Paper 100

A Simple SAS/AF® Program To Manage SAS/CONNECT®Sessions

David D. Chapman, U.S. Bureau of the Census

ABSTRACT

The Census Bureau is a large organization with a varied and changing computing environment consisting of DEC clusters with Alpha and VAX computers, all types of UNIX workstations, WINDOWS/NT servers, and desktop PCs connected together with TCP/IP. For an individual connecting to a single remote computer, it is easy to use SAS/CONNECT. However, for individuals using many different computers and operating systems that sometimes require different script files with the same computer, it can be complicated and tedious.

This simple SAS/AF program installed on the toolbar gives the individual user a point-and-click way to sign onto and sign off of different computers without knowing or remembering the IP address or script file name of the computer used. The user only needs to know the common name for the desired computer. For computer support staff the program organizes all information in a SAS catalog that stores all program code and other user defined information and files needed to use SAS/CONNECT.

INTRODUCTION

Large organizations, such as the Census Bureau, have a mix of computer systems. These different computers often differ greatly in speed, ease of use, accessability, available files, and general disk space. Files on one computer have to be converted to be used on another computer. SAS/CONNECT is designed to allow different computer systems with SAS installed on both to operate together. Many times desktop PCs are used to submit SAS programs to a larger and more powerful remote computer where they are executed. Also, SAS/CONNECT is used to access files on a remote computer using remote library services, RLS, or to move SAS data sets, catalogs, and non-SAS files using PROC UPLOAD and PROC DOWNLOAD.

BACKGROUND

For simple situations, when there are one or more remote computers of the same type, SAS/CONNECT is easy to set up and use. Often all the necessary statements can be placed in the autoexec.sas file and initialized at startup. In simple situations, where you always connect to one or more computers of the same type and use the same script file, a standard set of statements can be used. The following code can be placed in the autoexec.sas file and used to set up a connection between three different computers.

.

```
%let EPCV21=168.132.159.65
%let EPCV22=168.132.159.66
%let EPCV23=168.132.159.67
Options comamid=tcp ;
filename rlink
   'c:\sas\connect\saslink\tcpvms.scr';
```

The macro variables EPCV21, EPCV22, and EPCV23 are used to hold the IP addresses. The filename statement tells SAS/CONNECT what script file to use and where it is located. The script file, which is different for each type of computer, handles the dialog between the local and remote computer. These SAS statements are often placed in the autoexec.sas file and are executed at startup so that SAS is always ready to connect to the remote computers. To sign onto a remote computer, the user needs to enter the SAS statement "SIGNON EPCV22;RUN" in the program editor and to submit the SAS statements for execution. This starts the sign on process.

Use of SAS/CONNECT becomes more difficult for both the user and their computer support staff when there are multiple computers from different venders (Digital, Sun, HP, IBM, etc.) running different operating systems (e.g. Windows/NT, OpenVMS, Solaris, Digital Unix, etc.). In many situations, it is desired to run a remote job using the default parameters (e.g. local SASWORK); but, in special situations different parameters are necessary (e.g. a special disk for SASWORK) for different jobs on the same remote computer. This often requires a different version of the basic script file. Also, at times, it is necessary to access multiple computers at the same time.

Many SAS users are analysts — not computer professionals — with limited computer skills in dealing either with hardware or software. This combination often requires a bit of hand-holding by computer staff. Using SAS/CONNECT to standardize and simplify the set up and connection offers the potential to reduce the workload on an often overworked computer support staff.

All of this is complicated in large organizations like the Census Bureau because the computers are constantly changing. The Census Bureau has just moved into a new computer facility resulting in many changes to both computer names and IP addresses. In addition, the Census Bureau is evolving from a computer environment based on OpenVMS to one based on UNIX.

DOMAIN NAME SERVICE (DNS)

To use SAS/CONNECT the TCP/IP communication protocol requires an IP address. An IP address is a number such as 168.132.159.65 that uniquely identifies the computer. The Internet and many large organizations use a domain name service (server) to manage the IP addresses. DNS associates a name with a computer's IP address and users use the name rather than the IP number to reference a computer. The Domain Name Service (DNS) is usually set up and managed by an organization's central IT staff assigns a common name to each IP address. The advantage of this approach is that when the computer changes are made the IT staff corrects IP addresses or add information about a new computers to the DNS table. Since users work with just the name of the computer, most changes of the actual changes associated with name go unnoticed and have no (or minimal) effect on the user. The DNS name functions like the SAS Macro variable that holds the IP address. For Census Bureau SAS® users – and probably many users in a corporate environment – DNS names can be used by SAS/CONNECT as a source of an IP address. One potential problem is that DNS names are not required to be valid SAS 6 name. While most DNS names (at least at the Census Bureau) are valid SAS names, some are not. A DNS name can not be automatically assumed to be valid SAS names. At the Census Bureau, often DNS names are longer than 8 characters (e.g. "ECON7PROD"). Entering a name for a remote computer longer than six characters results in an error message that says " ERROR: Missing or invalid remote session id. Set OPTIONS REMOTE=session_id." This may not be a problem with version 7 which allows for names longer that 8 characters.

PURPOSE OF THE PROGRAM

This program was developed to meet the needs of both SAS users and the computer staff that supports them. For the individual users, the program goals were to: (1) give a point-and-click method to sign on and sign off remote computers, (2) allow easy access to any available computer by knowing only the name, (3) never require users to know either IP address or what script file to use, and (4) display the results of a sign on or sign off to the user in a easily understood way. For computer support staff, the program's goal was to put all the files,

scripts, and parameters needed by the SAS/AF® program in a single location so that any change to the program's files or parameters can be made for all users at one location by one person.

This program was developed using SAS 6.12 on an IBM compatible computer using the Windows/NT 4.0 operating system. No attempt was made to test this program on other computers or operating systems. All computers used in the development were connected with TCP/IP communications software. During development, the local computer was an IBM compatible and the remote computer was a Digital computer running either OpenVMS on a VAX or ALPHA. There is no reason to believe that this program would not work the same on other hardware and operating systems.

HOW THE PROGRAM WORKS

To use this program, the user needs to know the name of the computer that they want to either connect to or disconnect from. To start the program, after starting SAS, the user does a left mouse click on the program icon located on the toolbar. When the icon is selected, the SAS/AF® Frame program starts and the initial screen is displayed. The user can then select the computer to connect to from a list of all available computers. After a computer is selected, a display window shows the computer selected and a brief descriptive note. If it is the correct computer, the user left mouse clicks a control icon at the bottom of the screen to designate whether he wants to sign on, sign off or quit. When the user decides to either sign on or sign off of the selected computer, the program starts the SAS/CONNECT procedure and the initial display window disappears and is replaced by a window showing the type of action taken (sign on or sign off) and the result of the action. From the window the user can tell whether the sign on or sign off was successful or not. The user can than either sign off or on another computer or quit the program.

All program code is contained in a SAS catalog named CONNECT. The directory where the SAS catalog is located must be associated with a LIBNAME called *UTILITY*. All data is contained in the catalog and is used to create a reference data set IP_LIST when the program is started. All script files are also stored in the catalog as source entries. A list of objects used in this SAS/AF program is given in figure 4. The SCL code associated with these objects is given in Appendix A.

STARTING THE PROGRAM

The program is set up to start by a left mouse click on the program icon on the toolbar. The complete toolbar is show below.
The toolbar is the default toolbar with the SAS/ASSIST icon replaced by a

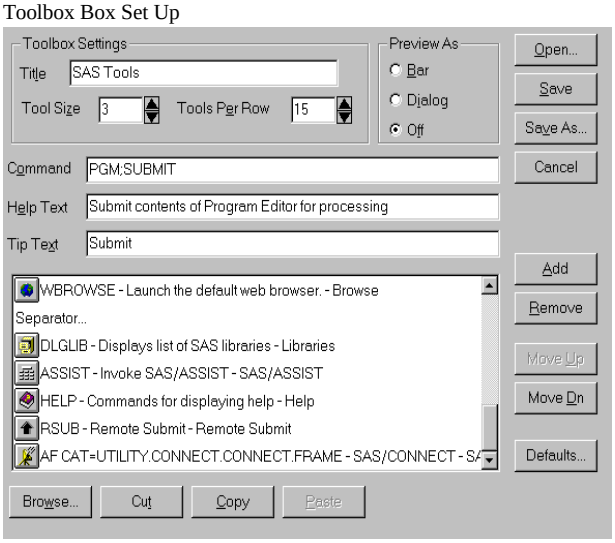
"program" icon and a "remote submit" icon. The program icon starts the SAS/AF®. The program can also be started the same way you would start any SAS/AF® program. Associating the program with an icon on the toolbar gives the user point-and-click access to the program.

The set up of the toolbar is shown below. When the user double clicks on the program icon, a SAS/AF program is started. Before the initial screen is shown, the program does initial housekeeping chores. It checks to see if a parameter data set named IP_LIST exists. If it doesn't exist, it creates it from data stored in the catalog. The parameter data set IP_LIST contains the names of all available computers, their IP addresses, the script file to use, and a brief descriptive note that is displayed to the user. The program also opens the reference file, and initializes variables at start up.

SELECTING THE REMOTE COMPUTER TO USE

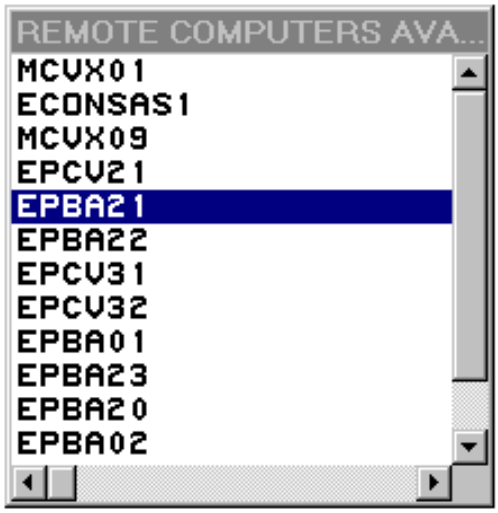
Once the program starts, a computer is selected from the list box object named *PICK*. The list box is showed at the bottom left in Figure 1 and in detail below.
The list box is populated using data from the data set IP_LIST created at start up. When an item is selected, the name of the selected computer and selected information are displayed for user verification in a container box

Toolbox Box Set Up



named *PICKED*. If the user selects the wrong computer, the user can just go back to the list box and make another selection without any adverse impact.

List box



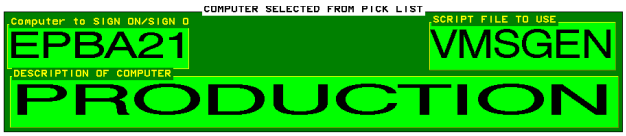
STORING REMOTE COMPUTER INFORMATION

Information on the computers that are available given the initial list box and stored in the SAS data set called IP_LIST. This data set contains the common name of the computer, the script file to be used, and information to be displayed with the computer is selected in the list box. In an earlier version of this program the IP address was also stored in the IP address. Incorporating the use of DNS names no longer requires saving the IP addresses and reduces program maintenance.

The basis for information in the list box in the SAS data set stored in the reference SAS catalog. Each time the program is invoked that the program checks to see whether the data set exists. If it doesn't exist it is recreated from the SAS catalog. This is possible because of the "catalog" option on the file statement. It is the same option that allows the script file to be stored in the reference catalog.

DISPLAYING THE COMPUTER SELECTED

Information about the computer selected using the list box is displayed in the container box *PICKED*. The container box is hidden by the program until the selection is made. The location of the container box in the initial screen is shown in Figure 2 and in detail below.



The container box contains three graphics text boxes used to display the information. The objects are placed in a container box to control background color behind the three objects and to hide all objects with a single program statement. Graphics text boxes were used because the sole purpose was to display text and they allowed the color and font of the text could be controlled.

The information displayed in the graphics text boxes are : (1) Name of the computer (this is the same name displayed in the list box), (2) name of the script file to be used in the SAS/CONNECT session, and (3) a brief description of the computer selected. All of the information comes from the reference data set IP_LIST. This information is displayed in a special box so that the user can verify that they have selected the correct computer before starting the sign on process.

CONNECTING AND DISCONNECTING WITH A REMOTE COMPUTER

The initial selection of the remote computer is displayed in the container box *PICKED*. At this point the user has four choices: (1) sign on the selected computer, (2) sign off the remote computer, (3) pick a different computer for sign on or sign off , or (4) quit the program. These choices are made with a left mouse click on either the SIGN ON, SIGN OFF, or HIDE image icons or the list box.

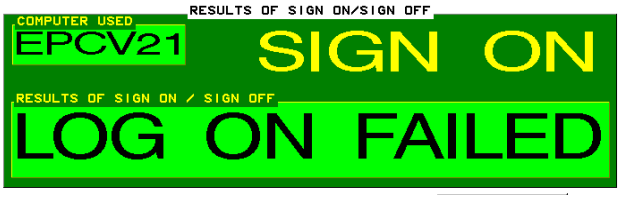
When either the SIGN ON or the SIGN OFF image icon are selected, the *PICKED* container box is hidden and the *USED* container box displayed with information from the SAS/CONNECT session. When the SIGN ON image icon is selected the following code is submitted.

```
SUBMIT CONTINUE
    FILE RLINK CATALOG
        "UTILITY.CONNECT.&SCRIPT.SOURCE";
    OPTIONS COMAMID=TCP REMOTE=&COMPUTER;
    SIGNON ;
ENDSUBMIT
```

The results of the sign on or sign off is captured in a macro variable named *LINKED*. The variable identifies whether a successful sign on or sign off has occurred.

DISPLAYING THE RESULTS OF THE SAS/CONNECT ACTION

The USED container box is shown in Figure 3 and in detail below. The *USED* container box contains three graphics text boxes: (1) *SIGNON*, (2) *SIGNOFF*, and (3) *HIDE*. The *USED* container box has three graphics text boxes.



The computer that is submitted for SIGNON or SIGNOFF is displayed in the *SIGNON* box. The type of SAS/CONNECT action ("SIGN ON" or "SIGN OFF") is displayed in the *TYPE* box. A message (e.g. " SUCCESSFUL LOGON") on the status of the SAS/CONNECT action is displayed in the *RESULTS* box.

A special macro variable named *LINKED* is used to monitor the results. The macro variable identifies when the local and remote computers are linked together. The code to determine whether a remote computer has been successfully "logged on" is given below.

```
CALL SYMPUT('linked','LOG ON FAILED');
submit continue;
   rsubmit &name continue;
      %sysrput linked="SUCCESSFUL LOGON";
   endrsubmit;
endsubmit;
results1=symget('linked');
call notify
   ('RESULTS','_set_text_',RESULTS1);
```

This code sets the macro variable to "LOG ON FAILED" and then connects to the remote computer and sets a special macro variable to "SUCCESSFUL LOG ON." The special macro variable returns to the local computer values set on the remote. The only way the value of the variable *LINKED* can be 'SUCCESSFUL LOG ON" is if there is a good connection between the two computers. This technique was suggested in SAS NOTE -- V6-SCL-A959 dated 29AUG95.

STOPPING THE PROGRAM

The program is stopped with a left mouse click on the HIDE image icon. When this is done, the display is removed from the screen and all files are closed. This image icon is shown at the bottom of figures 1, 2, and 3.

UPDATING THE PROGRAM

When a computer changes its attributes such as the script file to use or a new computer needs to be added, the information must be changed in the basic data set IP_LIST.SD2. This can be done using PROC FSEDIT, VIEWTABLE from the command line, a data step, or any method for changing or adding SAS data set records. When changes are made, it is important that the changes be saved in the SAS catalog. If the old SAS data set is deleted, the program will automatically recreate the data set from information stored in the catalog. Care should be taken that any value for script variable matches a source element in the catalog with the same name. The source element in the catalog must contain a valid script file for the matching remote computer and the communications methods used.

Once the data set is corrected, it must be changed in the catalog by running the following SAS program from the program editor. The SAS code is included in the CONNECT catalog as a SOURCE entry

```
FILENAME NODELIST CATALOG
      'UTILITY.CONNECT.NODELIST.CATAMS';
DATA _NULL_;
LENGTH Name $10 IP $14 SCRIPT $8
     NOTE $39. ;
FILE NODELIST;
SET UTILITY.IP_LIST;
/* NOTE:ALL VARIABLES MUST HAVE DATA ***/;
 PUT NAME IP SCRIPT NOTE;
 RUN;
```

SUBMITTING A SAS JOB TO THE REMOTE COMPUTER

A common way to use SAS/CONNECT is to connect to the appropriate computer, enter code in the program window, and then

execute the “REMOTE SUBMIT” or “RSUB” command from either the command line or a pull down menu. Another alternative is to add the “RSUB” command to the toolbar. All are valid. For me, the toolbar worked well.

SAS, SAS/AF, and SAS/CONNECT are registered trademarks of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration.

CONCLUSION

A simple SAS/AF program can be used to give users an uncomplicated way to access a variety of remote computers and support staff a way to efficiently support it.

DISCLAIMER

This paper reports the results of research and development undertaken by Census Bureau staff. It has undergone a more limited review than official Census Bureau publications. This report is released to inform interested parties of research and to encourage discussion

.AUTHOR CONTACT INFORMATION

David D. Chapman is a mathematical statistician with the Economic Statistical Methods and Programming Division of the U.S. Census Bureau. Any constructive comment -- good or bad -- is always welcomed.

David D. Chapman,
Chief, Frame Development Group
Economic Statistical Methods and Programming Division
U.S. Census Bureau
Washington, D.C. 20233

(301) 457-4904
E-Mail: dchapman@census.gov
FAX: (301) 457-1382

Figure 1: Initial Program Screen

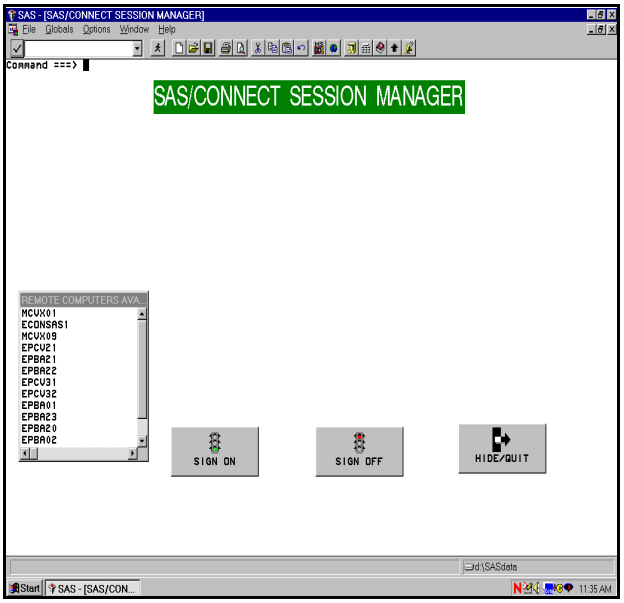


Figure 2: Remote Computer Selected

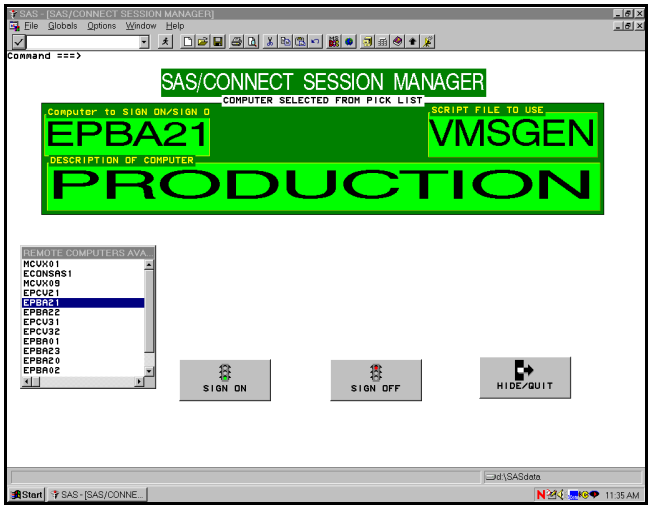


Figure 3: Results of Sign ON/OFF

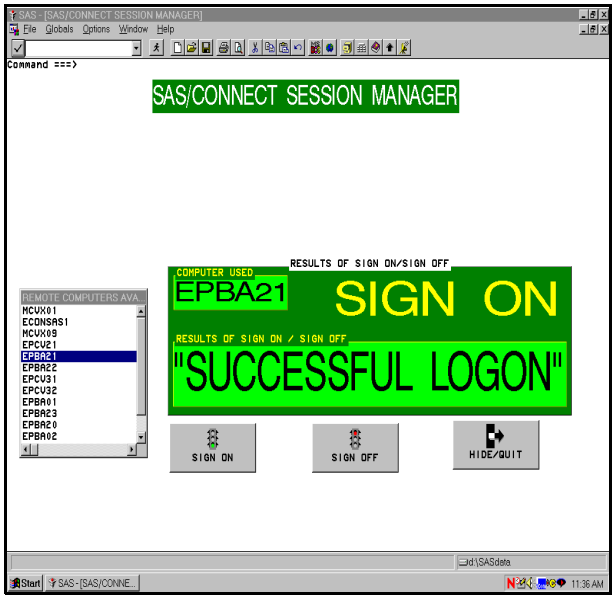


Figure 4: Objects in the SAS Program

OBJECTS		
Name	Type	Parent
FRAME	FRAME	FRAME
TITLE	GRAPHICS TEXT BOX	FRAME
PICK	LIST BOX	FRAME
PICKED	CONTAINER BOX	FRAME
USED	CONTAINER BOX	FRAME
SIGNON	IMAGE ICON	FRAME
SIGNOFF	IMAGE ICON	FRAME
HIDE	GRAPHICS TEXT BOX	FRAME
COMPUTER	GRAPHICS TEXT BOX	PICKED
SCRIPT	GRAPHICS TEXT BOX	PICKED
NOTE	GRAPHICS TEXT BOX	PICKED
SIGNON1	GRAPHICS TEXT BOX	USED
TYPE	GRAPHICS TEXT BOX	USED
RESULTS	GRAPHICS TEXT BOX	USED

APPENDIX A

SCL Program Code (UTILITY.CONNECT.CONNECT.SCL)

```
/*******************************************/;
/* S A S   S A M P L E   L I B R A R Y     */;
/*  NAME: SESSION MANAGE                   */;
/*  TITLE: SAS/CONNECT SESSION MANAGER     */;
/*  PRODUCT: SAS, SAS/AF, SAS/CONNECT      */;
/*  SYSTEM: SAS 6.12 using TCP/IP          */;
/*    DATA: UTILITY.IP_LIST                */;
/* SUPPORTED BY: DAVID D. CHAPMAN          */;
/*     UPDATE:JUNE 30,1998                 */;
/*     UPDATE:DECEMBER 10,1998             */;
/* In December,1998,the program was modified */;
/* so the value of the REMOTE option was the */;
/* computer name contained in the data file  */;
/*******************************************/;
INIT:
    CONTROL ENTER LABEL;
    RC=EXIST('UTILITY.IP_LIST');
    SUBMIT CONTINUE;
       OPTIONS COMAMID = TCP;  RUN;
     ENDSUBMIT;
     IF RC=0 THEN DO;
     SUBMIT CONTINUE;
         OPTIONS COMAMID = TCP;
         FILENAME NODELIST CATALOG
              "UTILITY.CONNECT.NODELIST.CATAMS";
         DATA UTILITY.IP_LIST;
          LENGTH NAME $10. IP $14. SCRIPT $8.
             NOTE $39.  ;
     INFILE NODELIST MISSOVER;
      INPUT
        NAME   $          IP     $
        SCRIPT $          NOTE   $ ;
      RUN;
      ENDSUBMIT;
      END;
    NAME   ='----------';    SCRIPT ='--------';
    NOTE   ='--------------------------------';
    IPOPEN = OPEN ('UTILITY.IP_LIST', 'U');
    CALL SET (IPOPEN); ROW=0;  TEXT='  ' ;
    CALL NOTIFY ('PICKED', '_HIDE_' ,'ALL');
    CALL NOTIFY ('USED'  , '_HIDE_' ,'ALL');
RETURN;
MAIN:
RETURN ;
TERM:
    RC=CLOSE(IPOPEN);
RETURN;
PICK:
  CALL NOTIFY ('USED'  , '_HIDE_' ,'ALL');
  CALL NOTIFY
     ("PICK",'_get_last_sel_',row,issel,text);
       SYSREC=FETCHOBS(IPOPEN, row);
  CALL NOTIFY ('computer','_set_text_',text );
  CALL NOTIFY ('note','_set_text_',    note );
  CALL NOTIFY ('script','_set_text_',  script);
  CALL NOTIFY ('PICKED','_unhide_');
  CALL NOTIFY ('COMPUTER','_unhide_');
  CALL NOTIFY ('SCRIPT','_unhide_');
  CALL NOTIFY ('NOTE','_unhide_');
RETURN;
HIDE:
  CALL NOTIFY ('PICKED', '_HIDE_' ,'ALL');
 CALL NOTIFY ('USED'  , '_HIDE_' ,'ALL');
LINK TERM;    _STATUS_='H';
RETURN;
SIGNON1:
 CALL NOTIFY('SIGNON','_set_text_',COMPUTER);
CALL NOTIFY ('TYPE','_set_text_',"SIGN ON");
  SUBMIT CONTINUE ;                ;
    FILENAME rlink CATALOG
       "UTILITY.CONNECT.&SCRIPT.SOURCE" ;
OPTIONS COMAMID = TCP REMOTE=&NAME; RUN ;
SIGNON ; RUN;
ENDSUBMIT;
LINK RESULTS1;
CALL NOTIFY ('USED'    , '_UNHIDE_' );
CALL NOTIFY ('SIGNON'  , '_UNHIDE_' );
CALL NOTIFY ('RESULTS' , '_UNHIDE_' );
CALL NOTIFY ('TYPE'    , '_UNHIDE_' );
CALL NOTIFY ('PICKED'  , '_HIDE_' ,'ALL');
RETURN;
SIGNOFF:
 CALL NOTIFY('SIGNON','_set_text_',COMPUTER);
CALL NOTIFY ('TYPE','_set_text_',"SIGN OFF");
   SUBMIT CONTINUE ;                ;
     FILENAME rlink CATALOG
     "UTILITY.CONNECT.&SCRIPT.SOURCE" ;
OPTIONS COMAMID = TCP REMOTE=&NAME ;
SIGNOFF ; RUN;
ENDSUBMIT;
LINK RESULTS2;
CALL NOTIFY ('USED'  ,   '_UNHIDE_' );
CALL NOTIFY ('SIGNON'  , '_UNHIDE_' );
CALL NOTIFY ('RESULTS' , '_UNHIDE_' );
CALL NOTIFY ('TYPE'    , '_UNHIDE_' );
CALL NOTIFY ('PICKED', '_HIDE_' ,'ALL');
RETURN;
RESULTS1:
 CALL SYMPUT('linked','LOG ON FAILED');
SUBMIT CONTINUE;
  RSUBMIT &NAME CONTINUE;
     %sysrput linked="SUCCESSFUL LOGON";
  ENDRSUBMIT;
ENDSUBMIT;
RESULTS1=SYMGET('linked');
CALL NOTIFY('RESULTS','_set_text_',RESULTS1);
RETURN;
RESULTS2:
 CALL SYMPUT('linked','SUCESSFUL LOG OFF');
   SUBMIT CONTINUE ;
     RSUBMIT &NAME CONTINUE;
        %SYSRPUT LINKED="LOG OFF FAILED";
     ENDRSUBMIT;
   ENDSUBMIT;
RESULTS2=SYMGET('linked');
CALL NOTIFY('RESULTS','_set_text_',RESULTS2);
RETURN;
```