

End user report generation on the IBM System RISC/6000: Integrating BASE SAS and UNIX shell

David Courtney, Southwestern Bell Telephone
Stephen Graf, Southwestern Bell Telephone

ABSTRACT:

This paper discusses the creation of an end user menuing system utilizing UNIX Shell and BASE SAS. The menuing system allows management to generate and subsequently view/print or email reports across multiple platforms on our network.

SAS code is imbedded in UNIX Shell "HERE" documents which allow the end user to specify the selection criteria of their choice. In addition, UNIX Shell variables are imbedded in the SAS code, which allows a single SAS script to be tailored at run time.

The output of this report can be viewed, printed, or sent as email across the network. This is transparent to the end users, who typically are not aware that they have actually logged in to a remote SAS server.

Problem Overview:

We needed to quickly create an application for the generation and handling of reports, and make it available to more than one hundred users on our network, in different departments, each of whom typically maintains a login on only one of several UNIX servers, and only one of these servers has SAS software installed. We wanted users to be able to run the application on their UNIX server as if it were resident there, the remote login being handled in the background without the user being aware of it. We wanted a menu system to guide the user through the selection process, and we wanted to give the user a friendly way to print or store the report after creation. Most users do not have graphic capability, but use one of several models of 'dumb' terminal, so the application needed to operate in line by line mode, while simulating as closely as possible a full screen application. We wanted to use the power of SAS software for data handling and report generation, but we also wanted to take advantage of the capabilities of UNIX Shell programming to accomplish if-then-else processing in open code, and to make title statement substitution at run time based upon user report selection.

The report options needed to guide the user through the selection of report type and which entity to report on. We have 247 Wire center geographic areas, which are aggregated into 13 units, which are aggregated into 2 districts, which make up our one Market Center. We wanted the user to be able to select any one of these four entity levels, and then choose any entity within that level. We wanted the user to choose from a selection list at each stage, rather than having to remember what to type or refer to a manual each time. This was particularly important, because most users will make only occasional use of this application.

Figure 1 (see also exhibit 6)

UNIX SHELL Script

Queries user for menu selections, builds prompts based on data stored in flat files, stores responses, and passes the information to the SAS code at run time.

SAS command line

Starts SAS, and orders it to receive the batch code that follows as if it were line by line input.

SAS batch code.

Revised at run time according to the value of variables passed from the Shell script.

ENDOFSCRIPT line

Notifies the Shell that we have reached the end of the script that is to be passed to the SAS command.

Our Solution:

We decided to try blending a UNIX Shell script and batch SAS code into a single UNIX Here document, and found that the two languages make a wonderful marriage. We found that we only needed BASE SAS and the UNIX operating system to accomplish all our wants for this application. Building from the bottom up, we started with SAS code to generate the basic report using data step report processing, and then added UNIX Shell script (in the same file, but before the SAS code) to perform run time substitution, and handle part of the user interface tasks. The UNIX Shell script stores variables based on user response, and these are later passed to the SAS code. The basic layout of the file is as illustrated in figure 1.

After we established that the UNIX Shell and SAS code would work together, we created another Shell script to handle the main user menu

and printed or stored output. A 'csqrpt' login was set up for use by users logging in remotely, and then a small C program was written for their local machine to handle the login. As you read the highlights of each of these programs, please refer to the corresponding exhibit on the following pages.

Exhibit 1 -- Sample Session Dialog

This is provided to demonstrate how the application looks to the user. Multiple blank lines that are used to simulate the erasure of one screen and the drawing of the next have been deleted to save space.

Exhibit 2 -- Sample .rhosts file

In order to facilitate the user being able to log in and run a menu, and at the same time passing the info as to who the user is, and the host they came in from, and their default printer, a C program is created on the user's local machine, which works in tandem with the .rhosts file on the SAS server.

On the SAS server, a user id of 'csqrpt' was established as a generic login in order to run the csqmenu. Normally passwords contain 'aging' that requires the user to change the password every 30-90 days. This aging was turned off, (though the password can still be changed by the administrator without affecting the login process).

In order that the users on the remote hosts could log in as csqrpt and run the csqmenu, a .rhosts file was set up in csqrpt's home directory on the SAS server.

Commonly the .rhosts file is user by individual users to allow them in to one host from another, without the need for a password. The .rhosts file however has several options which will allow many users to use the same login. Please note that in a very secure environment, each user should have their own account, and be required to enter a password.

In our particular application, the users needing access to the SAS server were considered "trusted" and with this in mind a .rhosts file was established to allow all users on remote hosts to login as csqrpt.

This is an example of how the csqrpt .rhosts were established
host1 + <- all users on host1 can login as csqrpt
host2 -root <- root is not allowed in from host2
host2 + <- all other users are allowed in from host2

Please refer to your particular UNIX documentation in order to see how to establish a .rhosts file.

Exhibit 3 -- Program code on user's local machine

Since all users on the remote hosts were now authenticated in the .rhosts file, any of these users could login as csqrpt and get a command prompt.

In order to prevent this, a 'C' program was written to place on the remote hosts, in order to alter the users TERM environment variable and provide a means that a user could log in and go directly to the csqmenu as user csqrpt and NOT be able to get a command line prompt.

In the 'C' program, the users TERM environment variable is altered to include the users id, the host they came in from and the local printer designation. It also placed in the TERM environment variable an authentication string that was unique for each host coming in.

Once the Term environment variable was altered, a system call was made to login to the SAS server as csqrpt.

(Side note. . . The TERM environment variable is the only item passed to a remote host when using rlogin)

Exhibit 4 -- Sample .profile file

The csqrpt user's .profile examines the variable TERM and checks for the authentication string. If it is not valid or is missing, the user is immediately logged off. If the remote user and host are authenticated, then the .profile calls csqmenu. After it finishes executing, the user is logged off and the line is hung up.

Exhibit 5 -- UNIX Shell for main menu and output

Lines 0006 to 0009 -- REMOTE is a UNIX shell variable that flags if a user has come in via rlogin or not.

Line 0015 -- TMPFILE is the temporary storage location for the SAS report output until the user views it and decides to scrap it, save it to a file, email it, or print it. This is also referred to in Exhibit 6.

Line 0018 and 0045 -- The output() and routput() are UNIX shell functions. These functions are passed the name of a file to output. The functions present different options to the user as to where the SAS output file should go, (email, print, file).

Lines 0077 to 0095 -- This section of code is the menu that is displayed to the user. The user enters the menu # they desire, which is run through the UNIX shell case statement (a switch). See lines 0098 to 0109.

Exhibit 6 -- Merged UNIX Shell and SAS code

Lines 0028 to 0031 -- LOGFILE sets the file location for standard output that otherwise would be seen on the user's screen when running SAS in line by line mode. This variable is normally set to the UNIX file /dev/null unless the administrator runs the program in debug mode. The UNIX file /dev/null - is basically a sinkhole that swallows output to never be seen again.

Lines 0033 to 0037 and 0140 -- TMPFILE is the temporary storage location for the SAS report output until the user views it and decides to scrap it, save it to a file, email it or print it.

Lines 0083, 0084, 0144, and 0156 -- WCODE is a UNIX shell variable that contains the selection criteria which will be used later as a subset selection variable.

Lines 0085 and 0156 -- WCNAME is a UNIX shell variable that contains the full name of the wire center to later be displayed in the SAS output report title. It is extracted from an ASCII file, based on a lookup of the WCCODE variable.

Lines 0019, 0020, and 0101 -- REP_TYPE is a UNIX shell variable which contains the first argument that was passed by the csqmenu shell program.

The command

SAS -<<ENDOFSCRIPT 1>\$(LOGFILE) 2>&1
is interpreted as follows:

sas - the SAS executable
- - tells SAS to expect input from standard
input (which is normally the keyboard)

<<ENDOFSCRIPT - tells the Shell to take all lines up to the first line including ENDOFSCRIPT (see also line 0198) and perform variable substitution and then feed the lines to SAS.

1>\${LOGFILE} - all output that normally goes to the screen, the UNIX Shell will send instead to the file named by the variable LOGFILE. This output would be the normal SAS log.

2>&1 Tells the Shell to send all errors to the same place that standard output is going, which in the case is the file named by \${LOGFILE}.

Lines 130 to 196 -- This is the SAS code that produces the report. It is a simple data step report which will be self explanatory to most SAS users with the exception of a few special features which will be detailed in the following paragraphs.

Note the variable substitution that takes place at run time on lines 0140, 0144, 0156, 0158, 0168, and 0169.

The proc printto statement on line 0140 sends the output to a file named according to the value of the UNIX Shell variable TMPFILE, which value is assigned by UNIX (see exhibit 5, lines 0014 and 0015). This permits UNIX to control output, and to dispose of the file when finished with it.

Line 0144 -- The dataset is subset based on the value stored in the UNIX Shell variable WCCODE. See also lines 0040 to 0083 for the Shell script that queries the user for this value and verifies its validity.

Lines 0156 and 0158 -- The title statement could have been handled in SAS, but since we were using the UNIX Shell anyway, it was simpler to use Shell variables. See also lines 0102 through 0118.

Lines 0168 and 0169 -- At this point, we wanted to do 'if-then-else' processing in open code, only SAS tech support said it couldn't be done. The Shell substitution provides the functionality we wanted. See also lines 0103 through 0120.

Line 145 -- Since we wanted to do a weekly report, beginning with Sunday and ending with Saturday, and we wanted to see whole weeks regardless of where the end of the month fell, we devised this little formula. Since January 1, 1960 was a Friday, and SAS stores date values as the number of days since January 1, 1960, Subtracting 2 from any date value yields the number of days since January 3, 1960, which was a Sunday. Dividing that number by 7 yields a number whose integer is the number of weeks since the week beginning January 3, 1960. We subtracted 1,768 from that number just to force the count to begin with the first week of December, 1993. See the sample report in exhibit 2.

Exhibit 6A -- Reference files

This exhibit contains two of the fourteen reference files that are used by the wc.sassh file to prompt the user for report selection criteria.

EXHIBIT 1 -- SAMPLE SESSION DIALOG

\$csqmenu

I-10 Reports Weekly Summary Reporting Menu Kansas City Market Center

Market Center	District	Unit (Area-Mgr)	Wirecenter
1. Residence	11. Residence	21. Residence	31. Residence
2. Business	12. Business	22. Business	32. Business
3. Coin	13. Coin	23. Coin	33. Coin
4. Res/Bus	14. Res/Bus	24. Res/Bus	34. Res/Bus
5. Res/Bus/Coin	15. Res/Bus/Coin	25. Res/Bus/Coin	35. Res/Bus/Coin

Enter # of report to generate (return=exit): 34

Enter the 2-3 character wire center code to select (1 = list): 1

Unit	Name
1	KANSAS CITY CENTRAL
2	KANSAS CITY EAST
3	KANSAS CITY MIDTOWN
4	KANSAS CITY NORTH
5	KANSAS CITY OTST
6	KANSAS CITY SOUTH
7	KANSAS CITY SPFLD
8	WICHITA MTRO EAST
9	WICHITA MTRO WEST
10	WICHITA NRTH CTRL
11	WICHITA NRTH EAST
12	WICHITA SOUTHEAST
13	WICHITA SOUTHWEST

Enter the unit number to list: 4

KANSAS CITY NORTH

1 ATK ATCHISON	6 LSK LANSING
2 BAK BASEHOR	7 LVK LEAVENWORTH
3 BSK BONNER SPRINGS	8 PK PLANK KNOB
4 DSK DESOTO	9 SUK SUNSET
5 DXK DREXEL	10 TKG TONGANOXIE

Enter the # of the wire center to select: 8

Working...

Wirecenter: PK PLANK KNOB

1

Installation report rate by week

Business and Residence Orders (Coin Excluded)

(Not official result, for week to week comparisons only)

19:39 Sunday, May 8, 1994

		Field Work			Non-Field Work			Combined		
Week	Begin Date	N, T and C Orders	I-10 Rep orts	Reports Per 100 Orders	N, T and C Orders	I-10 Rep orts	Reports Per 100 Orders	N, T and C Orders	I-10 Rep orts	Reports Per 100 Orders
1	12/01/93	50	2	4.00	248	9	3.63	298	11	3.69
2	12/05/93	61	4	6.56	384	17	4.43	445	21	4.72
3	12/12/93	65	3	4.62	384	12	3.13	449	15	3.34
4	12/20/93	54	4	7.41	384	19	4.95	438	23	5.25
5	12/26/93	66	3	4.55	269	13	4.83	335	16	4.78
6	01/02/94	69	6	8.70	349	12	3.44	418	18	4.31
7	01/10/94	45	3	6.67	581	18	3.10	626	21	3.35
8	01/16/94	64	2	3.13	371	16	4.31	435	18	4.14
9	01/24/94	68	7	10.29	482	9	1.87	550	16	2.91
10	01/30/94	81	6	7.41	485	18	3.71	566	24	4.24
11	02/06/94	50	3	6.00	449	12	2.67	499	15	3.01
12	02/14/94	48	0	0.00	359	22	6.13	407	22	5.41
13	02/21/94	52	3	5.77	501	15	2.99	553	18	3.25
14	02/27/94	87	5	5.75	523	16	3.06	610	21	3.44
15	03/07/94	70	2	2.86	337	13	3.86	407	15	3.69
16	03/14/94	52	7	13.46	252	16	6.35	304	23	7.57
17	03/21/94	49	2	4.08	308	9	2.92	357	11	3.08
18	03/28/94	76	4	5.26	439	6	1.37	515	10	1.94
19	04/03/94	73	7	9.59	412	23	5.58	485	30	6.19
20	04/11/94	62	1	1.61	315	8	2.54	377	9	2.39
21	04/18/94	57	8	14.04	328	16	4.88	385	24	6.23
22	04/25/94	85	3	3.53	430	19	4.42	515	22	4.27
23	05/02/94	89	1	1.12	405	11	2.72	494	12	2.43

Enter S to save to a file, P to print (return=exit):p

Enter printer queue to use (default = 48): 48

Press return to continue:

I-10 Reports Weekly Summary Reporting Menu
Kansas City Market Center

Market Center	District	Unit (Area-Mgr)	Wirecenter
1. Residence	11. Residence	21. Residence	31. Residence
2. Business	12. Business	22. Business	32. Business
3. Coin	13. Coin	23. Coin	33. Coin
4. Res/Bus	14. Res/Bus	24. Res/Bus	34. Res/Bus
5. Res/Bus/Coin	15. Res/Bus/Coin	25. Res/Bus/Coin	35. Res/Bus/Coin

Enter # of report to generate (return=exit):

\$

EXHIBIT 2 -- SAMPLE RHOSTS FILE

```
host1 +
host2 -root
host2 +
```

EXHIBIT 3 -- csqmenu.c -- PROGRAM CODE ON USER'S LOCAL MACHINE

```
#include <stdio.h>

#define SIZE 48
#define RCMD '/usr/bin/rlogin' /* remote login command */
#define RHST 'sashost' /* sas server hosts */
#define RUSR 'csqrpt' /* sas server login id */
#define LHST 'lohost' /* the local host name */
#define AUTH 'z=1-2#' /* authentication (not a password) */

void main(void)
{
    char cmd[SIZE]; /* system call command buffer */
    char user[SIZE]; /* this user on this host */
    char term[SIZE]; /* to hold new altered TERM variable */
    char lpdest[SIZE]; /* printer on this host */

    strcpy(user, getenv("LOGNAME")); /* get user id */
    if ( user[0] == '\0' )
        fprintf(stderr, "failure -> no LOGNAME");

    strcpy(lpdest, getenv("LPDEST")); /* get the lpdest */
    if ( lpdest[0] == '\0' )
    {
        strcpy(lpdest, getenv("PRINTER")); /* or printer */
        if ( lpdest[0] == '\0' )
            strcpy(lpdest, "none"); /* or none */
    }

    /* set TERM to "TERM LOCALHOST USERID PRINTER AUTHENTICATION" */
    sprintf(term, "%s=%s %s %s %s", "TERM", LHST, user, lpdest, AUTH);
    /* place TERM in environment */
    putenv(term);
    /* load command buffer */
    sprintf(cmd, "%s %s -l %s", RCMD, RHST, RUSR);
    /* execute rlogin to SAS server */
    system(cmd);
}
```

EXHIBIT 4 -- SAMPLE .PROFILE

```
trap "" 2 3          #trap BREAK and CONTROL-C
SHELL=/bin/false export SHELL #prevents escape to shell from 'pg' command

#get what host they are REALLY coming in from
INBOUND="/usr/bin/who am i | /usr/bin/awk -F'[][]' ' { print $2 }'"

#log the entry
echo "${INBOUND}\t${USER}\t${TERM}\tdate" >> ${HOME}/userlog 2>&1

set -k ${TERM}        #break apart TERM variable
rhost="${1}"          #arg 1 is the remote host
ruser="${2}"          #arg 2 is the remote user
rdest="${3}"          #arg 3 is the remote printer
rauth="${4}"          #arg 4 is the remote authentication

case "${INBOUND}" in
    #check authentication, logoff if not correct
    host1) if [ "${rauth}" != "z=1-2#" ] ; then
            exit 0
        fi
    ;;
    host2) if [ "${rauth}" != "z=2-1#" ] ; then
            exit 0
        fi
    ;;
    #unknown host - log them off
    *)
        exit 0
    ;;
esac
#change terminal to line by line mode
```

```
TERM=dumb
export TERM rhost ruser rdest
/home/nsa/analysis/bin/csqmenu
trap 2 3
stty 0          #hangup line
exit 0          #logoff
```

EXHIBIT 5 -- csqmenu -- UNIX SHELL FOR FIRST MENU SELECTION

Note: Line numbers are added for reference in the narrative section of this paper. The must be omitted for the code to run properly.

```
0001 #!/bin/sh
0002 # /home/nsa/analysis/bin/csqmenu
0003
0004 # is user on this host, or coming in via rlogin
0005 if [ "${ruser}" ] ; then
0006     REMOTE=/bin/true
0007     CLEAR='\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n'      #dumb term clear
0008 else
0009     REMOTE=/bin/false
0010     CLEAR='tput clear'
0011 fi
0012 export CLEAR
0013
0014 # temporary SAS report output file, used later in SAS scripts
0015 TMPFILE=/tmp/csq$$ export TMPFILE
0016
0017 # Local user file output function
0018 output () {
0019     outfile=$1
0020     if [ -s "${outfile}" ] ; then
0021         pg ${outfile}
0022         echo "Enter S to save to a file, P to print (return=exit): \c"
0023         read ans
0024         ans=${ans:-noans}
0025         case $ans in
0026             s|S) echo "Enter filename to save to (return=exit): \c"
0027                 read savefile
0028                 if [ "${savefile}" ] ; then
0029                     mv -f ${outfile} "${HOME}/${savefile}"
0030                     echo "Output saved to ${HOME}/${file}"
0031                 fi
0032             ;;
0033             p|P) echo "Enter printer to use (default=${PRINTER:-none}): \c"
0034                 read printer
0035                 cat $outfile | ${printer:-${PRINTER}}
0036             ;;
0037             esac
0038             echo "Press return to continue: \c"
0039             read trash
0040         fi
0041         return 0
0042     }
0043
0044 # Remote user report output function
0045 routput () {
0046     outfile=$1
0047     if [ -s "${outfile}" ] ; then
0048         pg ${outfile}
0049         echo "Enter E to email report, P to print (return=exit): \c"
0050         read ans
0051         ans=${ans:-noans}
0052         case $ans in
0053             e|E) echo "Email address format -> userid@host\n"
0054                 echo "Enter the email address to use \c"
0055                 echo "(default=${ruser}@${rhost}): \c"
0056                 read address
0057                 mail ${address:-${ruser}@${rhost}} < ${TMPFILE} > /dev/null 2>&1
0058             ;;
0059             p|P) echo "Printer address format -> host:printer\n"
0060                 echo "Enter the printer address to use \c"
0061                 if [ "${rdest}" = "none" ] ; then
0062                     echo ": \c"
0063                 else
```

```

0064         echo '(default= ${rhost}:${rdest}) : \c'
0065     fi
0066     read printer
0067     rprinter='${rhost}:${rdest}'
0068     cat $outfile | rlp -d '${printer:-${printer}}'
0069     ;;
0070   esac
0071   echo 'Press return to continue: \c'
0072   read trash
0073   fi
0074   return 0
0075 }
0076
0077 # csq report menu
0078 while : ; do
0079   echo $CLEAR
0080   cat - <<EOS
0081
0082
0083       I-10 Reports Weekly Summary Reporting Menu
0084       Kansas City Market Center
0085
0086   Market Center      District      Unit (Area-Mgr)      Wirecenter
0087   1. Residence       11. Residence      21. Residence        31. Residence
0088   2. Business        12. Business        22. Business         32. Business
0089   3. Coin             13. Coin            23. Coin              33. Coin
0090   4. Res/Bus         14. Res/Bus         24. Res/Bus           34. Res/Bus
0091   5. Res/Bus/Coin    15. Res/Bus/Coin    25. Res/Bus/Coin     35. Res/Bus/Coin
0092
0093
0094   EOS
0095   echo 'Enter # of report to generate (return=exit): \c'
0096   read num
0097   num=${num:-noans}
0098   case $num in
0099       {12345}) mk.sassh ${num}          # Market Center Report
0100       ;;
0101       1{12345}) dist.sassh ${num}       # District report
0102       ;;
0103       2{12345}) am.sassh ${num}         # Area Manager reports
0104       ;;
0105       3{12345}) wc.sassh ${num}         # Wire Center reports
0106       ;;
0107       noans) echo $CLEAR ; exit 0
0108       ;;
0109   esac
0110   # Now output the report
0111   ${REMOTE} && routput ${TMPFILE} || output ${TMPFILE}
0112
0113   rm -f ${TMPFILE} > /dev/null 2>&1
0114   done

```

EXHIBIT 6 -- wc.sassh -- MERGED UNIX SHELL AND BASE SAS CODE

Note: Line numbers are added for reference in the narrative section of this paper. The must be omitted for the code to run properly.

```

0001 #!/bin/sh
0002 # /home/nsa/analysis/bin/wc.sassh (WireCenter.SASShell)
0003 # A SAS Shell to generate a report based on a Wirecenter code (WCCODE).
0004 #
0005 # This program first has a loop to request a valid Wirecenter code (WCCODE).
0006 # If the user cannot remember the Wirecenter code they want, they can enter
0007 # L or LIST to break out of the first loop in order to enter a loop that
0008 # displays a list of the Unit Numbers (ascii file) under which Wirecenter
0009 # codes are arranged. Once a valid Wirecenter code has been chosen the
0010 # Wirecenter name (WCNAME) is set by lookup into the same ascii file.
0011 # Once WCCODE and WCNAME have been set, then the REP_TYPE (first argument)
0012 # is looked at to determine the title and select statement to use in the
0013 # SAS code.
0014 #
0015 cd /home/nsa/analysis/bin
0016
0017 # REP_TYPE is first argument. If REP_TYPE is empty, the programmer is running
0018 # from shell, so remind programmer they have to pass a report type

```

```

0019 REP_TYPE=${1}
0020 if [ -z "${REP_TYPE}" ] ; then                                # is REP_TYPE empty?
0021     echo "\nusage: wc.sasash {31|32|33|34|35} [debug]"
0022     exit 0
0023 fi
0024
0025 # if 'debug' is second argument a log file will be kept in /tmp/sas.debug,
0026 # otherwise the log file is sent to /dev/null (nowhere)
0027 if [ "${2}" = "debug" ] ; then
0028     LOGFILE="/tmp/sas.debug"
0029 else
0030     LOGFILE="/dev/null"
0031 fi
0032
0033 # if TMPFILE is NOT set (by csmenu), then the programmer is running this
0034 # from the shell, so set it for them
0035 if [ -z "${TMPFILE}" ] ; then
0036     TMPFILE="/tmp/sas.output"
0037 fi
0038
0039 emesg=""                #error message
0040 while : ; do            #this loop gets wc code
0041     echo "\n$emesg"
0042     echo "Enter the 2-3 character wire center code to select (l=list): \c"
0043     read WCCODE
0044     WCCODE=$(echo ${WCCODE:-none} | tr 'a-z' 'A-Z') #upshift user entry
0045     case $WCCODE in
0046         NONE) exit 0                                #no entry - exit
0047         ;;
0048         [A-Z][A-Z][A-Z] | [A-Z][A-Z][A-Z])
0049             WCNAME=$(grep "^${WCCODE}:" unit.[0-9]* | cut -d: -f3)
0050             if [ -z "${WCNAME}" ] ; then
0051                 emesg="Unknown wire center code ${WCCODE}"
0052                 continue #thru loop
0053             fi
0054             break #valid WCCODE - break from loop
0055             ;;
0056         L|LIST) list=y ; WCCODE=""
0057                 break #they want to see list - break from loop
0058                 ;;
0059         *) emesg="Unknown wire center code ${WCCODE}"
0060            ;;
0061     esac
0062 done
0063
0064 # If the user has entered L or LIST, then they want to see a list of codes.
0065 while : ; do
0066     if [ -z "${list}" ] ; then                                # is variable list empty?
0067         break                                                  # break from loop
0068     fi                                                         # otherwise
0069     echo "\n\n"                                                # display the lists
0070     echo "\n Unit Name\n"
0071     pr -t unit.name -e:4 -o4                                #unit.name is ascii file
0072     echo "\nEnter the unit number to list: \c"
0073     read unitnum
0074     unitnum=${unitnum:-none}
0075     case $unitnum in
0076         {1-9}|{10-3})
0077         unitname=$(grep "^${unitnum}:" unit.name | cut -d: -f2)
0078         echo $CLEAR ; echo "\n\t\t\t${unitname}"
0079         nl -s: unit.$unitnum | pr -t -e:2 -2
0080         echo "\nEnter the # of the wire center to select: \c"
0081         read wcnnum
0082         wcnnum=${wcnnum:-none}
0083         WCCODE=$(nl -s: unit.$unitnum | grep " ${wcnnum}:" | cut -d: -f2)
0084         if [ "${WCCODE}" ] ; then
0085             WCNAME=$(grep "^${WCCODE}:" unit.$unitnum | cut -d: -f2)
0086             break
0087         else
0088             continue
0089         fi
0090     ;;
0091     none) exit 0
0092     ;;
0093     *) continue

```

```

0094      ;;
0095      esac
0096  done
0097
0098  echo 'Working...\c'
0099
0100  #now figure out the report type....
0101  case $REP_TYPE in
0102      31)      title3='Residence Orders (Business and Coin Excluded)';
0103              exec1='fworders+orfw; fwrpts+irfw;'
0104              exec2='nfworder+ornfw; nfwrpts+irnfw;'
0105      ;;
0106      32)      title3='Business Orders (Residence and Coin Excluded)';
0107              exec1='fworders+obfw; fwrpts+ibfw;'
0108              exec2='nfworder+obnfw; nfwrpts+ibnfw;'
0109      ;;
0110      33)      title3='Coin Orders (Business and Residence Excluded)';
0111              exec1='fworders+ocfw; fwrpts+icfw;'
0112              exec2='nfworder+ocnfw; nfwrpts+icnfw;'
0113      ;;
0114      34)      title3='Business and Residence Orders (Coin Excluded)';
0115              exec1='fworders+obfw+orfw; fwrpts+ibfw+irfw;'
0116              exec2='nfworder+obnfw+ornfw; nfwrpts+ibnfw+irnfw;'
0117      ;;
0118      35)      title3='Business, Residence, and Coin combined';
0119              exec1='fworders+obfw+ocfw+orfw; fwrpts+ibfw+icfw+irfw;'
0120              exec2='nfworder+obnfw+ocnfw+ornfw; nfwrpts+ibnfw+icnfw+irnfw;'
0121      ;;
0122  esac
0123
0124  # The Shell now takes the following code, performs variable substitution,
0125  # then feeds the lines to sas as if it were keyboard input until the Shell
0126  # hits the word ENDOSCRIP. The Shell takes any screen output, including
0127  # errors, that SAS displays and sends it to LOGFILE which was defined above.
0128
0129  sas --<<ENDOSCRIP 1>${LOGFILE} 2>&1
0130
0131  /*-----*
0132  | Wire Center I-10 Reports by week |
0133  |-----*
0134  */
0135  libname general '/home/nsa/analysis/general';
0136  libname dirtft '/home/nsa/analysis/dirtft';
0137
0138  options pagesize=60 ls=78 notes source;
0139
0140  proc printto new print='${TMPFILE}';
0141
0142  data templ;
0143      set dirtft.dailyi10;
0144      where date ge '01dec93'd and wccode = '${WCCODE}';
0145      weeknum = int((date-2)/7)-1768;
0146
0147  proc sort data=templ;
0148      by weeknum date;
0149
0150  data _null_;
0151      set templ;
0152      by weeknum date;
0153
0154  file print header=head;
0155
0156      title1 'Wirecenter: ${WCCODE} ${WCNAME}';
0157      title2 'Installation report rate by week';
0158      title3 ${title3}
0159      title4 '(Not official result, for week to week comparisons only)';
0160      title5 '';
0161
0162  if first.weeknum then do;
0163      fworders=0; fwrpts=0; fwrte=0;
0164      nfworder=0; nfwrpts=0; nfwrate=0;
0165      put @5 weeknum 3. @10 date mmddyy8. @;
0166      end;
0167
0168      ${exec1}

```

```

0169      s(exec2)
0170
0171  if last.weeknum then do;
0172      fwrte = (fwrpts/fworders)*100;
0173      nfwrate = (nfwrpts/nfworder)*100;
0174      torders = fworders + nfworder;
0175      trpts = fwrpts + nfwrpts;
0176      trate = (trpts/torders)*100;
0177      put @20 fworders 5. @27 fwrpts 4. @32 fwrte 6.2
0178          @40 nfworder 5. @47 nfwrpts 4. @52 nfwrate 6.2
0179      @60 torders 5. @67 trpts 4. @72 trate 6.2;
0180  end;
0181
0182  return;
0183
0184  head:
0185      put @2 ' ';
0186      put @24 'Field Work' @43 'Non-Field Work' @65 'Combined';
0187      put @21 '-----' @41 '-----'
0188          @61 '-----';
0189      put '          Week | N, T |I-10|Reports| N, T |I-10|Reports| N, T |I-10|Reports';
0190      put '          Begin |and C |Rep |Per 100|and C |Rep |Per 100|and C |Rep |Per 100';
0191      put '          Week Date |Orders|orts|Orders
0192  |Orders|orts|Orders|Orders|orts|orders ';
0193      put '          -----';
0194
0195
0196  run;
0197
0198  ENDOPSCRIPT
0199  exit 0

```

EXHIBIT 6A --REFERENCE FILES FOR USER FRIENDLY SELECTION

file unit.name
1:KANSAS CITY CENTRAL
2:KANSAS CITY EAST
3:KANSAS CITY MIDTOWN
4:KANSAS CITY NORTH
5:KANSAS CITY OTST
6:KANSAS CITY SOUTH
7:KANSAS CITY SPFLD
8:WICHITA MTRO EAST
9:WICHITA MTRO WEST
10:WICHITA NRTH CTRL
11:WICHITA NRTH EAST
12:WICHITA SOUTHEAST
13:WICHITA SOUTHWEST

File unit.4
ATK:ATCHISON
BAK:BASEHOR
BSK:BONNER SPRINGS
DSK:DESOTO
DXK:DREXEL
LSK:LANSING
LVK:LEAVENWORTH
PK:PARKVILLE
SUK:SUNSET
TGK:TONGANOXIE