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

South Central SAS Users Group

SAS Educational Forum 2007

Austin, TX

Gabe Cano, Altarum Insitute

## ABSTRACT

Suppose you are required to deliver a report system via the internet. 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… well… you know the story.

Getting started with SAS/IntrNet requires a fair amount of knowledge of the operating system you are working with as well as the environment work structure of SAS/IntrNet. 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 questions need to be asked and what types of issues might be encountered by SAS programmers. 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 literally multitudes of 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.

Some assumptions are needed such as 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 machine work structure. 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 questions need to be asked and what types of issues might be encountered by SAS programmers. 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 that there are no mysteries along the way. As opposed to running SAS programs interactively (or in batch) there are many underpinning concepts that the programmer must be aware of. Some of these general concepts will be explained throughout this presentation along with some answers to the following questions: How SAS programs run via SAS/IntrNet? What can go wrong in the line of program processing? What the programmer's responsibility is to a successful system?

Ultimately, the hope is that SAS programmers will be better equipped to troubleshoot this type of system and to also be an effective part of the administration of it. In the end the bottom line to working with SAS/IntrNet is knowing the physical and virtual program locations, how to access the data and how to run your programs.

## THE PROBLEM

Suppose you have SAS programs that are used to create reports. Over time these reports become larger and more complex. You have come to the realization that it would be more efficient if these reports were produced over the internet. This way end users would only look at the portion of the report(s) they are interested in rather than 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 is anticipating this new system. Now the ball is in your court. You must deliver the system.

Starting out simple is always the best first step.

## 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                MECH02
CARY                RESCLK
CARY                FACMNT
CARY                FACCLK
...                 ...
```

## THE PROGRAM

The following SAS/IntrNet program 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;

title;
ods html close;
```

Note: this program uses the IA LIBNAME which should also be available upon installation. This program should work as is and will, as a result, indicate that the APPSTART.SAS program file (described later) is being loading properly.

## THE HTMSQL

The following is the htmSQL code file that can be used as the user interface.

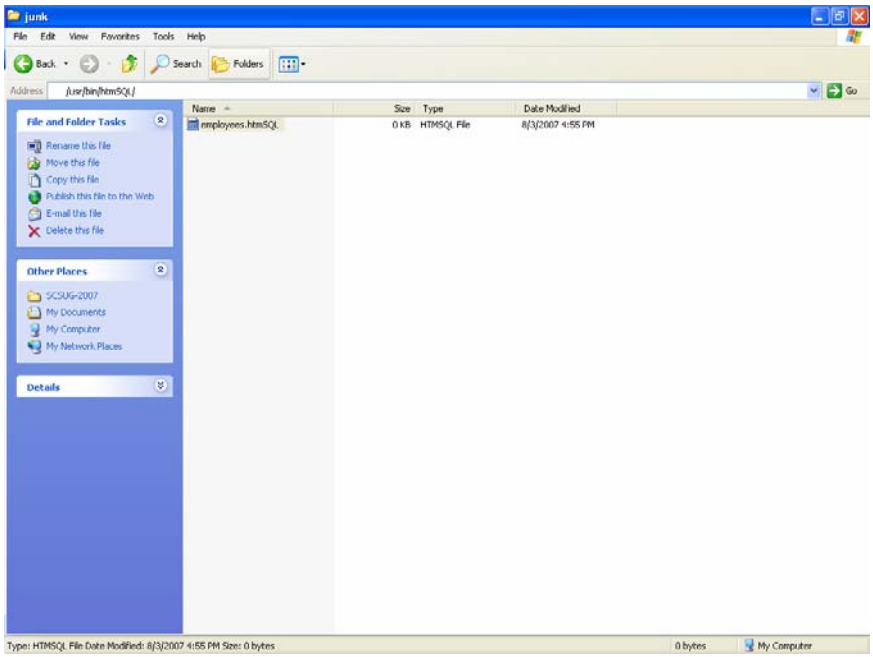
```
<html>
<head>
<title>Using an htmSQL Form Chapter 9 Exercise 2</title>
</head>
{query server="MachineDomainName:Port#"}
<form method=post name="TestForm" action="/cgi-bin/broker">
<h3 align=center>Select a Location</h3>
<h4 align=center>For the Job Code Report</h4>
<hr />
{sql}
select distinct EmpLocation 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 above bolded items have inherent attributes associated with them. See Appendix A for terminology definitions.

The htmSQL file should physically reside in a location that is accessible to an internet browser, for example: /usr/bin/htmSQL/employees.htmsql. Virtually, the htmSQL program file can be referred to as follows: http://MachineDomainServer/cgi-bin/htmSQL/employess.htmsql. (See the APPSTART.SAS program file and Appendix A for terminology definitions and information regarding file/ data locations.)

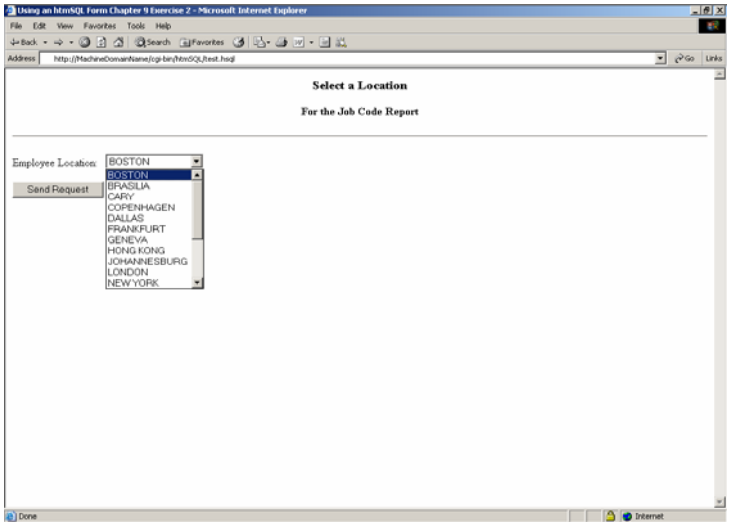
**SCREEN 1.** The Physical HtmSQL Program File in Your Internet Browser Reachable Location




## WHEN THINGS GO RIGHT

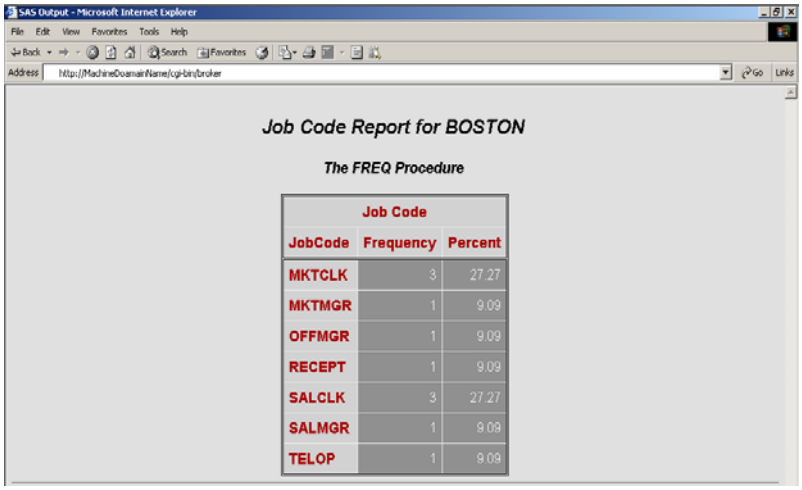
Screen 2 is what you will see via the internet browser. This browser page is also the point where the SAS program will be run from.

**SCREEN 2.** The Virtual HtmSQL Page via an Internet Browser




When everything is working properly the user can select a city from the pull down menu and hit the "Send request" button. This is what the result report might look like as shown in Screen 3.

**SCREEN 3.** Job Code Frequency Output



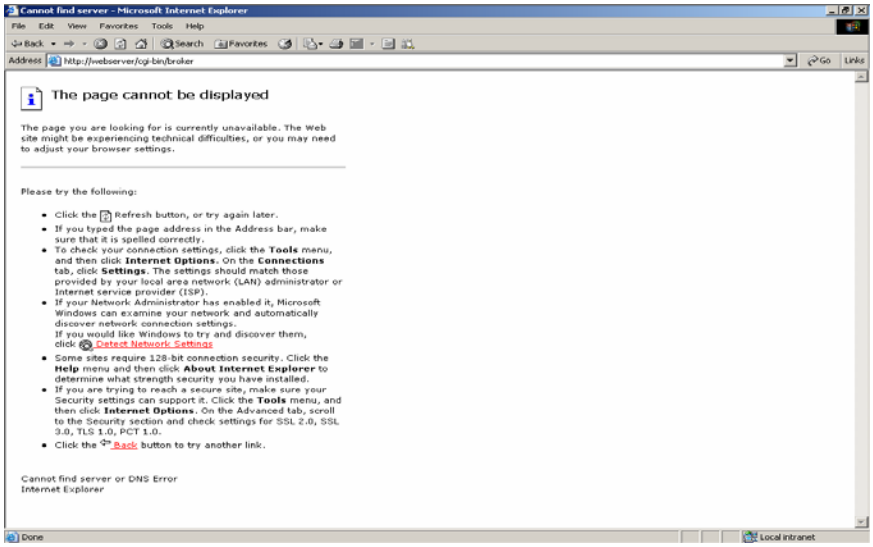

| Job Code | | |
|---|---|---|
| JobCode | Frequency | Percent |
| MKTCLK | 3 | 27.27 |
| MKTMGR | 1 | 9.09 |
| OFFMGR | 1 | 9.09 |
| RECEPT | 1 | 9.09 |
| SALCLK | 3 | 27.27 |
| SALMGR | 1 | 9.09 |
| TELOP | 1 | 9.09 |

## WHEN THINGS GO WRONG

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.

In fact, the first time you enter the URL for your program location using the defined aliases in APPSTART.SAS you will see something like this…

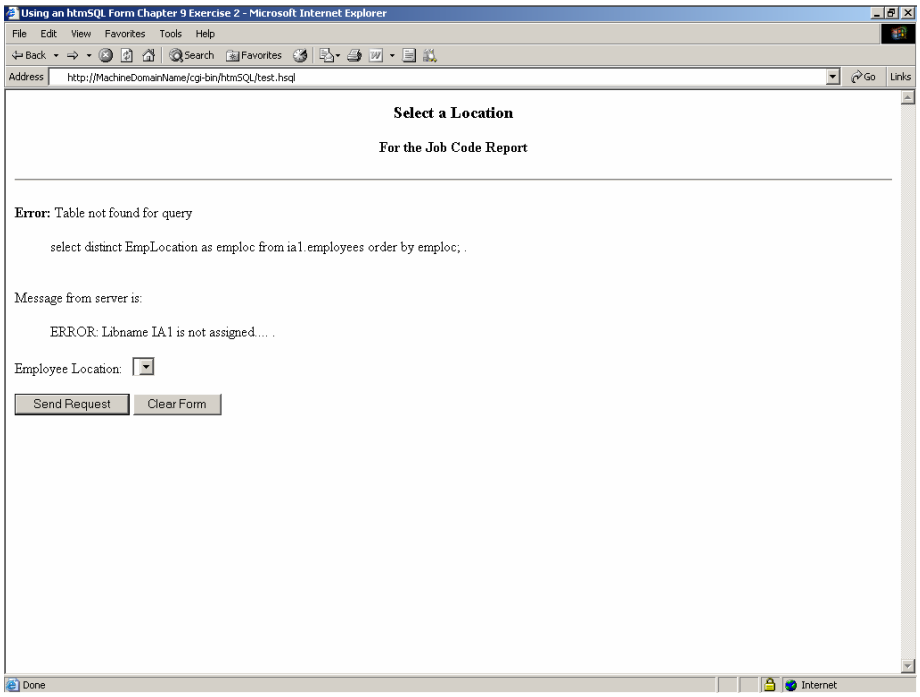
**SCREEN 4.** Unable to Access the HtmSQL Page Directory




This means that the htmSQL page and/or location is not viewable.

Or you could see a screen like this…

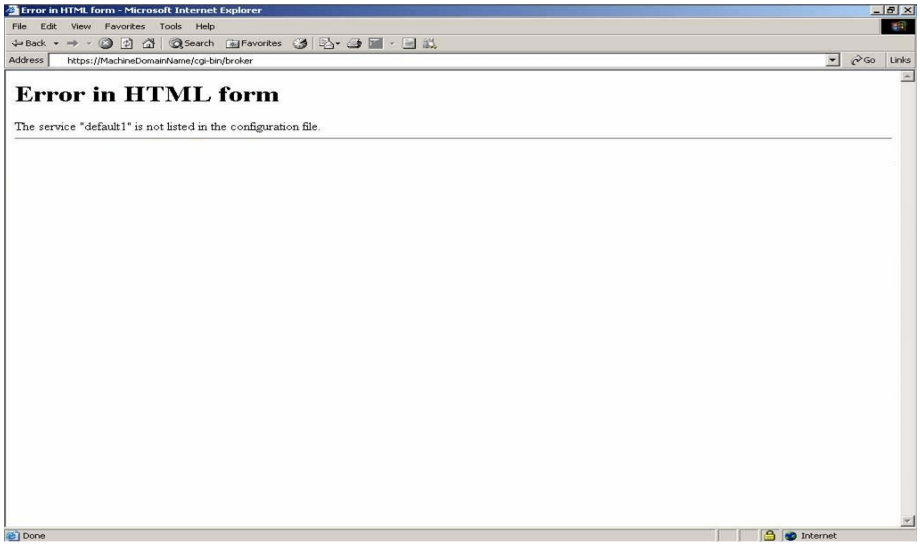
**SCREEN 5.** Unable to Access LIBNAME




This screen error is due to the data LIBNAME IA not being defined properly or that SAS/Share is not running.

You could also see something similar to this …

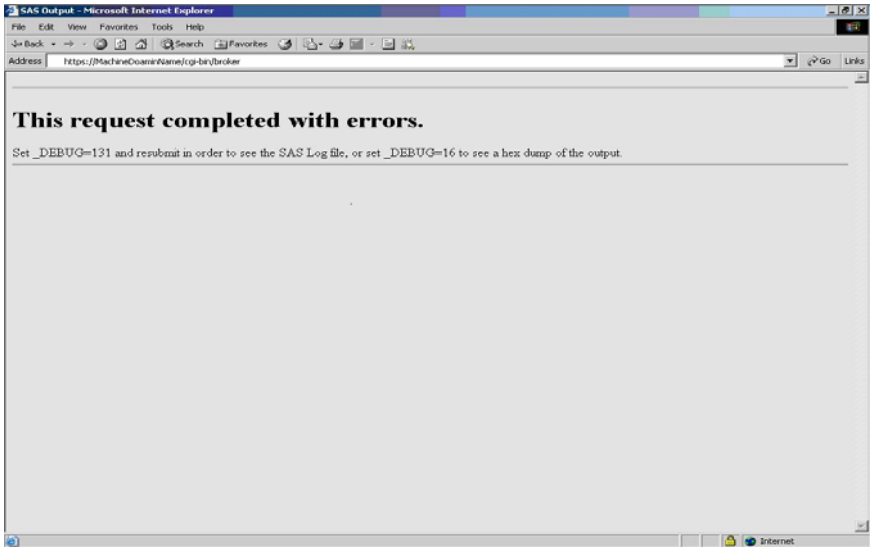
**SCREEN 6.** Default Service Not Available




This means that the specified SAS/IntrNet service, which should run your program, has not been defined.

But, if you are lucky you will see something like this…

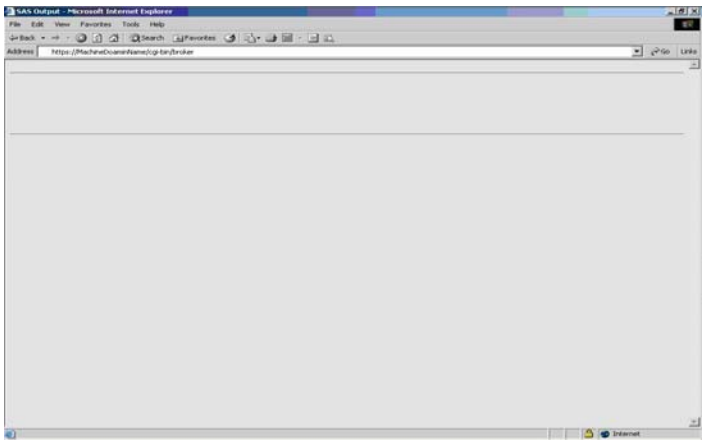
**SCREEN 7A.** Error in Output Report




This result screen means that everything is functional but that you have an error in your SAS program.

Or similarly, you could see this screen...

**SCREEN 7B.** The SAS Program Produced No Output



This result screen means that everything is functional but that your SAS program produced no output.

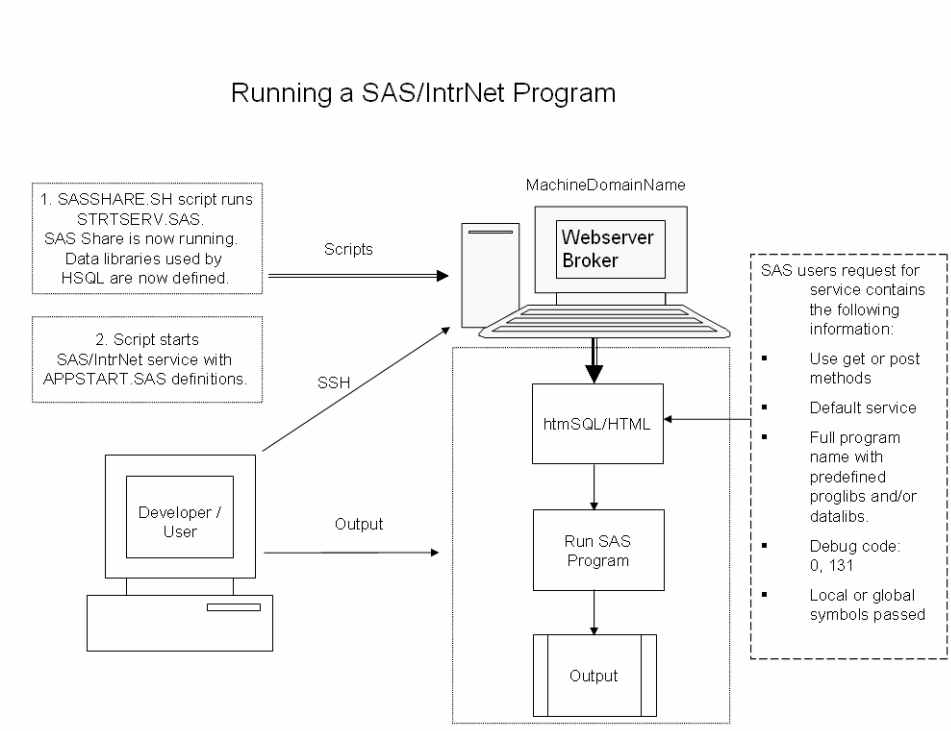
Now that we have seen examples of what can go wrong let's discuss how we might try to fix things.

First let's talk about how SAS/IntrNet works.

## HOW SAS/INTRNET PROGRAMS ARE RUN[2]

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

**GRAPHIC 1.**



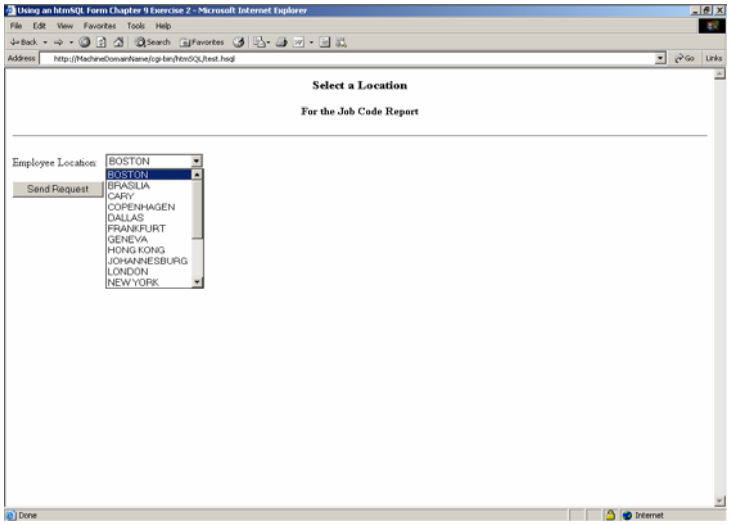

There is much documentation regarding SAS/IntrNet and how to run programs using this component of SAS. For this particular example the system requirement is that both SAS/Share and SAS/IntrNet must be running in order for the SAS program to be run properly.

1) Run the SAS/Share script to define data libraries.
2) Run the SAS/IntrNet service script with APPSTART.SAS definitions.
3) [Optional] (If data are located externally then SAS/ACCESS will also have to be running.)

These two items are very important for SAS programmers to know and understand. From the cases sited above many of the headaches associated with running programs via SAS/IntrNet orginate.

**THE SAS/SHARE SCRIPT**
SAS/SHARE is running in the background via a script which starts the server program. This program accesses the SAS datasets used for displaying information and will allow you to view SQL (see Appendix A for terminology) query result on that data within an HTML page.

**SCREEN 7.** SAS/SHARE Accessing a SAS Dataset.




The cities listed in the pull-down menu are displayed from a predefined SAS dataset.

**THE SAS/INTRNET SERVICE**
Now that SAS/SHARE is running the SAS/IntrNet service must also be running. This service, also known as the Application Dispatcher, is a suite of programs waiting for an application request. This request in carried out by the BROKER (see Appendix A for terminology) 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 file comes into play. This file contains the LIBANME allocation names of both the data you are accessing and the location of SAS programs. Here is what an APPSTART.SAS program file might look like:

```
proc appsrv unsafe='&";%'''   &sysparm ;
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;

/* program locations for htmSQL reference */
allocate library programs '/user/bin/programs';
allocate file programs    '/user/bin/programs';

/* program locations for htmSQL and SAS program reference */
allocate library ia       '/user/bin/ia';
allocate file ia          '/user/bin/ia';

/* program locations for SAS program reference */
allocate library library  '/sas/formats';
allocate file library     '/sas/formats';

proglibs programs;

datalibs ia library;
run;
```

Once the SAS program has been processed the final result is passed back in the same order and printed to the screen.

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

**LOCATION BECOMES THE ISSUE**

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:

1. The virtual and physical locations of your htmSQL pages.
2. The virtual and physical locations of your SAS/IntrNet programs.
3. The virtual and physical locations of your datasets.

(see Appendix A for terminology) location of your htmSQL pages.

HtmSQL pages should reside in a location viewable with an internet browser. This location is defined by the system administrator in the STRTSERV.SAS program.

SAS programs and datasets should be kept separate and reside securely within your system intranet. These locations are defined in the APPSTART.SAS program.

Furthermore, system maintenance requires the continuous synchronization between the STRTSERV.SAS and the APPSTART.SAS programs. This is can be a challenging task since in most cases the system administrator is the keeper of these files.

## TROUBLESHOOTING ERRORS

Errors with SAS/IntrNet can have much mystery associated with them. Why…? Because there are many things which can go wrong in the entire process that could contribute to the problem. Yes, many of the

following solutions are the responsibility of the administrator but as the developing programmer you have a better ability to diagnose the problem and a solution.

**TROUBLESHOOTING SCREEN 4 – UNABLE TO ACCESS THE HTMSQL DIRECTORY**
The possible cause of problems which can occur here include that (i) the virtual directory name is not defined or has been lost due to system restructuring, maintenance, etc. or (ii) the system/webserver was reset/rebooted but the SAS/IntrNet services were not restarted.

**TROUBLESHOOTING SCREEN 5 – UNABLE TO ACCESS DATASET DIRECTORY**
There are several possible errors which can cause errors like that listed in Screen 5. These include the following: (i) the LIBNAME for the dataset does not exist or is misspelled, (ii) that the SAS dataset either is not at that location anymore or the name is misspelled, or (iii) that the dataset appears to be a SAS dataset but might not be.

**TROUBLESHOOTING SCREEN 6 - DEFAULT SERVICE NOT AVAILABLE**
Again, several different types of problems can create errors of this nature and at this point in processing. These include the following: (i) the administrator started SAS/SHARE but did not start the SAS/IntrNet default service (START.PL), (ii) the error is a result of a programming error, e.g., you may have the wrong name as the _SERVICE VALUE parameter, or (iii) that the service might not be running over all ports for some unknown reason.

**TROUBLESHOOTING MISSING PROGRAMS**
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: (i) the program name is misspelled in the htmSQL, (ii) the virtual location is incorrect or not defined, or (iii) that the file does not physically exist.

**TROUBLESHOOTING SREENS 7A AND 7B – ERROR IN OUTPUT**
These are the easy ones. As far as SAS/IntrNet goes these solutions are more standard in nature and can be solved by typical SAS debugging practices using the debug codes (used in the htmSQL pages) listed in Appendix B.

## CONCLUSION
As data becomes more available and internet connections improve dynamic reporting systems will become more and more the tool 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 a program also defines the 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 /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. Traditionally used for producing web pages through an internet browser.

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

MachineDomainName:Port# - Generally, the same as the WEBSERVER machine name.

Query Server - See MachineDomainName:Port#.

SASSHARE.SH – A 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 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.

STRTSERV.SAS – Script sasshare.sh which runs SAS PROC SERVER that defines the data libraries used by the SAS/IntrNet hsql files. As new data libraries are identified, this file will need to be updated and restarted.

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

Virtual Directory (Name) – This is the alias defined by the administrator to locate a directory via your internet browser. Example: the BROKER program is accessed through a virtual directory name.

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.

All references to SAS and SAS products are trademarks of SAS Institute, Cary, NC 27513, USA.

## CONTACT INFORMATION

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