Paper 057-2012

A SAS® Macro to Zip and Unzip Files in MS Windows without Additional External Data Archiving Software

Kai Koo, Abbott Laboratories, Santa Clara, CA

ABSTRACT

A SAS macro is developed to compress/decompress files or folders in zip format without the helps from any third-party zip/unzip applications. It starts the default Compressed (zipped) Folder function exists in every PC running under Windows XP or later version through a short VBScript code activated by SAS %sysexec macro statement. This approach eliminates the dependence of installation of additional auxiliary files or external zip/unzip software, and it enhances the portability of SAS programs at different machines or working environments.

INTRODUCTION

It is a quite common practice to compress SAS datasets or outputs into a compact file archive after data extraction or processing, and the ZIP file is a very popular format for this purpose. Currently, Microsoft® Windows does not have built-in command-line interface to zip and unzip files, unless users install additional Windows resource kit tools offered by Microsoft.

In the SAS for Windows environment, programmers often use X command to call an external data compression application, such as WinZip, PKZip, or 7-Zip etc., in command-line mode for data archiving. However, those data compression applications use different command-line syntaxes and options (*e.g.*, application name and parameters) to perform the similar tasks. A SAS program that works smoothly with WinZip could fail to zip/unzip files at another machine with only 7-Zip installed due to the differences in command-line statements. The inconsistency of command-line syntax among those data archiving applications compromises the portability of SAS programs.

To improve the portability of SAS codes in file archiving under Windows, a standalone SAS statement or macro to zip/unzip files would be an attractive solution. Based on this requirement and knowledge in current Windows, a SAS macro is developed to meet this goal. The basic concepts in coding, testing and final implementation of this macro are presented in this article.

CONCEPTS AND IMPLEMENTATION

Windows XP and later versions (Vista and 7) contain basic built-in zip/unzip capability [*i.e.*, Compressed (zipped) Folder], allowing users to compress/decompress files or folders through a graphical user interface. Although the password encryption feature has been removed from the Compressed (zipped) Folder in Windows Vista and 7, it is still a convenient and free resource of data archiving for general purpose usages. For SAS datasets containing sensitive information and needed further protection, a simple SAS DATA step option, "`encrypt=yes`", could be an alternative solution to increase security measure in the latest Windows environment.

To further use Compressed (zipped) Folder for automatic data archiving, this feature has to be started by script. Although, Microsoft does not provide command-line mode for users to call this Compressed (zipped) Folder function programmatically, this feature can actually be activated by short VBScript codes. Based on this finding, a SAS macro is developed and tested to control Compressed (zipped) Folder through SAS X command or %sysexec macro statement. The basic concepts and real implementation are described in the following Concepts and Implementation sections, respectively.

CONCEPT 1 – ACTIVATE COMPRESSED (ZIPPED) FOLDER FEATURE PROGRAMMATICALLY THROUGH VBSRIPT

VBScript is a native and flexible scripting language developed by Microsoft for use in wide variety Windows environments. To activate the default zip function of Compressed (zipped) Folder by VBScript is quite straightforward, as demonstrated by following seven-line VBScript.

```
1.  Set ZipArgs = WScript.Arguments
2.  InputFile = ZipArgs(0)
```

```
3.  TgtFile = ZipArgs(1)
4.  CreateObject("Scripting.FileSystemObject").CreateTextFile(TgtFile, True).Write
    "PK" & Chr(5) & Chr(6) & String(18, Chr(0))
5.  Set objShell = CreateObject("Shell.Application")
6.  Set source = objShell.NameSpace(InputFile).Items
7.  objShell.NameSpace(TgtFile).CopyHere(source)
```

The first line assigns a memory space to store all command-line arguments. The second and third lines pass the location of source files and the full name of zip archive stored at the memory space, declared in the line one, to two variables (`InputFile and TgtFile`), respectively. The next line generates a preliminary file header of the designated zip file based on the information provided by line three. After this script prepared a short (22 bytes) header of the zip file with appropriate file name, the zip function existed in the Shell.Application is activated in line five. Line six just requests the Shell.Application to identify the number of files and their names existed in the source folder. The last line asks the system to compress all of the source files into a zip archive set up in line four.

For file decompression, users can also unzip file archive by using another six-line VBScript as shown below. These codes are very similar to the codes used to zip files.

```
1.  Set objArgs = WScript.Arguments
2.  InputFile = objArgs(0)
3.  TgtFolder = objArgs(1)
4.  Set objShell = CreateObject("Shell.Application")
5.  Set source = objShell.NameSpace(InputFile).Items
6.  objShell.NameSpace(TgtFile).CopyHere(source)
```

The fifth line extracts the number and names of files compressed inside a specific zipped archive and passes those file attributes of the zip file to a variable '`source`'. The last line of this script decompresses all the files described in the variable '`source`' and saves them into a target folder assigned in line three.

CONCEPT 2 – EXECUTE A VBSRIPT FILE FROM SAS

It is very easy to run a VBScript (.vbs) file under Windows. The syntax to execute a .vbs file is similar to start any command-line programs (.exe / .com) or batch (.bat) files. Like a batch (.bat) file, the VBScript (.vbs) file is also a text file, and user can simply type: [CScript or WScript] (optional) [*your_script_file*.vbs] [parameter 1] [parameter 2]…, to run a .vbs file with additional parameters. Because SAS users are able to execute external programs, scripts and commands by using either X command or %sysexec macro statement, these SAS features made it is possible to run any VBScript codes from a SAS program.

IMPLEMENTATION – CONVERT CONCEPTS TO A REAL SAS MACRO

Based on the understanding of how to control Compressed (zipped) Folder in Windows programmatically by using VBScript described above, a SAS macro for data archiving independent from other external zip/unzip applications can be developed. The following code (SASZip_Lite.sas) is an example of real-world implementation.

For program portability and maintenance simplicity, the macro shown below limits the use of VBScripts only in the core data archiving section. Furthermore, these two VBScript code segments are embedded inside a small SAS DATA step because of their similarity in coding.

Overall, this SASZip_Lite.sas macro can be divided into three main sections. First, the user declares local macro variables and passes the properties (names and directories) of source and target files to the VBScript via those variables. Then, this macro validates the properties of files and generates a VBScript file, XPZIP.vbs, at a temporary working folder. Later, this macro uses %sysexec statement to activate Windows' Compressed (zipped) Folder feature for file archiving through the VBScript generated previously, and then cleans up temporary folder and files at the completion of all the tasks.

```
%macro SASZip_Lite(zip=, sfdr=, fstyl=%str(*.sas*dat), tfdr=);
  /*****************************************************************
   The code posted below is provided "AS IS" with NO WARRANTIES.
   ZIP:   directory and file name of zip archive
   SFDR:  directory of source files (to be zipped)
   FSTYL: File type of source files; value: *.* as "zip a folder"
   TFDR:  Target directory for unzipped files (for unzip)
  *****************************************************************/
```

```
%local zip sfdr fstyl tfdr vbadir p q mode;

/*  Set up a temporary working folder for VBScript */
    %let vbsdir=c:\MyZi$Dir;

options noxwait;

/* To initiate a clean working space */
%if %sysfunc(fileexist("&vbsdir"))=1 %then %sysexec rd /s/q "&vbsdir";
%if %index(%upcase(&zip), .ZIP)=0 %then %let zip=&zip..zip;
%let mode=;

/* Compress (zip) files */
%if %length(&sfdr)>0 and (%length(&zip)>0) %then %do;
   /* Extract directory name of the zip file, if no such folder, generate one */
   %let q=%sysfunc(tranwrd(&zip, %scan(&zip, -1, %str(\)), %str( )));
   %let q=%substr(&q, 1, %length(&q)-1);
   %if %sysfunc(fileexist("&q"))=0 %then %sysexec md "&q";

   /* Copy all requested files from a validated source folder to a temporary folder,
         and keep their original time stamps */
   %if %length(&sfdr)>0 and %sysfunc(fileexist("&sfdr"))=1 %then %do;
      %let mode=z;
      %sysexec md "&vbsdir";
      %if %qupcase(&fstyl)^=%str(*.*) %then %do;
         %sysexec md "&vbsdir.\temp_zip";
         %sysexec copy "&sfdr.\&fstyl" "&vbsdir.\temp_zip";
      %end;
   %end;
%end;
%else %if %length(&tfdr)>0 and %length(&zip)>0 and %sysfunc(fileexist("&zip"))>0
%then %do; /* Unzip files */
  %let mode=u;
  %sysexec md "&vbsdir";
%end;

%if &mode=z or &mode=u %then %do;
/* Generate VBScript based on different modes */
   data _null_;
     FILE "&vbsdir.\xpzip.vbs";

     put 'Set ZipArgs = WScript.Arguments';
     put 'InputFile = ZipArgs(0)';
     put 'TgtFile = ZipArgs(1)';
     put 'Set objShell = CreateObject("Shell.Application")';
     put 'Set source = objShell.NameSpace(InputFile).Items';
     put 'soucnt = objShell.NameSpace(InputFile).Items.Count';

     %if &mode=z %then %do;
        put 'CreateObject("Scripting.FileSystemObject").CreateTextFile(TgtFile,
             True).Write "PK" & Chr(5) & Chr(6) & String(18, Chr(0))';
        put 'objShell.NameSpace(TgtFile).CopyHere(source)';
        put 'Do Until objShell.NameSpace(TgtFile).Items.Count = soucnt';
        put 'wScript.Sleep 3000';
        put 'Loop';
     %end;
     %else put 'objShell.NameSpace(TgtFile).CopyHere(source)'; ;
     put 'wScript.Sleep 3000';
   run;

    /* Run VBScript file for data archiving */
   %if &mode=z %then %do;
```

```
        %if %qupcase(&fstyl)=%str(*.*) %then %sysexec CScript "&vbsdir.\xpzip.vbs"
            "&sfdr" "&zip";
          %else %sysexec CScript "&vbsdir.\xpzip.vbs" "&vbsdir.\temp_zip" "&zip";
    %end;
    %else %sysexec CScript "&vbsdir.\xpzip.vbs" "&zip" "&tfdr";
  %end;

  /* Clean up */
  %if %sysfunc(fileexist("&vbsdir"))=1 %then %sysexec rd /s/q "&vbsdir";
%mend SASZip_Lite;
```

HOW TO USE

The SAS macro shown above is a fully functional version. It can compress single or multiple files located at a specific folder into a zip archive or decompress a zip file to a target folder.

To unzip a zip archive, users only need to provide the full name of zipped file and the directory of the target folder to two macro variables (`zip and tfdr`), respectively.

```
/* Usage demo 1: Unzip archived files in Raw_2011.zip to folder extrt1 */
%SASZip_Lite(zip=c:\work\study1\raw_ds\Raw_2011.zip, tfdr=C:\work\raw\extrt1);
```

For data compression, this macro has three basic options to generate zip files. By default, it zips all SAS datasets in the specified folder into a single file. It can also zip either a specific file type or whole contents (*i.e.,* files and subfolders) of a folder into a zip archive by specifying the file extension to the macro variable ‘`fstyl`’. Examples for these three options are displayed below.

```
/* Usage demo 2a: Zip all SAS datasets in a folder (Raw_ds) to a single zip file */
%SASZip_Lite(zip=C:\work\raw\test 1.zip, sfdr=C:\work\my_extract\Raw_ds);

/* Usage demo 2b: Zip all *.DOC files in a folder to a single zip file */
%SASZip_Lite(zip=c:\Doc temp\my_output.zip, sfdr=C:\work\5Y\Sasout\doc,
   fstyl=%str(*.doc));


/* Usage demo 2c: Zip all files & subfolders in a folder to a single zip file */
%SASZip_Lite(zip=c:\Doc temp\my_folder.zip, sfdr=C:\work\5Y\adhoc, fstyl=%str(*.*));
```

DISSCUSSION AND CONCLUSION

Basically, there are at least two ways to run VBScript from SAS, by either calling an existing VBScript (.vbs) file or executing a temporary .vbs text file generated from a SAS DATA step. To avoid any extra file copy or installation procedure to user’s PC, a temporary .vbs file generated by SAS DATA step, which is deleted after the completion of all the required tasks is a quite reasonable approach. This one program, no permanent auxiliary file setting not only keeps a clean working environment to clients’ disk drive, but also makes further maintenance and improvement tasks simple for coders.

In zipping small file, the SAS macro with VBScript described in “Concept 1" works well; however, it fails to compress a folder containing many large files. This early version program starts normally, but only generates an incomplete zip file with a temporary working file when the size or the number of source files increased. It seems the operating system terminates the last line of script “`COPYHERE(SOURCE)`” before the end of ongoing disk I/O for compression of some large files. To overcome this unsynchronized issue, a “DO-UNTIL” loop and “wScript.Sleep” command are added to the last section of VBScript. This improvement relies on a variable ‘`soucnt`’ as file counter. In a “DO-UNTIL” loop, this counter is used to guarantee the numbers of source and processed files matched each other. Furthermore, at the end of the script, an “unconditional” extra time (current setting: 3 seconds) is imposed as additional buffer for wrapping all unfinished file I/O to prevent this VBScript from terminating prematurely.

The revised SAS macro with “DO-UNTIL” loop has been tested successfully on different machine settings (from old Pentium III 1.2 GHz PC with Windows XP to dual-core CPU with Windows 7) under SAS 9.1.3 and 9.2. It is capable of compressing and decompressing files in Windows, and has been tested successfully in handling a folder with a total file size over one gigabyte containing 50 to100 SAS datasets without any errors. Although, this is only a “lightweight” version which equipped with a limited file/folder validation and error handling abilities, further enhancements, such as more user-friendly error-handling features or customized functions, can be added easily based on the current coding structure.

This report does not intent to answer all the needs of data archiving by using SAS for Windows; however, the concepts and implementation demonstrated here really benefit both end-users and SAS application developers. Because this macro takes advantage of the fact that script language and file archiving functions already existed in every PC with Windows XP or higher, there is no extra licensing cost to end-users for the file archiving software. Furthermore, this re-usable macro offers a simple solution in program portability by skipping any assistance from other third-party data archiving applications. A SAS code with these data archiving functions (*i.e.*, zip/unzip) can be used by different clients or machine settings naturally without any adjustments for the command-line statements discrepancies of different data archiving applications. It simplifies the programming logic and frees the SAS code developers from spending time and energy to such minor but tedious portability issues in coding.

REFERENCES

Sanbonmatsu, Lisa (2000) Batch Processing with SAS®: Beyond running programs overnight. NESUG 2000.

Llano, Jaime A. (2006) Reading compressed text files using SAS® Software. SUGI 31, Paper 155-31.

Hunt, Stephen, Sherman, Tracy and Brian, Fairfield-Carter (2005) An introduction to SAS® applications of the Windows scripting host. SUGI 30, Paper 226-30.

Running scripts from the command Line - http://technet.microsoft.com/en-us/library/ee156587.aspx

Folder.CopyHere method - http://msdn.microsoft.com/en-us/library/windows/desktop/bb787866(v=vs.85).aspx

Can Windows' built-in ZIP compression be scripted? - http://stackoverflow.com/questions/30211/can-windows-built-in-zip-compression-be-scripted

CONTACT INFORMATION

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

Kai Koo
Abbott Vascular
3200 Lakeside Drive
Santa Clara, CA 95054-2807
E-mail: kai.koo@av.abbott.com