Paper 297-2013

Back Up Your Sources During Development: A Stack of Base SAS® Scripts

Hans Sempel, Belastingdienst (Dutch Tax and Customs Administration), Apeldoorn, The Netherlands

ABSTRACT

If you're a Base SAS programmer and if you ever lost your code due to system crashes or overwriting your code, this might be the solution. The presented code provides a means of backing up your code during development, you can use it to save increments or you can use it for versioning and you can restore the code you're working on to an earlier version.

INTRODUCTION

In paper 037-2007, Suzanne Humphreys explained an easy way to create a backup of your code while programming in SAS. To fit the need of our programmers, the script needed some extra functionality as well as some improvements on the way the script changed the name of the code.

The extra functionality needed was a way to store backups on the same location and with the same name as the original script as well as the ability to restore the latest backup. Paper 023-2008 from Arthur L. Carpenter contained useful code to create the desired backup functionality. The script I wrote contains the backup functionality with version numbering as well as the functionality to restore backup code.

WHAT IT DOES

Imagine you are writing a piece of SAS code named testscript.sas

When creating a backup, the file testscript.sas is stored as testscript_1.sas and the script in the active editor is stored as testscript.sas. The number in the name of earlier backups is incremented by one.

EDITOR →TESTSCRIPT.SAS → TESTSCRIPT_1.SAS → … → TESTSCRIPT_n.SAS

When restoring a backup, the file testscript.sas is deleted and testscript_1.sas renamed to testscript.sas. The number in earlier backups is decreased by one.

EDITOR ←TESTSCRIPT.SAS ← TESTSCRIPT_1.SAS ← … ← TESTSCRIPT_n.SAS

The explained proces is quite similar to a LIFO stack where the backup key functions as a push command and the restore key as a POP command.

INSTALLATION

To make this script work, a few actions are needed.

Step one, store the programs 'backup.sas' and 'restore.sas' in a directory (e.g. C:\backup_script\).

Step two, define the function keys. To define the function keys, submit the program DM_COMMANDS.sas. This program contains four lines of code:

```
dm "keydef 'SHF F12' 'submit ''%nrstr(%let _product_path =
        %sysget(SAS_EXECFILEPATH);)'';'";
dm 'keydef "F12" "pgm; inc ""c:\backup_script\backup.sas""; submit;"';
dm 'keydef "CTL F12" "pgm; inc ""c:\backup_script\restore.sas""; submit;"';
dm 'keys; save;';
```

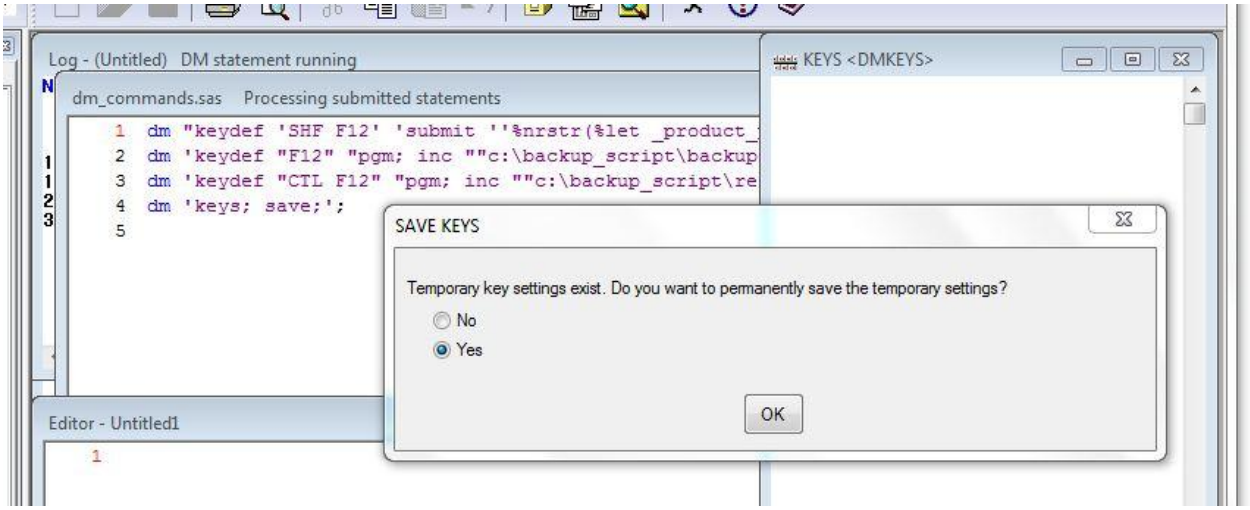
The command in the first line puts the complete path and filename of the code in the active editor window in the macro variable '_product_path'. This command is assigned to the key Shift-F12.

The command in the second line submits the program 'backup.sas' and this command is assigned to the key F12.

The third line assignes the key Control-F12 to submit the program 'restore.sas'.

The last line in the code opens the KEYS window and gives the command 'SAVE'.

A popup appears. Select 'Yes' and click on the OK button to confirm. The functionality is stored and ready to use in this and the following SAS sessions.



Display 1. The Popup

PREPARATION

To make the backup and restore program work properly you need to make the path and filename of the code you're working on known to the system. By making the editor window in which the code is opened active and then pressing Shift-F12, the full path and filename are known during the whole session.

It's important to do this step for each SAS session to make the scripts work!

BACKING UP YOUR CODE

You can backup your code by pressing the key F12. The contents of the file '`c:\backup_script\backup.sas`' is then loaded into an editor and submitted.

The first line is to make the x-commands function properly:

```
options noxwait xsync;
```

The next piece of code generates a DIR command which is then used with a PIPE to read the directory in which the source code is stored.

```
data _null_;
    p0 = length("&_product_path");
    p1 = findc("&_product_path",'\',(p0*-1));
    product=substr("&_product_path",p1+1,p0-p1-4);
    path=substr("&_product_path",1,p1);
    _command="'"||'dir "'||trim(left(path))||trim(left(product))||"*.sas"||'" /t:w
        /a:-d /OD'||"'";
    call symput('_command',_command);
    call symput('_path',trim(path));
    call symput('_product',trim(upcase(product)));
run;

filename backup pipe &_command;
```

With a data step the directory is read and the exact name of the file (in proper case) and the amount of backups is determined.

```
filename backup pipe &_command;
data _null_;
    infile backup missover pad length=len;
    input @01 line $varying200. len;
    if index(upcase(line)," &_product.");
```

```
    fname=compress(substr(line,index(upcase(line),"&_product"),length("&_product.")
        ));
    orda+1;
    call symput("orda",orda);
    call symput("fname",trim(left(fname)));
run;
```

The above code is also contained in the restore program.

The actual backup is done by a data step that renames all the files and increases the version number by 1 by means of rename statements and a copy statement

```
data _null_;
    format _command $200.;
    do i=&orda to 1 by -1;
        ordamin1 = i - 1;
        if (ordamin1 eq 0) then
            _command='copy "'||"&_path.&fname..sas"||'""'||
                     "&_path.&fname._1.sas"||'"';
        else
            _command='rename "'||"&_path.&fname._"||
                     trim(left(put(ordamin1,3.)))||".sas"||'""'||
                     " &fname._"||trim(left(put(i,3.)))||".sas";
        call system(_command);
    end;
run;
```

The last line of code saves the contents of the active editor window and closes the backup script.

```
dm 'wpgm; file "&_path.&fname..sas" replace; pgm off' editor;
```

RESTORING YOUR CODE

The restore program works similar and the code to determine the filename and the amount of backups is the same as the code in the backup program.

The actual backup is done by a data step that renames all the files and decreases the version number by 1 by means of rename statements and a delete statement

```
data _null_;
    format _command $200.;
    do i=1 to &orda - 1;
        ordamin1 = i - 1;
        if (ordamin1 eq 0) then do;
                _command='del "'||"&_path.&fname..sas";
                put _command=;
                call system(_command);
                _command='rename "'||"&_path.&fname._"||
                         trim(left(put(i,3.)))||".sas"||'""'||" &fname..sas";
                put _command=;
                call system(_command);
        end;
        else do;
                _command='rename "'||"&_path.&fname._"||
                         trim(left(put(i,3.)))||".sas"||'"'||
                         " &fname._"||trim(left(put(ordamin1,3.)))||".sas";
                put _command=;
                call system(_command);
        end;
    end;
run;
```

Close the restore script and go to the editor window.

```
dm 'pgm off' editor;
```

Clear the active editor window and load the restored file.

```
dm editor 'clear; include "&_path.&fname..sas"';
```

PROS AND CONS

PROS:

- The backups are saved in the same directory as your script.
- The name of the script doesn't change, so the save button still works.

CONS:

- When you restore a code, the actual code you're working on will be lost.
- When you backup or restore, the editor in which the code is edited must be active.

CONCLUSION

The method described in this paper can be very helpful in many different ways. It can be used during development to save increments. During exploitation where the same program has to run frequently with slightly different settings, it can be used to build up a history.

I'm sure that there are many more uses and its relative easy to adjust the code for your needs.

REFERENCES

- Humphreys, Suzanne., May 17, 2007 "Back Up with Each Submit and Save Your Sanity!" *SAS Global Forum 2007*. Orlando, Florida. Available at http://www2.sas.com/proceedings/forum2007/037-2007.pdf
- Carpenter, Arthur L., March 13, 2008 "The Path, the Whole Path, and Nothing but the Path, So Help Me Windows" *SAS Global Forum 2008.* San Antonio, Texas. Available at http://www2.sas.com/proceedings/forum2008/023-2008.pdf
- Carpenter, Arthur L., April 2, 2012 "Doing More with the SAS® Display Manager: From Editor to View Table – Options and Tools You Shouls Know" *SAS Global Forum 2012.* Orlando, Florida. Available at http://support.sas.com/resources/papers/proceedings12/151-2012.pdf

CONTACT INFORMATION

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

Name:	Hans Sempel
Enterprise:	Belastingdienst
Address:	Hoofdstraat 21
City, State ZIP:	7311 JT Apeldoorn, The Netherlands
E-mail:	hans@sempel.eu

APPENDIX

DM_COMMANDS.SAS:

```
dm "keydef 'SHF F12' 'submit ''%nrstr(%let _product_path =
            %sysget(SAS_EXECFILEPATH);)'';'";
dm 'keydef "F12" "pgm; inc ""c:\backup_script\backup.sas""; submit;"';
dm 'keydef "CTL F12" "pgm; inc ""c:\backup_script\restore.sas""; submit;"';
dm 'keys; save;';
```

BACKUP.SAS:

```
options noxwait xsync;

data _null_;
    p0 = length("&_product_path");
    p1 = findc("&_product_path",'\',(p0*-1));
    product=substr("&_product_path",p1+1,p0-p1-4);
    path=substr("&_product_path",1,p1);
    _command="'"||'dir "'||trim(left(path))||trim(left(product))||"*.sas"||
                                                    '" /t:w /a:-d /OD'||"'";
    call symput('_command',_command);
    call symput('_path',trim(path));
    call symput('_product',trim(upcase(product)));
run;

filename backup pipe &_command;

data _null_;
    infile backup missover pad length=len;
    input @01 line $varying200. len;
    if index(upcase(line)," &_product.");
    fname=compress(substr(line,index(upcase(line),"&_product"),
                                          length("&_product.")));
    orda+1;
    call symput("orda",orda);
    call symput("fname",trim(left(fname)));
run;

data _null_;
    format _command $200.;
    do i=&orda to 1 by -1;
        ordamin1 = i - 1;
        if (ordamin1 eq 0) then
            _command='copy "'||"&_path.&fname..sas"||'""'||
                     "&_path.&fname._1.sas"||'"';
        else
            _command='rename "'||"&_path.&fname._"||
                     trim(left(put(ordamin1,3.)))||".sas"||'"'||
                     " &fname._"||trim(left(put(i,3.)))||".sas";
        call system(_command);
    end;
run;

dm 'wpgm; file "&_path.&fname..sas" replace; pgm off' editor;
```

RESTORE.SAS:

```
options noxwait xsync;

data _null_;
    p0 = length("&_product_path");
    p1 = findc("&_product_path",'\',(p0*-1));
    product=substr("&_product_path",p1+1,p0-p1-4);
    path=substr("&_product_path",1,p1);
    _command="'"||'dir "'||trim(left(path))||trim(left(product))||"*.sas"||'" /t:w
/a:-d /OD'||"'";
    call symput('_command',_command);
    call symput('_path',trim(path));
    call symput('_product',trim(upcase(product)));
run;

filename backup pipe &_command;

data _null_;
    infile backup missover pad length=len;
    input @01 line $varying200. len;
    if index(upcase(line)," &_product.");

fname=compress(substr(line,index(upcase(line),"&_product"),length("&_product.")));
    orda+1;
    call symput("orda",orda);
    call symput("fname",trim(left(fname)));
run;

data _null_;
    format _command $200.;
    do i=1 to &orda - 1;
        ordamin1 = i - 1;
        if (ordamin1 eq 0) then do;
                _command='del "'||"&_path.&fname..sas";
                put _command=;
                call system(_command);
                _command='rename "'||"&_path.&fname._"||
                        trim(left(put(i,3.)))||".sas"||'"'||" &fname..sas";
                put _command=;
                call system(_command);
        end;
        else do;
                _command='rename "'||"&_path.&fname._"||
                        trim(left(put(i,3.)))||".sas"||'"'||
                        " &fname._"||trim(left(put(ordamin1,3.)))||".sas";
                put _command=;
                call system(_command);
        end;
    end;
run;

dm 'pgm off' editor;
dm editor 'clear; include "&_path.&fname..sas"';
```