Paper 293-2013

Implementing metadata-driven reports with SAS® Stored Processes

Toby Hill, Charles Marcus Group Services, London, UK

ABSTRACT

As more organizations that use SAS® software are implementing the full Business Intelligence reporting suite, many SAS programmers are becoming familiar with developing Stored Processes to deliver reports for the business. Developers are often required to implement content security in the reports or provide additional functionality for users with specific roles. How can all this be done? One approach is to make use of the SAS metadata. This paper demonstrates some techniques that you can apply to your SAS code in order to make use of the SAS metadata. This will allow you to implement security and role-based access in your Stored Process reports and minimize the amount of changes required as new users access the platform.

INTRODUCTION

Using a Stored Process to deliver reports allows you to surface your BI content via the web or with other SAS client tools such as the MS Office Add-In.

Large organizations with a varied user base will generally have different business teams with their own security requirements. With the correct logic you can build a Stored Process that will be aware of the SAS metadata and display only the appropriate content for the users in these different teams. In order to reduce the overall effort maintaining a growing collection of SAS Stored Processes it is good practice to adopt a template for the SAS code that comprises your reporting logic. The examples shown in this paper can be used as the basis for such a template.

By implementing your Stored Process reports to be “metadata-driven” you will:

- reduce maintenance
- provide a flexible and robust interface for reporting
- secure the content of your reports
- adapt to changing user roles

Two scenarios will be considered in this paper:

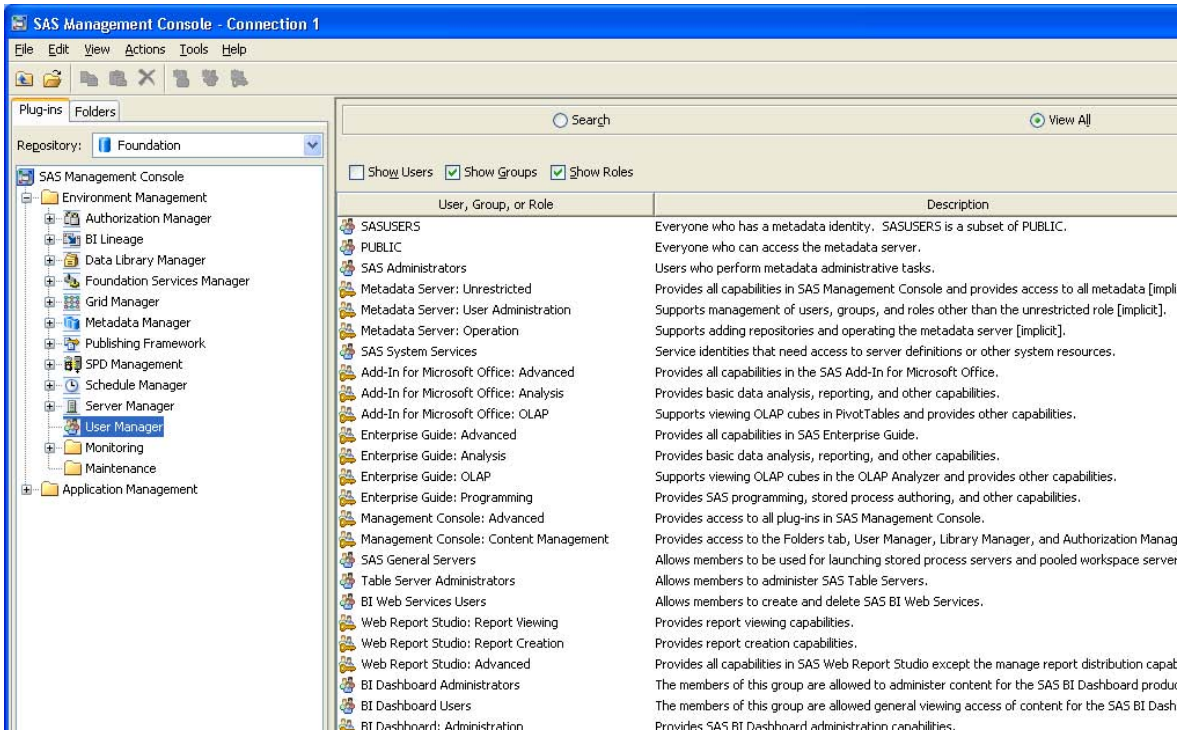
1. Controlling access to a data source at the row level.
2. Extra functionality for users with the appropriate role.

This paper does not cover in detail the steps required to build a SAS Stored Process. If the reader is completely unfamiliar with Stored Processes then it is recommended that you consult the documentation and/or one of the many other SAS papers on the subject.

SAS METADATA

When SAS first released their Version 9 software, it featured an important new module: the metadata server. The metadata server stores many different types of metadata content such as Stored Processes, Web Reports and Batch Job Definitions. This content is integral to the SAS BI Reporting platform and used by practically all the other SAS products and solutions. The metadata server also stores all the users, groups and roles on the platform and this information can be directly accessed from your SAS code.

When a SAS deployment is first implemented a support staff member will act as the SAS Administrator. They will create the users and set up their groups and roles. Full security can be applied to the SAS metadata; and permissions can be configured to prevent users from switching roles or adding another group to their profile.



Display 1. SAS Management Console showing the metadata groups and roles

In order to access the metadata, SAS has provided a set of functions that can be invoked in your SAS code. The full list of functions follows:

- METADATA_DELASSN
- METADATA_DELOBJ
- METADATA_GETATTR
- METADATA_GETNASL
- METADATA_GETNASN
- METADATA_GETNATR
- METADATA_GETNOBJ
- METADATA_GETNPRP
- METADATA_GETNTYP
- METADATA_GETPROP
- METADATA_NEWOBJ
- METADATA_PATHOBJ
- METADATA_PAUSED
- METADATA_PURGE
- METADATA_RESOLVE

- METADATA_SETASSN
- METADATA_SETATTR
- METADATA_SETPROP
- METADATA_VERSION

The code examples in this paper only utilize the METADATA_GETNOBJ and METADATA_GETATTR functions. Full details regarding all the functions can be accessed from the SAS online documentation. The code examples have been tested in a SAS 9.3 and SAS 9.2 environment.

When using the metadata functions you essentially run "queries" against the metadata server. Then you retrieve the results and use logic in your code to handle the result values. In order to effectively query the metadata you need to understand the structure of the SAS metadata and also learn the query syntax for the functions.

The metadata functions use several SAS system options to determine how to connect to the metadata server. These are:

- METASERVER
- METAPORT
- METAPROTOCOL
- METAREPOSITORY
- METAUSER
- METAPASS

Generally these settings are configured with appropriate values automatically when running a Stored Process. They are not detailed in this paper but the settings can be manually assigned with an options statement for testing purposes.

The reader may also note that there is a SAS procedure that provides access to the metadata: PROC METADATA. This paper focuses only on the use of the SAS metadata functions; consult the SAS online documentation for information about PROC METADATA if you are curious.

SAS STORED PROCESSES

When a Stored Process is created, usually it is configured to run on the Stored Process Server. This means whenever the Stored Process is invoked it will be run by a system account (the default system account name is: sassrv). This allows the Stored Process to mediate access to the underlying data. Security is configured to only allow the system account (sassrv) to access the SAS datasets then users run the Stored Processes to report on the data. Users cannot change the Stored Process code (usually they cannot even see the code!) so they will not be able to circumvent the logic in the Stored Process and can only see the content that is displayed for them.

The following reserved SAS macro variables are used by a SAS Stored Process to identify the user who is invoking the Stored Process:

_METAUSER – Userid of the person invoking the Stored Process

_METAPERSON – Metadata Identity of the person invoking the Stored Process

These are key parameters. If you want to identify the metadata groups for a user then you need to be able to identify who is invoking the Stored Process. The code examples in this paper make use of the _METAPERSON macro variable.

SCENARIO 1: CONTROLLING ACCESS TO A DATA SOURCE AT THE ROW LEVEL

Consider this scenario: You want to build a report that will display some tabular data. For example, sales figures, key economic indicators or patient health attributes. When different users run your report, you only want to display the appropriate data. This could be because you don't want to display irrelevant information to a user or it could be because of security concerns.

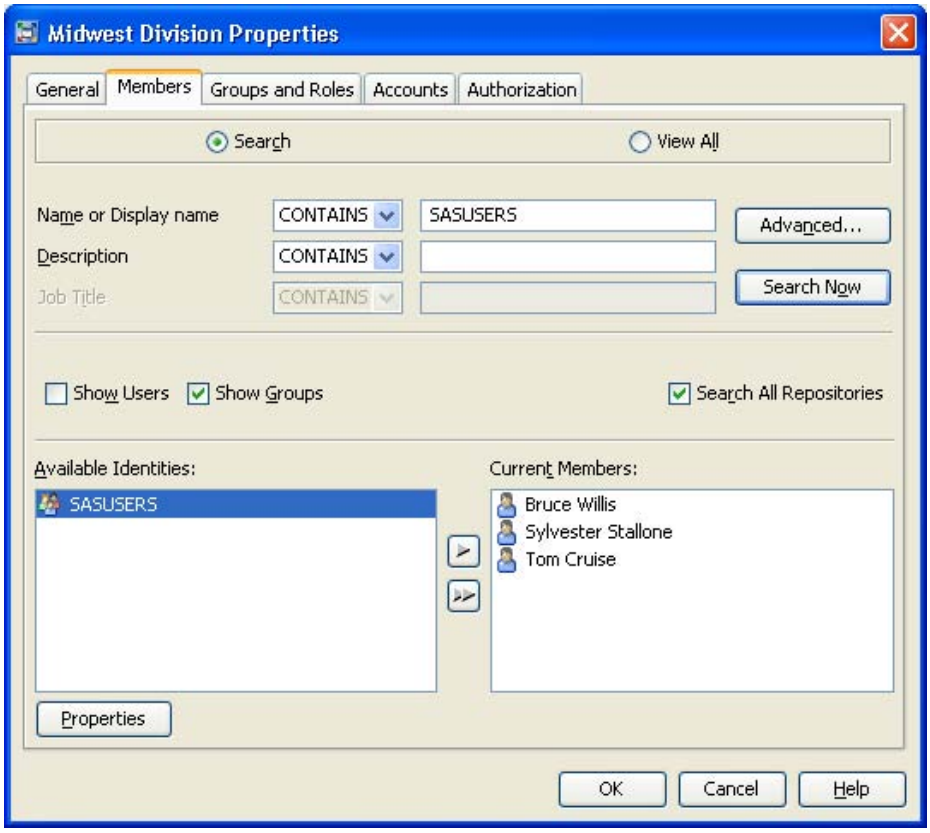
The following SAS code will demonstrate a technique to accomplish this goal. In this example a simple table of sales figures is used. The following display depicts a sample of the data (the dataset name in this example is: SASTEST.ALL_DATA).

	Division	Period	Product	Sales	Target	Diff
1	Northeast Division	2012-10-31	Appliances	1,005,969	702,285	-303,684
2	Northeast Division	2012-10-31	Electronics	1,032,776	295,017	-737,759
3	Northeast Division	2012-10-31	Furniture	1,265,419	1,110,452	-154,967
4	Northeast Division	2012-10-31	Sporting Goods	908,538	702,250	-206,288
5	Northeast Division	2012-11-30	Appliances	847,588	358,900	-488,688
6	Northeast Division	2012-11-30	Electronics	809,619	514,479	-295,140
7	Northeast Division	2012-11-30	Furniture	1,268,226	1,468,981	200,756
8	Northeast Division	2012-11-30	Sporting Goods	1,186,919	662,029	-524,890
9	Northeast Division	2012-12-31	Appliances	1,229,158	1,291,558	62,400
10	Northeast Division	2012-12-31	Electronics	1,078,511	993,238	-85,273
11	Northeast Division	2012-12-31	Furniture	994,495	739,370	-255,124
12	Northeast Division	2012-12-31	Sporting Goods	687,940	130,819	-557,121
13	Midwest Division	2012-10-31	Appliances	846,096	1,107,041	260,945
14	Midwest Division	2012-10-31	Electronics	1,219,504	906,189	-313,315
15	Midwest Division	2012-10-31	Furniture	984,326	715,859	-268,467
16	Midwest Division	2012-10-31	Sporting Goods	828,929	1,045,653	216,725
17	Midwest Division	2012-11-30	Appliances	580,719	547,976	-32,743
18	Midwest Division	2012-11-30	Electronics	430,795	88,665	-342,130
19	Midwest Division	2012-11-30	Furniture	989,386	1,224,691	235,305
20	Midwest Division	2012-11-30	Sporting Goods	919,189	521,194	-397,995
21	Midwest Division	2012-12-31	Appliances	894,058	942,580	48,523
22	Midwest Division	2012-12-31	Electronics	1,156,219	1,070,866	-85,353
23	Midwest Division	2012-12-31	Furniture	821,690	1,034,374	212,684
24	Midwest Division	2012-12-31	Sporting Goods	1,109,766	1,363,120	253,354
25	South Division	2012-10-31	Appliances	697,424	829,639	132,214
26	South Division	2012-10-31	Electronics	979,611	877,973	-101,638
27	South Division	2012-10-31	Furniture	1,399,786	1,134,468	-265,318

Display 2. Sample of the SASTEST.ALL_DATA table

In the metadata, each user is set up in different metadata groups. These groups correspond to the Division field in the data (eg. “Northeast Division”, “Midwest Division”, etc).

The following display shows how different users are members of the “Midwest Division” metadata group.



Display 3. SAS Management Console: Properties view for Midwest Division group

The basic overview of the Stored Process code is:

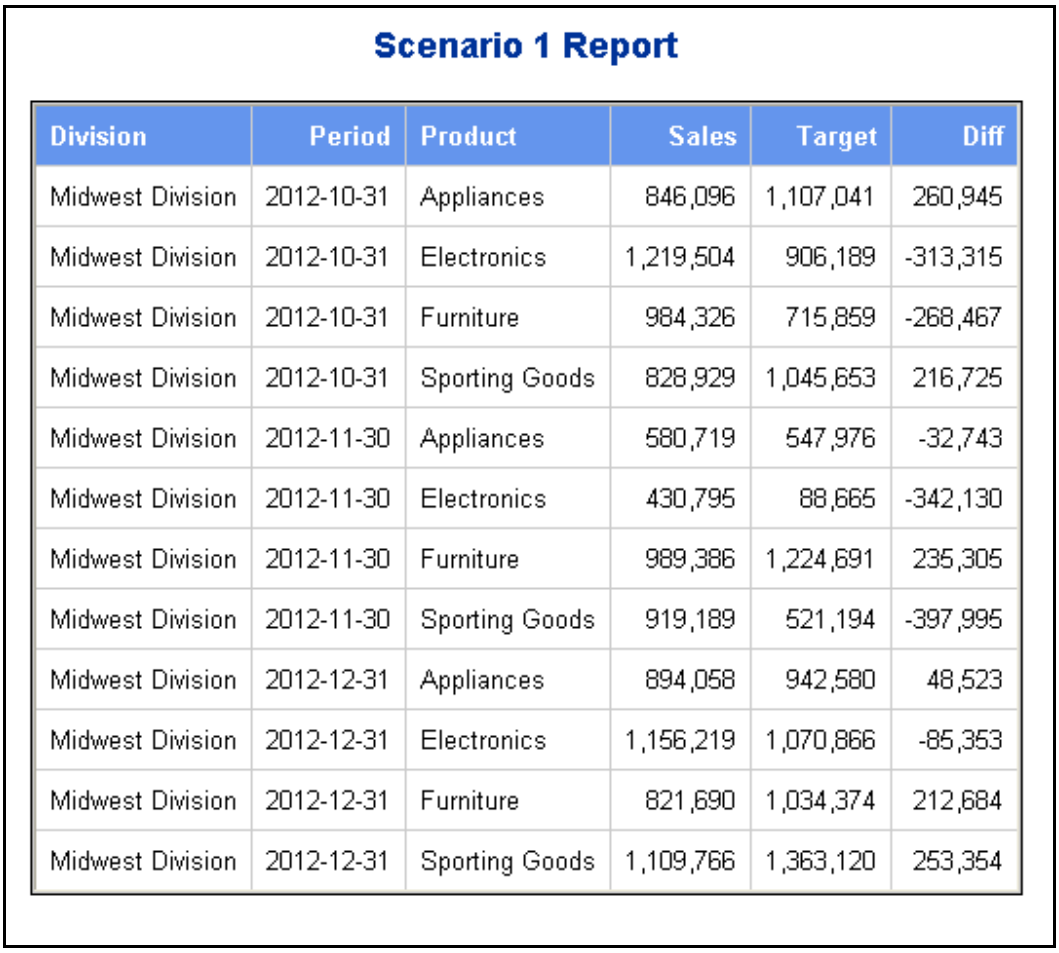
1. Use metadata functions to query the metadata. Use the _METAPERSON macro variable to find the groups for the user who is invoking the stored process.
2. Construct a WHERE clause based on these metadata groups.
3. Apply the WHERE clause when displaying the tabular data.

The Stored Process code follows:

```
*ProcessBody;

%stpbegin;

* this library is directly assigned here ;
* however it could be configured as a predefined metadata library or in an autoexec
file ;
libname SASTEST "/tmp/SASTest";

* use a datastep to build the WHERE clause into a macro variable ;
data _NULL_;
    length
           obj uri name metaperson $256
           division_list where_clause $1000
           ;

    * construct metadata query string ;
    metaperson = symget('_METAPERSON');
```

```
    obj = "omsobj:IdentityGroup?IdentityGroup[@GroupType ne
'ROLE'][MemberIdentities/Person[@Name='" !! strip(metaPerson) !! "']]";

    * loop through for each metadata group that is returned ;
    n = 0;
    nobj = 1;
    do while(nobj gt 0);
           n = n + 1;
           nobj = metadata_getnobj(obj, n, uri);
           if nobj gt 0 then do;
                  * retrieve the name of the metadata group ;
                  rc = metadata_getattr(uri, "Name", name);
                  if rc eq 0 then do;
                         * build the list of divisions
                           into a comma-separated string
                         ;
                         if division_list eq " " then
                                division_list = quote(strip(name));
                         else
                                division_list =
                                       strip(division_list) !! ", " !!
                                       quote(strip(name))
                                       ;
                  end;
           end;
    end;
    * build the WHERE clause and store it in a macro variable ;
    where_clause = "where DIVISION in (" !! strip(division_list) !! ")";
    call symputx("WHERE_CLAUSE", where_clause);
run;

* print the WHERE clause to the log - useful for debugging ;
%put WHERE_CLAUSE=&WHERE_CLAUSE;

* display tabular report data ;
title "Scenario 1 Report";
proc print data=SASTEST.ALL_DATA noobs;
    &WHERE_CLAUSE; * use the WHERE clause to filter the data ;
run;

%stpend;
```

The following display shows the output when a user in the “Midwest Division” metadata group invokes the Stored Process.



Scenario 1 Report

Division	Period	Product	Sales	Target	Diff
Midwest Division	2012-10-31	Appliances	846,096	1,107,041	260,945
Midwest Division	2012-10-31	Electronics	1,219,504	906,189	-313,315
Midwest Division	2012-10-31	Furniture	984,326	715,859	-268,467
Midwest Division	2012-10-31	Sporting Goods	828,929	1,045,653	216,725
Midwest Division	2012-11-30	Appliances	580,719	547,976	-32,743
Midwest Division	2012-11-30	Electronics	430,795	88,665	-342,130
Midwest Division	2012-11-30	Furniture	989,386	1,224,691	235,305
Midwest Division	2012-11-30	Sporting Goods	919,189	521,194	-397,995
Midwest Division	2012-12-31	Appliances	894,058	942,580	48,523
Midwest Division	2012-12-31	Electronics	1,156,219	1,070,866	-85,353
Midwest Division	2012-12-31	Furniture	821,690	1,034,374	212,684
Midwest Division	2012-12-31	Sporting Goods	1,109,766	1,363,120	253,354

Display 4. Report Output from Stored Process for Midwest Division group

The above scenario has been simplified for the sake of example. It could be that the relationship between SAS metadata groups and the values in the data is not a direct one-to-one match like this scenario. In this case, some form of lookup table could be used to map the groups to appropriate values in the data. You could also look at handling the situation where the user is not a member of any valid groups. In this case you could display some form of message rather than simply returning blank empty output. A further complication can arise where nested metadata groups are used, in order to correctly handle this situation you would need to perform additional metadata queries to traverse the metadata group hierarchy and determine the complete set of group memberships for the user.

SCENARIO 2: EXTRA FUNCTIONALITY FOR USERS WITH THE APPROPRIATE ROLE

Consider this scenario: You want to build a report that will display some tabular data similar to Scenario 1. However, you only want to display the report for certain types of users and you also want to display a chart for special types of users. Basically, you are looking at user roles and want to adjust the functionality of the report accordingly.

The following SAS code will demonstrate a technique to accomplish this goal. In this example the same table of sales figures is used as for Scenario 1.

The basic overview of the Stored Process code is:

1. Use metadata functions to query the metadata. Use the _METAPERSON macro variable to find the roles for the user who is invoking the stored process.

2. Use SAS %MACRO logic to control the code based on these metadata roles.
3. Display appropriate functionality in the report based on the %MACRO logic.

The Stored Process code follows:

```
*ProcessBody;

%stpbegin;

* this library is directly assigned here ;
* however it could be configured as a predefined metadata library or in an autoexec
file ;
libname SASTEST "/tmp/SASTest";

* use a datastep to build the list of roles into a macro variable ;
data _NULL_;
    length
           obj uri name metaperson $256
           role_list $1000
           ;

    * construct metadata query string ;
    metaperson = symget('_METAPERSON');
    obj = "omsobj:IdentityGroup?IdentityGroup[@GroupType eq
'ROLE'][MemberIdentities/Person[@Name='" !! strip(metaPerson) !! "']]";

    * loop through for each metadata role that is returned ;
    n = 0;
    nobj = 1;
    do while(nobj gt 0);
           n = n + 1;
           nobj = metadata_getnobj(obj, n, uri);
           if nobj gt 0 then do;
                  * retrieve the name of the metadata role ;
                  rc = metadata_getattr(uri, "Name", name);
                  if rc eq 0 then do;
                         * build the list of user roles into a '&'-separated string
                           (this is a suitable delimiter character because metadata
                           role names cannot contain '&')
                         ;
                         if role_list eq " " then
                                role_list = strip(name);
                         else
                                role_list = strip(role_list) !! '&' !! strip(name);
                  end;
           end;
    end;
    * store into a macro variable ;
    call symputx("ROLE_LIST", role_list);
run;

* print the list to the log - useful for debugging ;
%put ROLE_LIST=%superq(ROLE_LIST);

* use this macro to control the report logic ;
%macro report_logic;
    * define helper function to search the list of roles ;
    %macro roleSearch(role);
           %sysfunc(indexw(%superq(ROLE_LIST), &role, %str(&)))
    %mend roleSearch;

    %if %roleSearch(ReportingRole) ne 0 %then
    %do;
```

```
            * display tabular report data ;
            title "Scenario 2 Report";
            proc print data=SASTEST.ALL_DATA noobs;
            run;

            %if %roleSearch(ReportingAdvancedRole) ne 0 %then
            %do;
                  * display a chart as well - if the user has this role ;
                  proc gchart data=SASTEST.ALL_DATA;
                        vbar product / sumvar=sales;
                        run;
                  quit;
            %end;
      %end;
      %else
      %do;
            * display a message indicating no access ;
            data MESSAGE;
                  Note = "You do not have the ReportingRole.";
            run;
            title "Scenario 2 Report";
            proc print data=MESSAGE noobs;
            run;
      %end;
%mend report_logic;
%report_logic;

%stpend;
```

When a user invokes the Stored Process depending on their role they will see either:

- a table of data
- a table and a bar chart
- or, a message telling them they do not have the appropriate role for this report

The above scenario could be further enhanced to display web form controls and/or links providing access to other reports based on the user having the necessary roles. The code could also be combined with the template from Scenario 1 to provide row-level data security in the same report.

CONCLUSION

Using the above techniques, you can implement security in your Stored Process reports as well as role-based access. The two scenarios covered can be built upon and generalized to implement more advanced reporting requirements. As new users are added or existing users are removed there will be minimal change required to your Stored Processes due to the use of the SAS metadata.

ACKNOWLEDGMENTS

The author would like to thank Paresh Patel and Sebastian Scanzi for their contributions to this paper.

RECOMMENDED READING

- *SAS® 9.3 Stored Processes – Developer's Guide*
- *SAS® 9.3 Language Interfaces to Metadata*

CONTACT INFORMATION

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

Name: Toby Hill
Enterprise: Charles Marcus Group Services Ltd

Address: Cams Hall, Cams Hill
City, State ZIP: Fareham, Hants, PO16 8AB
E-mail: thill@charlesmarcus.com