PAPER CC05

USING SCREEN CONTROL LANGUAGE – WITHOUT A SCREEN

Steve Prust, Independent Consultant, Germany

ABSTRACT

SCL (Screen Control Language) is designed to run code for screen based applications in SAS/AF. It is a similar in look and feel to ordinary Data step code and shares many of its features. However, SCL has its own features - and these features can add valuable functionality to both SAS programs and the Display Manager To do so requires the "screenless" use of SCL. This paper demonstrates this approach together with some examples of how SCL functionality can be exploited.

SOME FEATURES OF SAS/AF

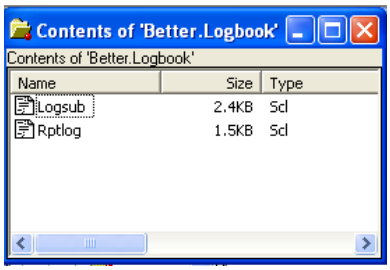
For the first time user SAS/AF has many similarities with the SAS Data Step language in terms of look-and-feel and syntax structure. What makes AF very different is the execution of the code. Firstly, AF is a compiled language rather than interpreted. Secondly, the Data Step typically runs iteratively through one or more input data sets, whereas AF has no such default processing. AF, however, has features that are either not available or difficult to obtain in the Data Step. Some examples of these could be:

- SCL lists
- access to Windows functions such as the Paste Buffer and Message Box
- ability to use OLE functionality e.g. to open and manipulate Word documents

EXAMPLE – SCL LISTS

The following example shows how one of these constructs - SCL Lists - can be used directly from within a program to construct a simple logbook.

SCL code needs to be entered in a SCL program entry within a SAS catalog.



SCL Lists are highly flexible storage areas that can be permanently saved within catalogs (for a full description see the SAS Help manual). In this application the SCL Lists are stored permanently in the SASUSER libref.

3. SAMPLE APPLICATION

3.1 CODING THE APPLICATION

Here is some SCL code to add an entry to a logbook:

```
init:
  dcl num rc;
  yearday  = juldate7(date());
  week     = ceil((yearday - (1000*int(yearday/1000)))/7);
  now      = put(date(),date9.) || ' ' || put(time(),time5.);
  job      = symget('progname');
  ActList  = makelist();
  rc       = fillist('catalog','sasuser.logbook.Week' || put(week,z2.) ||'.SList',ActList);
  rc       = insertc(ActList,'Job ' || job || ' run at ' || now, -1);
  NullList = makelist();
  rc       = savelist('catalog','sasuser.logbook.Week' || put(week,z2.)||'.Slist',ActList,
               NullList ,'Activity Logbook - Week' || put(week,z2.) );
  NullList = dellist(NullList);
```

```
  ActList = dellist(ActList);
return;
```

and some SCL code to report the logbook for a particular week:

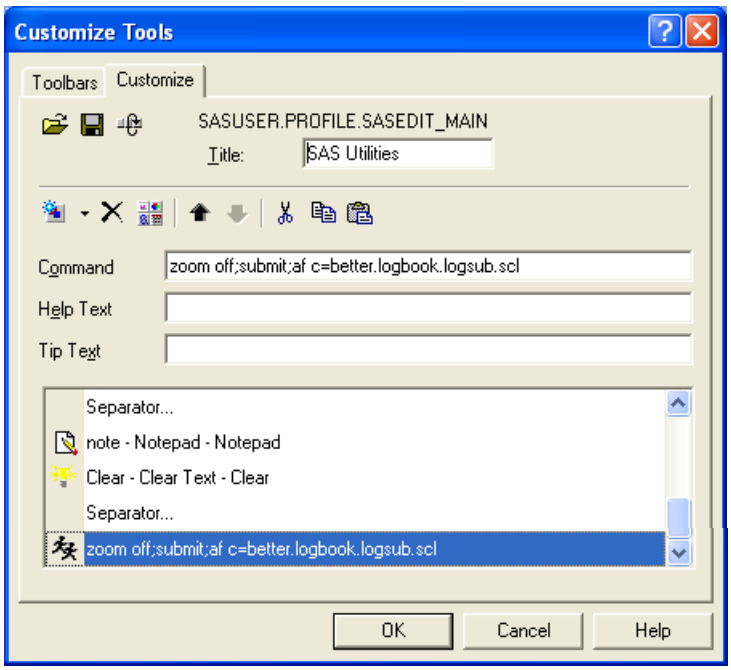
```
init:
  dcl num rc ;
  dcl char week;
  week    = symget('rptwk');
  ActList = makelist();
  rc      = fillist('catalog','sasuser.logbook.Week' || week ||'.SList',ActList);
  if rc = 0 then do;
    put 'Report for week ' week;
    do i = 1 to listlen(ActList);
      line = getitemc(ActList,i);
      put line;
    end;
  end;
  else
    put 'No report for week ' week;
  ActList = dellist(ActList);
return;
```

3.2 INVOKING THE APPLICATION

This is the code (written in either program editor or enhanced editor) to execute the SCL code that then writes an entry to the logbook

```
%let progname = easyprog;
dm 'AF c=better.logbook.logsub.scl';
```

But we can also attach the instruction to run the SCL code to a button, for example:



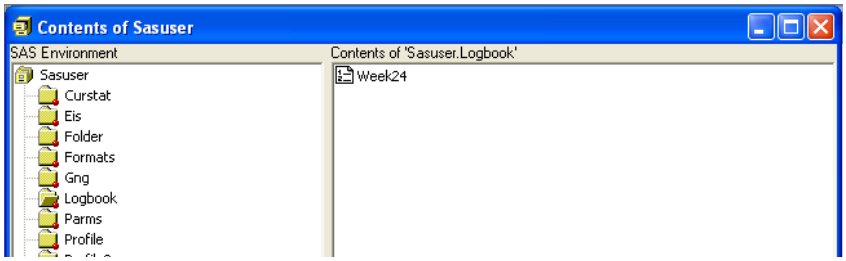
Here is the code to report a week's events from the log book

```
%let rptwk = 24;
dm 'AF c=better.logbook.rptlog.scl';
```

Results of the report in the log

```
10   dm 'AF c=better.logbook.rptlog.scl';
Report for week  24
Job myprog run at 16JUN2009 16:52
Job myprog run at 16JUN2009 16:52
Job myprog run at 16JUN2009 16:52
Job anotherprogrun at 16JUN2009 16:55
Job easyprog run at 17JUN2009  8:52
Job fred run at 17JUN2009  8:56
```

Or we can see the logbook directly through SAS Explorer



Double-clicking on the relevant item puts into the log the following:

```
SLIST(
   'Job myprog run at 16JUN2009 16:52' {C}
   'Job myprog run at 16JUN2009 16:52' {C}
   'Job myprog run at 16JUN2009 16:52' {C}
   'Job anotherprogrun at 16JUN2009 16:55' {C}
   'Job easyprog run at 17JUN2009  8:52' {C}
   'Job fred run at 17JUN2009  8:56' {C}
)[3]
```

4. OTHER SAMPLE CODE

4.1 PASTE AND MESSAGE BOX

This code shows an example of the Paste Buffer being instanced and also the Message Box functionality being used. The code looks for a variable name in the current paste buffer (clipboard), attempts to find the unique values of that variable in the last dataset created. If successful it places those unique values in the paste buffer and issues a message box command.

```
init:
  dcl num rc;
  dcl char crc;
  pbuf=loadclass('sashelp.classes.pastebuffer_c');
  dcl object pbufid;
  pbufid=instance(pbuf);
  if listlen(pbufid.text) then do;
    var = scan(getitemc(pbufid.text,1),1,' =);-*+:]');
    lastds = trim(symget('syslast')) ;
    dsid = open(lastds);
    vnum = varnum(dsid,var);
    if vnum then do;
      nlevels=0;
      unilist=makelist();
      rc=lvarlevel(dsid,var,nlevels,unilist);
      pbufid.text = copylist(unilist);
      rc = dellist(unilist);
      msglist=makelist();
      rc = insertc(msglist,"Unique values of '" || var || "' placed in clipboard",-1);
      crc = MessageBox(msglist,'I','O','Info');
      rc = dellist(msglist);
    end;
    rc = close(dsid);
  end;
  pbufid._term();
return;
```

This approach for using the Message Box is perhaps easier than the alternative of using the Windows API call via a SASCBTBL file method.

4.2 OLE AND WORD

This code shows how Word can be instanced, a document opened and text retrieved based on a location in the document.

```
hostcl = loadclass('sashelp.fsp.hauto');
call send(hostcl, '_new_', wordobj, 0, 'Word.Application');
call send(wordobj, '_set_property_', 'visible', 'True');
call send(wordobj, '_get_property_', 'Documents', docobj);
call send(docobj,  '_do_', 'Open',worddoc);
call send(wordobj, '_get_property_', 'ActiveDocument',adoc);
```

```
call send(wordobj, '_get_property_', 'Selection',selobj);
call send(selobj, '_get_property_', 'Find',findobj);
call send(findobj, '_do_', 'ClearFormatting');
call send(findobj, '_set_property_', 'Forward'          , -1); /* -1 = True */
call send(findobj, '_set_property_', 'Text', "Date of Birth: ");
call send(findobj, '_compute_', 'Execute', dobfrc);
if dobfrc = -1 then do;
  call send(selobj, '_do_', 'MoveRight',12); /* 12=wdCell i.e. move next cell*/
  call send(selobj, '_get_property_', 'Text',dobtext); /* get text from cell*/
etc etc ...
```

Note: OLE makes use of a standardised interfaces with a Word document and enables more robust interfacing than DDE.

5. CONCLUSION

Used appropriately AF has the ability to add significant new functionality to the standard SAS environment and to enable otherwise difficult problems to be resolved. This paper demonstrates just a few of the many techniques that are available.

6. REFERENCES

A. Darrell Massengill, SAS/AF: Running SCL Outside the Frame
http://support.sas.com/rnd/papers/sgf07/sgf2007-af.pdf

Curley, Lynn. 2006. “SAS/AF® Rises Again: Enhancements in SAS® 9.2”. SUGI 31 Proceedings. Cary, NC: SAS Institute Inc.
Available: http://support.sas.com/events/sasglobalforum/previous/online.html

Davis, Michael. 1998. “SCL for the Rest of Us: Nonvisual Uses of Screen Control Language”. SUGI 23 Proceedings. Cary, NC:
SAS Institute Inc. Available: http://support.sas.com/events/sasglobalforum/previous/online.html

CONTACT INFORMATION

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

Steve Prust
<to be advised>