

Using Dynamic Views as a Supplement to SAS Security to Enhance Multiple Levels of Access Requirements to Row-Level Data Upon SAS® Server Startup

Chris Bresson, Travelers, Hartford, CT and Marty Flis, SAS

Abstract

Many industries are challenged with regulatory requirements and customer demand for database access within the corporate environment. There must be restrictions in place to protect Personal Information and limit its access.

This paper will show a supplement to SAS security to allow for row level access to sensitive data and restrict access based upon dynamic views driven by datasets at the time of autoexecution as well as group settings in SAS Management Console®.

This supplement works well with user prompts in SAS Stored Processes®, and SAS Enterprise Guide®.

The code has been tested in SAS® 9.1.3 and 9.2 and on Windows, Unix and Linux operating systems.

Benefits include: transparency to users, masking sensitive fields, and using dynamic views capable of handling multiple users with different access requirements for limiting observation displays, limiting libraries in the server list as well as the list of tables to be displayed for incorporating row level security or full data access.

This method (or model) is only meant as a guide to reduce the access to sensitive data and must be coordinated with the security of SAS and the operating system's permissions to folders and files. It does not preclude a programmer's ability to code a libname statement and create a library to sensitive data, but the method (or model) does help reduce the footprint to where the sensitive data is stored and displays only the metapath to the data.

The display of sensitive data in a library can be redirected to a listing of views mimicking the actual protected library list of files. The alternate listing can point to a set of views incorporating the row level access and can be activated during the autoexec execution based on a user's identity and what metadata group their id belongs to

Introduction

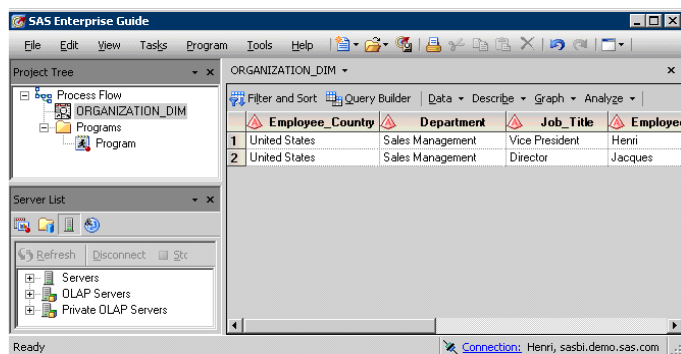
The purpose of a dynamic view is to show that one view, accessing a table, can be controlled to selectively output data by metadata group membership, using formats.

A side by side view of Ole's output and Henri's access to the same Organization_Dim table in the same Orion Star library can be filtered to render different results.



	Employee_Country	Department	Employee_Name	Salary
1	United States	Administration	Gloria	\$32,955
2	United States	Administration	Ole	\$25,195
3	United States	Administration	Susan	\$25,735

Image_1



	Employee_Country	Department	Job_Title	Employee
1	United States	Sales Management	Vice President	Henri
2	United States	Sales Management	Director	Jacques

Image_2

Assumptions

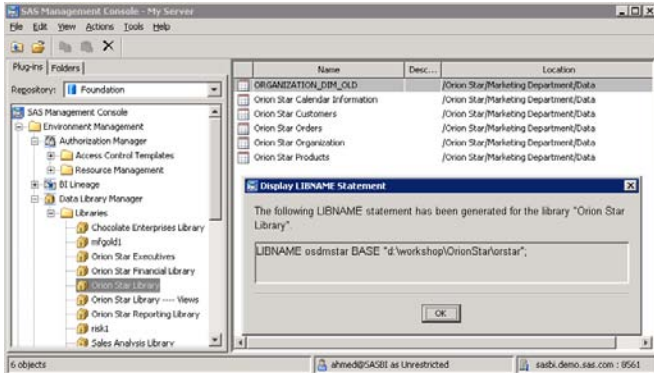
The assumption of this paper is that libnames, which must be regulated, are setup solely through the Autoexec_usermods.sas. Libnames may be present in individual programs, or embedded into EG projects, but the regulation of secured libraries, containing data which is to be filtered or not to be edited, would have to be executed before the SAS session begins. There also has to be a clear cut

grouping of individuals into metadata groups so that the logic of the autoexec has a mutually exclusive decision of who will receive which set of libnames and/or filtered views.

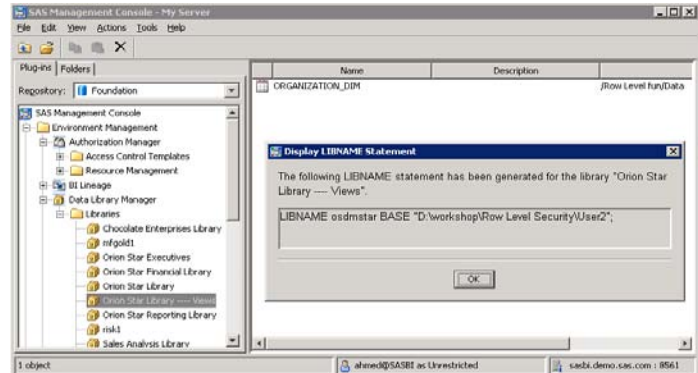
Pre-Existing Code

A requirement for adopting a supplemental security which affects libname statements is that the use of librefs is important to pre-existing code and projects and should not have to be edited.

To accomplish this there are two identical librefs created in SAS Management Console[®] with different library names for each. Once logic in the Autoexec_usermods.sas is executed, the libname statement to be redirected will be using the same libref but the 'meta library=' option will use the alternate metadata library name. The actual library path is hidden from the user in the metadata. In this way the libref is always consistent, although the location of the tables or the views may change.

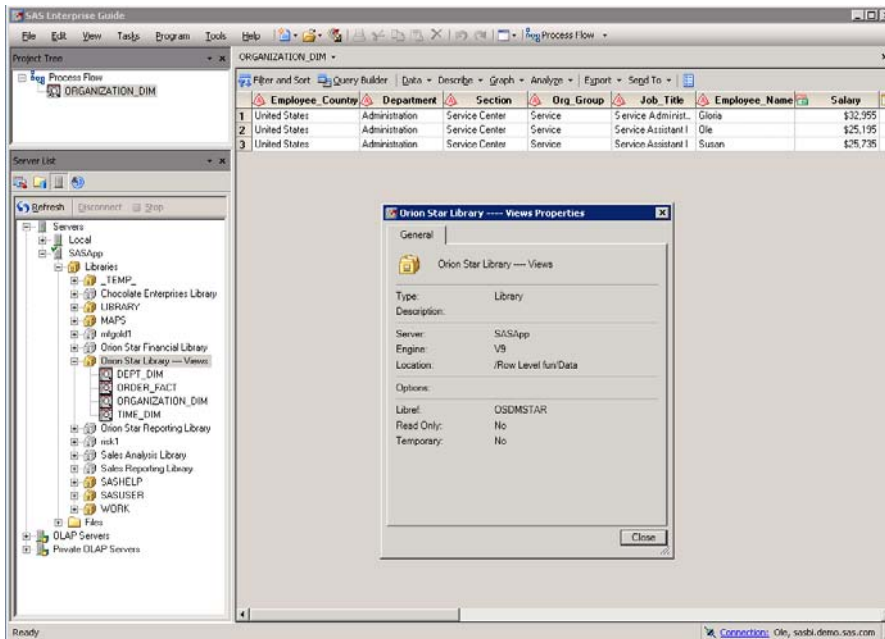


Image_3



Image_4

A user cannot determine a path from the SAS Enterprise Guide[®] Server List window once the Libname <libref> Meta Library= statement has assigned the library name from the Metadata.



Image_5

There is data step statement logic which can be used to pull the id of an individual starting a SAS session and incorporating the metadata groups that individual belongs to in order to assign a libref to a specific library name in metadata.

With the proper logic, library references can be controlled from autoexec_usermods.sas. In addition to controlling the location a libref points to, the libraries listed in the 'Server List' of an EG session can also be controlled. In other words, libraries can be turned on or off. Only the libraries that would be appropriate to an individual need be assigned, whereas other libraries would not even have to be displayed.

Creating the SQL Views Within Metadata Groups

Once the direction has been determined as to where to point a library reference, then a robust or dynamic view handling the output of the table's data for each individual, can be created. These views need to operate without hard coded values built into their where clauses. They must be flexible enough to provide a way of changing the content to be delivered, based on who is requesting access. Formats can be substituted into a where clause. Instead of a variable(s) being compared across an operator (ie. '=') to a defined string or numeric value, a format can take the defined value's place. A libname statement must be built into the view so it can handle the source of the data without relying on the autoexec and the libnames being redirected.

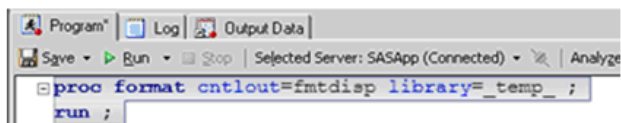
```
libname osdmsta_base "D:\workshop\Row Level Security\OrionStar\User2";

proc sql;
  create view osdmsta_organization_dim as
  select *,
  INPUT(put(UPCASE(department), $DPFILTR.),8.) as dept_flag,
  INPUT(put(UPCASE(section), $SCFILTR.),8.) as sec_flag,
  INPUT(put(UPCASE(org_group), $OGFILTR.),8.) as org_flag,
  INPUT(put(UPCASE(TRIM(LEFT(employee_name))), $RETURN.),8.) as return_val
  from _einfo.ORGANIZATION_DIM (rename=(salary=_salary))
  where
    (
      sum(
        INPUT(put(UPCASE(department), $DPFILTR.),8.), /* Returns 4 if TRUE */
        INPUT(put(UPCASE(section), $SCFILTR.),8.), /* Returns 2 if TRUE */
        INPUT(put(UPCASE(org_group), $OGFILTR.),8.), /* Returns 1 if TRUE */
      )
      =
      INPUT(put(UPCASE(trim(left(employee_name))), $RETURN.),8.) /* Returns Comparison Value */
    )
  and
  INPUT(put(UPCASE(trim(left(employee_name))), $RETURN.),8.) ne 0
  using libname _einfo BASE 'D:\workshop\OrionStar\orstar';
quit;
```

Image_6

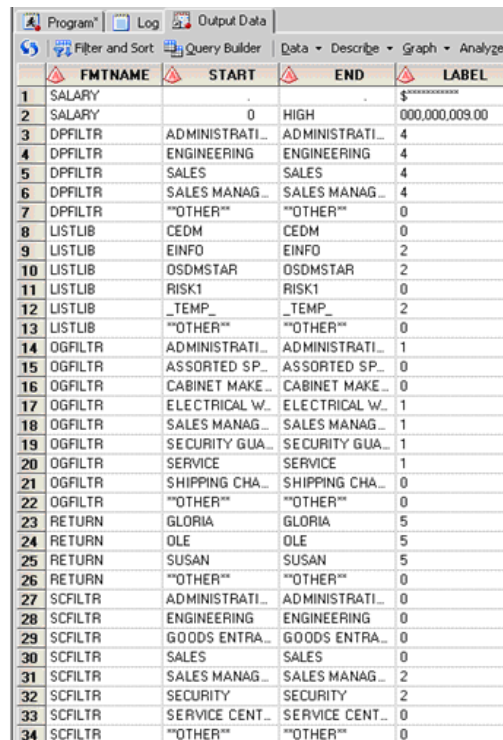
Loading the Formats

Defined formats would need to execute before the libname in autoexec_usermods.sas is assigned. They have the advantage of being held in a catalog and can be built into the Work library. Catalogs cannot be easily seen by the EG user and the original format, executed during autoexec_usermods.sas is held in memory and would still be active, even if the user manipulated the catalog. Since the format is built in the Work library, the format would disappear with the end of that user's SAS session, and a new format would then be assigned and independent of all other users accessing the same library or view.



```
proc format cntlout=fmtdisp library=_temp ;
run ;
```

Image_7



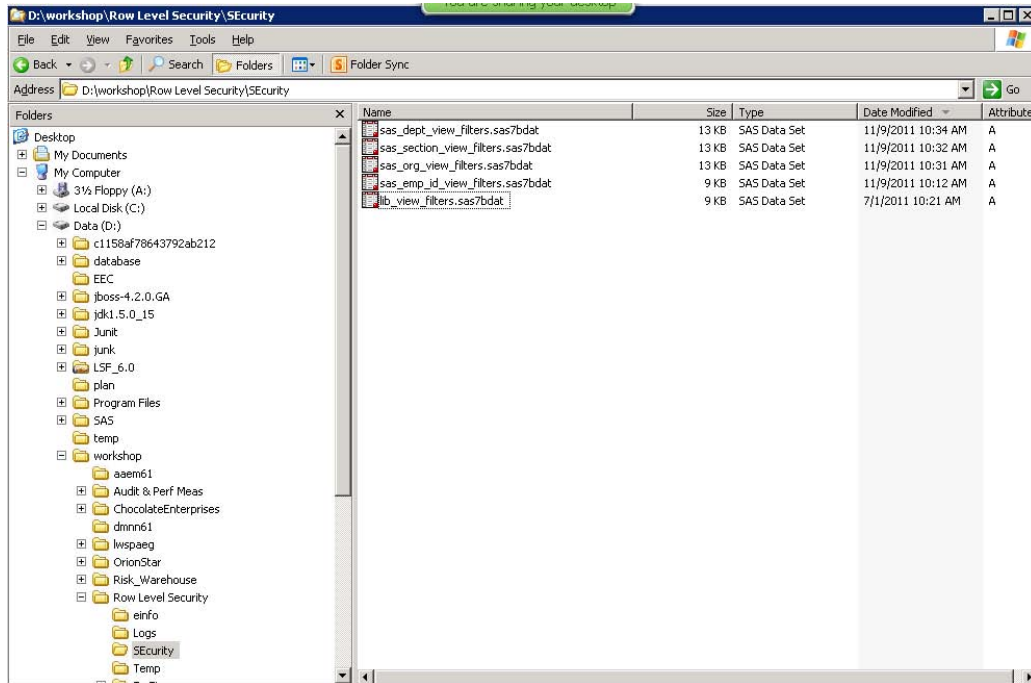
	FMTNAME	START	END	LABEL
1	SALARY			\$*****
2	SALARY	0	HIGH	000,000,009.00
3	DPFILTR	ADMINISTRATI...	ADMINISTRATI...	4
4	DPFILTR	ENGINEERING	ENGINEERING	4
5	DPFILTR	SALES	SALES	4
6	DPFILTR	SALES MANAG...	SALES MANAG...	4
7	DPFILTR	""OTHER""	""OTHER""	0
8	LISTLIB	CEDM	CEDM	0
9	LISTLIB	EINFO	EINFO	2
10	LISTLIB	OSDMSTAR	OSDMSTAR	2
11	LISTLIB	RISK1	RISK1	0
12	LISTLIB	_TEMP	_TEMP	2
13	LISTLIB	""OTHER""	""OTHER""	0
14	OGFILTR	ADMINISTRATI...	ADMINISTRATI...	1
15	OGFILTR	ASSORTED SP...	ASSORTED SP...	0
16	OGFILTR	CABINET MAKE...	CABINET MAKE...	0
17	OGFILTR	ELECTRICAL W...	ELECTRICAL W...	1
18	OGFILTR	SALES MANAG...	SALES MANAG...	1
19	OGFILTR	SECURITY GUA...	SECURITY GUA...	1
20	OGFILTR	SERVICE	SERVICE	1
21	OGFILTR	SHIPPING CHA...	SHIPPING CHA...	0
22	OGFILTR	""OTHER""	""OTHER""	0
23	RETURN	GLORIA	GLORIA	5
24	RETURN	OLE	OLE	5
25	RETURN	SUSAN	SUSAN	5
26	RETURN	""OTHER""	""OTHER""	0
27	SCFILTR	ADMINISTRATI...	ADMINISTRATI...	0
28	SCFILTR	ENGINEERING	ENGINEERING	0
29	SCFILTR	GOODS ENTRA...	GOODS ENTRA...	0
30	SCFILTR	SALES	SALES	0
31	SCFILTR	SALES MANAG...	SALES MANAG...	2
32	SCFILTR	SECURITY	SECURITY	2
33	SCFILTR	SERVICE CENT...	SERVICE CENT...	0
34	SCFILTR	""OTHER""	""OTHER""	0

Image_8

Setting Up the Filter Tables

The SAS formats which control the SAS SQL[®] views of a Libname <libref> Meta Library= statement are driven from SAS tables. These tables are called the filter tables and should be held in a non disclosed library, which then can be used to easily change and manipulate the format catalogs and therefore the content the SQL view would display.

SAS tables, which control the SQL view filters, can be easily maintained. Each variable built into a where clause to determine content would have a SAS filter table created for it. The table would drive the value for that individual variable in the where clause. The filter table would contain a minimum of three variables to assign values.



Image_9

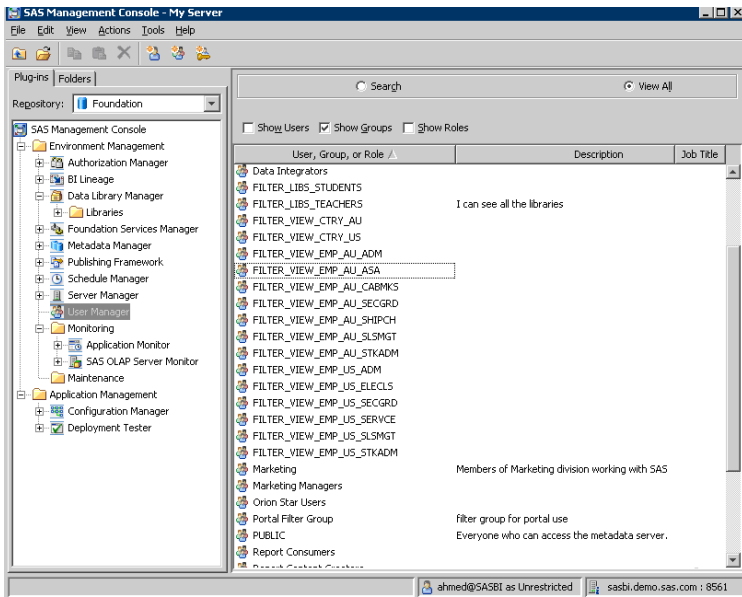
First Column of the Filter Table

The first variable would be for the group a SAS user can belong to. The variable group would contain the exact name of each group in the SAS metadata which is meant to control access for an individual. All groups do not need to be assigned in this table, just groups which will control libname assignment logic. Using a special prefix helps to differentiate the groups in the SAS Management Console[®].

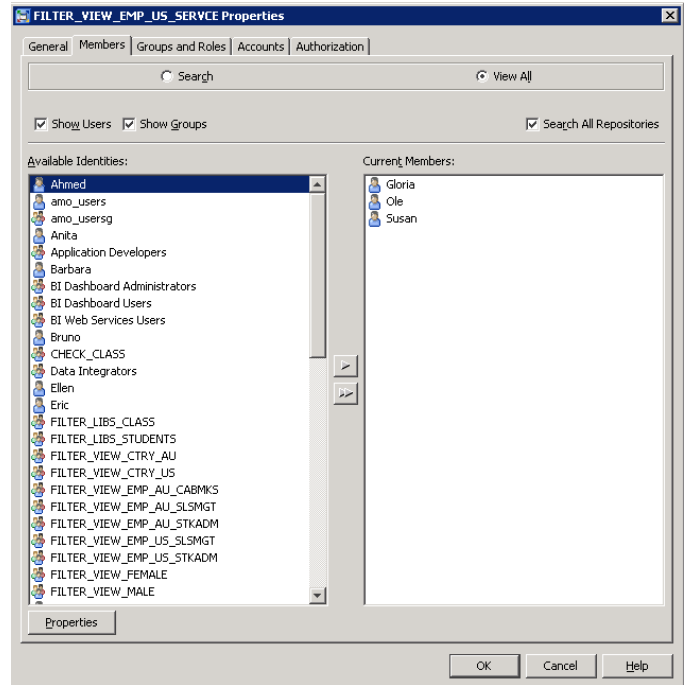
A suggestion would be to prefix the groups with 'FILTER_VIEW_<SHORTVAR>_' where <SHORTVAR> would represent the name of the variable in the where clause. The suffix would differentiate the groups from each other and can be the short string representation of the values of the variable in the where clause. An example for a department variable to be used in a where clause would have a group named 'FILTER_VIEW_DEP_D1' or 'FILTER_VIEW_DEP_D2'. This differentiation could be programically used to help check for group memberships when autoexec_usermods.sas runs.

Below are groups created for the 'Orion Star Library' (See *Image_10* below). The Organization_Dim table will be split out by the groups which are contained in the table. In this example, there are filter variables which can split the view based on the value each filter variable returns. The sum of all the filter variables in the view will be compared to a value called the 'Return' value. The variables used, to bring in values to sum on, will be Department, Section, Org_Group and the 'Return' value will be based on the user signing into the SAS session. The user's id will be picked up from the 'EID' variable in the 'SAS_Emp_Id_View_Filters' Filter Table.

If an individual belongs to the 'FILTER_VIEW_EMP_US_SERVICE' group, then the values assigned to the filter variables' values of Organization_Dim (Department, Org_Group, Section) are returned for that group and compared to the Employee_Name's return value. If they satisfy the operator of the where clause, in this case an '=' sign, then the observation will be allowed to pass the view.



Image_10

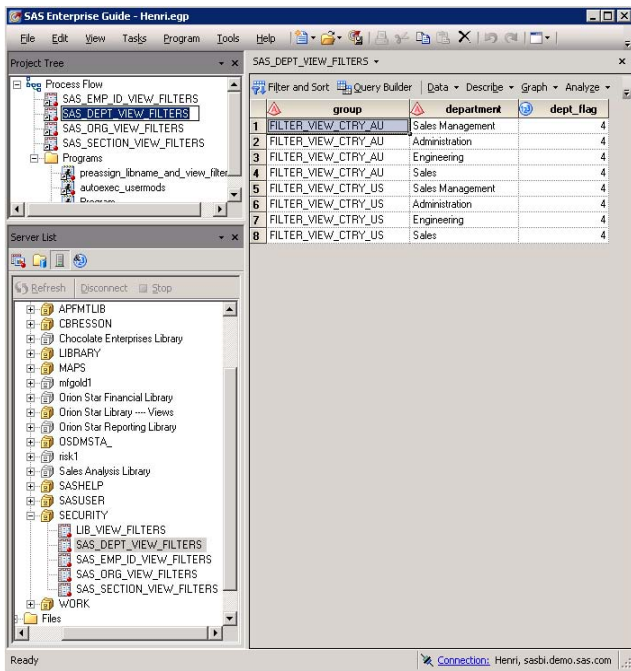


Image_11

The individuals in the Organization_Dim table are also organized into the Metadata groups created. When the groups are applied to the views, then the views will return the appropriate output (See *Image_11* above).

Second Column of the Filter Table

The filter variables' values, which are compared in the format of the where clause, are held in the second column of the filter datasets for each filter variable. The name of this column, which holds the values to appear in the original SAS table (Organization_Dim), can be named after the variable they represent, such as Department.



Image_12

group	org_group	org_flag
1	Sales Management	1
2	Administration	1
3	Shipping Charges	0
4	Stock Admin	1
5	Security Guards	1
6	Service	1
7	Cabinet Maker's Workshop	0
8	Electrical Workshop	1
9	Assorted Sports Articles	0
10	Sales Management	1
11	Administration	0
12	Shipping Charges	0
13	Security Guards	1
14	Service	1
15	Cabinet Maker's Workshop	1
16	Electrical Workshop	1
17	Assorted Sports Articles	0

Image_13

group	section	sec_flag
1	Sales Management	2
2	Administration	0
3	Goods Entrance	2
4	Security	2
5	Service Center	0
6	Engineering	0
7	Sales	0
8	Sales Management	2
9	Administration	0
10	Security	2
11	Goods Entrance	0
12	Service Center	0
13	Engineering	0
14	Sales	0

Image_14

The Third or Remaining Columns of the Filter Table

The next variable(s) in the filter tables would contain the flag value(s). This value is usually a numeric value and will be used to transform the values held in the second variable. The flag value is then returned to the SAS format in the SQL view and compared to the value of the variable for that observation which the SAS SQL® view is pointing to. If there is a Department variable in the SAS dataset and the view is subsetting on it, then the observation is loaded into that view, the value of that observation is submitted and transformed by the format for that variable and a returning value is then compared to for subsetting on. **NOTE:** It is important to set the return value for the user signing in, as the return value to be used in the format for subsetting all observations on. This information comes from the 'EID' variable in the 'SAS_Emp_Id_View_Filters' Filter Table.

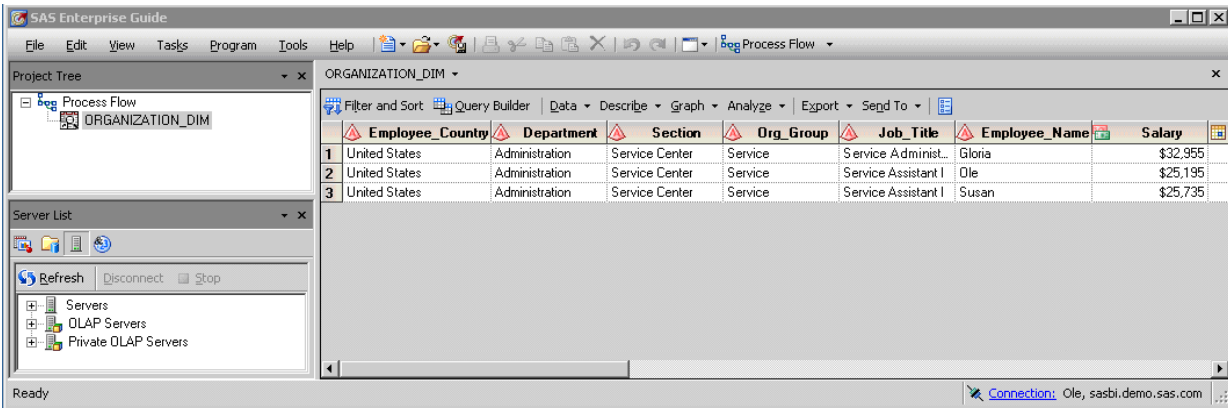
If boolean values are being returned from the filter table, then a subsetting statement in the SQL view can be written to accept any value equal to 1. A value of 1 can be set to correspond to the value of a filter variable, in the Filter table. If an individual belongs to the group for that filter value in the filter table, then the flag for that observation will be pulled into the view from that SAS dataset. Other groups may be assigned a value of 1 for that filter variable but anyone who has been assigned an explicit value of 0, or is not assigned in the filter table (implicit value of 0), will not see the row coming from the SAS table.

Below is an example of 'Return' values to which the filter variables can be compared to when summed on:

- 0 = No Access
- 1 = Only Org_Group Level Access
- 2 = Only Section Level Access
- 3 = Combine Org_Group and Section Level Access
- 4 = Only Department Level Access
- 5 = Combine Department and Org_Group Level Access
- 6 = Combine Department and Section Level Access
- 7 = Combine Department, Section and Org_Group Level Access

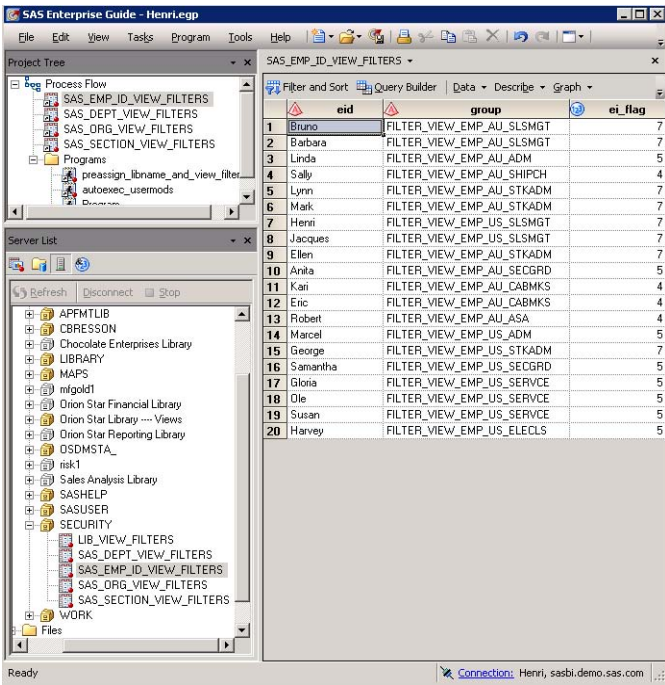
In this example, the view for an employee is not compared at a Boolean level of 0 or 1 but to a summation of values which allow more flexibility to access data.

In *Image_15*, Ole signed in to see a filtered view. Since he belonged to 'Administration' Department, 'Service Center' Section, and 'Service' Org_group, the value returned for Department was 4 (see *Image_12*), Org_group was 1 (see *Image_13*) and Section was 0 (see *Image_14*).



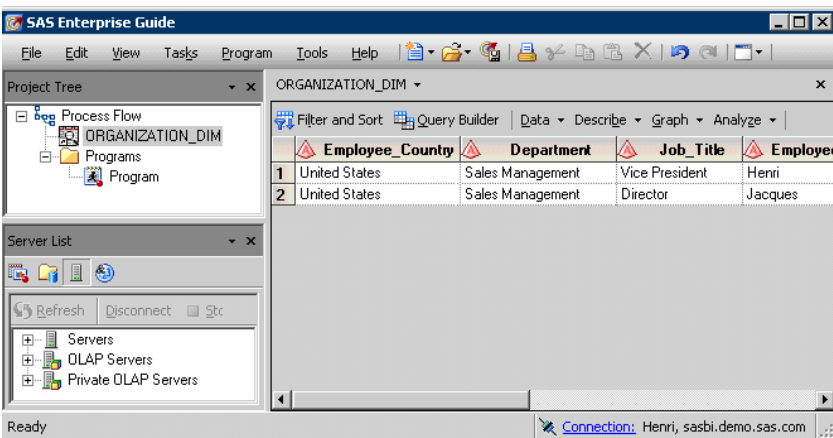
Image_15

As a result, a summed value for Ole's 'Return' was 5 (see Image_16 below) and a format was created (see Image_8) for Ole, during the kickoff of Autoexec_usermods.sas, to control the view's output.



Image_16

If Henri had signed in instead (see Image_17), a different format for that SAS session would have been created just for him in the Work library and he would have seen a different set of observations using the same dynamic view Ole was assigned.

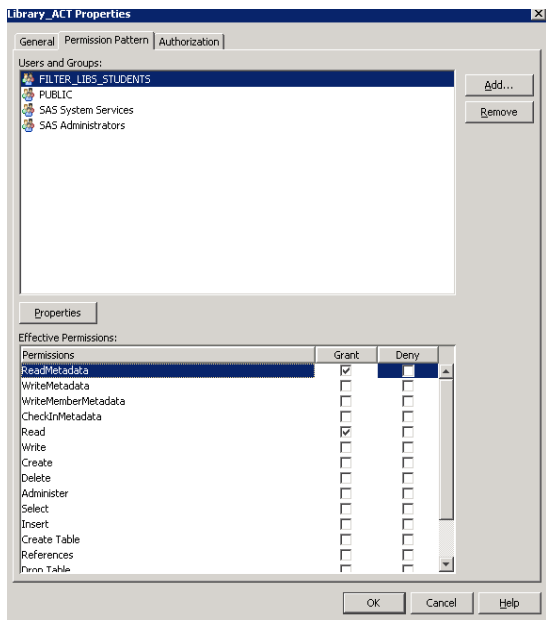


Image_17

Permissions

Permissions for the 'Orion Star Library' must be set to allow access as well as preventing two similar libraries from showing up in the 'Server List'. Note that it is possible to have two or more libraries set up in the SAS Management Console® with different library labels but the same libref (See *Image_3* and *Image_4*). One library has to be 'turned off' while the other is 'turned on'.

To maintain each view using the dynamic security, an ACT can be created and a group included, which will represent the individuals for whom security should be applied (See *Image_18*).

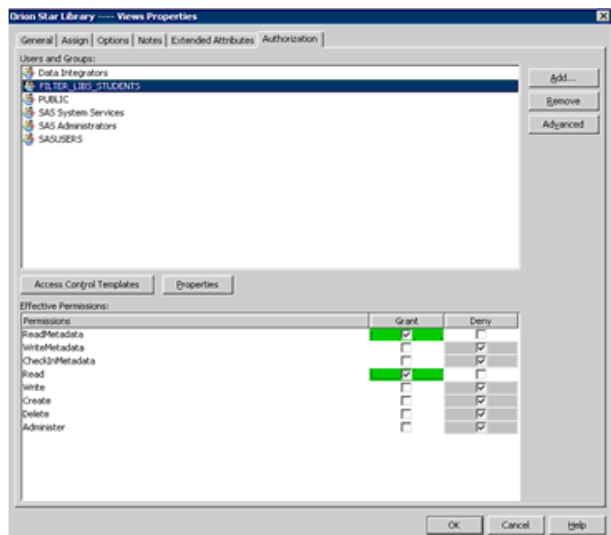


Image_18

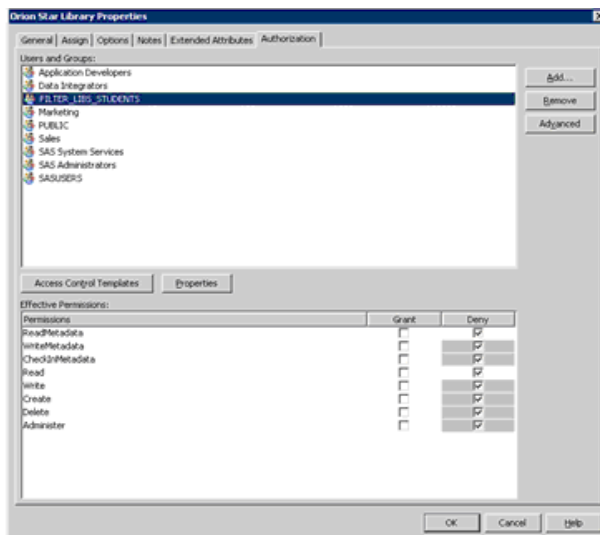
Once the new ACT is created, it can be applied to the library in the Metadata which will hold the dynamic views. Make sure the group created in Metadata, containing the names of all individuals whom security is applied, has the correct settings.

There is a higher level group checked in Autoexec_usermods.sas called "FILTER_LIBS_STUDENTS". The 'Orion Star Library ---- Views' library, displayed in the 'Server List' of EG, will be turned on for those individuals which belong to "FILTER_LIBS_STUDENTS" group. Access to ReadMetadata and Read Permissions should be set to Grant for the FILTER_LIBS_STUDENTS group and the Public and SASUSERS group settings should be all set to Deny (*Image_19*).

The 'Orion Star Library' library will not show up in the 'Server List' of EG because it will have been turned off for the individuals belonging to "FILTER_LIBS_STUDENTS". The SASUSERS and PUBLIC groups should have all permissions set to deny (*Image_20*).



Image_19



Image_20

Masking within Metadata Groups

The salary field can be masked using the dynamic SQL views. If a sum of the filters is greater than a specific value then that group should be given access to the salary data. If the sum is less than or equal to a specific value then those individuals will see the salary field masked (see *Image_21*).

```
libname osdmsta_base "D:\workshop\Row Level Security\OrionStar\User2";

proc sql;
  create view osdmsta_.organization_dim as
  select
  /*****
  Code for MASKING Salary
  *****/
  coalesce(
    case
      when
        (
          sum(
            INPUT(put(UPCASE(department), $DPFILTR.),8.) /* Returns 4 if TRUE */
            ,
            INPUT(put(UPCASE(section) , $SCFILTR.),8.) /* Returns 2 if TRUE */
            ,
            INPUT(put(UPCASE(org_group) , $OGFILTR.),8.) /* Returns 1 if TRUE */
          )
          le
            5 /* Returns Comparison Value */
        )
        and |
          INPUT(put(trim(left(employee_name)), $RETURN.),8.) ne 0
        )
      then .
      else _salary
    end
    , .) as salary format=salary15.,
  employee_country,
  department,
  section,
  org_group,
  job_title,
  employee_name,
  /* Flags for viewing Return Values and testing output */
  INPUT(put(UPCASE(department), $DPFILTR.),8.) as dept_flag,
  INPUT(put(UPCASE(section), $SCFILTR.),8.) as sec_flag,
  INPUT(put(UPCASE(org_group), $OGFILTR.),8.) as org_flag,
  INPUT(put(TRIM(LEFT(employee_name)), $RETURN.),8.) as return_val
  from _einfo.ORGANIZATION_DIM (rename=(salary=_salary))
  where
```

Image_21

Below we see the group which Gloria belongs to, as being able to view the salary data.

	Employee_Country	Department	Section	Org_Group	Job_Title	Employee_Name	Salary
1	United States	Administration	Service Center	Service	Service Administr...	Gloria	\$32,955
2	United States	Administration	Service Center	Service	Service Assistant I	Ole	\$25,195
3	United States	Administration	Service Center	Service	Service Assistant I	Susan	\$25,735

Image_22

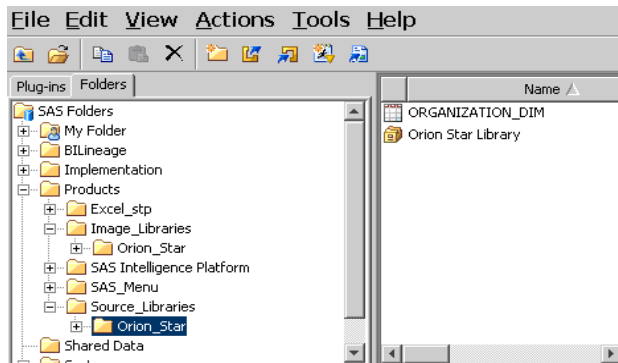
If we change the code to reflect a specific value, such as le 5 (see *Image_21*), to mask the data below a certain level, we can change the output of Gloria's view.

	Employee_Country	Department	Section	Org_Group	Job_Title	Employee_Name	Salary
1	United States	Administration	Service Center	Service	Service Administr...	Gloria	\$*****
2	United States	Administration	Service Center	Service	Service Assistant I	Ole	\$
3	United States	Administration	Service Center	Service	Service Assistant I	Susan	\$*****

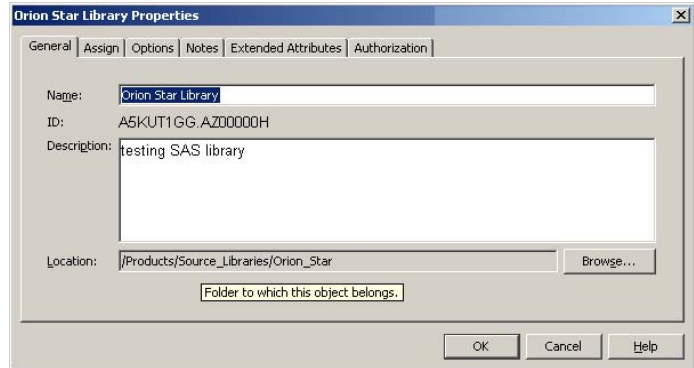
Image_23

Separating Source and Image Library Content

When registering views or tables in SAS metadata, separate folders must be created for 'Source' and 'Image' libraries to avoid duplicate registration names of a dataset and a view. Below, the folder tab (see *Image_24*) and the tree for the 'Source' library and its content is displayed in SAS Management Console®. The corresponding library properties window also displays the metadata path to the folder tree (see *Image_25*).

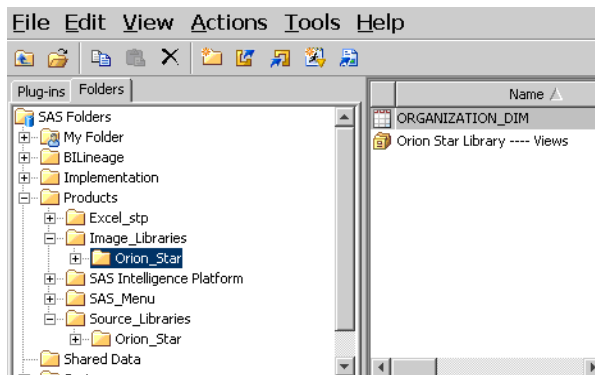


Image_24

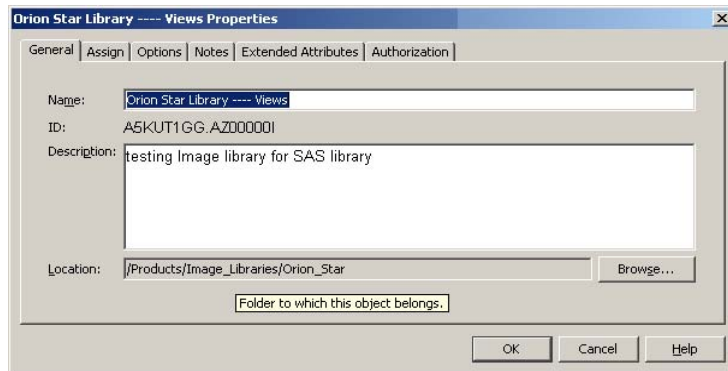


Image_25

A duplicate 'Image' tree is displayed under the folder tab in SAS Management Console® as well as the properties window for the views 'Image' library.



Image_26



Image_27

Displaying and Masking Output Across Metadata Groups

Displays of data across ranges of SAS metadata groups can be controlled for output by placing additional columns into the 'SAS_Emp_Id_View_Filters' Filter Table. Where as previous output was based on the 'ie_flag' to supply a 'Return' value to the right of the equation in the SAS SQL® view, we can supply a range of values to be returned (see *Image_28*). Adding a 'low_ie_flag' and a 'hi_ie_flag' to the code, filter tables and formats, as well as a between operator in the SAS SQL® view will control for ranges as well as masking data (see *Image_29*).

	Employee_Country	Department	Section	Org_Group	Job_Title	Employee_Name	Salary
1	United States	Administration	Administration	Administration	Office Administra...	Marcel	\$*****
2	United States	Administration	Goods Entrance	Stock Admin	Warehouse Man...	George	\$*****
3	United States	Administration	Security	Security Guards	Security Manager	Samantha	\$*****
4	United States	Administration	Service Center	Service	Service Administr...	Gloria	\$*****
5	United States	Administration	Service Center	Service	Service Assistant I	Ole	\$*****
6	United States	Administration	Service Center	Service	Service Assistant I	Susan	\$*****
7	United States	Engineering	Engineering	Electrical Works...	Electrician II	Harvey	\$*****
8	United States	Sales Managem...	Sales Managem...	Sales Managem...	Vice President	Henri	\$*****
9	United States	Sales Managem...	Sales Managem...	Sales Managem...	Director	Jacques	\$*****

Image_28

```
libname osdmsta_base "D:\workshop\Row Level Security\OrionStar\User2";
```

```
proc sql;
  create view osdmsta_organization_dim as
  select
  /*****
  Code for MASKING Salary
  *****/
  coalesce(
    case
      when
        (
          (
            sum(
              INPUT(put(UPCASE(department), $DPFILTR.),8.) /* Returns 4 if TRUE */
              |
              INPUT(put(UPCASE(section) , $SCFILTR.),8.) /* Returns 2 if TRUE */
              |
              INPUT(put(UPCASE(org_group) , $OGFILTR.),8.) /* Returns 1 if TRUE */
            )
            le
              7 /* Returns Comparison Value */
          )
          and
            /* Level of Cutoff For Viewing Masked Salary Data */
            /* Higher then 5 allows UnMasked data of Salary */
            INPUT(put(trim(left(employee_name)), $LRETURN.),8.) le 5
        )
      then .
      else _salary
    end
    , .) as salary format=salary15.,
  employee_country,
  department,
  section,
  org_group,
  job_title,
  employee_name,
  INPUT(put(UPCASE(department), $DPFILTR.),8.) as dept_flag,
  INPUT(put(UPCASE(section), $SCFILTR.),8.) as sec_flag,
  INPUT(put(UPCASE(org_group), $OGFILTR.),8.) as org_flag,
  INPUT(put(TRIM(LEFT(employee_name)), $LRETURN.),8.) as lo_return_val,
  INPUT(put(TRIM(LEFT(employee_name)), $HRETURN.),8.) as hi_return_val
  from _einfo.ORGANIZATION_DIM (rename=(salary=_salary))
  where
```

```

  /*****
  Code for gaining Row Level Security based on group membership
  *****/
  (
    sum(
      INPUT(put(UPCASE(department), $DPFILTR.),8.) /* Return value of 4 if TRUE */
      |
      INPUT(put(UPCASE(section) , $SCFILTR.),8.) /* Return value of 2 if TRUE */
      |
      INPUT(put(UPCASE(org_group) , $OGFILTR.),8.) /* Return value of 1 if TRUE */
    )
    between
      INPUT(put(trim(left(employee_name)), $LRETURN.),8.)
      and
      INPUT(put(trim(left(employee_name)), $HRETURN.),8.)
    )
    and
      INPUT(put(trim(left(employee_name)), $HRETURN.),8.) ne 0
  )

```

```

  /*****
  Change this hidden libname to something not pre-defined in preassign.
  ie: einfo einfo_
  *****/
  using libname _einfo BASE 'D:\workshop\OrionStar\orstar';

```

```
quit;
```

Image_29

Conclusions

In conclusion, it is important to stress that for this added layer of security to function properly, the libnames must be provided for the end users and that the libname statement embedded into the code or supplied by the power user could circumvent the views put into place. Care must be given not to disclose the location of sensitive data or the pathway to the views. The security library must also be protected.

Formats are loaded into memory. Even though the format catalog maybe changed within the work library, the new format will not be recognized unless the change is made in the Autoexec_usermods.sas.

The SQL dynamic view is a one to one view with a SAS or relational table in each library. The view needs its own folder where it and other security views will reside to mimic the library they point to. This library is the 'image library' and will contain all SQL views which need to protect their SAS table counterparts. If there is a SQL view for a table, then the view can be registered in the SAS Management Console® and be available to use in the 'Server List' of EG, thus allowing control of access to secured tables. A SQL view can be created, registered and included into the 'image library' but not contain the formats to filter the views. These SQL views would allow full data access to a user but still protect the data from editing. These 'image libraries' would not require much disk space since views contain no data and two or more 'image libraries' can be created for each 'source library' they point to. Two or more sub-directories can be created for an 'image library'. Each sub directory can contain a one to one view for each dataset to access. A sub-directory can contain views built without filters and another sub-directory can contain views built with filters. The libname statement in Autoexec_usermods.sas would then handle the logic of whether a user should be directed to the non-filtered 'image library' or directed to the filtered 'image library'. If a user does not belong to a higher level group, the logic can also clear a library with the libname statement 'Libname <ref> clear ;' or not redirect the original libname statement at all and allow full access and edit capability to the 'source library'.

The SAS administrator should be aware that registering a view embedded with user defined formats in the Work library can be tricky. The formats have to exist during the process of registering the SQL views. During registration, the Autoexec_usermods.sas is executed. The formats must be executed unconditionally during this time to be available for the registration of the view. This unconditional code can be called from Autoexec_usermods.sas, just during registration, and commented back out for production.

Caution: using Row Level Security will create only partial data output for users, and this data should not be used for full numeric summary reporting.

Acknowledgments

1. Tina Hobbs (2008). "Using statement in Proc SQL creating libref in SAS view", SAS Track 7610022639
2. Bubba Talley (2008). "Provide PROC METADATA and XML Libname engine code", SAS Track 7610009774
3. Bubba Talley (2008). "Describe the META engine for libnames", SAS Track 7610027836
4. Fran Olson (2008). "Code to list groups in the repository", SAS Track 7610131023
5. Joe DeRusso (2008). "How to get user information from login", SAS Track 7610088743
6. David Helwig for his assistance in contacting Marty Flis of SAS Institute.

Contact Information

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

Christopher Bresson
Travelers
Phone: 860.420.7121
E-mail: Christopher_Bresson@yahoo.com

Marty Flis
SAS
Phone: 919.531.9087
E-mail: Marty.Flis@SAS.com

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.