Abstract

In developing a production data collection environment, flexible access to raw operational data is a highly desirable feature for purposes of process optimization. Using SAS/INTRNET and SAS/CONNECT, we can allow dynamic reporting capability through a standard browser interface even in a cross-platform computing environment. This paper will discuss the methodology involved in rapidly developing such a reporting system in a production environment involving high-volume survey research data.

Through SAS/SHARE, SAS datasets of operational data for a telephone center are accessible to a Sun/Unix server doubling as an application server over a wide area network. An NT server functioning as a broker and housing HTML pages serves to process queries by analysts using a client web browser. Ultimately, SAS processes and returns the HTML output via the ODS utility in SAS Version 8. SAS/INTRNET serves as both the middleware and query builder in such an N-tier configuration.

While lacking the robust functionality of WebAF or a third-party interface such as Javascript or Perl, this approach using SAS/INTRNET and HTML allows for rapid application development in conjunction with serviceable analytic capability. We will discuss obstacles and solutions to finalize dynamic, web-enabled reports in a production environment, as well as balancing end-user needs with developer and environmental constraints.

Introduction

In today’s business world, web-enabled access to corporate data is a highly desirable thing, especially when this data can be used to optimize business processes or improve net profitability. SAS/INTRNET allows you the flexibility to access your SAS data sets from virtually any client through any web server to any data stores in your organization. And SAS can play the predominant role in this whole process without extensive programming. The two main tasks in setting up such systems are the implementation of the SAS/INTRNET back end, allowing a given client the physical access to data sources in any location, and the development of the HTML and SAS code that allows a given user to formulate requests for reports or queries through internet applications.

Setting up the SAS/INTRNET back end requires identifying existing hardware, software, network access and data sources, and then determining how to best integrate them. Having determined that, the actual installation and configuration of all necessary software is a relatively minor step. Developing the HTML pages to formulate data requests and developing SAS code on the server side to process those requests can be a straightforward task provided that the desired functionality is simple. Complex coding is also possible if desired. In any event, SAS provides many web-formatting utilities to reduce the coding burden.

Our pre-generated reports for in-house review are produced on a daily basis and were initially sent via email to all
parties, onsite and remote. These reports consisted of relevant crosstabs on various fields (call result, case result, number of calls, failure reason, interviewer, time zone, and hour of the day) sorted by day, week, and month using preset cutoffs. Our goal was to give the users the same report functionality, but with added choice of level, start date, and end date incorporated into their dynamic report.

The purpose of this paper is to demonstrate that functional web applications can be developed by individuals with basic HTML and SAS programming experience in a relatively brief timeframe. With a proper design approach, a basic overview of how SAS/INTRNET is deployed, and at least a basic knowledge of HTML and SAS coding, anyone can quickly and effectively develop web applications enabling anybody within the organization to easily access corporate SAS data.

Planning and Configuration of web infrastructure

Setting up a SAS/INTRNET web application is not an overly complicated process, thanks in part to the simplified setup wizards for the web server and the default programs and scripts for the SAS and web servers. However, it is important to identify the existing hardware and network configuration available to you and then select exactly how you will configure your various servers and data stores. SAS/INTRNET works seamlessly among various platforms.

A good starting point is to identify your SAS server. In many organizations, this may be an enterprise-level server with a full SAS installation that acts as a primary processor for SAS jobs, a repository for most corporate SAS data, and a server for remote SAS sessions. For this example, our SAS server is a Sun 5000E enterprise server running Solaris 2.6.1 (UNIX) and SAS version 8.01. It is this physical server that will run the SAS application server - a service running on a SAS server that facilitates interaction between queries from web (HTML) pages and the back-end SAS datasets.

Next, identify your web server. Normally, a web administrator will be responsible for the installation and administration of the SAS/INTRNET broker files, so it is often a matter of that person's discretion as to where SAS/INTRNET is configured. In our case, we were required to use a small NT server running MS Internet Information Services (IIS) web server. It is possible, and in some cases necessary, to use your SAS server as your web server as well. However, we kept our web server and application server separate. While there are some obvious administrative advantages to having a single web server/application server, it may be necessary to have them separate. SAS/INTRNET handles either configuration with no problems.

Lastly, identify your data location. In many cases, this will be the same as your SAS server. But in some cases it will not, and you will need to account for that one way or another. Our operational production data are stored on an NT server running SAS 6.12 in a remote location within our firewall. There are many ways to handle this situation, and the optimal way to handle such a configuration depends upon the various resources within the organization. One way to retrieve the remote SAS data is to set up a SAS/SHARE server or establish a SAS/CONNECT session through the application server. Another way would be to mount the NT server from the UNIX server using a third-party product. A third way would be to run a batch job to upload the SAS data from the NT server.
periodically. A fourth way would be to remove the UNIX application server from the picture altogether and make the remote NT server the application server. Each of these methodologies has distinct advantages and disadvantages regarding ease of administration, robustness, and processing efficiency. For purposes of proof of concept of a 4-tier configuration, we chose to access the data remotely by making use of a remote SAS/SHARE service on the NT server.

Once you have identified all of the components of your infrastructure and how you will configure your SAS/INTRNET environment, you can begin installing the SAS/INTRNET software and other components. For this paper, it will be assumed that SAS is already installed on the selected SAS server and the remote data server, and that web hosting software, such as Apache or IIS, is already installed on the web server. Now the SAS Common Gateway Interface (CGI) Tools must be installed and configured on the selected web server. A CGI program is simply one that allows your Internet server to run external files to perform a specific function; e.g. the SAS broker executables and configuration files. This is a very thin installation which requires no service to be running; rather, the broker executable is simply invoked every time a request is made. Some minor tweaks should be made to the configuration file to make it site specific, including selecting a port on the SAS server which will run the application server. This web server is also the physical device which will store the HTML code used to formulate queries or reports.

**Sample broker.cfg**

```
#Debug 2
#DebugMask 32767
#Fieldwidth 200
#PrependFile
"/usr/local/sas/IntrNet/broker/header.html"
#AppendFile
"/usr/local/sas/IntrNet/broker/footer.html"
#DefaultService default
#LoadManager
loadmgr.yourcomp.com:5555
#localAddress 111.222.333.444
#Export Environmental variables to SAS
#Export GATEWAY_INTERFACE _GATEWAY
#Set variables used by SAS/IntrNet
samples and applications
#set_mrvimg
"/sasweb/IntrNet8/MRV/images"

SocketService default "Reuse existing session"
ServiceDescription "Pages reference this generic server when +
they don't care which service is used."
ServiceAdmin "SAS/IntrNet
administrator"
ServiceAdminMail
"joe_the_admin@census.gov"
Server cmosun5.cmo.census.gov
port 5001
#Do not use following option for
servers before V8.1
FullDuplex True
```

Once the CGI files have been installed, the service type must be selected and configured on the SAS server. There are three types of services to choose from: socket, launch, and pool. Each has its own advantages and disadvantages. Socket services are the simplest to implement and most widely supported types of services, but they lack dynamic scaling to meet increased workloads and cannot be efficiently started and stopped. Launch services will start a separate application server every time there is a client request, so that multiple requests do not slow down or disrupt one another, providing for a more robust processing environment. But launch services are not supported by all platforms, and must be run
on the same system as the web server, rendering the unavailable in certain configurations. Pool services are the most robust and efficient of the three types, but are complex to configure.

In keeping with the idea of setting up our web reporting tool in the simplest way given the existing infrastructure, we chose to install a socket service, with the intention of later changing to a pool service once user workload increased. Setting up this service will give you a configuration file on your web broker, which you can modify as necessary to fit your site. Typically the default services are adequate, but you can specify any server, service name and port to suit your needs.

Below is a sample program to set up the application server as a socket service.

```
/********************************************
* This file starts an Application Server for
* the default service on port 5001.
********************************************/

/********************************************
The ifcexist macro is defined so that catalogs
can be conditionally included in a proglibs
statement.
********************************************/

%macro ifcexist(catname);
  %local catname;
  %if %sysfunc(cexist(&catname)) %then
    &catname;
  %mend;

/* Optional code to use SAS/CONNECT and
Remote Library Services
%let myhost=acsntsn1.cmo.census.gov;
  options remote=myhost comamid=tcp;
  filename rlink
'/SASROOT/connect/saslink/tcpwin1.scr';
  signon;
*/
proc appsrv port=5001 unsafe='&';
  allocate library tfu_ctrl
'd:\blaisedata\tfudata\proddata\sasdata\ctrldump' server=myhost.server1 access=readonly;
  allocate library tfu_blez
'd:\blaisedata\tfudata\proddata\sasdata\blezdump' server=myhost.server1 access=readonly;
  allocate file tfuprogs
'/ssroot1/sasprog';
  allocate file sample
'/SASROOT/samples/internet';
  allocate library samplib
'/SASROOT/samples/internet' access=readonly;

allocate library sampdat
'/SASROOT/samples/internet' access=readonly;
allocate library tmplib
'/usr/local/sas8/internet/default/tmp';
allocate file logfile
'/SASROOT/internet/default/logs/ka_%p.log';
proglibs sample samplib tfuprogs
%ifcexist(sashelp.webeis) sashelp.webprog;
adminlibs sashelp.webadmin;
datalibs sampdat tmplib tfu_ctrl tfu_blez;
log file=logfile;
```

The last step in configuring SAS/INTRNET is to configure and start an application server on your SAS server. Sample programs are provided for ease of configuration. The crux of this program is proc appsrv. This SAS procedure allows you to specify the identified port on the SAS server, and also your SAS data and program file locations. Once modified, this program is executed within a batch SAS job, and it will at that point run as a service on the SAS server, waiting for incoming requests from the web broker until the service is shut down.

Figure 1 shows the server and network configuration particular to this paper. Once this step is complete, the next step is to develop web applications (HTML or Java) and SAS server-side code.

**Development of HTML and SAS code**

The HTML code resides on the web server and must also have access to the SAS server and/or the data location. In our example, these are two different physical entities (one being remote), though they may be combined for simplicity or organizational purposes. HTML code may be developed via a variety of products designed for web publishing, including SAS AppDev Studio, many of which also contain scripting wizards for generation of custom scripts.

These custom scripts are typically used to pass data as parameters for action or verification. However, in the interest of
real-world time constraints, SAS provides several features for handling parameters and generating dynamic output to the web without having to build complicated custom scripts to handle your parameter passing and/or verification.

Following is the HTML code that was used to generate the page shown:

```html
<html>
<head>
<title>Our HTML Reports Generation Page</title>
</head>

<body bgcolor="#FFFFFF">
<hr>
<p align="center"><font size="7"><b>TFU Operational Reports</b></font></p>
<form action="/scripts/broker.exe">
<select size="8" name="rpttype">
<option selected value="0">Select one</option>
<option value="cumout">case_level</option>
<option value="outcom">call_level</option>
</select>
<select size="8" name="opvar">
<option selected value="0">Select one</option>
<option value="cedit">failure_reason</option>
<option value="tfuid">interviewer</option>
<option value="hour">hour_of_day</option>
<option value="tzone">time_zone</option>
</select>
<select size="8" name="datelvl">
<option selected value="0">Select one</option>
<option value="monthly">monthly</option>
<option value="daily">daily</option>
<option value="weekly">weekly</option>
</select>
<p>Start Month</p>
<select size="1" name="smonth">
<option selected value="00">Select one</option>
</select>
<p>Start Year</p>
<select size="1" name="syear">
<option selected value="00">Select one</option>
</select>
</form>
</body>
</html>

Figure 1: Physical/logical design of SAS/IntrNet on 4-tier network configuration
The parameter values designated in the HTML code are subsequently transmitted to your SAS program via the ODS HTML body=_webout(dynamic) option, using %superq(var) to capture the passed values.

Figure 2: Example of HTML page to select parameters
Following is an example of the SAS code used to prepare the data and then format output using the ODS HTML utility:

```sas
%macro subset;
/* library definitions are specified in the application server */
libname fmtlib '/ssroot1/sasprog';
options fmtsearch=(fmtlib);
%if &rpttype = cumout %then %do;
%if &datelvl = monthly %then %do;
proc sql;
/* create a data set of SAS data set names of data sets for a given month */
create table bwdata as
  select memname from dictionary.tables
  where libname = 'TFU_BLEZ' and
  substr(memname,7,2)="&syear" and
  substr(memname,3,2) = 
  "&month";
/* Select count of this data set into a macro variable for future iterative processing */
  select count(*) into :bwcnt from bwdata;
quit;
%if &bwcnt = 0 %then %do; /* if no data sets to aggregate */
  %put No input data exists for this date range;
%end;
%else %do;
/* Convert observations in the "bwdata" data set into macro variable names */
  %do i=1 %to %eval(&bwcnt);
  data _null_; 
m=&i;
  set bwdata point=m;
  call symput("ifile&i",memname);
  stop;
  run;
%end;
/* Now using macro variables just created, aggregate all into one data set */
  %do i=1 %to %eval(&bwcnt);
/* If it is the first observation */
    %if &i = 1 %then %do;
      data allbw;
      set tfu_blez.&ifile1(keep=cumout ncalls);
      run;
%end;
/* If it is not the first observation */
    %else %do;
      proc append base=allbw
      data=tfu_blez.&ifile1(keep=cumout ncalls)
      force;
      run;
%end;
/* Other code to aggregate datasets for various levels of reporting */
%end;
%end;
%mend;
%subset;

ods html body=_webout(dynamic) rs=none;
title "%superq(rpttype) x %superq(opvar)";
proc freq data=allbw noobs label;
tables %superq(rpttype) x %superq(opvar);
run;
ods html close;
```

Figure 3 shows an example of the resulting output.

Among these features are the SAS web publishing tools and the ODS utility. SAS web publishing tools allow you to display SAS datasets or subsets of datasets utilizing standard SAS procedures and options. These web views are generated dynamically, allowing users access to the latest data available.

Usage of the ODS utility also allows you to display SAS datasets or subsets of datasets utilizing standard SAS procedures and options. However, in addition, usage of the ODS utility allows for design of a web application with real-time user interface in which the user designates the parameters for report generation. These parameters are then passed to the SAS application server via the application broker utilizing the %superq(var) and &var (standard SAS macro options) to select and display the data and/or procedure(s) performed on the selected data.

As stated above, our goal was to give the users the same report functionality but with added choice of level, start date, and end date incorporated into their dynamic report. This functionality can be accomplished using the ODS utility in conjunction with streamlined HTML form code to designate the appropriate component(s), e.g. checkboxes, radio buttons, text entry boxes, and so on, contained within the form.
For our straightforward report delivery needs, a more time-consuming and complicated but more robust approach utilizes htmSQL in construction of HTML forms to collect the values selected, and construction of HTML tables to receive the parameters and display the results of the query containing those values. Accomplishing this task involves more complicated programming that combines both SQL and HTML, utilizing htmSQL with the requisite SQL query syntax. For SAS programmers, it is of note that SQL query syntax refers to resulting column names with &colvar_name, where this is NOT a SAS macro variable. The server= option in the SQL query section must point to the appropriate SAS/SHARE server in order to pass the query information for resolution.

Once the interface has been successfully constructed, it is often helpful to utilize select query directives (such as {norows}) to indicate information of interest to the user, e.g., that the query was successfully evaluated but that no data rows met the criteria. Either the simpler ODS or the more complex htmSQL approach can meet the need to rapidly web-enable business applications in an N-tier environment, depending on your various external time and resource constraints. Our approach, and one we recommend in the interest of rapid development, is to first successfully utilize the simpler approach and enable the users, transitioning later to the more robust solution.
Considerations

In our N-tier, cross-platform environment, we found that CPU performance appeared to decrease with the use of select web publishing macros but not others. This held true regardless of whether the data were accessed via SAS Remote Library Services (RLS) or directly through the SAS/SHARE remote connection. Additionally, with SAS/SHARE an integral part of our development environment, we recommend upgrading to SAS Version 8.1 or higher, since Version 8.0 has had known problems with IntrNet. Lastly, wherever possible, our SAS network administrators recommend maintaining parallel SAS versions on your web server and your application server. In general, they recommend trying to avoid conflating 8.2 with lower 8.x versions of SAS, although we have found that 8.x and 6.12 appear to coexist relatively peacefully in our production environment. Also, for format catalogs that need to address datasets in different versions of SAS, you will need to remember to translate your original format catalogs using PROC CPORT and CIMPORT. Additional performance considerations deriving from alternate configurations may also need consideration.

References


Contact Information:

Amanda F. Sardenberg
Bureau of the Census
Room 1657, FOB-3
Suitland, MD 20746
(301) 457-1040
amanda.f.sardenberg@census.gov

Kenneth P. Stulik
Bureau of the Census
Room 1657, FOB-3
Suitland, MD 20746
(301) 457-1040
kenneth.paul.stulik@census.gov