Paper SAS264-2014

Reading and Writing ZIP Files with SAS®

Rick Langston, SAS Institute Inc., Cary, NC

ABSTRACT

The ZIP access method is new with SAS 9.4. This paper provides several examples of reading from and writing to ZIP files using this access method, including the use of the DATA step directory management macros and the new MEMVAR= option.

INTRODUCTION

You can use the new ZIP access method to read and write standard ZIP files. You use the FILENAME statement as the typical way to use the access method, similar to how you use other access methods, such as CATALOG and FTP. You have various ways in SAS syntax to access the ZIP files, depending on what is most convenient. And you can use the ZIP files created by the access method by another other facility that handles ZIP files.

DEMO ZIP FILE

For the sake of demonstration, here are the UNIX commands to create two small files, mytest1.txt and mytest2.txt, and a ZIP file to contain them:

```
cat > mytest1.txt <</
First line of mytest1.txt
Second line of mytest1.txt
Last line of mytest1.txt
/
cat > mytest2.txt <</
First line of mytest2.txt
Second line of mytest2.txt
Last line of mytest2.txt
/
rm -R mydemo.zip
zip mydemo.zip mytest1.txt mytest2.txt
```

The mydemo.zip file can be used on any UNIX or Windows system, and it can be used by the ZIP access method. Note that the ZIP file is a binary file, and if you transfer it to another system via FTP, you must ensure that you transfer it in binary mode.

USING THE FILENAME STATEMENT

You can use the FILENAME statement to associate a ZIP file with the ZIP access method. Here is an example, using the MEMBER= option:

```
filename fromzip zip 'mydemo.zip' member='mytest1.txt';
data _null_; infile fromzip; input; put _infile_; run;
filename fromzip zip 'mydemo.zip' member='mytest2.txt';
data _null_; infile fromzip; input; put _infile_; run;
```

The following appears in the SAS log:

```
3          filename fromzip zip 'mydemo.zip' member='mytest1.txt';
4          data _null_; infile fromzip; input; put _infile_; run;

NOTE: The infile FROMZIP is:
      Filename=mydemo.zip,
      Member Name=mytest1.txt
```

```
First line of mytest1.txt
Second line of mytest1.txt
Last line of mytest1.txt
NOTE: 3 records were read from the infile FROMZIP.
      The minimum record length was 24.
      The maximum record length was 26.


5          filename fromzip zip 'mydemo.zip' member='mytest2.txt';
6          data _null_; infile fromzip; input; put _infile_; run;

NOTE: The infile FROMZIP is:
      Filename=mydemo.zip,
      Member Name=mytest2.txt

First line of mytest2.txt
Second line of mytest2.txt
Last line of mytest2.txt
```

When you use the MEMBER= option, you are treating the FROMZIP file as a single sequential file.

You can refer to the members of the ZIP file by using parentheses in the INFILE statement in a DATA step. For example:

```
filename fromzip zip 'mydemo.zip';
data _null_; infile fromzip(mytest1.txt); input; put _infile_; run;
data _null_; infile fromzip(mytest2.txt); input; put _infile_; run;
```

In this case, you are treating the FROMZIP file as a directory.

You can also use the new MEMVAR= option, which is similar to the FILEVAR= option. The MEMVAR= option enables you to reference individual members of a directory-based file. For example:

```
filename fromzip zip 'mydemo.zip';
data _null_;
     length memname $200;
     input memname $;
     infile fromzip memvar=memname end=eof;
     eof=0;
     do while(not eof);
        input;
        put memname= _infile_;
        end;
     datalines;
mytest2.txt
mytest1.txt
mytest2.txt
;
```

The SAS log includes the following:

```
memname=mytest2.txt First line of mytest2.txt
memname=mytest2.txt Second line of mytest2.txt
memname=mytest2.txt Last line of mytest2.txt
memname=mytest1.txt First line of mytest1.txt
memname=mytest1.txt Second line of mytest1.txt
memname=mytest1.txt Last line of mytest1.txt
```

```
memname=mytest2.txt First line of mytest2.txt
memname=mytest2.txt Second line of mytest2.txt
memname=mytest2.txt Last line of mytest2.txt
```

In this example, you read the member name mytest2.txt from datalines. The memname variable is set to that value, and is associated with the MEMVAR option in the INFILE statement. When the INPUT statement is executed for the first time with that INFILE statement in effect, the mytest2.txt member is opened. All records are read from it until end-of-file. The process is repeated with mytest1.txt, and then again for mytest2.txt.

If you have Write permissions for the ZIP file, you can add new members to it, or replace existing members. For example, to add/replace the member called newmem.txt:

```
data _null_; file fromzip(newmem.txt);
     put 'first line of newmem.txt';
     put 'second line of newmem.txt';
     put 'last line of newmem.txt';
     run;
```

USING THE DIRECTORY-BASED FUNCTIONS

You can use the DOPEN, DNUM, DREAD, and DCLOSE functions with the ZIP access method. You provide the DOPEN function with a fileref associated with a ZIP file as a directory. The DNUM function tells you how many members are in the ZIP file. The DREAD function reads through the directory to give you the names of all of the members of the ZIP file. You can use MOPEN, FREAD, or FGET to read the contents of individual members. And when you have completed reading the directory, you use the DCLOSE function to close out the process. For example:

```
filename fromzip zip 'mydemo.zip';
data _null_;
     length memname $200;
     fid=dopen("fromzip");
     if fid=0 then stop;
     memcount=dnum(fid);
     do i=1 to memcount;
        memname=dread(fid,i);
        put memname=;
        fid2 = mopen(fid,memname,'s');
        if fid2=0 then do;
           put 'could not open ' memname;
           continue;
           end;
        do while(fread(fid2)=0 and fget(fid2,record,200)=0);
           put memname= record=;
           end;
        rc = fclose(fid2);
        end;
     rc=dclose(fid);
     run;
```

USING MULTIPLE DIRECTORY LEVELS

A ZIP file can contain many directory levels. You can access the files by separating the directory levels with a forward slash. For example, if you want to create a directory level called level2 with a new member called newmem2.txt, you

can use the following code:

```
data _null_; file fromzip(level2/newmem2.txt);
     put 'first line of newmem2.txt';
     put 'second line of newmem2.txt';
     put 'last line of newmem2.txt';
     run;
```

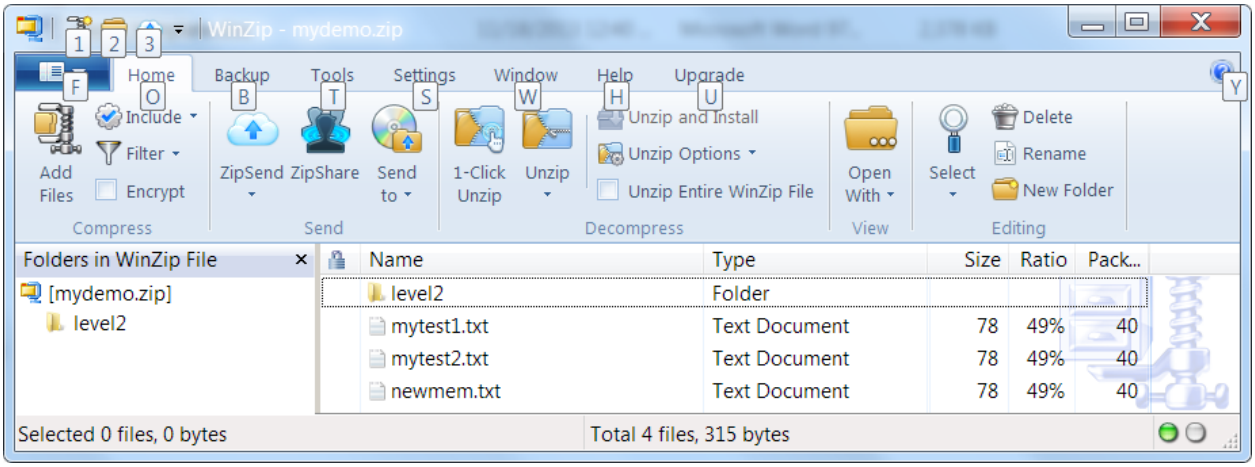
SUPPORTED OPERATING SYSTEMS

You can use the ZIP access method on any supported operating system in SAS 9.4. On MVS, a ZIP file must be in HFS space and cannot be an OS data set. (That is, the PATH= option must be used in JCL to reference it). Remember that if you copy a ZIP file from one platform to another, you must ensure that it remains a binary file. Also remember that the ENCODING= option might be necessary if you are reading from a ZIP file. For example:

```
filename fromunix zip '/myfiles/mydemo.zip' encoding=latin1;
data _null_; infile fromzip(mytest1.txt);
     input; list;
     run;
```

Suppose that you run on MVS, you do not use the ENCODING= option, and the ZIP members contain ASCII text. The listing will show unprintable characters because there is no transcoding to EBCDIC. Note that you do not want to do this if the ZIP members were originally written with binary data (such as JPG files).

The ZIP file will be fully recognizable by standard ZIP tools on your operating system of choice. For example, here is a view of the mydemo.zip file on Windows. Note that level2 is shown as a folder.



Note: A ZIP file does not have to be named with a .zip or .ZIP extension, although it is certainly a best practice to do so. There are common extensions for files that are actually ZIP files, such as docx, xlsx, and jar. These extensions might be readable by the ZIP access method, although there is no compelling reason to process them. Other archive forms, such as .gz, are not supported by the Zip access method. Also, the ZIP access method does not support password-protected ZIP files.

OTHER ACCESS METHODS

You can use the techniques described in this paper with other SAS access methods that are directory-based. For example, you can use the CATALOG access method this way with parentheses specifying the catalog member:

```
filename mycat catalog 'work.mycat';
data _null_; file mycat(sample.source);
     put 'line 1';
     put 'line 2';
     put 'line 3';
     run;
```

The SAS log includes the following:

```
NOTE: The file MYCAT(sample.source) is:
      Catalog Name=work.mycat.SAMPLE.SOURCE,
      Catalog Page Size=4096,
      Number of Catalog Pages=4,

NOTE: A total of 3 records were written to the file library MYCAT.
      The minimum record length was 6.
      The maximum record length was 6.
NOTE: 3 records were written to the file MYCAT(sample.source).
```

Other applicable access methods include FTP, SFTP, HADOOP, WEBDAV, and the standard DISK access method (if you are referring to a directory). Note that you cannot use the SOCKET or HTTP access methods to refer to directories.

CONCLUSION

The ZIP access method is a convenient way to read ZIP files generated by other processes, and to write ZIP files that can be used elsewhere. The access method is compatible with other access methods such as CATALOG in that the standard syntax and tools are available for ZIP.

CONTACT INFORMATION

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

Rick Langston
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Rick.Langston@sas.com