Paper 022-2008

Getting Started with SAS/IntrNet® – A Quick and Dirty Tutorial

Gabe Cano, Altarum Institute, San Antonio, TX

ABSTRACT

Suppose you are required to deliver a report system that is accessible to users. Through some research, you have decided that the optimal way to make this happen is using SAS/IntrNet. Theoretically, everything should work out nicely once the system is in place, i.e., installed and functional. But in reality, things frequently turn out otherwise.

Getting started with SAS/IntrNet requires a fair understanding of the operating system you are working with, as well as the general work structure of the SAS/IntrNet environment. Programmers need to work with administrators to insure that the configuration is functional. System integrity can be tested quickly with some simple programs.

This presentation will briefly demonstrate how SAS/IntrNet works, what types of issues might be encountered during configuration and how SAS® programmers can be an effective part of creating a successful system. An example of a system will be shown along with setting up htmSQL and SAS programs. Some maintenance, development, and troubleshooting tips will also be presented.

Keywords: APPSTART.SAS, SAS/SHARE®, htmSQL.

INTRODUCTION

There are many ways to install, configure, access and run SAS programs with SAS/IntrNet. This is a brief tutorial of a working example and is in no way a substitute for other manuals. This presentation is meant to consolidate documentation cited for running SAS/IntrNet for a single configuration instance.

In this tutorial, it is assumed that an administrator is available to install and answer system configuration questions. The EMPLOYEE example cited here is taken directly from the *SAS Course Notes Instructor-based training*[1] manual.

Running SAS programs with SAS/IntrNet requires a fair amount of knowledge of the operating system you are working on as well as the general work structure of the SAS/IntrNet environment. Programmers invariably need to work together with administrators to insure that the configuration is functional. System integrity can be tested quickly with some simple programs. This presentation will demonstrate what types of issues SAS programmers might encounter, and what questions they may need to ask. An example of a system will be shown along with setting up htmSQL and SAS programs. Some maintenance, development, and debugging tips will also be presented.

As a programmer, this paper will assist in guiding you to asking questions regarding system configuration as a whole so there are no mysteries along the way. As opposed to running SAS programs interactively (or in batch), there are many underpinning concepts of which the programmer must be aware. Some of these general concepts will be explained throughout this presentation along with some answers to the following questions: How do SAS programs run via SAS/IntrNet? What can go wrong in the line of program processing? What is the programmer's responsibility for a successful system?

Ultimately, the hope is that SAS programmers will be better equipped to troubleshoot this type of system and be an effective part of its success.

THE PROBLEM

Suppose you have SAS programs that are used to create reports. Over time, these reports become larger and more complex. You come to the realization that it would be more efficient to produce these reports using the Internet. As a result, end users could look only at the portion of the report(s) they are interested in rather than having to weed through the entire document. Given a minimal set of information, it is your job to deliver these customized reports as a system. Everyone applauds your ingenuity and looks forward to this new system. Now the ball is in your court — you must deliver the system.

Starting out simple is always the best first step.

Here are some of the first steps to begin delivering the system.

THE HTMSQL

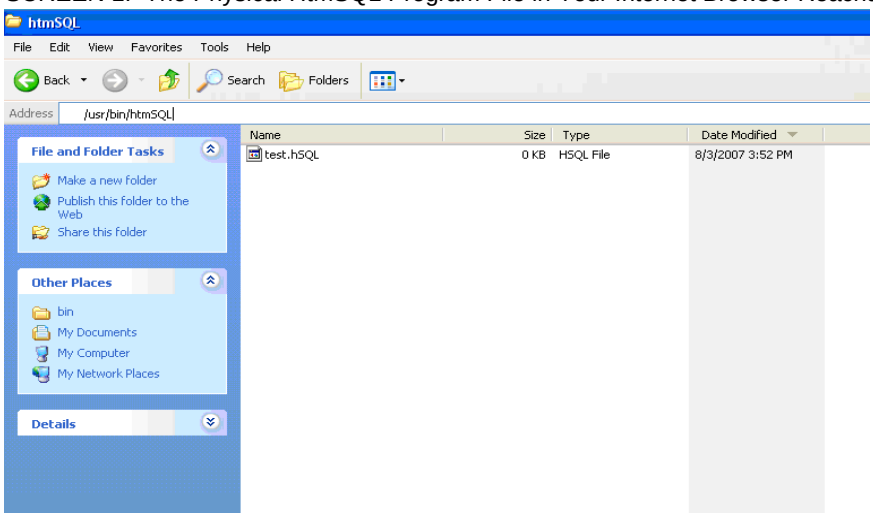
The following is the htmSQL code file that can be used as the user interface. Note that htmSQL and hSQL are used synonymously throughout this paper.

```
<html>
<head>
<title>Using an htmSQL Form Chapter 9 Exercise 2</title>
</head>
{query server="MachineDomainName:Port#"}
<form method=post name="TestForm" action="/cgi-bin/broker">
<h3 align=center>Select a Location</h3>
<h4 align=center>For the Job Code Report</h4>
<hr />
{sql}
select distinct EmpLocation as emploc
from ia.employees
order by emploc;
{/sql}
<p />
Employee Location:  
<select name=emploc>
{eachrow}
<option value="{&emploc}">{&emploc}
{/eachrow}
</select>
<p />
<input type=hidden name=_service value=default>
<input type=hidden name=_program value=programs.c09ex3.sas>
<input type=hidden name=_debug value=0>
<input type=submit value="Send Request">
<input type=reset value="Clear Form">
</form>
{/query}
</html>
```

The htmSQL file should physically reside in a location that is accessible to an Internet browser, for example: /usr/bin/htmSQL/test.hsql. Logically, the htmSQL program file can be referred to as follows: http://MachineDomainServer/cgi-bin/htmSQL/test.hsql. The administrator should be able to assist in locating where and how this location can be accessed. When this Webpage is loaded there are at least two items which must be defined in APPSTART.SAS: (1) IA (DATALIBS) and (2) PROGRAMS (PROGLIBS).

The bolded items in the htmSQL above have some terminology associated with them. See Appendix A for terminology definitions.

SCREEN 1. The Physical HtmSQL Program File in Your Internet Browser Reachable Location



THE PROGRAM

The following SAS/IntrNet program (`C09EX3.SAS`) can be used to create a simple report from the EMPLOYEES dataset and print it to the screen:

```
ods html body=_webout rs=none;
```

```
proc freq data=ia.employees;
title “Job Code Report for &emploc.”;
where EmpLocation eq “&emploc.”;
tables JobCode / nocum norow nocol;
run;

ods html close;
```

Note: This program uses the IA libname and the EMPLOYEES dataset, which should also be available upon installation. This program should work as is and as a result will, in part, indicate that the APPSTART.SAS program file is being loading properly.

THE DATASET

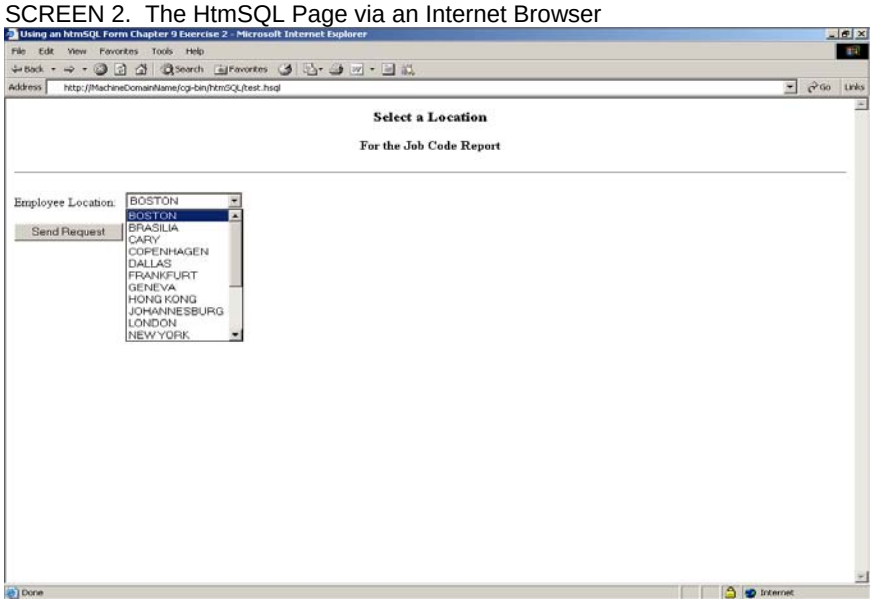
Take the EMPLOYEES data set installed with SAS/IntrNet. It contains about 2000 records and should be readily available for testing. This is a snippet of that SAS dataset.

```
/*************************/
/* Dataset: EMPLOYEES    */
/* Variables:            */
/* EmpLocation   JobCode */
/*************************/
CARY                FLTAT3
CARY                VICEPR
COPENHAGEN          GRCREW
TORONTO             OFFMGR
CARY                MKTCLK
BOSTON              RECEPT
CARY                RESCLK
CARY                FACMNT
CARY                FACCLK
...              ...
```

WHEN THINGS GO RIGHT

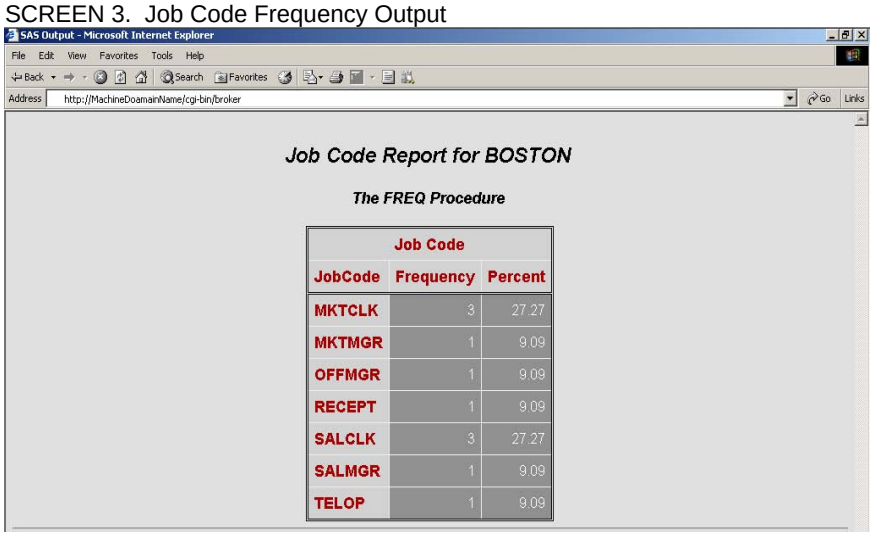
SCREEN 2 is what you will see via the Internet browser when you enter the URL for your htmSQL page. This browser page is also the point from which the SAS program will be run.

SCREEN 2. The HtmSQL Page via an Internet Browser



When everything is working properly, the user can select a city from the pull down menu and click the “Send Request” button. The resulting report appears as shown in SCREEN 3.

SCREEN 3. Job Code Frequency Output



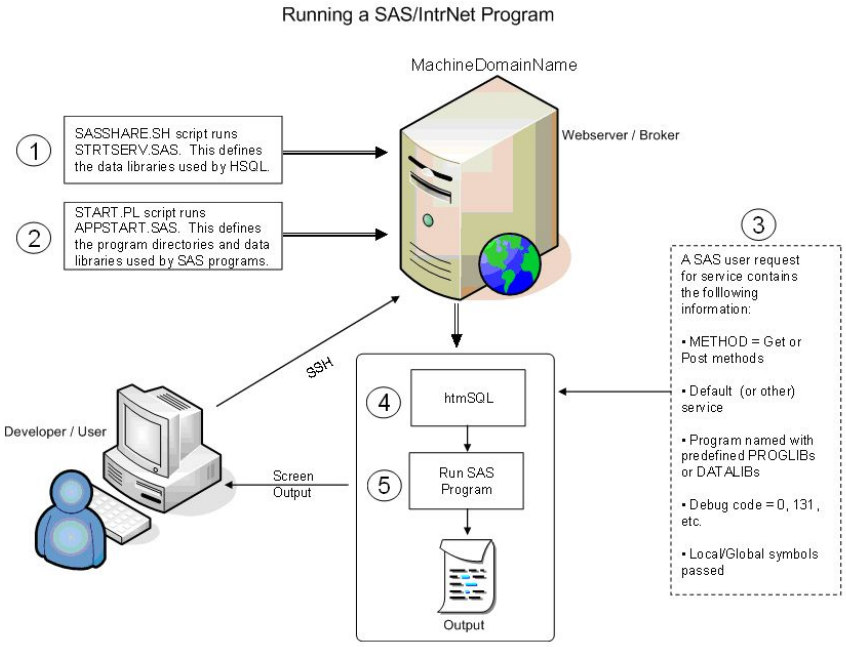
Job Code		
JobCode	Frequency	Percent
MKTCLK	3	27.27
MKTMGR	1	9.09
OFFMGR	1	9.09
RECEPT	1	9.09
SALCLK	3	27.27
SALMGR	1	9.09
TELOP	1	9.09

Now that we have seen a working example let's explore how SAS/IntrNet programs are run.

HOW SAS/INTRNET PROGRAMS ARE RUN[2]

Here is a graphical explanation of how SAS/IntrNet programming works.

GRAPHIC 1.



There is much documentation regarding SAS/IntrNet and how to run programs using this component of SAS. For this particular example, the system requirement is that both SAS/SHARE and SAS/IntrNet must be running in order for the SAS program to run properly. The following numbered items can be considered a checklist of processes which must execute in order for SAS/IntrNet to successfully carry out program requests.

1) Run the SAS/SHARE script to define data libraries for data display and queries.
2) Run the SAS/IntrNet service script with APPSTART.SAS definitions.
3) HtmSQL page should be properly loaded with all SAS program information.
4) Bring up your htmSQL page with an Internet browser and make data selections.
5) Submit request to run the program.

This entire line of processing is very important for SAS programmers to know and understand. From the model that GRAPHIC 1 represents so many things can (and do!) go wrong.

Understanding how the system works and where your program files are located is important.

1 – THE SAS/SHARE SCRIPT

SAS/SHARE is running in the background via a script (SASSHARE.SH) which starts the server program by running a SAS program STRTSERV.SAS. This program defines at least one item of interest to us – It defines the location of the IA library used by the htmSQL within the Internet browser. As seen in SCREEN 2 the cities listed in the pull-down menu are displayed from a predefined SAS dataset. It is also because SAS/SHARE is running that we are able to query the dataset within the Webpage for display.

Here is part of what a STRTSERV.SAS program file might look like:

```
proc server;
allocate   library  ia   '/user/bin/ia';
run;
```

2 – THE SAS/INTRNET SERVICE

Now that SAS/SHARE is running the SAS/IntrNet service must also be running. The service is run via a script (START.PL) which runs a SAS program called APPSTART.SAS. This service, also known as the Application Dispatcher, is a suite of programs waiting for an application request. The request is carried out by the BROKER program and sent to the Application Server where the program is processed. From the Web browser all the way down to the final SAS program, all symbols are passed along as macro variables.

Here is where the APPSTART.SAS (see below) file comes into play. This file contains libname allocations for both the data you are accessing and the location of the SAS programs. Note that IA and PROGRAMS used in the htmSQL are defined here.

```
proc appsrv unsafe='&";%'''   &sysparm ;
/* standard SAS/IntrNet setup */
allocate  file    sample  '!SASROOT/samples';
allocate  library samplib '!SASROOT/samples' access=readonly;
allocate  library sampdat '!SASROOT/samples' access=readonly;
allocate  library tmplib  '.';
allocate  file    logfile '../logs/%a_%p.log';
proglibs  sample  samplib sashelp.webeis sashelp.webprog;
proglibs  sashelp.websdk1;
adminlibs sashelp.webadmn;
datalibs  sampdat tmplib;
/* SAS program locations for htmSQL reference */
allocate library programs '/user/bin/programs';
allocate file programs    '/user/bin/programs';
/* SAS program locations for htmSQL and SAS program reference */
allocate library ia       '/user/bin/ia';
allocate file ia          '/user/bin/ia';
/* SAS program locations for SAS program reference */
allocate library library  '/sas/formats';
allocate file library     '/sas/formats';
proglibs programs;
datalibs ia library;
run;
```

3 – LOADING THE HTMSQL PAGE

When creating the HTML Webpage (i.e., TEST.HSQL) the programmer must remember that the page must be loaded with all the necessary machinery for SAS/IntrNet to translate into information usable by the underlying SAS programs. Understanding your machine's work structure becomes another key to solving the brunt of your SAS/IntrNet programming problems. In this regard the intimate knowledge of the following locations will pop up again and again:

- The logical and physical locations of your SAS programs.
- The logical and physical locations of your SAS datasets.
- The htmSQL page should contain the machinery to pass symbols to SAS program.

SAS programs and datasets should be kept separate and reside securely within your system intranet. Doing this, will help avoid confusion in how you define your PROGLIBS and DATALIBS definitions in the APPSTART.SAS program.

As development evolves you or the administrators will relocate the SAS/IntrNet system for various reasons, namely maintenance. System maintenance requires the continuous synchronization with the STRTSERV.SAS and the APPSTART.SAS programs to keep data and program libraries up to date. This is can be a challenging task, since in most cases the system administrator is the keeper of these files.

4 – BRING UP THE HTMSQL PAGE WITH A BROWSER AND MAKE DATA SELECTIONS

You will be able to tell right away if your machine's htmSQL location is defined properly. HtmSQL pages should reside in a location viewable with an Internet browser. On newer versions of SAS/IntrNet this location may already be known to the Webserver as the following URL:

http://www.webserver.com/cgi-bin/htmSQL/test.hsql

or

http://www.webserver.com/test.hsql

This virtual location can also be customized by the system administrator in the Webserver configuration files. Redefining these locations are areas that administrators and SAS programmers need work closely together on.

Next, you will be able to tell if the SAS datasets being used for data selections are accessible or if the SAS datasets being used for selections were created correctly. This will be apparent if you can see all the expected data selections in the pull-down menu (as shown in SCREEN 2).

5 – SELECT DATA AND SUBMIT REQUEST TO RUN THE PROGRAM

Once the data selections have been made from the screen the user (or developer) can submit a request to run the SAS program. When the program has been run, the final results are passed back in the same order and printed to the screen. At this point there could be more mysteries at hand. A few examples are the following: (1) that the libraries you defined in APPSTART.SAS are either not accessible or don't exist, (2) symbols (e.g., htmSQL statement lines such as the following: <input … >) from the htmSQL page did not get sent properly to the SAS program, or (3) there is an error in the SAS program.

WHEN THINGS GO WRONG – TROUBLESHOOTING ERROR

Unfortunately, this is not a perfect world and the nice report shown in SCREEN 3 is probably not what you will see when you first run your SAS/IntrNet programs.

Errors with SAS/IntrNet can have much mystery associated with them. Why? Because there are many points at which things can go wrong. Many of the following problems and solutions are the responsibility of the administrator; but as the developing programmer, you have a better ability to diagnose the problem and discover a solution.

Here are five basic case points at which things can go wrong:

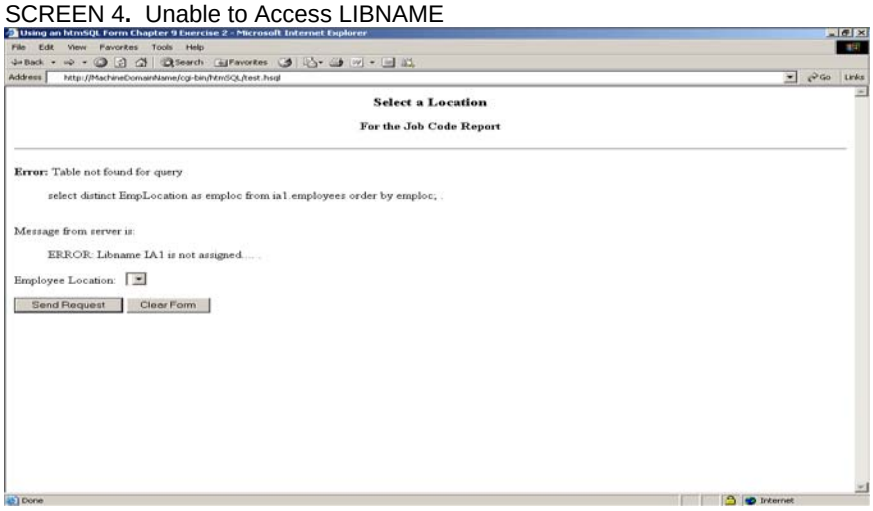
1. Logical and physical location names defined for data in the SAS/SHARE startup script.
2. Logical and physical location names defined for SAS programs and data in APPSTART.SAS.
3. PROGLIBS/DATALIBS location names and other service information used in the HtmSQL page.
4. Reading the htmSQL page and handing off symbols to the SAS program.
5. Running the SAS program with all of the above information.

These numbered points correlate, respectively, to those used in the steps to describe GRAPHIC 1. It should also be noted here that in troubleshooting problems within this model there is much overlapping in how to solve problems within such a system.

CASE 1 – LOGICAL AND PHYSICAL LOCATION NAMES DEFINED FOR DATA IN SAS/SHARE

The first time you enter the URL for your program location using the defined aliases in APPSTART.SAS you may see something like this:

SCREEN 4. Unable to Access LIBNAME



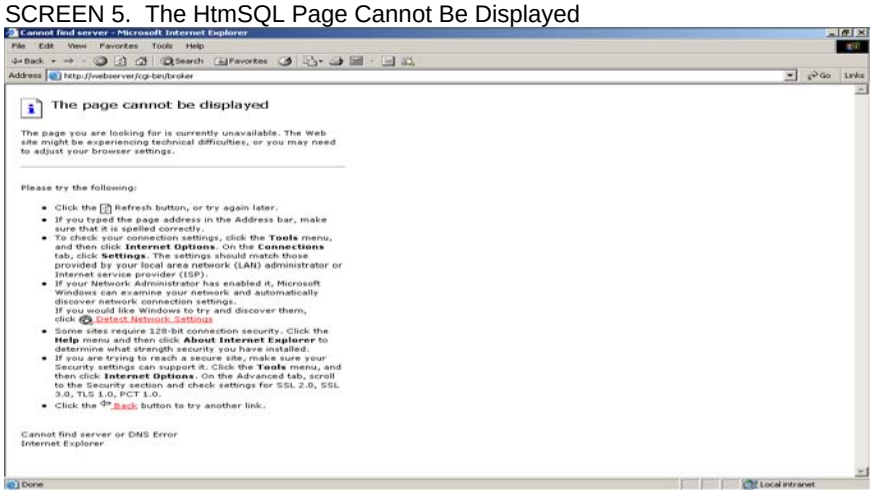
What went wrong? The data libname IA was not defined properly or SAS/SHARE is not running.

TROUBLESHOOTING: There are several possible errors that can cause a result as displayed in SCREEN 5. These include the following: (1) the libname for the dataset does not exist or is misspelled, (2) the SAS dataset is either not at that location anymore or the name is misspelled, or (3) the dataset to be used for htmSQL data selections may not be a SAS dataset. There could also be a deeper administration problem with the SAS/SHARE service.

CASE 2 – LOGICAL/PHYSICAL NAMES DEFINED BY SASSHARE.SH OR APPSTART.SAS

You could also see something similar to SCREEN 5.

SCREEN 5. The HtmSQL Page Cannot Be Displayed



What went wrong? The htmSQL page and/or location is not viewable.

TROUBLESHOOTING: The possible causes of problems which can occur here include (1) the logical directory name is not defined or has been lost due to system restructuring, maintenance, etc., (2) the system/Webserver could be off or was reset (rebooted) but the SAS/IntrNet services were not restarted or (3) the SAS/IntrNet user process does not have permission to access that location.

CASE 3 – LOADING LOGICAL AND PHYSICAL NAMES USED IN THE HTMSQL PAGE

It is important to keep in mind that when developing your webpages that you include all the necessary items which the SAS/IntrNet service will need to successfully cary out your program request. The 3rd numbered item in GRAPHIC 1 can be considered a checklist for the necessary information which is expected to be defined in the webpage. Items such as the DEFAULT service and the defined METHOD (GET, POST, or other?) used to carry out the programmers request are the responsibility of the administrator. This is another point where programmers must work closely with administrators to understand the system configuration.

SCREEN 6. Default Service Not Available

What went wrong? The specified SAS/IntrNet service (DEFAULT1), which should run your program, has not been defined.

TROUBLESHOOTING: Several different types of problems can occur at this point in the system. These include the following: (1) the administrator started SAS/SHARE but did not start the SAS/IntrNet default service (START.PL), (2) the error is a result of a programming error (e.g., you may have the wrong name as the _SERVICE VALUE parameter), or (3) the service might not be running over the specified port for some unknown reason. There could also be deeper administration problems with the service definition.

CASE 4 – READING THE HTMSQL PAGE AND HANDING OFF SYMBOLS TO THE SAS PROGRAM

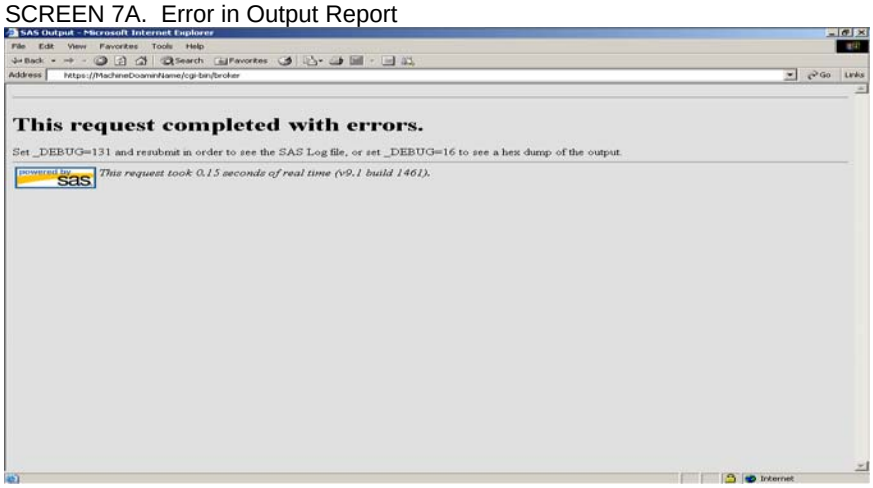
What else can go wrong? The SAS program can be missing.

OTHER TROUBLESHOOTING: Though there is no graphic listed for a missing program case the result looks similar to SCREEN 6. Problems of this nature can be due to the following: (1) the program name is misspelled in the htmSQL, (2) the logical location is incorrect or not defined, or (3) the SAS program file does not physically exist or moved.

CASE 5 – ERRORS IN THE SAS PROGRAM

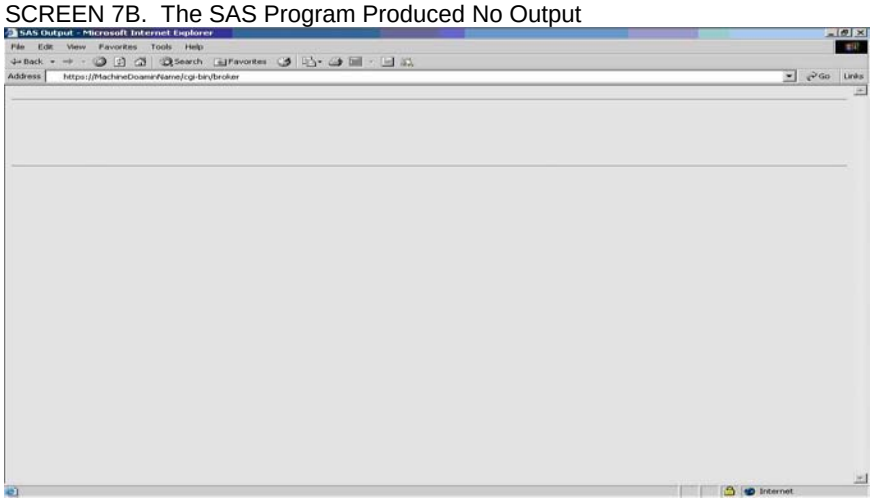
If you got to this point then your system may be in pretty good shape. The following items detail programming scenarios which are more programs based. This means that your system configuration appears functional and that the errors you are seeing are based in your SAS programs. The following items are some cases of what you might see on your screen when you encounter an error in your SAS program.

SCREEN 7A. Error in Output Report



What went wrong? Everything is functional, but you have an error in your SAS program.

SCREEN 7B. The SAS Program Produced No Output



What went wrong? Actually SCREEN 7A & B are good errors. Everything appears functional, but your SAS program produced an error or no output.

TROUBLESHOOTING: There could be two problems here. Either (1) there is an error in the SAS program (the solution would be to change to higher debug code, e.g., 128, to see the log output) or (2) the SAS/IntrNet process does not have access permissions to write to an output directory.

CONCLUSION

As data becomes more available and Internet connections improve, dynamic reporting systems will become more and more the tools of choice. SAS/IntrNet offers a viable option to do this.

Once a simple system is in place it becomes more apparent that the power of real-time reporting can be extended to other datasets, programs or even applications. Any of these may be external to the SAS/IntrNet system but accessible to the working intranet. You, or others, may desire to apply this working model in order to create a more extensive system. This will surely create more complex problems. Such problems will come in the form of firewall issues, permissions issues, general accessibility (i.e., SQL queries, running external applications, etc.) and others. See the SAS Global Forum Webpage (or your regional SAS user group Webpages) [3] for more SAS/IntrNet examples[4].

Hopefully, this presentation has offered some insight into (i) the types of problems you will encounter while developing SAS/IntrNet programs, (ii) how you as a programmer can be proactive in anticipating those problems, (iii) how you can assist system/network administrators in troubleshooting , and finally (iv) being successful in such a programming environment.

APPENDIX A – TERMINOLOGY

Action - See BROKER.

Application Dispatcher – A product of SAS/IntrNet that is a CGI program containing other programs which carry out a SAS program request.

Application Server – A program which carries out the application request from the BROKER program.

APPSTART.SAS – This file runs to start SAS PROC APPSERV. This is a program which defines program and data libraries that are used by the Application Dispatcher.

BROKER – This is an executable program. This is the application program that runs SAS programs with all the user specified inputs, or symbols. The program is referred to as the /cgi-bin/broker.

BROKER.CFG – The file broker.cfg defines the port that broker will use and other information for setting up the broker capabilities.

FORM METHOD - Methods of submitting a SAS/IntrNet program. Usually, GET or POST methods are used.

HTML – Hyper Text Mark-up Language. Used for producing Webpages via an Internet browser.

htmSQL – A product of SAS/IntrNet which allows you to embed SQL queries in HTML pages. These pages can also be identified with any other suffixes, e.g., .hsql, .htm etc. htmSQL as a logical directory is defined in STRTSERV.SAS.

Logical Directory – This is the alias defined by the administrator to locate a directory via your Internet browser. Example: the htmSQL and SAS programs are accessed through a logical directory names.

MachineDomainName:Port# - Generally, the same as the WEBSERVER machine name. Administrators can recommend a safe port #. Under htmSQL this is called the query server value.

SASSHARE.SH – The script, sasshare.sh, that runs strtserv.sas to start SAS/SHARE which continually runs in the background. The log of the SAS/SHARE goes to strtserv.log.

Service (Input) Value - A default service (or socket) should be created upon installation. Other customized services may be created for different groups or as backup services.

SQL – Structured Query Language. This is a generic term for any language which provides the ability to perform traditional predicate calculus functions on a database.

START.PL – The script used to run SAS/IntrNet Application Dispatcher with APPSTART.SAS.

STRTSERV.SAS – Script SASSHARE.SH which runs SAS PROC SERVER which defines the SAS data libraries used by the SAS/IntrNet hsql files. For new data libraries, this file will need to be updated and service restarted.

Symbols – SAS term for the SAS/IntrNet program macro variables passed from the htmSQL page.

Virtual Directory – An alias defined by the administrator to locate a directory via your Internet browser.

WEBSERVER – This is the machine running the CGI BROKER program that will ultimately run your SAS/IntrNet program.

APPENDIX B – SAS/INTRNET DEBUG CODES

SAS programs are run when the submit command is issued in an HSQL screen from an Internet browser. The following results are rendered when a SAS/IntrNet HSQL page submits the following debug values with broker:

- debug=0 will allow you to see the resulting HTML output.
- debug=1 will allow you to see the symbols passed from the broker to the SAS program and the resulting HTML output.
- debug=2 will allow you to see the resulting HTML output with the SAS Logo and the time it took to execute the SAS run.
- debug=3 will allow you to see the symbols passed from the broker to the SAS program, the resulting HTML output with the SAS Logo, and the time it took to execute the SAS run.
- debug=128 will allow you to see the output listing and log together.
- debug=129 will allow you to see the symbols passed from the broker to the SAS program, a listing (if one printed), and the log.
- debug=130 will allow you to see the output listing and log together with the SAS Logo, and the time it took to execute the SAS run.
- debug=131 will allow you to see the symbols passed from the broker to the SAS program, a listing (if one printed), the log, the SAS Logo, and the time it took to execute the SAS run.

Note: If your programs create any type of HTML output you will not be able to see this with debug=128 or higher. You will only see the log [and symbols].

Other debug codes exist but the above codes are those primarily related to viewing the SAS log, listing and symbols.

REFERENCES

1. SAS Web Tools: Static and Dynamic Solutions Using SAS/IntrNet Software Course Notes. Copyright 2001 by SAS Institute, Cary, NC 27513, USA.

2. SAS Web Tools: Advanced Dynamic Solutions Using SAS/IntrNet Software Course Notes. Copyright 2001 by SAS Institute, Cary, NC 27513, USA.

3. SAS Global Forum (and previous forums) URL: http://support.sas.com/events/sasglobalforum/index.html. (SAS regional groups, other SAS user groups and various examples can be found at the SAS Support Website: http://support.sas.com/).

4. An example of a SAS/IntrNet system with complex problems: SUGI 29 Paper 026-29: Gabriel Cano, Bob Cameron, Kevin McGowan, Jean Orelien. *Developing a System with SAS/IntrNet, Accessing an Oracle Database, Some ODS and a Whole Lot of Problems*. http://www2.sas.com/proceedings/sugi29/026-29.pdf.

ACKNOWLEDGMENTS

Special thanks to Lisa Matthews, Altarum Institute, for her editing assistance.

CONTACT INFORMATION

Gabe Cano
Altarum Institute
3737 Broadway, Suite 205
San Antonio, TX 78209
(210) 832-3000
(210) 832-3099
gabe.cano@altarum.org