

Tying it Together: Using ODS to Insert EXCEL® Hyperlinks

James E. Snider, Regions Financial Corporation, Birmingham, AL

Garland D. (David) Maddox, Regions Financial Corporation, Birmingham, AL

ABSTRACT

This paper explores the use of ODS to insert hyperlinks in Microsoft EXCEL spreadsheet output. Although Microsoft Office® provides some examples of how to insert hyperlinks, these methods do not work when using ODS without some additional manipulation. This paper will demonstrate linking to other spreadsheets as well as to other file types.

INTRODUCTION

A common task in report development is to generate EXCEL spreadsheets that have logical links between the various worksheets. Quite often, it's desirable to insert hyperlinks in selected cells to link a particular row to related data in other worksheets creating a type of drill-down report structure. The specific need in this instance involved the requirement to link various data elements pertaining to bank loan applications. But the methodology is pertinent in any situation involving the development of multiple spreadsheet-based reports that contain logical relationships. This paper outlines the use of the EXCEL function HYPERLINK in conjunction with SAS® to produce linked spreadsheets.

LINKING SPREADSHEETS

The Microsoft EXCEL function HYPERLINK is used here to provide the linking mechanism from one cell to another cell within the same or a different worksheet. The syntax is:

```
=HYPERLINK(link_location, friendly_name)
```

where link_location is the path and file name of the document to link to. Link_location can refer to a specific cell, a named range, a file name, or a URL. In our example, the link_location will be represented as specific workbook and worksheet. The friendly_name is the jump text or numeric value that is displayed in the cell. The following example creates a hyperlink to cell C22 on the worksheet named Monthly in the workbook Budget.xls:

```
=HYPERLINK("[http:// example.xxx.com/budget.xls]Monthly!C22", "Click to See  
the Dept Detail").
```

The cell containing the hyperlink displays "Click to See the Dept". In order to generate EXCEL HYPERLINKS through SAS processing, it is necessary to perform some calculations in order to derive the appropriate link_location.

GENERATING THE EXAMPLE SET OF SPREADSHEETS

To illustrate the process of setting hyperlinks in EXCEL spreadsheets, we must first generate a set of spreadsheets that have some logical attributes that can be linked together. The methodology used in this paper makes use of the dictionary tables to generate three spreadsheets that contain metadata information about the libraries, SAS data files, and variables that are pertinent to a specific WINDOWS® directory. This is a useful utility in its own right, and the code to generate the spreadsheets is included in the Appendix A. Each file is processed first through SAS, and the outputs are written out in spreadsheet format. Here are a few rows of each of the three files:

LIBRARIES

Library Name	Path Name	Dataset Link
CURRENT	\\sesug\blc_whse\current	No SAS files
DATASOR	\\sesug\blc_whse\Dhub\Tech - process\data\data_store_routine	SAS Files
DW	\\sesug\Blc_Whse	No SAS files
E_CCP2	\\sesug\blc_whse\jsnider\CCP2	No SAS files
FMTSRC	\\sesug\blc_whse\Dhub\Tech - process\data\format input	SAS Files

The second spreadsheet includes a row for each file within the libraries. Only a few of the variables are shown here, but many more can be included from the dictionary table output.

DATA SETS (FILES)

Library Name	Member Name	Compression Routine	Column Link
DATASTOR	REPORT_DS_CHGOFF_RPT	BINARY	Column Info
DATASTOR	REPORT_DS_CHGOFF_RPT_BKUP	CHAR	Column Info
DATASTOR	REPORT_DS_LLANAL_TOC_RPT	NO	Column Info
DATASTOR	REPORT_DS_LLANAL_TOC_RPT_BKUP	NO	Column Info
FMTSRC	ALL_SAD	NO	Column Info

VARIABLES (COLUMNS)

Library Name	Member Name	Column Name	Column Type	Column Length	Column Format
DATASTOR	REPORT_DS_CHGOFF_RPT	crit1	char	40	
DATASTOR	REPORT_DS_CHGOFF_RPT	crit2	char	40	
DATASTOR	REPORT_DS_CHGOFF_RPT	item_desc	char	40	
DATASTOR	REPORT_DS_CHGOFF_RPT	report_dt	num	8	MMDDYY10.
DATASTOR	REPORT_DS_CHGOFF_RPT	report_name	char	40	
FMTSRC	ALL_SAD	Officer_cd1	char	1	
FMTSRC	ALL_SAD	Officer_cd2	num	8	

The third spreadsheet includes variable information for each file within the libraries.

It is in the "Dataset Link" column in the Libraries sheet and the "Column Link" in the Files sheet that we want to populate with a hyperlink that when clicked will take us to the appropriate data in the accompanying sheet.

SETTING THE LINKS

The following code illustrates the processing of the libraries file to associate the appropriate files with the appropriate libraries. The variable "ds_link" is added to serve as the link variable and includes the HYPERLINK function. The link_location argument is specifically coded to SAS_data.xls workbook and SAS_files.xls worksheet, but it necessary to derive the target cell portion of the argument by introducing the counter variable CNT . The first.libname variable can be used to identify the first occurrence of each libname in the list.

```
data libraries(keep=libname path ds_link )
  datasets(drop=path ds_link cnt) ;
  length path ds_link $100. ;
  label ds_link='Dataset Link' ;
  *** we will start the counter at 2 to allow for the column headings. ;
  retain cnt 2 ;
  merge libs(in=a) files2(in=b) ;
  by libname ;
  if a ;
  if first.libname then
    if not b then
      do;
        ds_link = 'No SAS files' ;
        output libraries ;
        return;
      end;
    else
      do;
        ds_link =
          "=HYPERLINK('||'|'||'|'|'[SAS_data.xls]SAS_files!A'||trim(left(put(cnt,5.)))||'|', 'SAS
          Files')' ;
        output libraries ;
      end;
    output datasets ;
    cnt+1 ;run;
```

The files (data sets) and variables (columns) data sets are then created in a similar manner.

```

data datasets2(keep=libname memname memlabel crdate modate nob nvar compress col_link
)
    columns3(drop=memlabel crdate modate nob nvar compress col_link cnt) ;
length label col_link $100. ;
label col_link='Column Link' ;
*** we will start the counter at 2 to allow for the column headings. ;
retain cnt 2 ;
merge datasets(in=a) columns2(in=b) ;
by libname memname ;
*drop cnt;
if a ;
if first.memname then
    if not b then
        do;
            col_link = 'No Columns' ;
            output datasets2 ;
            return;
        end;
    else
        do;
            col_link =
"=HYPERLINK("||'|"'||"[SAS_data.xls]SAS_vars!A"||trim(left(put(cnt,5.)))||'|"', "Column
Info")' ;
            output datasets2 ;
        end;
    output columns3 ;
    cnt+1 ;
run;

```

The SAS_DATA.XLS may then be created as a multiple sheet workbook by writing the libraries, files, and variables sheets using the ExcelXP tagset. The final caveat is that when the data sets are written out in EXCEL formats the Define statement includes left justification and FLOW options. For example, the PROC REPORT definition is as follows for the libraries data set.

```

COLUMN libname path ds_link;

DEFINE libname / DISPLAY FORMAT= $8. WIDTH=8 SPACING=2 LEFT "Library Name" ;
DEFINE path / DISPLAY FORMAT= $100. WIDTH=60 SPACING=2 LEFT "Path Name" ;
DEFINE ds_link / DISPLAY FORMAT= $100. WIDTH=60 SPACING=2 FLOW LEFT 'Dataset
Link' ;
run;

```

This methodology can be used to effectively tie together EXCEL spreadsheets generated by a SAS program. In addition, this methodology can be used to tie together other file types from other applications such as Microsoft WORD by referencing a bookmark in the HYPERLINK link_location argument.

CONCLUSION

The practice of linking spreadsheets provides a powerful and robust reporting mechanism especially in large data set settings while providing output in the popular Microsoft formats.

REFERENCES

Microsoft Corporation. 2003. Microsoft EXCEL Online Help.

DelGobbo, V. 2008. "Tips and Tricks for Creating Multi-Sheet Microsoft EXCEL Workbooks the Easy Way with SAS". Proceedings of SAS Global Forum 2008. Paper 192.

Appendix A

```
proc sql;
    create table wl.cols as
    select * from dictionary.columns ;
quit;

proc sql;
    create table wl.files as
    select * from dictionary.members ;
quit;

proc sql;
    create table wl.tables as
    select * from dictionary.tables ;
quit;

data libs(keep=libname path) ;
    set wl.files;
    where upcase(path) like '\\\sesug\cc%' ;
run;

proc sort data=libs nodupkey ;
    by path ;
run;

proc sort data=libs ;
    by libname ;
run;

proc sort data=wl.tables(where=(memtype = 'DATA' and libname ne 'SASHELP' and
    datepart(modate) ge '01apr2008'd))
    out=files(keep=libname memname memlabel crdate modate nobs nvar
compress);
    by libname ;
run;

data files2;
    merge files(in=f) libs(in=l) ;
    by libname ;
    if f and l;
run;

proc sort data=files2 nodupkey ;
    by libname memname ;
run;

proc sort data=wl.cols
    out=columns(keep=libname memname name type length label format) ;
    by libname memname ;
run;

data columns2 ;
    merge columns(in=c) files2(in=f) ;
    by libname memname ;
    if c and f;
run;

proc sort data=libs nodupkey ;
    by libname ;
run;
```

CONTACT INFORMATION

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

James E. Snider
Regions Financial Corporation
Credit Analytics
19th Floor
1901 6th Avenue North
Birmingham, AL 35203
Work Phone: 205-263-5027
Email: James.Snider@Regions.com

Garland D. (David) Maddox
Regions Financial Corporation
Business Services Administration
RCS-3
250 Riverchase Parkway East
Birmingham, AL 35244
Work Phone: 205-560-6339
Email: Garland.MaddoxJr@Regions.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