	Databases	Granted Explicitly	Denied Indirectly	Denied Indirect
EDW SASRPT	/Databases/DWHPREP/EDW_SASRPT/	Databases	Granted Indirectly	Granted Indirectly	Granted Indire
EDW Training	/Databases/DWHT/Train/	Databases	Granted Indirectly	Denied Indirectly	Denied Indire

Figure 15B. Library Metadata-Permissions/Grants

Since much of the information in the rows repeats and most of our use cases only require parts of the data we can simply select one row of data per member when we generate a report. Figure 15-C. shows a non-duplicated set of rows. As noted earlier all unix groups as space delimited values in the "unixgroup" field and all SAS Meta Data groups in pipe delimited "smc_allgrouplist" field, but here the fields are together in the same dataset shown along with the HRIS data.



4 Total columns: 26 Rows 1-4

netid	unixgroups	smc_allgrouplist	EmployeeName	Department	division	SupervisorName
jjer[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Enterprise Guide: Advanced | MCM IM Datam	Je[redacted]	Management Consulting	Marketing	Ha[redacted]
km[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | EDW Group | Enterprise Guide: Advanced | Finance Group | HCA Group | M	M[redacted]	Govt Program Analytics	Marketing	Car[redacted]
pch[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | BI Dashboard Administrators | DWP Group | EDW Group | EM Group | En	Cl[redacted]	Finance Government Progra	Finance	Go[redacted]
pto[redacted]	finance hca sasusers	Add-In for Microsoft Office: Advanced | DWP Group | EDW Group | Enterprise Guide: Advanced | HCA Group | NF	T[redacted]	Management Consulting	Marketing	Ha[redacted]

Figure 15C. MOAM File Distinct Rows

Staging the Data

Having the MOAM file created weekly is generally adequate for our needs. The data from the SAS Metadata is compiled at run time each week. The unix data is extracted daily for another process so it is available for our weekly MOAM file processing. The HRIS data is extra is run once per week. We run the entire process using the SAS® Schedule Manager every Sunday morning. The weekly run includes the SAS Metadata extract. Each weekly run of the MOAM process creates a permanent file with a date stamp suffix. The date stamp suffix allows us to go back and retrieve a snap shot of our user metadata for a given point in time.

Code Review

For purposes of brevity I will only be looking at the code we borrowed from SAS Institute. The full set of code including the code for processing and integrating our additional data sources is available with my paper. We used three sets of code borrowed from SAS Institute. The soure URLs are bolded in the code comments. The first set of code queries and extracts the library permissions metadata. There are few things to note about the library permission metadata. First, as previously noted, when running the macro, we specified three metadata folder objects of interest (Divisional Folders, Datamarts and Databases). Second, databases objects can return a very large volumes of data, so you may consider updating it only periodically (versus every time you run your process). Third, the query returns a lot data that was not useful to us so after spending time reviewing the data we limited record types. Lastly, the mdsecds_join is the key source table in the extract. We do three passes on the metadata and then concatenate the resulting output files.

```
/***************** Start: library & permission data set up-staging.                    ************/
/* Data set up-staging.   This block gets library & permission data.                              */
/* This program queries the sas metadata (which is store as xml) and returns detailed            */
/* data on our metadata libraries and their respective authorizations. so, we can see we can see libraries and
what groups have permissions on the libraries.  the code is a sas macro from SAS Institute.  b/c the data is
stored in xml the queries are kind of hard to understand and if you spend too much time thinking    about it
your hair will hurt. HGM 20170628                                                                 */

/* connect to the metadata server                                                                 */
  options    metaserver=saprdvmet.excellus.com
  metauser="username"
  metapass="{SAS003}XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

/* Run the main report macro against a target folder the folder option points to SAS Mgt Concole folder(s) I
run 3 of these: /Divisional Folders, /Datamart and /Databases.  Heads up!  the databases folder returns 30
million rows.  it is row/col level & very detailed. I dump column level data so it is more manageable the
mdsecds_join  is the JACKPOT file here I am just dumping the data out to permanent sas data sets  I am
attaching a prefix to the data set name. This is canned code from sas that queries the metadata to find
this macro and documentation search %mdsecds            */
/*http://support.sas.com/documentation/cdl/en/bisecag/63082/HTML/default/viewer.htm#n0l1
mpdt430djgn1bl1c3euei85w.htm */

%mdsecds(folder="/Datamart");

data authdata.datamart_mdsecds_join ;
```

```
set work.mdsecds_join   ;
run;

%mdsecds(folder="/Divisional Folders");
data authdata.DIVFLDR_mdsecds_join ;
set work.mdsecds_join   ;
run;

/* As mentioned this returns 30 MILLION rows so I am leaving it commented out until          */
/* we feel that the oracle DB info need to be updated.                                        */

/*%mdsecds(folder="/Databases");  * updated 20181207  ;
data authdata.DATABASES_mdsecds_join ;
set work.mdsecds_join   ;
run;*/

/* after the meta for each data type is extracted it needs to be combined into one data set. that happens
next                                                                                          */

 /* 20170706 hgm: I profiled this metadata & figured out that in authorizations               */
 /* data, identityname is the SMC Group! Johnny was very helpful with this too.                */
 /* he suggested that i wasnt getting the correct rows and we should be able to join on  objname from here
(authdata) to the library data . may need to include identityname in the join.  hang on to identitytype b/c it
has "person"s SMC Groups                                                                      */

data authdata.libraries_groups_perms;
set  authdata.DATABASES_MDSECDS_JOIN (in=databases)
    authdata.DATAMART_MDSECDS_JOIN  (in=datamart)
    authdata.DIVFLDR_MDSECDS_JOIN   (in=divfldr);
length objecttype $ 20 ;

/*  HGM -  lots of metadata object types are returned. We thin it down here.                 */
where PublicType in ("Folder", "Library")  and metadatatype  in ("SASLibrary") and
   identityname not in ("PUBLIC" , "SAS System Services" , "SASAdministrators") ;

if databases then objecttype = "Databases" ;
if datamart then objecttype = "Datamart" ;
if divfldr then objecttype = "Division Folder" ;
run ;
/***************** End of library & permission data set up-staging.              ***************/
```

The second set of code we borrowed, extracts user level information. The only point to mention here is that we limit the data returned to name dispname ExtLogin MDUpdate group.

```
/*************** Start: User SAS MetaData extract, set up-staging.               **************/
/*  source:                                                                                   */
/*http://support.sas.com/documentation/cdl/en/lrmeta/63180/HTML/default/viewer.htm#p1k9zipe59h
a2an1pq34gu143lay.htm */
```

```
data tmp1day.smc_users_grps;
/* The LENGTH statement defines variables for function arguments and assigns the maximum length of each
variable.  */

  length uri  uri2 uri3 uri4 name dispname group groupuri ExtLogin $256 id MDUpdate $20;

/* The CALL MISSING routine initializes output variables to missing values.                                    */

  n=1;
  n2=1;
  call missing(uri, uri2, uri3, uri4, name, dispname, group, groupuri, ExtLogin, id, MDUpdate);

  /* The METADATA_GETNOBJ function specifies to get the Person objects in the repository. The n argument
specifies to get the first Person object that is returned. The uri argument will return the actual uri of the
Person object that is returned. The program prints an informational message if no Person objects are found.
*/

     nobj=metadata_getnobj("omsobj:Person?@Id contains '.'",n,uri);
  if nobj=0 then put 'No Persons available.';

/* The DO statement specifies a group of statements to be executed as a unit for the Person object that is
returned by METADATA_GETNOBJ. The METADATA_GETATTR function gets the values of the object's Name
and DisplayName attributes.                              */

  else do while (nobj > 0);
     objrc=metadata_getattr(uri, "Name", Name);
     objrc=metadata_getattr(uri, "DisplayName", DispName);

/* The METADATA_GETNASN function gets objects associated via the InternalLoginInfo association. The
InternalLoginInfo association returns internal logins. The n2 argument specifies to return the first associated
object for that association name. The URI of the associated object is returned in the uri2 variable. */

objrc=metadata_getnasn(uri,"InternalLoginInfo",n2,uri2);

/* If a Person does not have any internal logins, set their IntLogin variable to 'No' Otherwise, set to 'Yes'. */
IntLogin="Yes";
DomainName="**None**";
if objrc<=0 then
do;
put "NOTE: There are no internal Logins defined for " IdentName +(-1)".";
IntLogin="No";
end;

/* The METADATA_GETNASN function gets objects associated via the Logins association. The Logins
association returns external logins. The n2 argument specifies to return the first associated object for that
association name. The URI of the associated object is returned in the uri3 variable. */

objrc=metadata_getnasn(uri,"Logins",n2,uri3);
```

```
/* If a Person does not have any logins, set their ExtLogin variable to '**None**' and output their name. */
if objrc<=0 then
do;
put "NOTE: There are no external Logins defined for " IdentName +(-1)".";
ExtLogin="**None**";
output;
end;

/* If a Person has many logins, loop through the list and retrieve the name of each login. */
do while(objrc>0);
rc=metadata_getattr(uri3,"UserID",ExtLogin);

/* If a Login is associated to an authentication domain, get the domain name. */
DomainName="**None**";
objrc2=metadata_getnasn(uri3,"Domain",1,uri4);
if objrc2 >0 then
do;
 objrc2=metadata_getattr(uri4,"Name",DomainName);
end;

/*Output the record. */
output;
n2+1;

/* Retrieve the next Login's information */
objrc=metadata_getnasn(uri,"Logins",n2,uri3);
end;
 /*do while objrc*/

/* The METADATA_GETNASN function gets objects associated via the IdentityGroups association. The a
argument specifies to return the first associated object for that association type. The URI of the associated
object is returned in the groupuri variable.  */

  a=1;
    grpassn=metadata_getnasn(uri,"IdentityGroups",a,groupuri);

/* If a person does not belong to any groups, set their group variable to 'No groups' and output their name.
*/

    if grpassn in (-3,-4) then do;
          group="No groups";
       output;
    end;
 /* If the person belongs to many groups, loop through the list and retrieve the Name and MetadataUpdated
attributes of each group, outputting each on a separate record. */

    else do while (grpassn > 0);
                rc2=metadata_getattr(groupuri, "Name", group);
                rc=metadata_getattr(groupuri, "MetadataUpdated", MDUpdate);
                a+1;
                output;
          grpassn=metadata_getnasn(uri,"IdentityGroups",a,groupuri);
        end;

          /* Retrieve the next person's information */
        n+1;
        n2=1;
        nobj=metadata_getnobj("omsobj:Person?@Id contains '.'",n,uri);
      end;

    /* The KEEP statement specifies the variables to include in the output data set. */

      keep name dispname ExtLogin MDUpdate group;  *** Thinning file to key fields ;
    run;
    /*************** End of User MetaData extract, set up-staging. *******************/
```

The last set of code that we borrowed, extracts the physical directory path for a resource's location. The important fields in this extract are name libref path and we use libref as the key when we join it with our other data.

```
/**** Start of Code to find the physical location assoc with SMC Libraries  ***/

/* this is some more borrowed code that queries the metadata. This is pirated code             */

/* https://communities.sas.com/t5/Administration-and-Deployment/Listing-Metadata-libraries/td-
p/359558 */

 data authdata.librarylocations;

/* The LENGTH statement defines variables for function arguments and assigns the maximum length of each
variable.  */

  length liburi upasnuri $256 name $128 type id $17 libref engine $8 path
mdschemaname schema $256;

/* The KEEP statement defines the variables to include in the output data set.  */

  keep name libref engine path mdschemaname schema;

/* The CALL MISSING routine initializes the output variables to missing values.  */

  call missing(liburi,upasnuri,name,engine,libref);

  /* The METADATA_GETNOBJ function specifies to get the SASLibrary objects in the repository. The
argument nlibobj=1 specifies to get the first object that matches the requested URI. liburi is an output
variable. It will store the URI of the returned SASLibrary object. */
```

```
  nlibobj=1;
  librc=metadata_getnobj("omsobj:SASLibrary?@Id contains '.'",nlibobj,liburi);

  /* The DO statement specifies a group of statements to be executed as a unit for each object that is
returned by METADATA_GETNOBJ. The METADATA_GETATTR function is used to retrieve the values of the
Name, Engine, and Libref attributes of the SASLibrary object.  */

  do while (librc>0);

    /* Get Library attributes */
    rc=metadata_getattr(liburi,'Name',name);
    rc=metadata_getattr(liburi,'Engine',engine);
     rc=metadata_getattr(liburi,'Libref',libref);

 /* The METADATA_GETNASN function specifies to get objects associated to the library via the
UsingPackages association. The n argument specifies to return the first associated object for that association
type. upasnuri is an output variable. It will store the URI of the associated metadata object, if one is found.
*/

      n=1;
      uprc=metadata_getnasn(liburi,'UsingPackages',n,upasnuri);

 /* When a UsingPackages association is found, the METADATA_RESOLVE function is called to resolve the
URI to an object on the metadata server. The CALL MISSING routine assigns missing values to output
variables.  */

      if uprc > 0 then do;
        call missing(type,id,path,mdschemaname,schema);
        rc=metadata_resolve(upasnuri,type,id);

        /* If type='Directory', the METADATA_GETATTR function is used to get its
path and output the record */
        if type='Directory' then do;
                rc=metadata_getattr(upasnuri,'DirectoryName',path);
                      output;
          end;

        /* If type='DatabaseSchema', the METADATA_GETATTR function is used to get
the name and schema, and output the record */

        else if type='DatabaseSchema' then do;
           rc=metadata_getattr(upasnuri,'Name',mdschemaname);
           rc=metadata_getattr(upasnuri,'SchemaName',schema);
          output;
          end;
             /* Check to see if there are any more Directory objects */

        n+1;
        uprc=metadata_getnasn(liburi,'UsingPackages',n,upasnuri);
```

```
                end; /* if uprc > 0 */
        /* Look for another library */
        nlibobj+1;
        librc=metadata_getnobj("omsobj:SASLibrary?@Id contains '.'",nlibobj,liburi);
      end; /* do while (librc>0) */
    run;
    /**** End of Code to find the physical location assoc with SMC Libraries  ***/
```

Use Cases

Here is a list of use case examples that we have seen or could quickly respond to:

- Corporate Compliance has asked us to provide a list of Excellus SAS users.
- What resources can Lara Croft access?
- What users, business teams have access to the XYZ library?
- Profile business units' unix group configurations and develop a minimum configuration for new user for each business unit. This is a best practice for security provisioning.
- John Wick's accounts were accidently deleted yesterday, what unix groups did he have? Need to rebuild his unix account and SAS Metadata account.
- Evelyn Salt changed departments two years ago, (dropping SAS and unix accounts) but has returned, what unix and SMC Groups did she have before he left.
- The SAS Administration Team is migrating from unix to a linux platform and we have an opportunity to re-architect and optimize our security model. The implementation consultant and project manager need an overview of the current security model implementation.
- The New York State Department of Financial Services is auditing a regulatory submission from 2014. The equalization SAS code was written by analyst Robert McCall who left the company in 2016. Locate McCall's equalization code and data.
- Excellus IT Department is decommissioning the Legacy Mainframe Data Warehouse data. Provide a list of all users who have been granted permission to use this Legacy Mainframe Data Warehouse data.

Conclusion

The goal of this paper has been to show that while identity, authorization and security data may be available from different sources it is disparate, and likely presented in a transactional format. Going to multiple sources and working with individual level data is acceptable for certain purposes but for meta level reporting, analysis and decision making it is unworkable. By making use of SAS® Institute provided macros and code and harvested SAS® Metadata, human resources, and LDAP data we have been able to create a consolidated source of information on our users and data resources. This consolidated source of information allows us more effectively manage users and data resources, to provide better customer service and respond quickly to management and inquiries from internal and external regulators.

Contact Information:

Hugh G. McCabe
Excellus Blue Cross Blue Shield
hugh.mccabe@excellus.com