Paper 12

How I Converted a Batch Application System to Client-Server and Lived to Tell About It

Curtis A. Smith, *Defense Contract Audit Agency, La Mirada, Ca.*

ABSTRACT

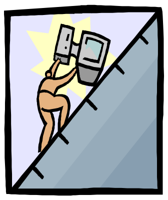
After years of using batch applications, though they were user friendly, my users wanted something GUI, something that runs on their desktop, and something that interchanges with their other MS-Windows applications. So, I set out to convert my batch application system to something my users wanted. Armed with Base SAS, SAS/CONNECT®, SAS/AF®, a TCP/IP connection, Windows 95, my large-scale server, my existing batch code, and the will to succeed, I endeavored to build a user friendly, point-and-click, drag-and-drop, client-server environment. Within this paper I will describe the steps I took, the tools I used, and the lessons I learned. However, I will not discuss how to write SAS code, how to write SCL®, or how to create a SAS/AF FRAME. I will just describe how I used them within my client-server environment.

INTRODUCTION

In the beginning, was SAS code running under IBM MVS® in batch mode. Most of my users did not relish the thought of writing or modifying program code, modifying Job Control Language® (JCL), and submitting batch jobs. To give them something user friendly, I created an application environment that consisted of a PC, a 3270-emulation board and software, a DOS-based micro-to-host macro programming language, and an active connection to an IBM mainframe. To create this environment, I did the following. I used the DOS-based micro-to-host macro programming language to create DOS-based user interfaces. From these, the user selected a desired application to run and responded to prompts to select desired specifications. I stored the user's specifications from the user interfaces to a text file on the local hard drive. Next, I connected the PC to the host IBM mainframe and uploaded the text file to the host. Then, I launched a CLIST that made needed allocations and submitted a batch SAS job. The batch SAS job would read the text file containing the user specifications and pass that information into macro variables. The macro variables would then modify the SAS job, per the user's specification. This all worked quite well, but my users still wanted more.

WHY CONVERT?

My batch environment worked well, back in the old DOS days. Yet it did not have a graphical user interface (GUI), it could not point-and-click, it did not drag-and-drop, - heck, it did not do Windows! IT just not user friendly enough. Another short-coming was that it lacked capability. Output went to a mainframe printer in another building. Output SAS data sets that the user might want to analyze further had to be converted on the mainframe to a text format, then downloaded, then read into another application (like MS-Excel). To make things worse, mainframe computing is expensive, so the batch jobs were costly. What my users wanted and needed was a new environment that would be MS-Windows based, would have lots of user data analysis tools, could get to all the data they needed (wherever it is stored), would have data interchange capability with other MS-Windows applications, and would have a facility to create user applications. However, I had to overcome many technical obstacles before I could give my users what they wanted and needed.



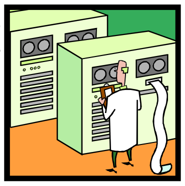
TECHNICAL OBSTACLES

Before I could set up a client-server data analysis environment, there were many complex and complicated technical problems to overcome Accessing and successfully converting enterprise data to information is not necessarily an easy process. However, an easy to use, well designed, well developed, and well implemented user environment can make the process of converting data to information easy for the end-user.

If an enterprise created and stored all of its data in one system, in one place, and in one format, then setting up such an environment might be easy. However, enterprises do not create and store all of their data in one system, nor in one place, nor in one format. Beyond the logistics problems created by having data disbursed, the volume of enterprise data creates enormous technical obstacles. Below I will discuss some of the more prevalent obstacles I had to overcome to implement a user friendly data analysis environment.

MULTIPLE HARDWARE PLATFORMS

Most, if not all, of the data the enterprise generates is likely created and stored electronically. Further, the enterprise's electronic data is likely created and stored on more than one type of computer, or 'hardware platform'. This can be true despite the size of the enterprise. Companies use different hardware platforms (such as IBM mainframes, DEC mainframes, Cray mainframes, HP mid-range computers, Sun Workstations, and PCs) because different hardware platforms offer different advantages. Some are better business machines, others are better scientific machines, and others are better modeling machines.



With the electronic data my users need scattered on multiple hardware platforms, the access issues that I had to overcome was multiplied. That is, logistically, the electronic data is in more than one place. So, whatever hardware, software, and coordination issues I had to know, I had to overcome, I had to do so for each hardware platform I encountered.

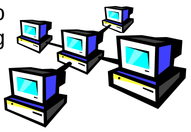
MULTIPLE OPERATING SYSTEMS

Each of the hardware platforms storing the data my users need has its own operating system. In fact, some hardware platforms run more than one operating system. For example, IBM mainframes run MVS/TSO® and VM/CMS® while PCs run DOS, OS2®, and MS-Windows. The different operating systems can have different capabilities. For example, MS-Windows allows file interchange capabilities not found in MVS/TSO.

Having to access multiple operating systems created an obvious problem for me. Without a common tool to access my data, I had to be able to know how to work in more than one operating system. Tasks that are otherwise simple become complicated when a having to know how to do the tasks in different systems. For example, the program editor in VMS is nothing like the program editor in MVS/TSO. Likewise, the file download syntax for MVS/TSO is much different from that for VM/CMS. Further, some tasks that may be simple in one operating system may not be possible in another, or may require a sophisticated work-around.

MULTIPLE ACCESS METHODS

The enterprise's electronic data my users need may be found on multiple hardware platforms under multiple operating systems. The hardware and software methods to access different hardware platforms and operating systems vary. Thus, to access all the electronic data my users need required me to have a variety of access methods.



To access an IBM mainframe (the host) I might use a 3270-adapter circuit board in a PC (the client), or I might use a TCP/IP line connected to an Ethernet circuit board in a PC. To connect to an HP mid-range computer (the host) I might use a modem connected to a PC to dial-up via telecommunications software. Each different access method could have different software to allow the client to act like (emulate) a host terminal. So, I may need to use multiple access methods simultaneously.

These different access methods required me to have expertise in the hardware and software connectivity issues necessary to maintain access to the multiple platforms. Also, the I had to learn more than one software emulation package to access the needed enterprise electronic data.

MULTIPLE FILE FORMATS

On each of the enterprise's hardware platforms and operating systems, many application programs create and store the enterprise's electronic data. Different application programs create and store data in different file formats. Many electronic data files my users need are stored in sequential files of either fixed or variable length records. These files are known as 'flat files'. Other application programs create and store data in structured, relational formats, such as DB2 files and Oracle files. The enterprise uses different file formats because each has its own advantage. Some formats are faster and more efficient for certain functions, while other formats are less expensive. Some file formats are easy to understand. Others are not. Sequential files may contain all the needed data in one file, while relational files may have the needed data spread across several, related files.

The different electronic data files my users need may be in different file formats. The various formats required me to need various methods to access the data and different software tools to convert the data to information. Obviously, this created additional strain – having to know multiple access methods, multiple software tools, and multiple format quirks.

MULTIPLE STORAGE MEDIA

Enterprises create and store most, if not all, of their critical accounting and other data electronically. All that electronic data is stored somewhere. The storage media used can vary – and it can vary by type of data, by hardware platform, and by use. Typically, the type of media used is chosen based on size and performance issues.

Disk storage is a better means to access data for speed and functionality. Access to files stored on disk is fast, multiple users can access files on disk simultaneously, files on disk can be indexed for faster record retrieval, and files on disk can be accessed by interactive applications. However, disk space is expensive. [While we see that PC hard disk drives are getting to the point of being dirt cheap, their capacity pales compared with that of the drives used on larger-scale computers. Remember, while PCs keep getting better, so do larger-scale computers.]

Tape storage is much cheaper than disk storage. Therefore, the enterprise can store much more data on tape. In fact, tape can be used to store files of about any size. However, tape media complicates the data access process and does not allow some functionality that exists with disk drives. This is because tape files must be read sequentially, that is, one bit at a time, moving through the physical tape.

An example of tape limitations is that files stored on tape cannot be indexed as indexing requires files to be read randomly. There are different types of tape media and different tape loading and management systems. At some locations tapes are still loaded manually. This slows the process even more, compared with locations that use robotic tape loading systems. Reading files stored on tape is mush slower than reading files stored on disk. This is so because all the characters in the tape file must be read as compared with a disk file that can be read randomly.

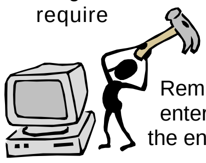
With data on different types of devices, the process of accessing the enterprise's data is complicated. I had to know how to access data on various storage media.

AMOUNT OF DATA

Enterprises have been creating and storing electronic data for decades. The history data my users need is enormous. Thousands of employees, millions of individual time charges each months, millions of material transactions, hundreds of thousands of journals each month. Billions of characters of data. It adds up. The largest of seven company segments my users analyze has a monthly work-in-process cost ledger data file that contains about 20 million records. That is 20 million records each and every month, just for the cost ledger, just for one company segment. Add to that indirect cost data, labor transactions, accounts payable, travel, material, and more - it gets enormous. Even at small enterprises, the data gets big because of the number of sources, the various level of detail, and the endless cycles. All of this data produces computer processing and storage difficulties.

The problem with the amount of data is not just the size of the data files, in and of themselves. To process the data, the user needs working space. Sorting a file, for example, can take between 1 ½ and 2 ½ times the size of the source file. "But we have PC hard drives bigger then 10 GB now, and CD-ROMs hold 650 MB, aren't these big enough?", some may ask. Remember that 20 million record file? It has 174 characters per record. That makes it 3,480,000,000 bytes big. That is more than 3.5 GB! While my users could get a PC hard disk drive big enough to hold one month's work-in-process cost ledger data file, they still might not have enough have room to sort it! Getting the data to the PC can also be quite a chore. Downloading files of only a few million records can literally take many hours.

Summarizing and subsetting the data does reduce the size of the files. However, the reason the enterprise's files are so many and so big is because they maintain a certain level of detail that provides insights into the data. While many of my users' needs do not require that level of detail, many of my users' needs do. So, I cannot simply access summary level data and find my users' data needs satisfied. Remember, if all that data was not useful, the enterprise would not maintain it. And if it is useful to the enterprise, it is useful to my users.



DATABASE CONCEPTS

Implementing a solution for my users required that I shield them from certain concepts and technical aspects that are obstacles to the end users. One such obstacle for many users is trying to understand database concepts. The data retrieval and analysis environment with raw data files, programming, and operating system commands are more difficult to grasp than other electronic forms, like spreadsheets and word processing documents. With a spreadsheet, the user sees the information on the screen the same way it looks on paper. Need another column for totals? Just insert the column and add a formula. Getting one's hands on an electronic spreadsheet is easy, figuratively, that is. Not so with data files.

The enterprise's electronic data is stored somewhere, which was created by some type of application program that the user cannot use to view the data. So, the user must use an independent means to convert the data to information. For example, the user may want to create a tabular report. To do so, there has to be something between the electronic data (that the user cannot touch, figuratively) and the desired output. That something likely is a computer program.

Believe it or not, the relationship between electronic data, computer program, and output is frequently hard to understand. This can be demonstrated by trying to explain that by accessing the data behind the a printed report the user can get more information out of the data. Some users have trouble understanding why the electronic data would be of value when, for example, resorting the data shown on the printed page would not reveal anything new. What some users do not understand is that the printed report is only a template that picks up the data it needs from the electronic data files and selects particular elements and records to format into a report. Some users are believe that the electronic data is nothing more than an electronic image of the printed report.

PROGRAMMING

A significant obstacle that I had to shield my users from is computer programming. To get from data to information, the user needs some type of computer program. Program code is something that someone must write. There are many computer programs for data retrieval and analysis, and the one that is best can depend on what the user wants to accomplish. Also, not all tools are available on all platforms and all operating systems. In fact, few are available for different platforms and operating systems. Better known tools for converting data to information are SAS, DYL 280, MS-Access, MS-Excel.

In some circumstances, using these tools can require the use of other programming languages. For example, to run SAS under MVS/TSO in batch mode (a very typical situation) requires Job Control Language (JCL) to tell the computer what to do. Yet on a DEC/VAX machine, to run SAS you need DEC Control Language (DCL) to tell the computer what to do. And doing some tasks on the PC with MS-Access or MS-Excel will require Visual Basic. Both the data retrieval programming language and the languages needed to run the data retrieval software make the whole thing rather complex.

THE TECHNICAL SOLUTION

USER REQUIREMENTS

My users need to access the enterprise's data no matter where it is located. At many locations the data is too big to data warehouse on the PC, so my users need to access the data from the larger-scale server. Because they need to have a GUI that will permit data interchange with other applications, my solution needs to run on the PC under MS-Windows or other GUI operating system while reading the data from the large-scale server. This requires more than PC to large-scale computer connectivity. Simply connecting the PC to a mainframe with emulation hardware and software does not make a client-server environment. Without a client-server environment my users would have to download the needed data first, then process it on the PC. However, my users' PCs do not have the capacity to hold the enterprise data warehouse. The requirement for a true client-server environment is probably the most important obstacle I needed to overcome.

My users must be able to read the enterprise's electronic data from the PC, no matter where the data is located. This means the software solution must communicate across multiple platforms and understand many data structures and formats. The solution must also provide the means to data warehouse the enterprise's data to make the process of converting data to information easy, efficient, and user friendly.

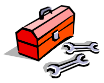
The solution must provide the capability to develop user applications that run from a GUI desktop. Also, the software solution must provide the user with tools to data retrieve, data visualize, data mine, data present, and statistically analyze the data.

Because any one software solution may not satisfy all of my users' needs all of the time, my users must be able to interchange data and results with other applications. My users need the capability to read the data warehouse with other tools, such as MS-Excel, using advanced, simple techniques, like Open DataBase Connectivity (ODBC). They need the ability to dynamically update results into applications like MS-Word, using Dynamic Data Exchange (DDE). My users must also be able to embed results, such as graphs, into applications like MS-Word using Object Linking and Embedding (OLE). And, of course, my users must be able to cut and paste results to other applications.

SOFTWARE SOLUTION

It is not necessary that one software package meet all of my users' requirements. A single software package that does would be the optimum solution. The software needed to meet users' needs might vary by enterprise. For example, users' data needs at a small enterprise may fit handily into a PC (it could happen). In such cases, the software solution might not need a client-server approach. However, in many, if not most, cases users' data warehouse will be too big for the PC.

Having worked as an Information Technology Specialist for ten years, I have seen only one software package that gives me the data warehousing and client-server capabilities I need. That software package is the SAS System. Even better, the SAS System meets all of my users' other requirements for their application environment. The SAS System runs on virtually every hardware and operating system platform (although I have not yet seen a version for the Commodore 64 or the Trash 80) and the SAS System connects between the PC and the other platforms creating a true client-server environment.



The SAS System provides the capabilities I needed to create a user friendly GUI environment using the following modules of the SAS System.

- ★ Client-server connectivity and processing via Base SAS and SAS/CONNECT
- ★ Source data access via SAS/ACCESS®
- ★ Data retrieval via Base SAS and SAS/ASSIST®
- ★ User applications via SAS/AF
- ★ Data Mining via SAS/INSIGHT® and SAS/Spectraview®
- ★ Data Visualization via SAS/INSIGHT, SAS/Spectraview, and SAS/Graph-N-Go (using SAS/GRAPH®)
- ★ Quantitative Methods via SAS/Analyst (using SAS/STAT® and SAS/QC®)
- ★ Graphic Presentation via SAS/INSIGHT and SAS/Graph-N-Go (using SAS/GRAPH)
- ★ Report Presentation via SAS/Enterprise Reporter®
- ★ Interchange capabilities using ODBC, DDE, OLE, and cut and paste

These SAS System components are briefly described below. The SAS System is an integrated suite of software for enterprise-wide information delivery. The functionality of the system is built around the four data-driven tasks common to virtually any application -- data access, data management, data analysis and data presentation. Applications of the SAS System include executive information systems; data entry, retrieval, and management; report writing and graphics; statistical and mathematical analysis; business planning, forecasting, and decision support; operations research and project management; statistical quality improvement; computer performance evaluation; and application development. The SAS Systems is installed at more than 30,000 sites, in 120 countries, and used by more than 3.5 million users.

BASE SAS SOFTWARE The foundation of the SAS System, providing data access, management, analysis, and presentation capabilities within a powerful application development environment. Base SAS software includes a powerful fourth-generation programming language and ready-to-use programs called procedures. These procedures handle data manipulation, information storage and retrieval, basic descriptive analysis, and report writing.

SAS/ACCESS SAS provides direct access to data, despite the file type. SAS accomplishes this through SAS/ACCESS modules. The SAS System then reads the data values just as if they were in a SAS data file.

SAS/AF SAS/AF provides an object-oriented application development environment for building highly interactive and intuitive applications driven by graphical user interfaces. The software supports a variety of objects (push buttons, radio boxes, list boxes, extended tables, icons, sliders, scrollbars, images, etc.), making applications development much easier and faster. Included are ready-to-use procedures for constructing screens, controlling the user's path through an application, transporting screens between operating environments, creating help windows, and developing computer-based training courses.

SAS/ASSIST SAS/ASSIST provides the power of the SAS System at the user's fingertips, despite experience level. The software provides a point-and-click interface that guides the user to access, manage, analyze, and present data. SAS/ASSIST software also serves as an application generator as it builds the underlying code for each completed task. The code is reusable and customizable.

SAS/CONNECT SAS/CONNECT enables clients running the SAS System to establish communications with one or more SAS applications or programs running in remote environments on different platforms. This software allows the client machine, such as a PC with MS-Windows 95, to read data dynamically from the server, such as MVS/TSO.

SAS/FSP SAS/FSP provides easy-to-use facilities for interactive data entry, editing, browsing, and retrieval. The user can create customized data entry screens that simulate business forms; create table-like data entry screens for quick updating; browse external files; and create data entry applications that include cross validation of field values, table lookup capabilities, and other data manipulations. Computed fields, field protection, and hidden fields are also supported.

SAS/GRAPH SAS/GRAPH is an information and presentation color graphics tool that can be used to produce charts, plots, and maps in a variety of colors and patterns. Graphics components can be created, stored in catalogs, retrieved as needed, and combined with other graphics. With the interactive graphics editor, graphics output can be modified interactively without re-running the code.

SAS/INSIGHT SAS/INSIGHT is a dynamic tool for data exploration and analysis. The user can explore data through interactive histograms, box plots, scatter plots, and 3-D rotating plots. The user can examine distributions and explore correlations between variables. Principal components can be calculated and plotted in two or three dimensions to reveal the essential structure of the data. All graphs and analysis are linked, so changes to data in one graph or analysis show immediately in all others. The software supports multiple regression; analysis of variance with classification variables; and the generalized linear model that includes logistic regression, poisson regression, and other models.

SAS/QC SAS/QC for statistical quality improvement offers many tools for establishing statistical quality control and reducing variability. A menu-driven environment is provided for producing Shewhart charts, performing process capability analysis, and histograms. A variety of designs can be constructed, including two-level fractional factorial, response surface, orthogonal array, and mixture designs.

SAS/SPECTRAVIEW SAS/Spectraview is a tool for multidimensional data visualization and modeling. The software enables the user to create, analyze, and modify graphical models representing multidimensional data. Through an interactive, menu-driven interface, the user can create models of volumes, isometric surfaces, cutting planes, and more, and then enlarge, rotate, and move images to explore the data thoroughly.

SAS/STAT SAS/STAT is a comprehensive statistical analysis tool that provides state-of-the-art statistical capabilities for regression analysis, analysis of variance, categorical data analysis, multivariate analysis, cluster analysis, survival analysis, psychometrics analysis, and nonparametric analysis. Also included are techniques for scoring and linear and nonlinear transformations of variables.

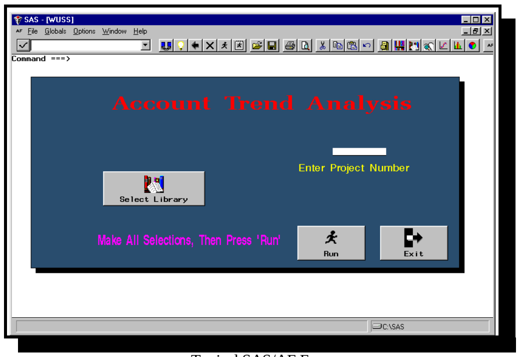
ENTERPRISE REPORTER Enterprise Reporter simplifies the reporting process by enabling users to create richly formatted reports and distribute the results in a variety of formats.
Other software tools that my users have, including the MS-Office suite, have some of these capabilities. My users can do some graphing against PC based data within MS-Excel using the MS-Graph module. They can subset PC based data with MS-Access. However, no other software tool that I know of can do all the things noted above.

HOW TO IMPLEMENT THE SOLUTION

Converting a batch application environment to an interactive client-server SAS System solution requires many of the same essential principles as existed in my batch environment. I use SAS System code to process data and produce the desired output. Anywhere within the code where the user will want to customize the running of the application (such as the use of a WHERE statement), I use macro variables to insert user specifications. I use a user interface to allow my users to make specifications to customize the application processing. And I use a user interface to pass the user's specifications to the SAS System macro variables and to submit the code for processing.

USER INTERFACE

Unlike my old batch environment, rather than using a DOS based scripting language to create a user interface, I use a SAS/AF FRAME. An obvious advantage to the SAS/AF FRAME is that it is MS-Windows based and, therefore, GUI. My users then have a point-and-click, drag-and-drop environment from which to make the necessary selections and specifications. From the SAS/AF FRAME, my users can make selections from radio boxes, list boxes, push buttons, and the like. For me, the developer, SAS/AF offers fast object oriented programming (OOP). With experience, the developer can create interfaces using SAS/AF in minutes.



Typical SAS/AF Frame

PASSING VARIABLES

In my old DOS-based user interfaces, I had to store the user specifications as text information into a text file that SAS would later read into macro variables. However, my SAS/AF interface allows me to store the user specifications and selections directly into Screen Control Language® (SCL®) variables. SCL is a command language used with SAS/AF. Using an SCL statement, I can transfer the contents of an SCL variable into a macro variable. I use another SCL statement to launch my SAS System program code that will use the macro variables to customize the processing.

I now have an MS-Windows based environment with a GUI interface allowing my users to make selections to customize the processing of their applications. From the interface, I launch my SAS System program code for processing. My users need only start the application and respond to the prompts. However, my SAS System applications need data, so I need to access my data warehouse.

CLIENT-SERVER

My data warehouse is too big for PC storage devices. So, I need to access my data warehouse on the server but read it from the PC (my client). I need a true client-server environment that will connect the client and the server in an interactive environment. Such an environment will permit the PC based software to access the data on the server interactively. No downloading here! From the user's point of view, the data could just as well be stored on the PC hard drive.

I can accomplish the client-server environment using SAS/CONNECT and an active TCP/IP connection to the server. SAS/CONNECT comes with the necessary scripts to establish the needed connection to the server. Typically, these scripts need only minor modification to work at specific sites. This type of connection has many benefits. One huge benefit is that I can have the client do the processing, reducing expensive large-scale processing costs, or have the large-scale processor do the work when speed is the ticket. Back to my GUI interface. Within the SCL associated with the SAS/AF FRAME, I can issue commands to establish a connection to the server. While the user must supply a user ID and password, the rest of the client to server connection process can happen behind the scenes.

I now have an MS-Windows based client-server environment with a GUI interface allowing my users to make selections to customize the processing of their applications. From the interface, I launch my SAS System program code for processing. My users need only start the applications and respond to the prompts. When the applications run, my users will be prompted for a user ID and password. The applications will connect to the server and interactively read the data stored in my data warehouse on the server.

INTERCHANGEABILITY

Often, users want to analyze the outcome of a SAS System application further. This can be for many reasons. They may want to use another MS-Windows application to present their final product. Or, they may have needs somewhat different from the output generated by the SAS System application and do not want to write SAS System program code (unbelievable, isn't it!). Therefore, it becomes effective to save the results of the application as a SAS data set on a client storage device. This way, if the user wants to move the resultant SAS data set into any one of many other MS-Windows applications, the user can use Open DataBase Connectivity (ODBC) to pull the data into the non-SAS MS-Windows application. Or, the user can use interactive modules of the SAS System (such as SAS/ASSIST, SAS/INSIGHT, Analyst, Graph-N-Go, or Enterprise Reporter) to analyze further and present the data without having to be connected to the server.

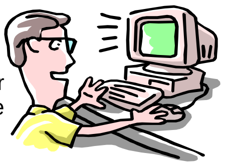
When I create my SAS System applications reading SAS data sets from my server, I can easily store the resultant SAS data set(s) on the server. If I want instead to store the resultant SAS data set(s) on the client, this is a simple matter. I only have to open two SAS data libraries within the application: one for the input data on the server and one for the output data on the client. The SAS System can access data libraries on different platforms simultaneously.

I now have an MS-Windows based client-server environment with a GUI interface allowing my users to make selections to customize the processing of their applications. From the interface, I launch my SAS System program code for processing. My users respond to a few prompts before the applications run. The applications will connect to the server and interactively read the data stored in my data warehouse on the server. Resultant SAS data sets can be stored on client storage media to be further analyzed or interchanged with other applications.

NICE OUTPUT

A huge advantage to converting my batch environment to client-server is that my users' output is no longer directed to a large-scale printer, but instead to a SAS System window. This provides the obvious added capability of being able to print to a local printer, save as a MS-Windows readable file, or cut-and-paste to other MS-Windows applications.

I now have an MS-Windows based client-server environment with a GUI interface allowing my users to make selections to customize the processing of their applications. From the interface, I launch my SAS System program code for processing. My user responds to a few prompts before their applications run. The applications will connect to the server and interactively read the data stored in my data warehouse on the server. Resultant SAS data sets can be stored on the client storage media to be further analyzed or interchanged with other applications. The output from the applications will be routed to a window within the SAS System.


Typical Happy User

LAUNCHING SHORTCUT

On the workstations I develop for my users, we have many applications that run from several software programs, including the SAS System. Expecting my users to know which MS-Windows software program is needed to run the application they want becomes confusing. My users do not necessarily have to know which software program does the work. So, after developing my SAS System applications, I create an MS-Windows desktop shortcut to launch the applications. To do so, I only need a shortcut that runs the SAS System with a particular SAS/AF applications specified. This way, my users will only have to double-click an icon on the desktop or in a folder to launch the desired SAS System application.

CONCLUSION

I now have an MS-Windows based client-server environment with a GUI interface, launched from a desktop icon, that allows my users to make selections to customize the processing of their applications. From the interface, I launch my SAS System program code for processing. My users respond to a few prompts before the applications run. The application will connect to the server and interactively read the data stored in my data warehouse on the server. Resultant SAS data sets can be stored on the client storage media to be further analyzed or interchanged with other applications. The output from the applications will be routed to a window within the SAS System.

REFERENCES

SAS MACRO Variables:
SAS Guide to Macro Processing, Version 6, Second Edition, Chapter 2
SAS Language Reference, Version 6, First Edition, Chapter 20
WHERE Statement:
SAS Language Reference, Version 6, First Edition, Chapter 9
DATA Step:
SAS Language Reference, Version 6, First Edition, Chapter 2

Questions: e-mail the author at casmith@mindspring.com