

Photos with Text Reports on the Same SAS/AF Screen

Michael Shreve, American Honda Motor Company, Torrance, California

Abstract

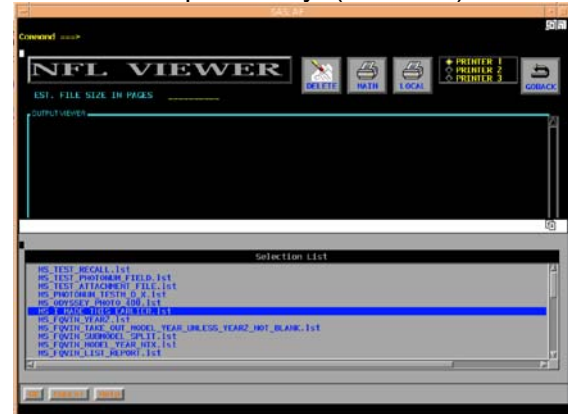
The day of the digital media has arrived! Our field reps are armed with digital cameras. The menu system for text reporting is fully functional, so that users can create reports from the data sent in. But now the field reps are sending in supporting digital photographs (better than a thousand words!), and I hear "Can I see the photos that go with this text, at the same time?" I found a way using the file viewer, Image class and downloaded digital photos, on a version 6 engine, to produce a SAS/AF screen showing the text report and the photos for a selected record. Super-sizing of the photos is available!

Introduction

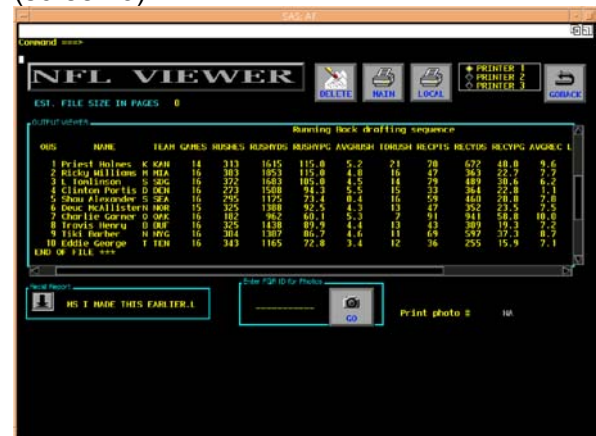
Due to the sensitive nature of the original text and photos in the photo/text viewer, I will be using fictional subject matter to illustrate the concepts and use of this SAS/AF menuing system. It may even be more interesting.

First, the final product shows the magic that the user will see. The impact, of course, is greater when viewed interactively, rather than this static representation.

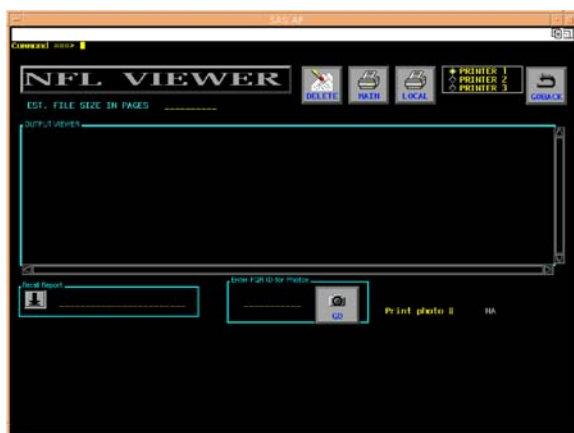
button, and a list appears of reports that the user created previously. (screen 2)



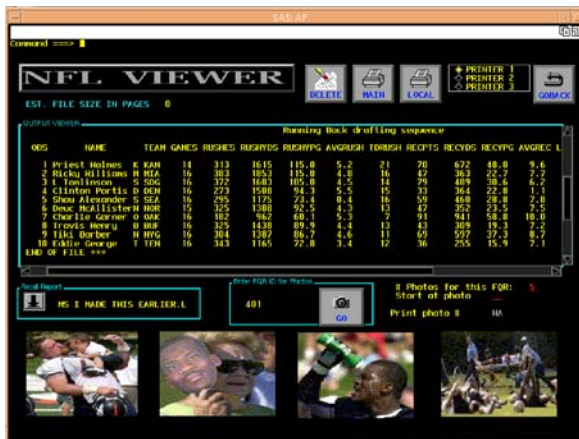
Selecting one puts it in the file viewer. (screen 3)



The report fills the fileviewer. After perusing the data, you decide a photo may shed more light on the subject. To view the photos, you enter the record ID in the box with the camera button, and click on the button. Immediately, or sometime approximating it, the photos associated with this record present themselves on the screen.



This is the menu when it first comes up. (screen 1) Click on the 'Recall Report'



These may clarify details or add information not easily contained in text or statistical variables.

When more than the four photos that fit on the screen are available, a field appears that designates how many photos are available and another text entry field that allows you to tell the screen, "start with photo this number and display the next four photos."

For a closer look at a photo, click on it and it will appear full screen size.

That is the magic from the user viewpoint, and it is fairly impressive to see on screen interactively.

The backbone of the magic is the External File Viewer and the Image classes. These objects are available in the normal pull down menu under Action/make when creating a frame.

External File Viewer

The External File Viewer Class displays the contents of ordinary text files in a FRAME entry. To fill the file viewer with the text, I use

```
call notify
('FILEVIEW', '_SET_FILE_', 'VIEWF', 'F')
where FILEVIEW is the name of the screen
object (so called 'widget name'),
_SET_FILE_ is the 'method name' that says
assign the following file to the object
FILEVIEW, VIEWF is the fileref of the text
file, and F tells _SET_FILE_ that VIEWF is a
fileref instead of a filename.
```

After I've coded the listbox fill and retrieved the selection, I code the following to fill the viewer.

```
ivalue=getitemc(listref,select);
**** retrieve select ****
```

```
Value=trim(left(dequote(ivalue)));
**** clean up ****
```

```
call notify('FILEVIEW', '_SET_FILE_', 'F');
** clear viewer **
```

submit continue;

```
filename viewf "/directory/&value";
```

```
call notify
('FILEVIEW', '_SET_FILE_', 'VIEWF', 'F');
*** fill the viewer ****
```

*** needed to fix external file viewer bug ***;

```
call notify
('FILEVIEW', '_VSCROLL_', 'MAX', 1);
*** max to bottom *** REFRESH;
```

```
call notify
('FILEVIEW', '_VSCROLL_', 'MAX', -1);
***max to top ****
```

Now I have a text report displayed in my File Viewer Object on the screen that I can scroll horizontally and vertically.

Image Class Object

The Image class allows you to display images on your AF screen. Be aware that this class does not support all file types, one unsupported type being the common jpeg. The downside of my menu system is that I had to FTP my jpeg photos to my PC, open them in the Image editor, save them as gif files, and FTP them back to the UNIX box directory. There are software packages that will convert photos in a batch mode, such as 2GIF that you can find by searching in google. Fortunately, I receive a mere handful per week, so that the manual conversion is not a huge burden. File types supported, according to the SAS/AF Software: FRAME Class Dictionary, are BMP, CAT, DIB, EMF, EPSI, GIF, MET, PBM, PCX, PICT, PS, TGA, TIFF, WMF, XBM, XPM, and XWD.

I want to use a fileref for my photo files, so that I can use the following code to fill the screen image object.

```
Call notify('photo1', '_read_fileref_', 'img1');
```

Photo1 is the name given to the screen object, `_read_fileref_` is the method to use a file ref, and `img1` is the fileref for the file where the photo resides.

Other methods can fill or manipulate the object, using file paths, directories, and even the clipboard. See the SAS/AF Software: FRAME Class Dictionary. For instance, to initialize the image objects, I use:

```
Call
notify('photo1', '_READ_FILEPATH_',
'directory/nfl001.gif');
```

This uses the entire path name of a default photo to fill photo1.

Putting it together

In my working menu system, the text file contained a field that displayed how many, if any, photos were associated with that record. Each record also had a unique index. This index was part of the file name for the photos so that the text record and the photo file could be tied together. For instance, in the NFL file, for record 333, the associated photos were `nfl333-1.gif`, `nfl333-2.gif`, `nfl333-3.gif`, etc. So the following code shows how the photos are retrieved, displayed, and enlarged on the screen.

It all starts in the INIT section where I create a pipe to execute the `wc` command (word count) on the Unix directory.

```
Filename photos pipe "wc '/directory/'nfl*";
```

The file looks something like this:

```
[/u/cg51/wuss cg51] wc */u/cg51/wuss/*nfl*
 138   1080   36281 /u/cg51/wuss/nfl400-1.gif
 159   1084   36611 /u/cg51/wuss/nfl400-10.gif
 136   1040   35825 /u/cg51/wuss/nfl400-2.gif
 110    755   28920 /u/cg51/wuss/nfl400-3.gif
 145    952   33125 /u/cg51/wuss/nfl400-4.gif
 112    959   33591 /u/cg51/wuss/nfl400-5.gif
 116    910   32640 /u/cg51/wuss/nfl400-6.gif
 180   1202   37864 /u/cg51/wuss/nfl400-7.gif
 148   1097   35992 /u/cg51/wuss/nfl400-8.gif
 145   1056   37135 /u/cg51/wuss/nfl400-9.gif
.. 1389   10135 347984 total
```

When read as an input file to a datastep, I have what I need to get the number of photos for this record, the file extension, and the full name of every photo file.

Once the frame is up, I choose my report to fill the Viewer, find the record ID I want and type it in the text entry field in the 'enter for photos' box. (The code for filling the File Viewer is at the end of this paper.) Clicking on the 'go' button takes me to the SCL code under the label

PHOTOS:

```
Data fotos(keep=filename);
infile photos pad;
input linesize wordsize bytesize
filename $120.;
length nflen 3;
*****
```

`fqrid` is the screen variable containing the record ID typed in

```
*****;
fqlen=length(&fqrid) + 3;
len='$' || fqlen;
Call symput('end',0);
poutname = putc('nfl' || &fqrid, len);
if index(filename, poutname) > 0; run;
```

The dataset 'fotos' now contains all of the filenames that contain a string 'nfl' plus the ID I typed on the screen. (i.e if I typed 400, all files with `nfl400` in the name.)

To get the exact filename of the photo, I still need the sequence number and the file extension. This code will obtain this from the complete pathname and file name saved in the dataset `fotos`.

```
Data _null_ ;
set fotos end=last;
start=index(filename, '.');
start=start+1;
extens=substr(filename, start, 4);
call symput('ext', trim(extens));
if last then call symput('end', left(_n_));
run;
```

Now I can use `&ext` for the extension and `&end` as the number of photos for my record ID. Using them in a macro and a do loop

allows me to assign every photo a fileref ending in a number so that I can easily call it later in a similar do loop.

```
%macro fotoname;
%do i=1 %to &end;
filename img&i.
"/directory/nfl&nflid-&i.&&ext.";
%end;
%mend;
%fotoname;
```

The bottom line now is that I can call 4 photos and fill in the image objects on the screen with the following do loop; (the objects on the screen are called photo1, photo2, photo3, photo4)

```
Do i=1 to 4;
call notify('photo'||left(put(i,8.)),
_READ_FILE_REF_', 'img'||left(put(i,8.)));
end;
```

Making it big

One cool thing about Image Objects is that if you click on them, SCL will direct you to a label in your code with the same name, if you have one. So I made one. And I put this code in it:

```
Photo1:
fotofram=0;
if startfot ne _blank_ then
snum=inputn(startfot,'8. ');
else snum=inputn(0,'8. ');
```

```
Call
display('photobig.frame',snum,fotofram);
return;
```

This calls another frame that has a large image object in it, and, using the startfot field, figures out which photo to call to fill the frame. The startfot field is necessary when more than 4 photos exist and PHOTO1 object may contain photo-5 instead of photo-1. Note that fotofram is used to add to the photo start number, hence in photo1 it will be set to 0, in photo2 it will be set to 1, etc.

Here is the entire program for PHOTOBIG.FRAME:

```
entry snum fotofram 8;
init:
if snum > 0 then do;
/**snum=putn(startfot,8.);****/;

    call notify('PHOTO','_READ_FILEREf_',
'img'||LEFT(PUT((SNUM+FOTOFRAM),8.
)));
end;
else do;
    call
notify('PHOTO','_READ_FILEREf_',
'img'||LEFT(PUT((FOTOFRAM+1),8.)));
end;
return;
main: return; term: return;
```

To complete the SCL, I had to address other questions like:

What if you have more than 4 photos for one record?

What if you have less than 4 photos for one record?

How does the user know if you have more photos to show?

How do you show the next 4 photos?

The last 4 photos?

I had to use a text entry field (startfot) and some manipulating to answer these questions. I'll let you search out the dirty details in the complete SCL code that I wrote for the text and photo viewer.

Details details

INIT:

```
call notify('numfotos','_hide_');
call notify('numtext','_hide_');
call notify('starttxt','_hide_');
call notify('startfot','_hide_');
IDD = SYSGET('USER');
IDDLC = LOWCASE(IDD);
```

```

/*****
create pipe to unix and pull in current list of
output file names. create a sas data set
containing these names. all pgms need
headers with max size limit.
*****/

```

```
submit continue;
```

```

filename size pipe "wc
'app/sei/&idd/menu_system_output/output/
wus/*";

```

```

data files(keep=linesize wordsize bytesize
foutname pagesize)
total(keep=linesize wordsize bytesize
foutname pagesize);
infile size pad;
input linesize
wordsize
bytesize
filename $120.;

```

```

foutname = substr(filename,46,74);
if foutname =: 'tput' then foutname = 'NO
FILES FOUND';
pagesize = round((linesize/59),1);
if foutname = ' ' then do;
foutname = 'total';
output total;
end;
else output files;
run;
endsubmit;

```

RETURN;

PHOTO1:

```

fotofram=0;
if startfot ne _blank_ then
snum=inputn(startfot,'8. ');
else
snum=inputn(0,'8. ');
call
display('photobig.frame',snum,fotofram);
return;

```

PHOTO2:

```

fotofram=1;
.....same as PHOTO1.....

```

PHOTO3:

```

fotofram=2;
.....same as PHOTO1.....

```

PHOTO4:

```

fotofram=3;
.....same as PHOTO1.....

```

PHOTOS:

```

if fqr = ' ' then do;
msg = 'MUST ENTER NFL ID FOR
PHOTOS';
call send(_SELF_,'_SET_MSG_',Msg);
RETURN;
end;
/*****

```

```

create pipe to unix and pull in all fqr photos
for that id. Get # of photos (end) and
extension (ext). create filerefs img&i for 1 to
&end *****/

```

```

submit continue;
filename photos pipe "wc
'/u/cg51/wuss/'nfl*";

```

```

data fotos(keep=filename);
infile photos pad;
input linesize
wordsize
bytesize
filename $120.;

```

```

length fqr 3;
fqr=length(&fqr)+ 3;
len = '$'||fqr;

```

```

call symput('end',0);
poutname = putc('fqr'||&fqr,len);
pouttrim = trim(poutname);

```

```

IF index(filename,poutname) > 0;
run;

```

```

data _null_ ;
set fotos end=last;
start = index(filename,' ');
start = start + 1;
extens = substr(filename,start,4);
call symput('ext',trim(extens));
if last then call symput('end',left(_n_));
run;
%macro fotoname;

```

```

%do i=1 %to &end;
  filename img&i.
"/u/cg51/wuss/nfl&fqid-&i.&&ext.";
%end;
%mend;
%fotoname;
*****
initialize screen photo images
*****;
endsubmit;
DO i=1 to 4;
  call
notify('PHOTO'||left(put(i,8.)),'_READ_FIL
EPATH_'||'/u/cg51/wus/wussdrill.gif');
  END;
  fotend = symget('end');
length photoid 8;

if modified(fqid) and fqid ne _blank_ then
call notify('startfot','_set_text_');

  IF startfot ne _blank_ THEN DO;

    fotplus = startfot + 3;
    subtract= startfot - 1;
    IF fotplus > fotend then DO;
    DO i=startfot to symgetn('end');
    call notify ('PHOTO'||left(put((i-
subtract),8.)),'_READ_FILEREF_'
'img'||left(put(i,8.)));
    END;
    END;
    ELSE DO;

    DO i=startfot to fotplus;
    call notify('PHOTO'||left(put((i-
subtract),8.)),'_READ_FILEREF_'
'img'| |left(put(i,8.)));
    END;
    END;

END;
ELSE
IF symgetn('end') > 4 THEN DO;

call notify('numfotos','_UNHIDE_');
call notify('numtext','_UNHIDE_');
call notify('starttxt','_UNHIDE_');
call notify('startfot','_UNHIDE_');
call notify('numfotos','_set_text_',fotend);

  DO i=1 to 4;
call notify

```

```

('PHOTO'||left(put(i,8.)),'_READ_FILEREF
_',img'||left(put(i,8.)));
  END;
END;
ELSE DO;

  DO i=1 to symgetn('end');
call notify
('PHOTO'||left(put(i,8.)),'_READ_FILEREF
_',img'||left(put(i,8.)));
  END;

call notify('numfotos','_HIDE_');
call notify('numtext','_HIDE_');
call notify('starttxt','_HIDE_');
call notify('startfot','_HIDE_');

  END;

put 'The End of Photos';
RETURN;

```

Conclusion

That's the basic way I used the File Viewer and Image Class to show data and photos associated with it. SAS has made putting text or photos on a frame simple. Tying them together and using them to solve your users' needs is the challenging and fun part. And there is a Video Player class.....

References

SAS Institute Inc., SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1995, 1155 pp.

Smokin' with Unix Pipes, Kimberly J. LeBouton & Thomas Rice, SUGI 25 Proceedings, SAS Institute Inc., 1999

Contact Information

Michael Shreve
1919 Torrance Blvd.
Ms 500 2S 775
Torrance, CA 90501-2722
(310) 783-2678
Fax: (310) 783-3575
Email: michael_shreve@ahm.honda.com