Paper 214-2012

Put a Little Zip in Your SAS® Program

Louise S. Hadden, Abt Associates Inc., Cambridge, MA
Christianna Williams, Independent Consultant, Chapel Hill, NC

ABSTRACT

SAS® programmers are frequently called upon to perform repetitive processes. To deal with repetitive processing SAS macros are indispensable. If a process calls for using software outside of SAS, it might be hard to figure out how to use SAS macros. This paper presents a solution to passing SAS macro code through to an external software call. A bonus is the ability to generate a listing of the resulting zip file and to output it to an external file within the SAS program so that processing done in batch can be easily checked. This example is run on a Microsoft Windows x64 server using WinZip. However, the process can be used on other platforms and with other software.

INTRODUCTION

This paper really originated many years ago after we had experimented with unzipping flat files within SAS programs using unnamed pipes (on an AIX UNIX system.) The ability to do this allowed us to use large data files which would have been impossible to house on our system unzipped. At the time, the scale of data storage possibilities we have now wasn't even dreamt of. Fast-forward a decade or a few, and our desktop PCs' storage capacity greatly exceeds our mainframe and mini systems storage capacity in the past. Nonetheless, space-saving is still a valuable ability. Saving time is still more valuable, and that is the goal of the technique shown here. The example shown is run in batch mode using WINZIP and SAS 9.2 M2 on a Windows x64 server, but it should be noted that the technique works with other zipping programs such as gzip, PKZIP, and SECUREZIP, as well as on other platforms.

The specific project which inspired the use of WINZIP CLI within SAS programs is a very large and complex one, with new processing of data occurring each month. The processing is repetitive – a suite of programs is run each month with minor modifications. We took advantage of SAS Macro language to simply change macro variables for date values at the top of each program. In order to save space and time, we also wished to build in zipping routines to our programs, and to use those same macro date variables within the zipping routines.

X MARKS THE SPOT

There are a couple of methods of performing operating system commands such as copying files or running external programs within SAS programs: the X command (eXit to operating system), and the SYSTASK command. We chose to use the X command because the SYSTASK command requires the use of the WAIT and WAITFOR options and it is slightly more complex. However, the SYSTASK command is better for multiple external commands. In our case, we had one or two related commands so greater simplicity won out. Below is a simple sample of usage of the X command. This will copy all files with the XLS extension from D:\DATA to C:\DATA.

```
X "copy d:\data\*.xls c:\data\*.xls";
```

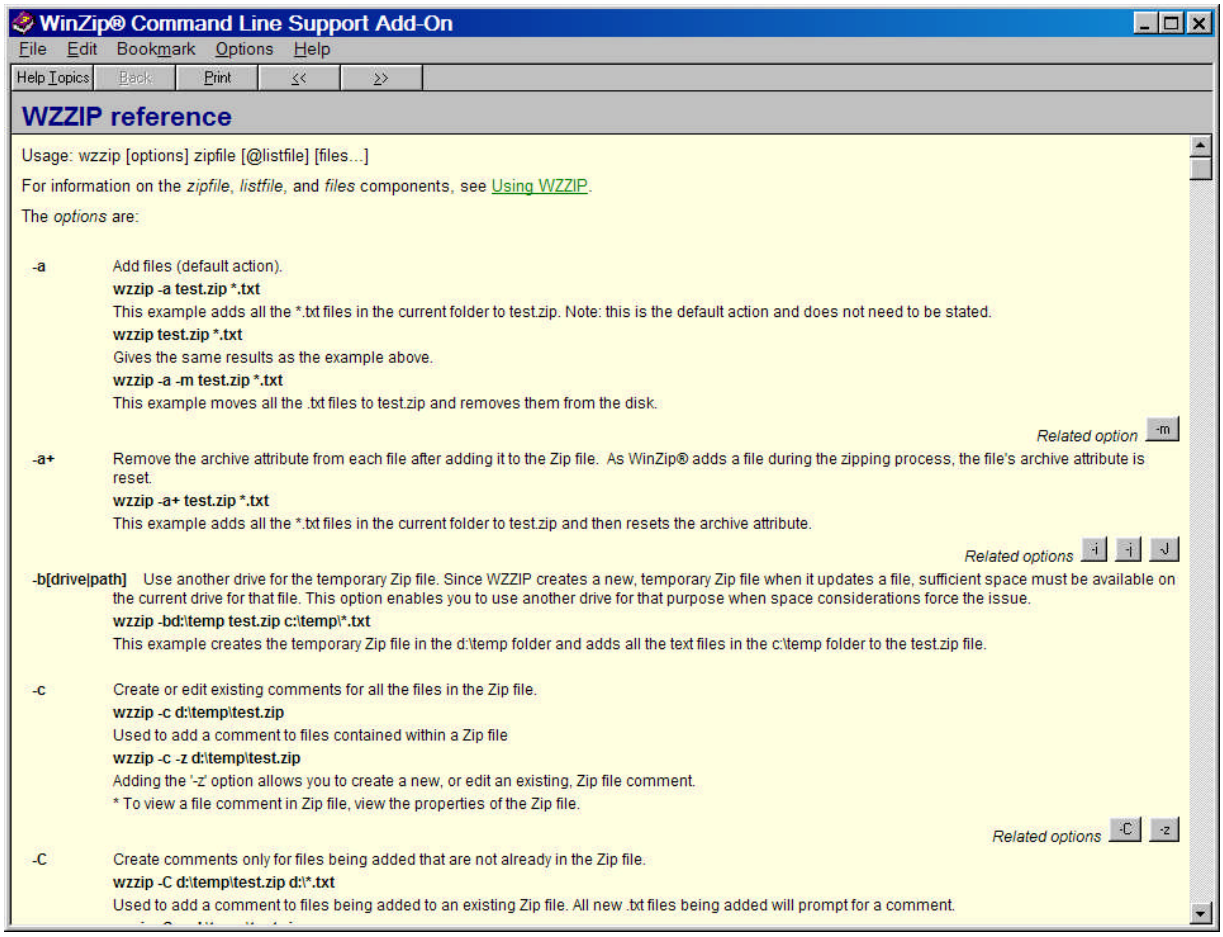
There are some drawbacks to the use of external commands within a SAS program. There is no SAS log report on external activities within the program. It is necessary to build in safeguards to make sure that your program steps execute in the way that you expect. If you have interspersed external commands throughout the program, and further steps in the program are dependent on the successful execution of a prior external command, you will have program errors that may be hard to identify and resolve. Luckily, through a combination of SAS return codes, operating system commands and external software commands these safeguards are relatively easy to implement.

THE TASK AND SOLUTION

For our complex project, we wished to build programs to create zipped deliverables and archive a month's worth of processing. We also wanted to include zipping routines in some other programs. Our SAS server environment is Windows x64 based. We have WINZIP, the WINZIP CLI (which is sometimes included in a WINZIP deployment, but if not, is a free download given a licensed, recently released WINZIP product), and SECUREZIP available to us on the server. We run SAS exclusively in batch while in production mode for this project. Programmers on the project had greater familiarity with WINZIP and its CLI, so WINZIP was chosen for our processing although SECUREZIP

performed in similar ways during testing. PKZIP and GZIP work as well, in testing done on other systems and operating platforms. Although ODS PACKAGES is now available in higher releases of SAS and would accomplish similar goals while operating WITHIN SAS (and give us a log!), we needed our zip archives to be accessible to our clients, not all of whom have SAS.

The first step in our process was to consult the documentation for the WINZIP CLI. This gave us access to a wealth of information about WINZIP CLI commands and some great ideas for managing the external processing. A screenshot of the documentation for WZZIP (as opposed to WZUNZIP) is presented below. Users interested in using the WINZIP CLI are encouraged to review the documentation in full.



When using the WINZIP CLI within SAS, you need to specify the full path for the WZZIP or WZUNZIP executable in your command. For example:

```
X "c:\progra~1\winzip\wzzip c:\data\zipit.zip c:\data\*.xls";
```

The ~1 is an artifact of command line processing, harkening back to the days of when directory names had to be 8 characters or less. Because the full path name “program files” is more than 8 characters, the command prompt cuts the name off at 6 characters and inserts the ~ and a number. On most systems, program executables are in progra~1. When we first moved our processing for this project from the PC environment to the x64 server environment, our zipping code did not work. Some exploration revealed that 64 bit programs on the server were in “program files” (progra~1) but 32 bit programs were in “program files (x86)” (progra~2). For our system, then:

```
X "c:\progra~2\winzip\wzzip c:\data\zipit.zip c:\data\*.xls";
```

Just being able to zip files up within our SAS programs was a significant enhancement to our processing and a real time saver given that there were multiple zip steps in our monthly processing. However, we still had to edit the filenames (which changed each month) by hand. The zip steps were always at the end of the programs, and it was

easy to make a mistake and fail to edit a file name – and there was no error in the log to alert us to a problem. Ryan Whitworth of RTI presented a Coders' Corner at SGF 2010 (see the References section at the end and read his valuable paper!) and gave us a solution.

```
x " ""c:\progra~2\winzip\wzzip"" ""archive_5star_&fileyear.&filedate..zip""
"".\files&fileyear.&filedate\*.*"" ";
```

The double quotes surrounding all command strings allows the use of macro variables. In our case, all our programs have macro variables for file dates at the top of each program that are edited each month. This allowed us to edit the macro variables once, and not have to worry about the filenames in the zip routine at the bottom of the program.

The above x command statement worked great, except that in our monthly archive program we did NOT want to include some very large files that were to be backed up on DVD separately. A review of the WINZIP CLI documentation revealed that it is possible to exclude files with the –x option.

```
wzzip -x*.txt d:\temp\test.zip *.*
```

The above command excludes all TXT files from the zip file.

```
wzzip -x*\ d:\temp\test.zip *.*
```

The above command excludes all empty folders from the zip file. The form of the –x option we chose to use was:

```
x " ""c:\progra~2\winzip\wzzip""  -x@""exclude_&fileyear.&filedate..txt"" -p -r
      ""archive_5star_&fileyear.&filedate..zip"" "".\files&fileyear.&filedate\*.*"" ";
```

We built the exclude_yyyy_mmdd.txt file earlier in the program using some simple data set programming. This preserves a record of what files were excluded from the archive. The @ sign tells WINZIP to use a text file as input to the –x command. The –p and –r options specify that the archive should go into subfolders and retain subfolder name information in the filenames in the archive.

```
filename xx ".\exclude_&fileyear.&filedate..txt";
filename yy ".\files&fileyear.&filedate.\ExcludeFromArchive_&fileyear.&filedate..txt";

run;

/* first, create a text file with the exclusion information with the appropriate dates
*/

data exclude;
    length linevar $ 80;
    linevar=".\files&fileyear.&filedate.\reporting\facilitypdfs\*.zip"; output;
    linevar=".\files&fileyear.&filedate.\reporting\facilitypdfs\*.exe"; output;
    linevar=".\files&fileyear.&filedate.\reporting\facilitypdfs\*.pdf"; output;
    linevar=".\files&fileyear.&filedate.\reporting\fac_rating_report*.log"; output;
run;

data _null_;
    file xx lrecl=80;
    set exclude;
    put linevar;
run;

data _null_;
    file yy lrecl=80;
    set exclude;
    put linevar;
run;
```

Our project has 100% peer review of all programs, logs and output prior to delivering files to the client. Our review process indicated that our zip solution still wasn't perfect. We needed a solution that allowed us to still maintain our 100% review, but make it a little easier to identify problems in our zip files by providing a listing of what went into the zips which can be reviewed. We used the -@ to redirect a listing of the contents of the zip archive to a text file which could then be reviewed. This command follows the creation of the zip archive.

```
x " ""c:\progra~2\winzip\wzunzip"" -@"".\Archive_&fileyear.&filedate._listing.txt""
```

```
""archive_5star_&fileyear.&filedate..zip"" ";
```

The text file that was output is reviewed along with the program logs and zip file itself. This allows us to do an extra quality check that is easy to accomplish.

CONCLUSION

With some research and a little creativity, you can easily incorporate external processes within your SAS programs in conjunction with macro coding to save a lot of time and reduce errors in your processing! The example shown is for the WINZIP CLI, but the possibilities are endless.

REFERENCES AND RECOMMENDED READING

Gebhart, Eric. "ODS Packages: Putting Some Zip in Your ODS Output. Proceedings of SAS Global Forum 2009 Conference. March 2009.

Hunt, Stephen and Fairfield-Carter, Brian. "Zipping Right Along: Push-button SAS® Transfers via Command-line Invocation of the WinZip32 Executable." Proceedings of SAS Global Forum 2007 Conference. April 2007.

Walsh, Irina. "Pros and Cons of X command vs. SYSTASK command." Proceedings of Western Users of SAS® Software 2009. September 2009.

Whitworth, Ryan. "Zip and Email Files Using SAS® To Reduce Errors and Make Documentation Easy." Proceedings of SAS Global Forum 2010 Conference. April 2010.

CONTACT INFORMATION

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

Louise Hadden louise_hadden@abtassoc.com

Christianna Williams christianna_williams@abtassoc.com (do you want to use a different email?)

Code samples are available upon request. Appendix 1 below contains a full program example.

Appendix 1: SAMPLE PROGRAM

```
/* set the dates for the current round of processing */
/* needs to be set for each round of processing */

%let filedate=0401;
%let fileyear=2011;

%let yy=%substr(&fileyear,3,2) ;
run;

 /*******************************************
 Program:   archive.sas
 Purpose:   (1) Archive current round of processing

 Path:      TOP LEVEL Data from CMS
 Ancestor:  None
 Notes:    MUST run from top level or zip file will be corrupt trying to write to
           Itself.  Actual file date information must be edited into the zip file
           name as code in X-window does not parse macro variables
           SERVER version
           Removes provider preview zip and log from zip archive

 Input Data:  Folders containing all programs and data for the current round of
              processing

 Output Data:  (1) Zip File containing all programs and data for the current round of
               processing

 Created:  01/28/2009 / lsh
 Edited: 03/10/2010 / lsh - added a listing of what is in zip archive to program
         04/20/2010 / lsh - revised so that provider preview log is excluded - was
                            excluding an older version of program
                            with file dates in it so it didn't actually exclude the log.

runs:  01/26/2010 - files20101004
       02/12/2010 - files20100201
       03/10/2010 - files20100302
       04/19/2010 - files20100401
       . . .


 *******************************************/
run;

title1    "NH Compare - Archive Current Round of Processing" ;
footnote1 "%sysfunc(getoption(sysin)) - &sysdate - &systime";
run;

libname dd '.';
filename xx ".\exclude_&fileyear.&filedate..txt";
filename yy ".\files&fileyear.&filedate.\ExcludeFromArchive_&fileyear.&filedate..txt";

run;

/* first, create a text file with the exclusion information with the appropriate dates
*/

data exclude;
    length linevar $ 80;
    linevar=".\files&fileyear.&filedate.\reporting\facilitypdfs\*.zip"; output;
    linevar=".\files&fileyear.&filedate.\reporting\facilitypdfs\*.exe"; output;
    linevar=".\files&fileyear.&filedate.\reporting\facilitypdfs\*.pdf"; output;
    linevar=".\files&fileyear.&filedate.\reporting\fac_rating_report*.log"; output;
run;
```

```
data _null_;
    file xx lrecl=80;
    set exclude;
    put linevar;
run;

data _null_;
    file yy lrecl=80;
    set exclude;
    put linevar;
run;


%macro zipit;

/* note that the progra~2 is vital.... */

*** ZIP THEM UP *** *** CHANGE FILE DATES ***;

  x " ""c:\progra~2\winzip\wzzip""  -x@""exclude_&fileyear.&filedate..txt"" -p -r
      ""archive_5star_&fileyear.&filedate..zip"" "".\files&fileyear.&filedate\*.*"" ";

run;

  x " ""c:\progra~2\winzip\wzunzip"" -@"".\Archive_&fileyear.&filedate._listing.txt""
      ""archive_5star_&fileyear.&filedate..zip"" ";

%mend;

%zipit;

endsas;
```