Let SAS/SHARE® Deliver Formatted Data to Microsoft Office
Hsiwei Yu, Custom Software Systems, Leesburg, VA
Tao Dong, Pragmatics, Reston, VA

ABSTRACT
When working in a company with multiple platforms, it is preferred to use Windows SAS, accessing SAS, DB2, Oracle, and SQL Server data, etc., from different platforms as if they are all local data. This paper showcases a relatively new feature since 9.2 — making remote data and formats automatically available to the Windows SAS session. We will discuss ways to deliver formatted values; e.g., from a low-tech solution building a new table of formatted values, to a more advanced macro solution that dynamically delivers formatted values for any table. We will examine the merits and applicability of each solution. Showing formatted values for Microsoft Office products improves user acceptance of SAS values and their productivity.

INTRODUCTION
A colleague in the Management Accounting Department needs to report on financial accounting information stored on a mainframe in SAS data, and her chosen tool is Excel. We ran mainframe SAS/SHARE and enabled SAS ODBC drivers on her Windows machine for Excel to retrieve SAS data. The colleague was happy and continued her work for awhile, and then I received a call from her:

“What is this zero and one? In the “Employees” table, “Gender” column?”

I told her that zero stands for female and one for male, and the problem was resolved.

THE REASON IS SAS FORMAT
The reason is that in SAS we can associate a format with a column to decode the cryptic values, zero and one, to be meaningful descriptions, female and male. Had the client been using SAS, the formats would have been available, and the client could have seen the descriptions.

Next came another phone call.

“What is this 1, 2, 3, 4, 5, 6? In the “Sales” table, “Region” column? But the corporate standard is Northeast, Central, Midwest, etc. Why don’t I see Northeast, Central, or Midwest, but rather 1, 2, 3? Who knows what these 1, 2, 3 stand for?”

After careful review, it became clear it was because SAS/SHARE did not deliver formatted values, even if the format was readily available for the SAS/SHARE session to use it. The challenge, therefore, is how to deliver meaningful formatted or decoded values from an underlying cryptic SAS column to non-SAS clients (i.e., Excel).

WHY NOT USE SAS ON WINDOWS?
It is easy for Windows SAS to retrieve corporate data from DB2, Oracle, or SAS. Also, a new enhancement in SAS 9.3 makes remote data and format appear as if it were local. The text below is from Windows SAS for signon to a remote host and accessing remote data as if local.

/* Substitute remotehost to be a valid host name, and tcp_port_number to be a number for TCP port. */
/* Note a space, i.e. blank, between remotehost and tcp_port_number in macro variable assignment. */
%let sassrv= remotehost tcp_port_number;
/* Note the sassrv does NOT have ampersand, &, as prefix. */
options remote= sassrv;
/* *** Depending on SAS/CONNECT configuration, the RLINK filename may NOT be needed. */
filename rlink '../.. tcptso.scr';
*** */
signon;
rssubmit;
libname datalib '../.. ../.. ';
endsubmit;
libname local server= remohost slibref= datalib;
options insert= ( fmtsearch= ( local ));

Or, since we are already running SAS/SHARE on a remote host, there is no need to sign on at all, but users can
directly use the SAS/SHARE library.

/* Substitute remoteshare to be a valid host name, and tcp_port_number to be a number for TCP port. */
/* Note a period, i.e. dot, between remoteshare and tcp_port_number in macro variable assignment. */
%let sassrv= remoteshare.tcp_port_number;

/* Note the sassrv is prefixed WITH ampersand, &. */
libname local server= &sassrv slibref= datalib;

options insert= ( fmtsearch= ( local ));

This approach requires Windows SAS; alas, the management accounting colleague prefers to stay with Excel, the
tool of choice. For more information on this solution, see Reference 1.

CHALLENGES TO SAS/SHARE

This problem exists because the SAS data set of interest has columns with formats associated with it. SAS/SHARE, when responding to an external client such Microsoft Office products, merely returns raw values, NOT formatted values.

Figure One. Excel Data Retrieval

Compare Figure One, above, in Excel, with Figure Two, below, from SAS. The employee SAS table uses one
custom format for gender and several SAS-supplied formats, such as dollar, week date, date time, and time.

Note the “Gender” column is 0 and 1; i.e., SAS format is lost in Excel. Excel correctly handles “date” and “date
time” columns, but the “time” column becomes zero in Excel. The salary dollar format is lost, but the value is cor-
How can Excel retrieve data just like SAS? See Figure Three.

SAS FORMAT, A UNIQUE CONCEPT

SAS format is a unique concept, though similar to a lookup table in relational database terms.

```
Proc format;
Value gendr
0= ’Female’
1= ’Male’; run;
Data employee;
```
Oracle and DB2 practitioners would use a lookup table similar to the one below:

```plaintext
Data genderLookup; /* The lookup table. */
  Key= 0; value= 'Female'; output;
  Key= 1; value= 'Male'; output;
run;
Data employee; /* The raw data table. */
  name= 'Hsiwei Yu'; gender= 1; output;
attrib gender Label= 'Gender';
run;
proc sql; /* A user-friendly view is created from the raw data AND lookup table. */
create view employeeDisplay as
  select E.name, L.value as Gender
  from employee E
    , gnderLookup L
  Where E.gender = L.key;
quit;
```

The lookup table approach is good for user-defined formats such as 0 and 1 for female and male. What is needed next is a method for dealing with SAS-supplied formats.

**Note:** The SAS variables label is also an unfamiliar concept to other IT professionals. Oracle does allow adding comments to columns, which serves the same purpose as the SAS variables label.

**MR. LANGSTON’S MACRO**

One of the authors, Hsiwei Yu, was communicating with Rick Langston of SAS Institute about this particular need, and Langston graciously offered a macro as a solution (see Appendix). The idea is for columns having formats associated with them to then return formatted values, or return original values if there are no formats.

```plaintext
/* Sample use of %GetFormattedValues( table ) in SAS/SHARE session context. */
Proc sql;
  Create view employeeDisplay as select %GetFormattedValues( nesug12.employees ); quit;

/* The actual code in Italic that is executed from the %GetFormattedValues( table ) macro. */
proc sql;
create view employeeDisplay as select
  Full_Name ,
  put (Gender,GENDR.) as Gender ,
  put (Salary,DOLLAR15.2) as Salary ,
  put (HireDate,WEEKDATE18.) as HireDate ,
  put (ETL_Record_Add_DtTm,DATETIME22.2) as ETL_Record_Add_DtTm ,
  put (ETL_Update_Tm,TIME12.2) as ETL_Update_Tm
```
from nesug12.employees; quit;

The above illustrates %GetFormattedValues usage in SAS/SHARE for returning formatted values for every column of a SAS table. This macro takes a single positional input parameter, the table to be displayed.

Note the macro returns desired column statements but NOT the SELECT statement required by SQL because only this way it fits in with SAS/SHARE context.

Users interested in macro techniques, and specifically the SYSFUNC, can look this up on SAS Online Help and Reference 2 for sample codes.

ALTERNATIVE SOLUTION, VIEW FOR EACH DATA SET

It is possible to create a view for every SAS data set, using %GetFormattedValues such as:

    Proc sql; Create view libref1.view1 as select %GetFormattedValues( libref1.dataset1 ); quit;
    Proc sql; Create view libref1.view2 as select %GetFormattedValues( libref1.dataset2 ); quit;

Stop and restart the SAS/SHARE server. This way the client only needs to access these views from ODBC, and there is no need to manually enter anything in the Microsoft Query Window tools. The SAS programmer must create all views needed. This is a solution to be performed by central IT personnel. Below is a distributed client solution.

USE MICROSOFT QUERY WINDOW TOOL

The Microsoft Query tool is available in Excel, and in it we can specify SQL for execution by SAS/SHARE. In the SQL window we can enter:

    select %GetFormattedValues( nesug12.employees )

Detailed steps for Windows 7 Microsoft Professional Plus 2010 are:

Open Excel
Click on “Data”
Click on “From Other Sources” dropdown list
Click “From Microsoft Query”
Highlight the SAS data source and click “OK”

Highlight the SAS data set, move it to the right, and click next two more times
In the “Query Wizard – Finish” window, choose “View data or edit query in Microsoft Query,” then click “Finish.”
Now click on SQL so that we can edit what is to be executed by SAS/SHARE.

```
SELECT EMPLOYEES_0.Full_Name, EMPLOYEES_0.Gender,
      EMPLOYEES_0.Salary, EMPLOYEES_0.HireDate,
      EMPLOYEES_0.ETL_Record_Add_DtTm,
      EMPLOYEES_0.ETL_Update_Tm
FROM NESUG12.EMPLOYEES EMPLOYEES_0
```

We can edit and change it to

```
select %GetFormattedValues( nesug12.employees )
```

then click “OK” twice. Ensure the proper data is returned.
From “File,” “Return Data to Microsoft Excel,” and click “OK.” Excel then receives the data (see Figure Three, above).

The author is not able to find the Microsoft Query in Access. Therefore, the workaround is to use Excel, as an intermediate step, for returning formatted values from SAS to Access.

CONCLUSIONS

Being able to return formatted values to Microsoft Office products makes the user community more productive by eliminating confusion. This macro solution can be used centrally to pre-generate required views, or it can be implemented by users on demand. Either way, the solution delivers corporate SAS data clearly to Excel and Access user, enabling them to be more productive.

REFERENCES:

SAS(R) 9.3 Drivers for ODBC: User’s Guide Hyperlink retrieved 11 August 2012
Communications Access Methods for SAS/CONNECT(R) 9.3 and SAS/SHARE(R) 9.3 Hyperlink retrieved 11 August 2012

ACKNOWLEDGMENTS

Thanks to Rick Langston of SAS Institute for taking an interest in this specific need and coming up with a cool solution! Brian Deitch of Federal Deposit Insurance Corporation reviewed and provided valuable comments.

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

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

CONTACT INFORMATION

Comments and questions are valued and encouraged. Contact the authors at:

Hsiwei Yu
Custom Software Systems
Leesburg, VA
hsiwei_yu@yahoo.com

Tao Dong (Tony)
Pragmatics, Inc.
Reston, VA
DongT@Pragmatics.com
APPENDIX

The GetFormattedValues Macro developed by Rick Langston of SAS Institute:

%macro GetFormattedValues(table);

%--------------------------------------------------------------*;
%* Originally developed by Rick Langston of SAS Institute. *
%* About December 2011. *
%* Used in NESUG 2102 paper *
%* Let SAS/SHARE Deliver Formatted Data to Microsoft Office *
%* For questions and support, contact paper author *
%* Hsiwei Yu, email hsiwei_yu@yahoo.com *
%* Sat, 11 Aug 2012 *
%* *
%* This macro will provide the necessary SQL to produce columns *
%* that are the formatted values for all variables, so that all *
%* columns are character. For numeric variables without formats *
%* BEST12. is used, and for character variables without formats *
%* the variable is used as is. *
%*--------------------------------------------------------------*;

%* First try to open the table. Generate just a star if not opened;

%let fid = %sysfunc(open(&table,i));
%if &fid = 0 %then %do;
    %str(*)
    %goto after_loop;
%end;

%* Loop through all variables in the table;
%let nvars = %sysfunc(attrn(&fid,nvars));
%do i=1 %to &nvars;

    %* Get the name of the variable, and n-literalize it if necessary;
    %let varname = %sysfunc(varname(&fid,&i));
    %if %verify(upcase(&varname),ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_)
        ne 0 %then %do;
            %let varname=%str('%')&varname%str('%')n;
            %let varname=%unquote(&varname); %end;

    %* Get the format name, and use BEST12. if numeric with no format;
    %let varfmt = %sysfunc(varfmt(&fid,&i));
    %if %length(&varfmt) = 0 and %sysfunc(vartype(&fid,&i))=N %then %let varfmt=BEST12.;

    %* If a format is determined, generate a PUT function;
    %if %length(&varfmt) ne 0 %then %do;
        put (&varname.,&varfmt.) as &varname.
        %end;

    %* Otherwise just select the variable as is;
    %else %do;
        &varname.
    %end;

%end;
/* Separate each specification with comma (except the last);*/
%if &i ne &nvars %then %do;
  ,
  %end;
%end;

/* Close the table;*/
%let rc = %sysfunc(close(&fid));
%after_loop:

/* Produce the FROM clause;*/
from &table

%mend GetFormattedValues;