Developing a Simple Application with SAS/DMI® Software
Dale D. Ingold, SAS Institute Inc., Cary, NC

ABSTRACT

A simple ISPF application using SAS® software is developed in this paper. The application allows you to enter and update events, such as meetings and appointments, in a SAS data set and to display the events scheduled for the current day along with a calendar.

The application consists of an ISPF panel and SAS source code. The Calendar and Notepad application distributed with SAS/DMI® software is also used. Edit models are used to ease the development.

INTRODUCTION

The first task is to decide how to access the components of the application. The components can consist of panels, messages, skeletons, and dialog functions that are written using the SAS DATA step language.

Since this is a simple application, it consists of a single panel and three SAS source code members.

The panel is put into a personal library that is concatenated ahead of the standard ISPF panel libraries. This makes it easier to use the SAS/DMI Application Development Tools distributed with SAS/DMI software. Alternatively, the application can use the LIBDEF ISPF service to tell ISPF to search the personal library before searching the standard libraries. There is some performance degradation with this method, but it does not require altering the standard allocation of the ISPF libraries. The choice is a matter of personal preference and convenience.

The SAS source code can be in any library; the library is allocated by the invocation CLIST. During development, the source code is included directly from the library using the Application Development Tools menu.

Later, a CLIST is written to allocate SAS libraries, issue the LIBDEF service request to ISPF for the ISPF panel library, and invoke the SAS System via the SASOMI interface.

Design

The goals of our application are as follows:

• to be able to enter events such as meetings and appointments into a SAS data set. (See the ADDENDUM later in this paper for a description of how the SAS data set can be used.)

• to display a calendar and a list of events for the current day.

• to be able to invoke the application at the beginning of the day and leave it active all day. The SAS data library must not be in use (allocated) while the calendar is being displayed.

• to add, delete, or change any scheduled event and then redisplay the calendar and notepad.

The application must allocate the SAS data library containing the calendar data set, extract those events scheduled for the current day, and display them (and a calendar). The Calendar and Notepad dialog distributed with SAS/DMI software displays a scrollable calendar and a list of notes. The notes are ISPF profile variables. If those notes are set to the day's events, the Calendar and Notepad dialog can be invoked to handle the display part. (See Screen 1.)

Screen 1 Calendar and Notepad

When that dialog terminates, you are asked if you want to update the calendar or exit the application. If you just press ENTER, the UPDATE function is invoked, followed by reinvoking the application from the beginning; if you execute the END command, then the application terminates.

One additional feature that would be useful is the ability to have regularly scheduled events, such as weekly and biweekly meetings. For repeated events, you need only enter the date of the first (or next) occurrence and the frequency, such as 1 for weekly, 2 for biweekly, and so on. Entering a frequency of 0 implies a one-shot, or non-repeated, event.

A final consideration is the environment in which the application will run. At the Institute, we use a multi-session product called Productivity Integrated Environment/Sessions (PIE/Sessions) from Technologic Software Concepts, Incorporated. PIE/Sessions allows us to appear to have several TSO sessions running, all under the same userid. In reality, each pseudo-session is a separate task running in the same address space. Thus, all files allocated in any pseudo-session are visible to all others. PIE/Sessions renames certain ISPF-related files, such as SPLIST, SPSPROF, and so on, when switching between pseudo-sessions so each pseudo-session can invoke ISPF.)
DEVELOPMENT

To develop the SAS code, it is helpful to use the PDF editor under the SAS/DMI Application Development System and, particularly, the SAS/DMI models.

The first step of the program is to dynamically allocate the SAS data library containing the calendar data set. The TSO statement can be used, but that requires that the application be invoked via SASCP, the SAS command processor. This application uses the CALL DMI statement in SAS/DMI software because it does not require the overhead of invoking the SAS System through SASCP. Selecting option Elan the Tools menu and specifying the data set name of the SAS source code library and member name CALEN000 brings up the PDF editor; MODEL CLASS SAS and MODEL DMI are specified to bring in model lines and notes describing the parameters and return codes for the CALL DMI statement. A simple DATA step is wrapped around the model lines, and some logic is added to shut the application down if it cannot get exclusive access to the library.

To make sure that the DDname is unique, a macro variable is defined that is a unique suffix for all DDnames used by the application. For example, if the value of the macro variable SUFFIX is 51, then the libref for the calendar data set is DCNDAT51.

The following DATA step allocates the library:

```
DATA _NULL;
  LENGTH ERROR INFO $2;
  CALL OMIDYNAM (ERROR, INFO, 'ALLOCATE', 'DCNDAT51.SUFFIX', 'SASDI.INGOOL.DAT', 'NEW);
  IF ERROR ^= 0 THEN DO;
    P='ALLOCATION FAILED FOR CALENDAR LIBRARY';
    CALL EXECUTE (' ENDSAS; ', ) ;
    PUT P;
  END;
RUN;
```

To finish the initialization member, a PROC FORMAT step is added to create a format for the days of the week. Finally, a %INCLUDE statement is added for the mainline member (CALEN010).

Next the mainline source code member, CALEN010, is created. First, a DATA step is added to update the dates for repeated events. The date for any repeated event is checked to see that it is either the current date or in the future. This way the date of the previous event can be entered, for example, and the application will project it to the next meeting date. The data set is also sorted by date and time, and the PRINT procedure is executed. This gives an overview of the schedule for the next several days. A minimum amount of time should be spent looking at the PROC PRINT output since this ties up the data library.

Next the code is added to extract events for the current day and create ISPF profile variables containing a description of each event. These variables are later displayed by the Calendar and Notepad application.

This is the code that creates the ISPF variables:

```
DATA _NULL;
  LENGTH NOTENAME $8;
  ARRAY DKINOTE(*) $46 DMINOTE1-DMINOTE9
  LENGTH DMINOTEB DMINOTEC DMINOTE DMINOTEI DMINOTEII DMINOTEIII
  DATA ...NULL;
  CALL OMIDYNAH (ERROR, INFO, 'FREE', 'DCNDAT51.SUFFIX', 'FREE');
  CALL EXECUTE (' ENDSAS; '); END;
RUN;
```

Next the code is added to query the user about whether to invoke the calendar update dialog or to terminate the application. The code consists of a DATA step that displays a panel instructing you to press ENTER to update the calendar data set and redisplay the calendar and notepad, or to execute the END command to exit the application. The ZSEL variable is set by the panel depending on the response. It is then passed to the EXECUTE CALL function to be executed by the SAS System immediately after the current DATA step. If you execute the END command, an ENDSAS statement is generated; the SAS System sees the ENDSAS statement and shuts down. If you just press ENTER, a %INCLUDE statement is generated that includes the calendar extraction and display functions.

Note the NORETURN option in the %INCLUDE statement; this tells the SAS System that you are through reading from the current file and the SAS System should not remember the current line and location on the line. Otherwise, the SAS System would remember all that information each time you passed through this file, taking up memory unnecessarily, since this is the last line in the file.

Finally the update dialog function (CALEN020) is added. First, the code from member CALEN000 is copied to allocate the library. To keep things simple, PROC FSEDIT is used for now. Later, a fancier ISPF dialog can be written, possibly using tables to display and manipulate the calendar. To delete an event, the DELETE command in PROC FSEDIT can be used. This sets all
variables to missing. If the EVENT variable is required on the
FSEDIT screen, then any observation with EVENT equal to blanks
must be a deleted one, so it can be deleted from the data set with
a simple DATA step:

```sas
PROC FSEDIT DATA = DCNDAT&SUFFIX.CALENDAR;
FORMAT DAY WEEKDAYS;
RUN;
DATA DCNDAT&SUFFIX.CALENDAR;
SET DCNDAT&SUFFIX.CALENDAR;
IF EVENT = ' ';
RUN;
```

**SUMMARY**

The complete SAS code, ISPF panel, and CLIST to invoke the
application are included in the APPENDIX. A flowchart of
the application is shown in Figure 1. Note that the CLIST uses com-
mands that may not be available at your installation; where possi-
ble, these have been indicated.

```sas
start
| 1
| V

;CALEN000: 
!- create format
!- alloc library

;CALEN010: 
!- update dates
!- sort calendar
!- print calendar
!- free library
!- extract today's
!- events and
!- build notes
!- invoke DMICALEN
!- display CALENQ

;CALEN020: 
!- allocate library

;CALENQ: 
!- delete events

;CALEN000: 
| !- update dates
| !- sort calendar
| !- print calendar
| !- free library
| !- extract today's
| !- events and
| !- build notes
| !- invoke DMICALEN
| !- display CALENQ

Figure 1 Flowchart
```

Member CALENOO0 is invoked first; it creates a format that is
used by PROC PRINT and PROC FSEDIT. Next it allocates the
calendar data library. Then it invokes the main function code
member. CALENOO0 is not executed again during this session.

Member CALENO10 updates the dates in the calendar data set
for repeated events. Then it sorts the data set by date and time
and displays it with PROC PRINT. Then it extracts the events for
the current day and creates ISPF profile variables to contain the
time, location, and event name. It is then finished accessing the
calendar data library, so it frees the library. Next it includes the
Calendar and Notepad application to display a calendar and the
list of events scheduled for the day. When that dialog terminates,
it displays a panel asking you if you want to edit the calendar data
set or terminate. If you want to edit the data set, members
CALENO20 and CALENO10 are included. Otherwise, an ENDVANAS
statement is generated, which stops the SAS session.

Member CALENO20 first allocates the calendar data library and
then invokes PROC FSEDIT. Next it deletes any observations that
had variables set to missing or blank by PROC FSEDIT. When
it finishes executing, the SAS System begins executing member
CALENO10 again.

**ADDENDUM**

This section explains why I need the calendar information in a
SAS data set. Another product installed at the Institute is
PIE/Network Access. This product allows a TSO user to actually
log on to TSO (or some other VTAM application). This can be
done from a PIE/Sessions pseudo-session; thus a pseudo-
session can be a completely separate TSO session. In such a
session, I start a noninteractive SAS session that reads the calen-
dar data set and selects the next event to occur on the current
day. It then invokes a very simple external program as a proce-
dure, with the time five minutes prior to the next event specified
in an OPTIONS statement as the value of the PARM= option. Fol-
lowing the procedure is a TSO statement that issues a SEND
command to all of my userids telling me that it is five minutes until
the next event, what the event is, and its location. Then the code
reincludes itself to go to sleep until the next event; if none is
found, the SAS session ends.

Thus, I am notified (if and only if I am active at the terminal) about
events such as meetings that are about to happen. This SAS ses-
Sion has to run in a separate TSO session and not a PIE/Sessions
pseudo-session because PIE/Sessions does not allow more than
one session to be active at a time. TSO does, so the separate
session executes completely independently of my other ses-
sions. That is also why I need to have the calendar data set allo-
cated for as little time as possible; the reminder session needs
to read the SAS data set, and the calendar session needs to
update it. The application can be enhanced. Use SAS/SHARE®
software to access the library.

**APPENDIX**

**SAS Source Code Member CALENOO0**

```sas
PROC FORMAT DD=
/.
/.
/.
/.
/.
/.
/.
!- create format

ALLOC library

Value WEEKDAYS {MIN = 1}
1 = 'Sunday' 2 = 'Monday'
3 = 'Tuesday' 4 = 'Wednesday'
5 = 'Thursday' 6 = 'Friday' 7 = 'Saturday';
RUN;
```

.member CALENO000 is invoked first; it creates a format that is
used by PROC PRINT and PROC FSEDIT. Next it allocates the
calendar data library. Then it invokes the main function code
member. CALENO000 is not executed again during this session.

Member CALENO10 updates the dates in the calendar data set
for repeated events. Then it sorts the data set by date and time
and displays it with PROC PRINT. Then it extracts the events for
DATA JULL;
LENGTH ERROR INFO $2;
CALL DKIDYNAK (ERROR, INFO, 'ALLOCATE', 'DCNDATSUFFIX', 'SASDDI.INGOLD.DATA', 'OLD');

IF ERROR ^= '0000' THEN DO;
  PUT 'ALLOCATION FAILED FOR CALENDAR LIBRARY';
  CALL EXECUTE ('ENOSAS');
END;
RUN;
/*
  /* Initialize the main code for the CALENDAR application. */
  */
SINCLUDE nCNIN'SUfFIX(CALENO10);

SAS Source Code Member CALEN010
/*
  /* Project the dates of any past repeated event to the date of the next occurrence, delete any past nonrepeated events, extract variables describing each event. */
  */
DATA DCNDATSUFFIX.CALENDAR;
  RETAIN THISDATE THISDAY NOW;
  SET DCNDATSUFFIX.CALENDAR;
  IF = THEN DO;
    THISDATE = TODAY;
    THISDAY = NOW;
  END;
  WEEKDAY (THISDATE);
  TIME (I);
  IF DAY THEN DAY WEEKDAY (DATE);
  SELECT (FREQ);
  WHEN {OJ DO;
    IF DATE < THISDATE THEN DELETE;
    IF DATE THISDATE THEN DELETE;
    END;
  WHEN {ll DO;
    ADJUST DAY - THISDAY;
    IF ADJUST < 0 THEN ADJUST = ADJUST + 7;
    DATE THISDATE + ADJUST;
    END;
  OTHERWISE DO;
    IF DATE THISDATE THEN DO;
      INT FLOOR {ABS {INTCK ('WEEK', THISDATE, DATE)} FREQ);
      DATE THISDATE + ADJUST;
    END;
  END;
  OUTPUT DCNDATSUFFIX.CALENDAR;
  DROP THISDATE THISDAY NOW ADJUST INT;
RUN;
/*
  /* Use PROC SORT to sort the updated calendar by date and time. */
  */
PROC SORT DATA = DCNDATSUFFIX.CALENDAR;
  BY DATE TIME;
RUN;
/*
  /* Use PROC PRINT to print the complete calendar. */
  */
PROC PRINT DATA = DCNDATSUFFIX.CALENDAR;
  ID DATE;
  VAR DAY FREQ TIME LOC EVENT;
  FORMAT DAY WEEKDAY3; TITLE 'Updated Calendar'; FOOTNOTE 'Press END (PF15) to continue';
RUN;
/*
  /* Allocate the SAS data library that contains the CALENDAR data set. */
  */
DATA _NULL;
 LENGTH ERROR INFO $2;
 CALL DKIDYNAK (ERROR, INFO, 'ALLOCATE', 'DCNDATSUFFIX', 'SASDDI.INGOLD.DATA', 'OLD');
 IF ERROR ^= '0000' THEN DO;
   PUT 'ALLOCATION FAILED FOR CALENDAR LIBRARY';
   CALL EXECUTE ('ENOSAS');
 END;
RUN;
/*
  /* Invoke the Calendar and Notepad application to display a calendar and the notes about today's events. */
  */
DATA _NULL;
 LENGTH ZSEL $80;
 CALL ISPLINK ('VDEFINE', 'ZSEL');
 CALL ISPLINK ('DISPLAY', 'CALENDAR');
 CALL EXECUTE (ZSEL);
 CALL ISPLINK ('VDELETE', 'ZSEL');
RUN;
/*
  /* Allocate the SAS data library that contains the CALENDAR data set. */
  */
DATA _NULL;
 LENGTH ERROR INFO $2;
 CALL DKIDYNAK (ERROR, INFO, 'ALLOCATE', 'DCNDATSUFFIX', 'SASDDI.INGOLD.DATA', 'OLD');
 IF ERROR ^= '0000' THEN DO;
   PUT 'ALLOCATION FAILED FOR CALENDAR LIBRARY';
   CALL EXECUTE ('ENOSAS');
 END;
RUN;
/*
  /* Use PROC SORT to sort the updated calendar by date and time. */
  */
PROC SORT DATA = DCNDATSUFFIX.CALENDAR;
  BY DATE TIME;
RUN;
/*
  /* Use PROC PRINT to print the complete calendar. */
  */
PROC PRINT DATA = DCNDATSUFFIX.CALENDAR;
  ID DATE;
  VAR DAY FREQ TIME LOC EVENT;
  FORMAT DAY WEEKDAY3; TITLE 'Updated Calendar'; FOOTNOTE 'Press END (PF15) to continue';
RUN;
PROC FEDIT DATA = DCNATSUFFIX.CALENDAR;
  SCREEN = DCNATSUFFIX...SCC.CALENDAR;
  FORMAT DAT WEKEDAT3.;
RUN;
/*
/* Delete any "deleted" events from the calendar data set.
/
*/
DATA DCNATSUFFIX..CALENDAR;
  IF EVENT = '' THEN +
RUN;
ISPF Panel Member CALENG

PROC 0;
  ISPFBLR (installation's ISPF table library) +
  ISPFBLR (user's ISPF panel library) +
  DMILLIB (SAS/UTI load library) +
  SASLIB (SAS utility data library) +
  SASMIG (SAS message library) +
  DMISCDB (SAS/DI SAS source code library) +
  SASSOURCE (user's SAS source code library) +
  PROG (SAS/UTI load module name) +
  ENTRY (SAS/UTI load module name) +
  OPTIONS () /* SAS options */ +
  TRACE /* CLIST debugging option */ +
  CONTROL NOFLUSH ENDO;

IF &SYSISPF = ACTIVE THEN ISFEXEC CONTROL DISPLAY ENDSAS;
ELSE DO
  IF &PIEID = 0 THEN SET PIEID =
ELSE SET PIEID = &SUBSTR( &EVALI &RC-1921 :1),AB);
ENDIF

EXIT CODE(LASTCC)
ENDO;

ISPSTART CMD(&SYSICMD USPTLIB &ISFPLIB &DKILLIB +
ENTRY &OPTIONS STRACE);

ENDO;

ELSE DO
  /PIS is not active; let SPF command allocate profile.*/
  SPF CHKKEK 0 THEN SET &ENTRY = "PROG ENTRY OPTIONS STRACE";
ELSE DO
  IF &PIEID = 0 THEN SET PIEID =
ELSE SET PIEID = &SUBSTR( &EVALI &RC-1921 :1),AB);
ENDIF

ENDO;

EXIT CODE(LASTCC)
ENDO;

ISPSTART ERRORS RETURN SET 5011=MEMO

/* Create a suffix that is unique to the PIE pseudo-session */
/* and ISPF logical screen. */
ISPSTART WAIT (SCREEN) SHARED PIAuthor
SET RC = LASTCC
IF &RC = 0 THEN DO
  IF &RC >= 240 THEN SET PIEID = &RC-240
ELSE SET PIEID = &SUBSTR( &EVALI &RC-1921 :1),AB);
ENDIF

ENDO;

ENDO;

ELSE DO
  IF &PIEID = 0 THEN SET PIEID =
ELSE SET PIEID = &SUBSTR( &EVALI &RC-1921 :1),AB);
ENDIF

ENDO;

ENDO;

ENDO;

ENDO;

ENDO;

ENDO;

ENDO;

ENDO;

ENDO;

ENDO;
/* Allocate the libraries needed by the SAS System. */
ALLOC FI(IINPUT ) DA(&INPUT ) SHR ESU1
ALLOC FI(IWORK ) DA(IWORK) SHR ESU1
ALLOC FI(ISASUTL ) DA(ISASUTL ) SHR ESU1
ALLOC FI(ISASKSGS) DA(ISASKSGS) SHR ESU1
ALLOC FI(ISASLIB ) DA(ISASLIB ) SHR ESU1
ALLOC FI(STEMPL) CYL SP(35) SHR ESU1
RECFM(U) BLKSIZE(32760) DSORG(PO)

/* NOTE: The SASCF command automatically adds the LIBRARY */
/* data set to the TASKLIB. Since we aren't using SASCF, we */
/* need to perform the same function (especially since we */
/* don't want to depend on the hardcoded DDname "LIBRARY"; */
/* that could cause interference between this SAS session */
/* and others running in the same address space). */
/* Define the ISPF libraries to ISPF. */
ISPEXEC LIBDEF ISPFLIB DATASET ID(DMILLIB)
ISPEXEC LIBDEF ISPFLIB DATASET ID(ISAMLIB)

/* Define the ISPF libraries to ISPF. */
ISPEXEC SELECT PGM(ISPF) NAMESPACE(IWP) PASSLIB +
        PARM(NEW = 'ENTRY SASLIB "SASLIB DMILLIB = 2 SA = C/+
        L = "LET SUFFIX = 632FF;SINC INPUT(CALM600);" +
        WORK = 'ENTRY SASKSGS = ISASKSGS ISASUTL = ISASUTL +
        NOPRINTINIT NOIMPLMAC NOHAUTOSRC +
        NODETONODE NOGRAPH NOWORK NOPRINT +
        NOSTIMER NOMEMRPT 50PTIONS)
ISPEXEC LIDDEF ISPFLIB
ISPEXEC LIDDEF ISPFLIB
/* Undefine the ISPF libraries. */
ISPEXEC LIBDEF ISPFLIB
ISPEXEC LIBDEF ISPFLIB
/* Unallocate the SAS libraries. */
CONTROL M一看EG
FREE FI(IINPUT WORK ISASLIB LIBRARY ATENP ISASUTL SASKSGS)

SAS, SAS/DMI, and SAS/SHARE are registered trademarks of
SAS Institute Inc., Cary, NC, USA.

1347