

The Virginia Higher Education Information System

Dennis B. Catley, Virginia Polytechnic Institute & State University, Blacksburg, VA., 24061
James M. Alessio, Virginia Commonwealth University, Richmond, VA., 23284

Abstract

Staff of the State Council of Higher Education for Virginia (SCHEV), Virginia's higher education coordinating board, and the Office of Institutional Research at Virginia Tech have developed an enterprise information system in a client/server environment. The system, using SAS®, provides access to data and information at the institution, state, and national levels. Data can be accessed from client PC's or mainframe sessions over established state and national networks. The Virginia Educational and Research Network (VERNet) provides the backbone within the state and access to the INTERNET. The model is flexible, providing users the ability to connect to and access data on a variety of computers and operating systems.

Introduction

There has been a continuous dialogue involving the need for access to quality management information and enhanced reporting capabilities for Virginia Tech administrators. A related development activity began at Virginia Tech during the spring of 1993. The goal of this activity was to improve management information by enhancing and expanding data access, and making the interface to the information more intuitive. Benefits realized thus far include time-savings, data quality improvements, synergism, and increased access to both state and federal higher education data.

The initial application was developed to provide institutional researchers and campus

administrators access to higher education data locally or at remote sites. A development team consisting of two SAS programmers and one non-SAS programmer was formed during late spring of 1993. A prototype worthy of internal demonstrations was available within approximately six weeks. The prototype was built using SAS/EIS software.

Development effort was shared between Virginia Tech and SCHEV by porting the application, via SAS/CONNECT to Richmond. The project evolved into a cross-state team development effort. A couple of trips to Richmond for in-depth discussions with the SCHEV staff, and demonstrations of the application supplemented the electronic communication and cooperation.

This paper will trace the key development efforts that have taken place during the past 12 months, with an emphasis on aspects relevant to SAS and to those who may benefit from what has been learned by the presenters. An attempt will be made to introduce some SAS client/server concepts, and segmentation strategies will be discussed. With the intuitive interface being a key to the success of an application, some interface design strategies and tips will be covered. It is not the intent of this paper to explore the vertical depth of possibilities that a SAS/EIS application offers.

The Virginia Higher Education Information System

Virginia Higher Education

The State Council of Higher Education for Virginia (SCHEV) is the state agency responsible for coordinating higher education activities among the more than 80 colleges

and universities in Virginia. SAS is used at SCHEV for storage, analysis, and reporting. The SAS data warehouse that results from the data supplied by institutions is stored on an RS-6000. SCHEV routinely receives extracts of the state-level personnel and finance data, and national higher education data, which are converted to SAS data sets. The result is an extremely robust SAS data warehouse containing more than a decade of valuable Virginia and national higher education data.

The Virginia Tech Perspective

Several implementation strategies surfaced early in the development process at Virginia Tech. They included:

- an attempt to fill informational voids
- accessing SCHEV-based data and integrating the SCHEV SAS data warehouse with on-campus data
- an ability to view Virginia Tech information relative to geographical peer institutions
- exploiting the graphical user interface

The development team hoped to demonstrate to Virginia Tech staff that information retrieval, data analysis and exploration, and access to quality and timely reports could be accomplished without a knowledge of programming, and without knowing that SAS is the underlying software. An early goal was to have minimal text entry. The use of SAS FRAME with its radio boxes, push-buttons, icons, dialogue boxes and scroll bars would keep users from having to enter any text or interact with a command line. The desired interface has been accomplished via four principal SAS/EIS objects.

- GRAPHICS MENU BUILDER
- BLOCK MENU
- LISTMENU
- SAS/AF FRAMES

The GRAPHICS MENU BUILDER uses SAS FRAME technology to construct SAS/EIS objects called DESKTOPs. A DESKTOP object presents users with graphic icons,

graphic text, and push-buttons. On the opening DESKTOP for the Virginia Tech application users are presented with several graphic icons representing information sources including Human Resources, Student, and SCHEV data. Two push-buttons offer users HELP and EXIT. Clicking on an icon will result in the display of a BLOCK MENU, offering the user a choice to Virginia Tech or State-Level data. Clicking on a BLOCK MENU selection then displays a LISTMENU which is a stack of user selections such as reports, listings, and graphs. Users can quickly and easily use a mouse to click through selections to reach the information they desire.

Several design tips related to the three menu structures described above follow. SAS provides an icon library to aid in the development of DESKTOPs. The supplied icon library proved somewhat limited, and for an extensive SAS/EIS application, developers may want to plan on expanding the icon library by importing CGM files. The BUILD environment offers both grid and snap facilities to enhance screen symmetry. When building DESKTOP objects it is recommended that one icon be built and then copied. This ensures that all icons within a DESKTOP are identically sized, so that screens maintain a consistent look and feel. Font and color selection are very important. For example it was found that several text colors were virtually unreadable.

LISTMENU objects can be used as a simple stack of choices, or the pop-up option can be turned on so that the LISTMENU 'pops up' onto the current screen. The pop-up option of the LISTMENU object proved very useful for 'hot-spot' applications. An example of such an application is a map of Virginia depicting the number of extension agents by county. Users can click on a desired county and are presented with a pop-up LISTMENU which offers several additional choices related to the county chosen. Additionally, users can click on the title and be presented with a pop-up selection for viewing the entire extension

population either graphically or in a tabular form.

Access to remote compute services is transparent and dynamic. User intervention is not required to connect to a remote host. At the click of a mouse, a user establishes a remote connection. The user is unaware that a remote server is being accessed. The userid and password necessary for establishing a remote connection are passed in a script file

The first screen that a user sees is a welcome screen that prompts for a userid and password. The WELCOME object in SAS/EIS is used at this point. The WELCOME object provides several capabilities:

- a chance to reference an SCL verification program
- an opportunity to include a company logo on the opening screen
- user supplied welcome window text
- a checkbox to suppress logon window
- a checkbox to suppress welcome window
- a checkbox to verify access to all reports

Capturing the userid and password via macro variables at this point permits their re-use later within the application. The WELCOME object is used throughout the application with the display suppressed, to access a block of SCL code for verification purposes. The &SYSJOBID for instance can be queried within a verification program to determine whether a user has permission to remotely connect to SCHEV. The verification program can also capture important information as to who is seeking the connection, for what subject matter, and so on. If the WELCOME object denies a user permission to establish the remote connection, a message is presented.

If the verification program grants remote connect privileges, a SAS/EIS EXECUTE object similar to the block of SAS code below then establishes the connection.

```
/* ipnode below is the ip address of the */
```

```
/* remote host, and can be assigned */
/* a value with a %let in autoexec.sas */

options remote=ipnode
        comamid=tcp;

/* the tcp2unix.scr file is the SAS supplied */
/* edited to include the passing of userid */
/* and password */

filename rlink
        'd:\sasos2\connect\saslink\tcp2unix.scr';

signon;

run;
```

Necessary libraries can then be allocated on the remote host with the following code. This code can also be stored in a SAS/EIS EXECUTE object, and thus reused throughout the application.

```
rsubmit;

**** remote libname allocations here****

run;

endrsubmit;
```

At this point the process becomes choice-specific. The three steps up to now are generic:

- we have tested for permission to establish a remote connection, using SCL in a verification program
- we have used an RLINK statement embedded within a SAS/EIS EXECUTE object to 'signon' to a remote host.
- we have allocated necessary libraries
- we are then ready to take care of the user request for information
- we can then 'signoff' using the RLINK statement with the signoff replacing signon.

With this modular, object approach, four of the above five steps can be used throughout the application.

The example above uses the remote compute services at SCHEV to execute SAS programs. Output is returned to the local user screen. In terms of the client/server models, this falls into the category of a local client requesting services, in this case both data and CPU, of a remote server (SCHEV RS-6000). There are situations in the Virginia Tech application where the local process makes a request for summary data, and downloads a SAS data set. Processing then resumes locally to complete the task. An example of this strategy involves the mapping of in-state enrollments by academic level. A FRAME entry uses a radio-box to allow users to click on an institution, an academic level, and a year. Clicking on a SUBMIT push-button initiates the process. Instead of output being returned as a result of remote processing, a SAS data set is returned and processing is then completed locally. The developer has the flexibility to design the client and server duties to optimize processing.

Sample code for requesting summary data from a remote site, then resuming local processing would be as follows.

```

rsubmit;

*** necessary SAS code to build your ***
    summary SAS data set goes here

PROC DOWNLOAD DATA=WORK.???
                        OUT=SASUSER.??;
RUN;

endrsubmit;

*** remote processing terminates ***

*** locally processed SAS code here ***

```

Conclusion

The SAS/CONNECT product coupled with application development modules such as SAS/EIS, SAS/AF and FRAME technology, and SCREEN CONTROL LANGUAGE have proved remarkable tools. Important management information is now instantly accessible even from remote servers around the state. An FTP site is available on the SCHEV RS-6000 which allows SAS code to be accessible to and shared among other colleges and universities.

Contacts

Dennis B. Catley
Senior Programmer/Analyst
Institutional Research
129 Smyth Hall
Virginia Tech
Blacksburg, Virginia 24061

phone - (703) 231-4189
fax - (703) 231-7219
irdennis@vtvm1.cc.vt.edu

James M. Alessio
Executive Director
University Enrollment Services
Virginia Commonwealth University
901 West Franklin Street
Richmond, Virginia 23284

phone - (804) 828-0366
fax - (804) 828-1464
alessio@wang.ucc.vcu.edu

SAS, SAS/EIS, SAS/AF, SAS/FSP, and SAS/CONNECT are registered 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.