

## **Client/Server Made Easy: Using SAS/AF® and SAS/CONNECT® SoftwareTogether**

**Tammy Gagliano, SAS Institute Inc.**

### **Introduction**

Whether you call it distributed processing or client/server technology, the concept of being able to access all of the data sources on your network and process those data on the optimal platform is a popular one. This paper will discuss only one aspect of client/server computing: making a request to a remote platform (like MVS) and receiving data at the local platform (like OS/2 or Windows).

Since early 1987, SAS Institute has offered tools which enable you to offload work from a remote system to a local machine for further processing, thus saving valuable and often costly host resources. Downloading data from one platform to another can be accomplished in a variety of ways with a vast set of tools that exist in the marketplace, so why use SAS® software?

The SAS System offers an integrated set of tools which enable an end-user to make data requests with little or no system knowledge. Using SAS/AF software, a menu-driven interface can be built to surround the functionality of SAS/CONNECT software and guide a user through a specific task.

This paper discusses the integration of these tools from an application developer's view and specifically addresses the following:

- providing dynamic access to data that exists on another platform
- designing a transparent, easy-to-use interface that does not require intervention of a data administrator to move the data across platforms
- giving users control over which rows and columns they need to download.

By making the data more accessible, less time will be spent on the logistics of getting the data to the right person and more time can be spent making effective business decisions. This accessibility can easily be accomplished by providing appropriate interfaces that match the skill level of your end-user audience.

### **Key Product Features**

**SAS/CONNECT software** is a cooperative processing product that enables a local SAS session to establish conversations with one or more remote SAS sessions, usually on different machines and operating systems. This gives you access to files, hardware resources, and SAS software located throughout your computing network allowing you to distribute the processing to the most appropriate machine in order to use your computing resources to their best advantage.

The following is a list of the major features that SAS/CONNECT software offers:

- remote processing of applications
- file transfer (uploading and downloading of data)
- local printing or display of graphics processed on the remote host
- the ability to establish connections to multiple remote hosts
- automated processing using the non-interactive mode of execution.

The sample application described in this paper focuses more heavily on the file transfer functionality found in SAS/CONNECT software; however, remote processing of data dictionary tables that reside on the host is also a key feature of this system.

DICTIONARY-tables, part of base SAS software, are special, read-only PROC SQL objects that contain system catalogs for all SAS data sets and other SAS files. These tables are generated at run time to retrieve information about SAS data libraries, SAS data sets, SAS catalogs, SAS system options and external files associated with your SAS session. SAS procedures can access this information by using PROC SQL views stored in the SASHELP data library. The information stored in these remote dictionary tables are queried by the application to provide the local session with a list of valid values to populate selection list windows.

SAS/AF software enables you to create user-friendly prompting systems. When you use SAS/AF software as your application development tool, you have access to all of the features within the SAS System to access, manage, analyze and present your data. Both new and experienced users can access the SAS System without writing any code or knowing the SAS language.

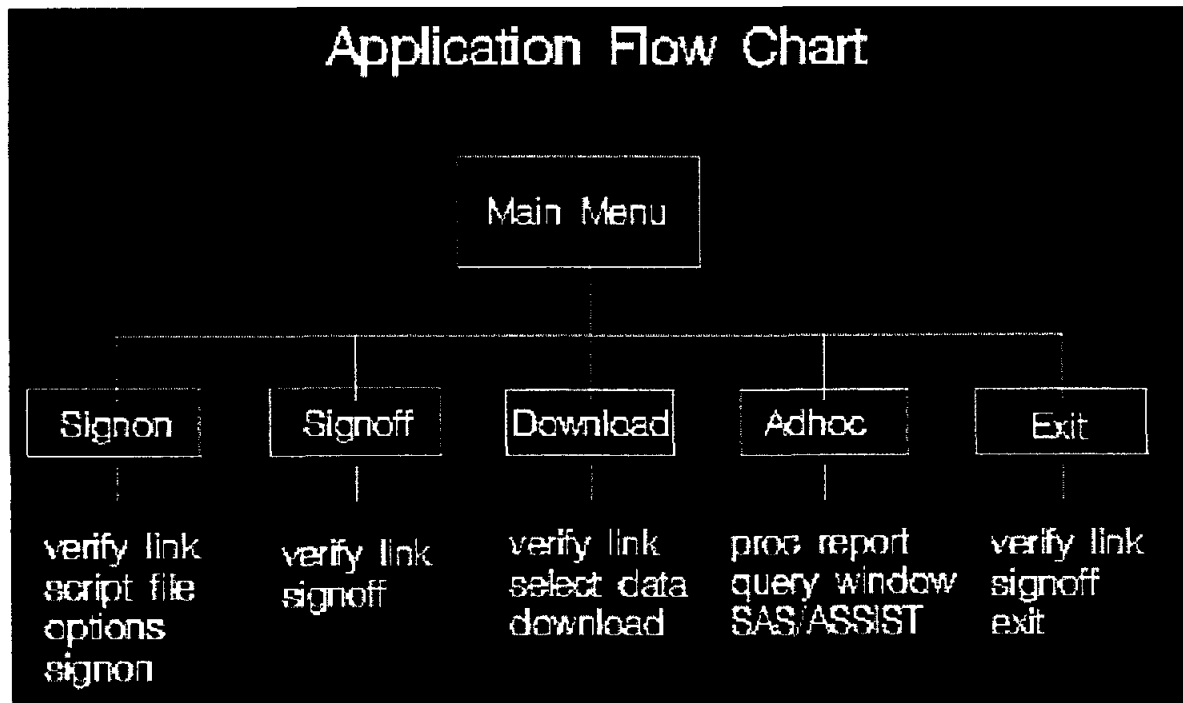
The latest release of SAS/AF software offers a new object-oriented applications development environment. This environment allows for more rapid development of graphical user interfaces (GUIs) using window elements such as graphics, icons, pull-down and pop-up menus, command buttons, scroll bars and sliders.

In addition to the predefined graphical and text objects for window design, SAS/AF software includes Screen Control Language (SCL). SCL combines the syntax of base SAS software with special statements and functions that enable application developers to control the flow of the application based on input from the users. GUI-based, fill-in-the-blank windows can be designed that validate information, insert information into program code, and submit the code for processing.

For the remainder of this paper it is helpful to have a basic understanding of how to build applications with SAS/AF software and how to establish a SAS/CONNECT session between two hosts. This paper was not written to teach, but rather to illustrate how the two products can be used in conjunction with one another. The reference list at the end of this paper will provide detailed information on any of the above product features.

## **Application Design Goals**

As the following flow chart depicts, the goal of this application is to offer the user five main selections.



Below is a brief description of what each selection will allow the user to accomplish.

➤ **SIGNON**

- verify whether or not a connection is currently in progress
- if not, point to the script file to be used and set SAS system options
- signon to the remote host.

➤ **SIGNOFF**

- verify whether or not a connection is currently in progress
- if yes, signoff.

➤ **DOWNLOAD**

- verify whether or not a connection is in progress
- provide the user with a window for selecting the remote table they want to download and, optionally, which columns and rows to include
- download the data.

➤ **ADHOC**

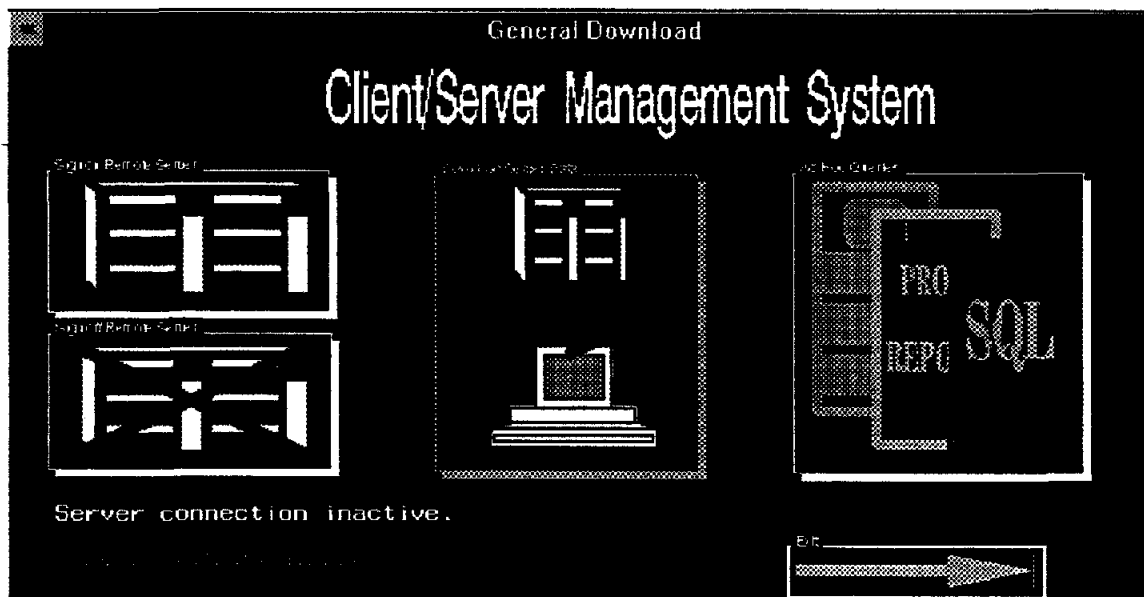
- give users access to report writing tools that exist on the local platform.

➤ **EXIT**

- verify whether or not a connection is in progress
- if yes, signoff
- exit system.

## Development Phase

Clip art and the SAS' graphics editor were used to design this graphical main menu. To make a selection, the user clicks on one of the graphic areas on the screen.



The main menu was built using a new entry type of FRAME. FRAME entries are similar to PROGRAM entries in that they both have associated SCL programs that execute behind the scenes. The choice of whether or not to use PROGRAM or FRAME entries will not be discussed in this paper. The paper, instead, will focus on SCL programming techniques. The following examples are excerpts taken from the SCL programs behind the SIGNON, DOWNLOAD and EXIT menu selections.

### SIGNON SELECTION

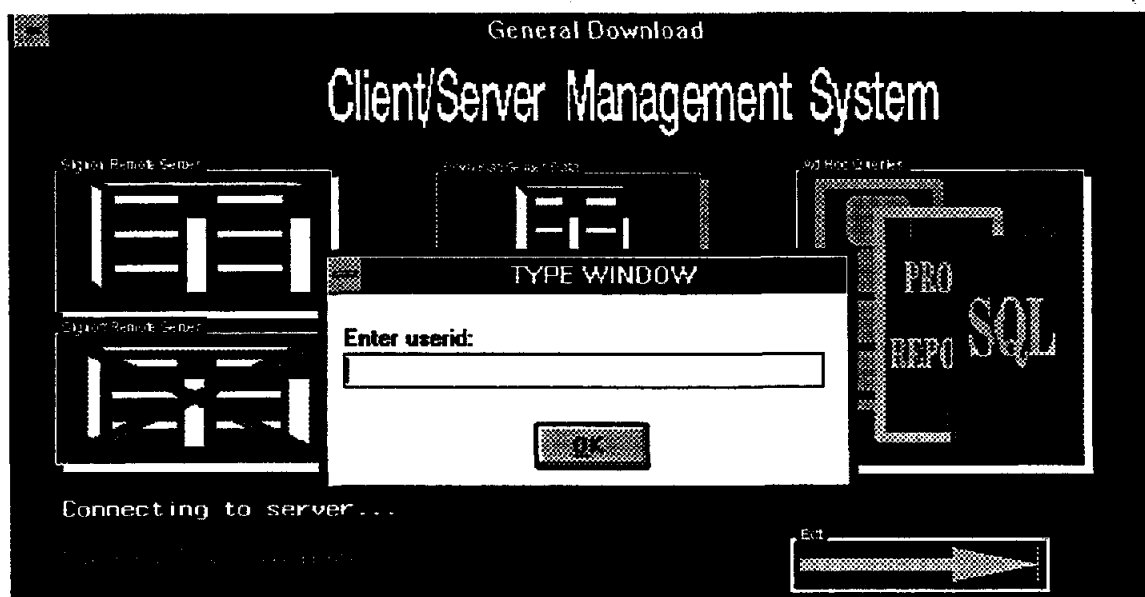
In order to establish a connection between two platforms, there are parameters that must be in place. Using SCL, there are several ways to set these options. The following example uses SCL functions exclusively to establish a link.

```
rc=RLINK('A');
if rc=1 then _MSG_='Session already established.';
else do;
  rc=FILENAME('rlink','c:\sas\connect\saslink\tsolog.scr');
  rc=OPTSETC('remote','A');
  rc=OPTSETC('comamid','ehllapi');
  call EXECCMDI('signon');
end;
```

The RLINK function determines if a link exists between the current SAS session and a remote SAS session. The value supplied as the function argument is the name of the device that is used by your communications software, in this case 'A'. If a value of 1 is returned, use the \_MSG\_ variable to print a message to the user that a session is currently active. Otherwise, begin to establish the connection.

Using the FILENAME function, supply the fileref and complete path name of the script file that you want to use. Sample script files that define the remote host logon process are shipped with SAS/CONNECT software. These script files need to be modified based on your site's logon procedure. In this example, we use the automatic logon script that is provided for the MVS environment. This script file prompts the user during the signon process for their userid and password as shown below.

The OPTSETC function sets the required values for the REMOTE and COMAMID SAS System options. The values supplied for these options vary depending on the communications method that you are using to establish a link at your site. The final step is to issue the SIGNON command. SAS commands can be issued immediately from within your SCL program by using the EXEC CMDI function and supplying the string of SAS commands, separated by semicolons.



Alternatively, you can use a SUBMIT block within SCL to submit standard SAS statements to be processed. Using this technique, the following code will accomplish the same task as the previous example.

```
rc=rlink('A');
if rc=1 then _msg_='Session already established.';
else
  SUBMIT continue;
  filename rlink 'c:\sas\connect\saslink\tsolog.scr';
  options remote=a comamid=ehllapi;
  signon;
  ENDSUBMIT;
```

The SUBMIT statement marks the beginning of a block of SAS statements to be submitted to the SAS System. In this example, when the SUBMIT statement is encountered, SCL collects all the statements between SUBMIT and ENDSUBMIT and passes them immediately to the SAS System for execution.

### **EXIT SELECTION**

The SIGNOFF and EXIT sections are similar with the only exception being to issue the BYE command when the user chooses to exit the system. Both sections first check to see if there is a connection in progress before issuing the SIGNOFF command.

```
rc=rlink('A');  
if rc=1 then call execcmdi('signoff');  
call execcmdi('bye');
```

### **DOWNLOAD SELECTION**

Most of the work in this application takes place in the DOWNLOAD section. Here, you may prefer to take the user to another window, as shown below, where additional selections can be made.

Limit Download

☐ Choose File to Download

☐ Choose Columns to Download (All Columns)

☐ Choose Rows to Download

Limit number of rows to download OBS=

Choose a file to download.

Download GoBack

In this window, the user can select "Choose File to Download" and a list of valid remote data files will appear. SCL provides over 15 functions that will automatically build a selection list window and populate that window based on the arguments that are passed to the function. For example, the DIRLIST function displays a selection list window that lists the members of the current, local SAS data libraries and returns the user's selections to the program.

In our case, the data files are located on the host. Downloading all of the data sets to the local session so that they can show up in a selection list is not a viable option. Downloading a single data set that contains the names of the individual data sets is a more efficient way of populating a selection list. This information can be found in the dictionary tables that are generated automatically when a SAS session is established.

SQL views are used to access the dictionary tables. These views are stored in the SASHELP data library. For example, SASHELP.VMEMBER is a view of the table DICTIONARY.MEMBERS which stores the following information:

```
libname char(8) label='Library Name'
memname char(8) label='Member Name'
memtype char(8) label='Member Type'
engine char(8) label='Engine Name'
index char(8) label='Indexes'
path char(80) label='Path Name'
```

By querying the view that lives on the host, you can download a relatively small data set that contains the libref, data set name and type of all the available remote data sets. This list set is used to supply values to a selection list window. For example,

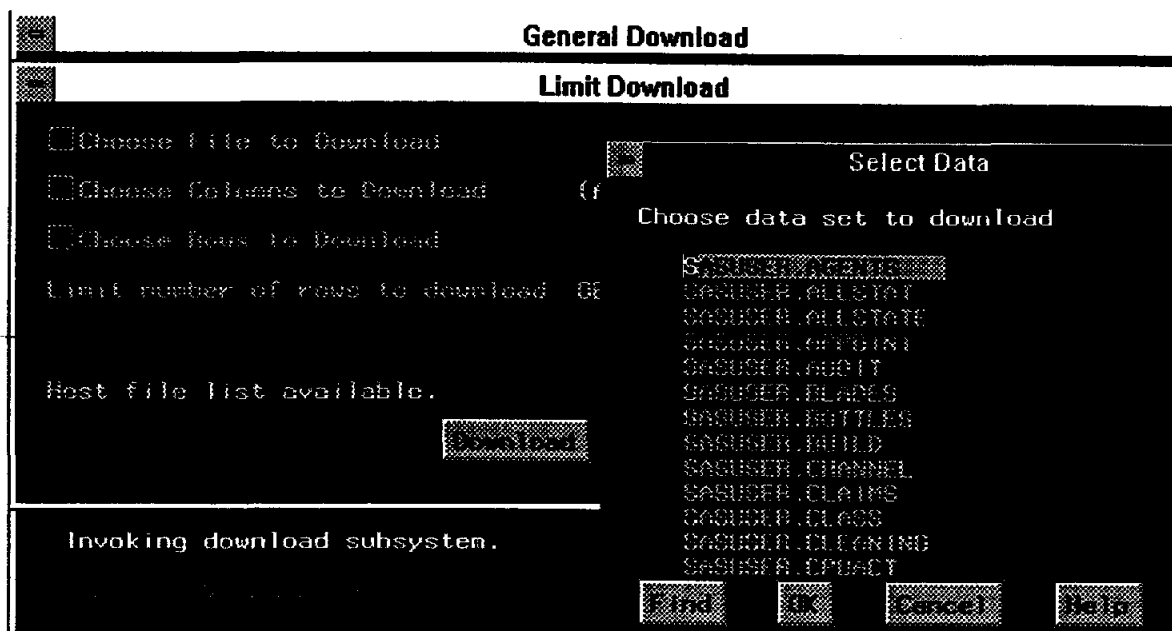
```
submit continue REMOTE;
  proc download data=SASHELP.VMEMBER(keep=libname memname memtype)
    out=DSLST STATUS=NO;
    where libname ^in ('MAPS' 'SASHELP') and upcase(memtype) in
      ('DATA' 'VIEW') and memname ne 'SASPARM';
  run;
endsubmit;
```

Notice the REMOTE option on the SUBMIT statement. This option tells SAS to submit these statements to the remote host for execution. As a result, the DOWNLOAD procedure queries the view called SASHELP.VMEMBER and creates a temporary data set called DSLIST on the local machine. A WHERE clause is used to select or exclude the rows being returned. Setting STATUS=NO on the PROC DOWNLOAD statement suppresses the status window which is usually displayed during the download process. In this part of the application, there is no need for the user to know that data are being transferred. By suppressing the status window and only downloading the information that is needed, the process is more efficient and completely transparent to the user.

One additional step might be added to the above submit block. This step concatenates the libname variable with the memname variable to create a complete two-level data set name. This new variable will then be shown in the selection list window. A short DATA step can accomplish this.

```
/* still part of the submit block above following the download */
data dslst;
  set dslst;
  fullname=trim(libname)||'.'||memname;
run;
```

This local data set is used to supply the values to the DATALISTC function. The DATALISTC function builds and displays a selection list window to the user as displayed in the following window.



```
dsid=OPEN('dslist','r');
dsname=DATALISTC(dsid,'fullname','Please make one selection','Y');
rc=close(dsid);
```

Before the DATALISTC function can read the DSLIST data set, the data set must first be opened for read access. The OPEN function does this by returning a data set identification number which is then passed to the DATALISTC function along with the variable to display in the selection list ( in this example the variable FULLNAME). The third argument supplies a message for the selection list window. The fourth argument closes the window automatically once the user makes a selection. The name of the data set selected will be returned to the variable DSNAME. As a general guideline, all data sets should be closed as soon as they are no longer needed by the application. Use the CLOSE function to accomplish this task.

The same process can be used to populate a window with the list of valid variables to choose from by querying the remote view SASHELP.VCOLUMN. Or the following technique might be even simpler.

```
submit continue remote;
  proc download data=&dsname(obs=0) out=emptyds status=N;
    run;
  endsubmit;
```

In this example, setting the data set option OBS=0 tells SAS to download just the descriptor portion of the remote data set, not the data. This saves resources since the user may not need all of the data and may supply subsetting criteria. The downloaded data set would contain the list of variables that exist in the host data set. The VARLIST function can then be used to read the local data set (EMPTYDS) and populate a variable selection list window.

Optionally, this sample application gives the user selections to "Choose Rows to Download" or "Limit the number of rows to download". The options are limitless. By designing a friendly, menu-driven interface, users can choose to download only the rows and columns that they need. This gives them more control and will hopefully lead to more efficient use of the organization's time and resources.

### **Summary**

This paper barely scratches the surface of the client/server features that SAS/CONNECT software can provide and only lists a few of the powerful SCL functions that can be used to build customized menu systems. Using SAS/AF software to front-end the steps necessary in signing on to a remote host, accessing remote data, subsetting and downloading a more manageable data set, and finally signing off makes the data more accessible to others -- more accessible data with less administrative burden.

### **Futures**

In the client/server data area, the connectivity capabilities will be enhanced to allow for access to remote data sources using LIBNAME references. This will make the access to these sources appear more transparent, and will allow applications to access remote data without making a local disk copy.

In our example above, this will eliminate the need to query the remote dictionary tables to provide the dynamic list of values to the local selection list windows. However, this will not eliminate the need for the data transfer and remote computing services illustrated above.

An example of where all three features are needed is in the case where there is a need to query large, production data files across platforms. In this situation, using the remote library reference (remote library services) to populate the local selection list windows from within a custom menu system makes sense. However, in the event that the queries will be numerous and will be sending large volumes of data back each time, the impact on the network load has to be considered. You may choose to distribute the work load by processing the data on the remote side until it is of a more manageable size (remote computing services). The resulting subset or summary data can then be downloaded to the local host for further analysis (data transfer services), truly being able to efficiently and effectively distribute the work load across your network.

This is just one of many case scenarios that you may be facing in implementing client/server technology within your organization. Each scenario may be different and therefore require a different model. The SAS System's client/server and application development tools are robust enough to back your decisions and implement solutions that will make your organization's strategic data accessible to people regardless of their technical expertise.

### **Reference List**

SAS/CONNECT Software: Usage and Reference, Version 6, First Edition

SAS Technical Report P-223, SAS/CONNECT Software: Changes and Enhancements for the CMS, MVS, and VMS Environments, Release 6.07, Cary NC: SAS Institute Inc.

SAS/AF Software: Usage and Reference, Version 6, First Edition

SAS Screen Control Language: Reference, Version 6, First Edition

SAS/AF Software: FRAME Entry, Usage and Reference, Version 6, First Edition

SAS Technical Report P-222: Changes and Enhancements to Base SAS Software, Release 6.07

Garner, Cheryl (1993), "Techniques for Distributed Application Development", Cary, NC: SAS Institute Inc.

McIntyre, John (1993), "Using the SAS System for Remote Data Access and Distributed Computing", Proceedings of the Eighteenth Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc.

### **Acknowledgments**

I would like to thank Bret Smith for helping design and implement the system used in this paper and in the formal demonstration area.

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

Other brand and product names are registered trademarks or trademarks of their respective companies.