Paper 36-28

# Producing American Community Survey Edit Analysis Reports Dynamically Using SAS/IntrNet®

Arumugam Sutha, US Census Bureau, Washington DC


## ABSTRACT

Each year the Data Processing Team (DP Team) of the American Community Survey (ACS) at the U. S. Census Bureau produces several reports to support subject-matter analysts in their review of the edited survey data. Since these reports are several hundred pages long and a particular analyst has interest in only a few pages, there have been complaints from the analysts on printing and viewing the reports. SAS/IntrNet® software offers a solution to this problem. It allows the users to select certain parameters from a browser at their desktop and submit a job to produce a nicely formatted HTML report without having SAS software on their machine. This paper will discuss how SAS/IntrNet is used to accomplish this task and will describe the necessary SAS code. It will also show how the report can be downloaded to a spreadsheet by the click of a button.


## INTRODUCTION

The ACS is being developed by the Census Bureau to replace the decennial census "long form" which collects the demographic, social, economic, and housing information required by hundreds of federal laws and court cases. The data collection for the ACS will occur throughout the decade rather than just once in ten years. The DP Team applies complex programs to the unedited ACS data to make the data consistent and complete. The programs look at the housing and population variables according to a predetermined hierarchy. They examine the data for inconsistencies and missing values where data should be present. During this process, the DP Team produces several reports in the form of SAS listings and/or datasets to aid the analysts in their review of the edited data. These reports can be several hundred pages long.

## PROBLEM

The analysts who review the data previously had to print or view an entire report even if they were only interested in a few pages pertaining to their variables of interest. Therefore, the analysts requested the DP Team to find an approach that would enable them to print or view only what they needed to see. The DP Team searched for a solution to satisfy their need for some time.

One solution was to provide them with SAS datasets against which they would write SAS programs to subset what they wanted. Since not all analysts are SAS literate this was not viable.

Another solution involved producing static HTML pages using ODS. This didn't solve the problem either because they are too long as well.

## SOLUTION

The term "SAS/IntrNet" came up repeatedly in NESUG and SUGI conferences. After taking "SAS Web Tools: Static and Dynamic Solutions Using SAS/IntrNet Software" offered by the SAS Institute, I saw a potential of SAS/IntrNet for our project.

The methods of accessing SAS systems from the Web fall into two major groups -- data services and compute services. The data services allow users to update and query data from a Web browser with SQL-type queries. SAS/IntrNet htmSQL is an example of this type. The compute services allow users to execute SAS programs from a Web browser. SAS/IntrNet Application Dispatcher is an example of compute services.

Selecting the method depends on the application and the experience of programmers in SAS and Web development. Although most of the application can be done using either method, certain applications require the Application Dispatcher. For example, one of our applications which uses PROC COMPUTAB to produce reports has to use the Application Dispatcher. The Application Dispatcher satisfied the needs of our entire product.

## REQUIREMENTS FOR PROGRAMMERS OR DEVELOPERS

### SOFTWARE

The SAS/IntrNet software is not included with the Base SAS software. You may have to purchase the software if you don't have it. During SAS/IntrNet software installation, the Application Broker should be installed on a Web server and the Application Server should be installed on a SAS server. The system administrators do the installation. You have to make sure that installation and setup are done correctly.

### SAS PROGRAMMING KNOWLEDGE

You should be able to write some basic SAS programs. You will be able to make a few changes to the program you are already using to produce the traditional listing reports. Some macro knowledge will also be very helpful to customize your program.

### HTML PAGE

You also should be able to create some simple HTML pages. These pages will allow the user to select some parameters that will be passed to the SAS program. Some knowledge of Javascript or some other scripting languages will help you to customize your HTML page; for example, to add a pop-up menu.

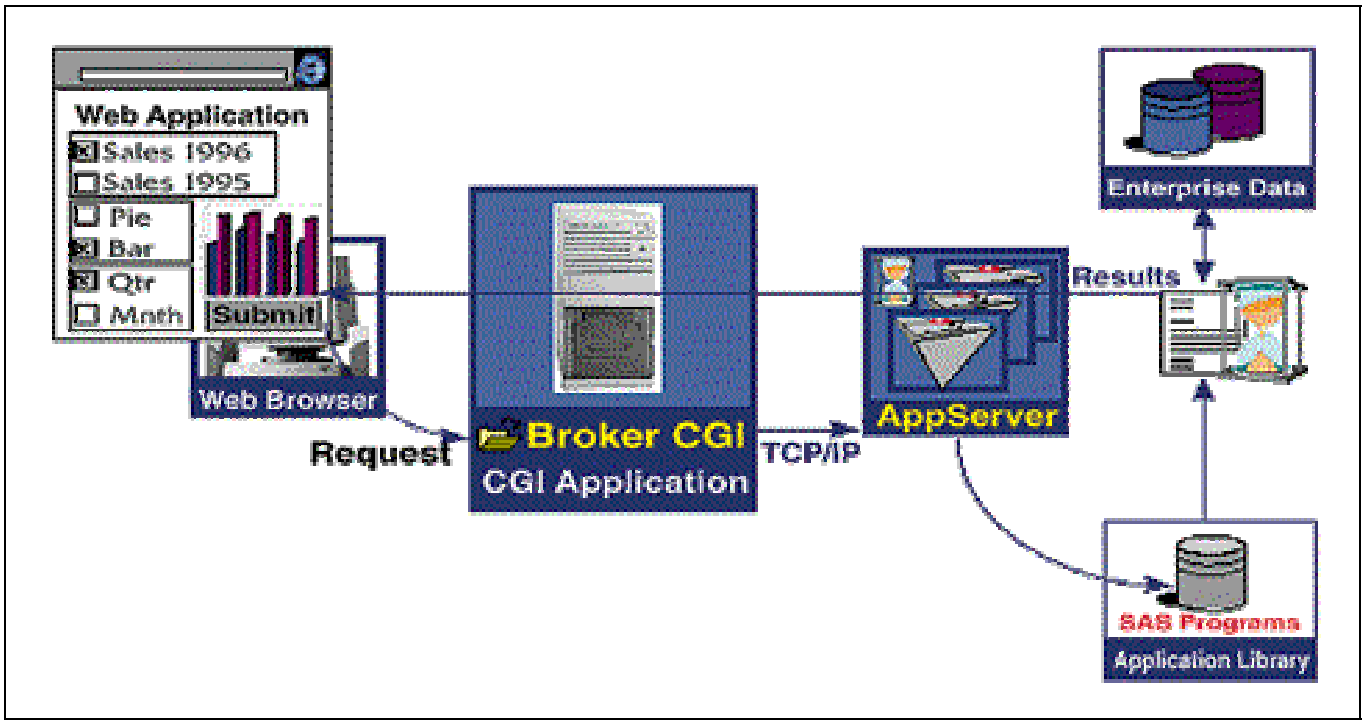
### WHAT THE USER NEEDS

The user will need only a Web browser. All of the users have a Web browser on their desktop computers. They only need to know the link to the IntrNet or website where the HTML pages reside. They don't need to write any SAS programs. They don't even need SAS software installed on their desktops.

## HOW DOES SAS/INTRNET WORK?

Application Dispatcher consists of two components. A Common Gateway Interface (CGI) program called the Application Broker resides on a Web server. The second component is the Application Server that is simply a SAS session running on a server that executes the SAS programs.

Web users fill out some fields in an HTML form from their Web browser and submit it. The information from the form is passed




*Note: The diagram above and most of the text in this section are taken from SAS Institute Web page "Application Dispatcher Version 2.0: How the Application Dispatcher Works". For more information visit the Web page http:/www.sas.com/rnd/web/intrnet/dispatch20/how.html.*

to the Web server, which invokes the Application Broker. The name of the Application Server that is to process the request and the name of the program to run are also sent from the Web form to the broker. These two items are normally hidden fields on the form that the user is unable to view from the HTML page. The broker passes the user selections along with the program name to the appropriate Application Server location that is defined in a configuration file.

The SAS program receives the parameters selected on the Web form as SAS macro variables that are resolved in the SAS program. The program is executed and the results are sent back to the user's browser.

## APPSTART.SAS PROGRAM

During the installation of the Application Dispatcher, a service name will be assigned for each service you establish. We named our service as ***cmo5-2dr***. You can take the system default by typing ***default***. The system administrators normally do the installation process.

Before building any application, the location of your SAS programs and the SAS data sets for your application must be defined on your ***service***. The service is started with SAS program name APPSTART.SAS. The APPSTART.SAS program contains a PROC APPSRV procedure that starts the service. The ALLOCATE FILE statement associates a SAS file reference with an external file or directory containing SAS programs and ALLOCATE LIBRARY statement associates a SAS library reference to SAS data library. The following statements will create an **asprgs** file reference and **asdata** will create a SAS dataset reference.

```
Allocate file asprgs '/export/home/sutha005';
Allocate library asdata '/export/home/sutha005';
```

Since the system administrators run the APPSTART.SAS program, they must have the location of your SAS programs and SAS datasets.

Html form creation

Once you have created the references using allocate statements, you are ready to create HTML pages. Only the main points of the HTML page construction will be discussed here. The details are included in Appendix 1.

The FORM tags as shown below are used to collect the parameters to be passed to the SAS program:

```
<FORM NAME = "tallyParam" METHOD=get
ACTION="../../scripts/broker.exe">
```

The ACTION above calls for the Application Broker Program. The path of the CGI program broker.exe should be correctly specified.

```
<TR>
<TD>Select the parameters for the report</TD>
</TR>

<TR>
<TD>Edit</TD>
<TD>
   <SELECT NAME="evar" >
      <OPTION VALUE="AGE">Age
      <OPTION VALUE="ANC">ANC
   </SELECT>
</TD>
</TR>
```

```
<TR>
<TD>State</TD>
<TD>
   <SELECT NAME="state" >
      <OPTION VALUE="001">Alabama
      <OPTION VALUE="002">Alaska
   </SELECT>
</TD>
</TR>
```

The value of the parameter **evar** is selected here. For example, the evar is AGE for edit variable age. The value of the parameter variable **state** is selected in the same way.

```
<TR>
<TD>
<INPUT TYPE=RESET VALUE="Clear Form">
<INPUT TYPE =SUBMIT VALUE="Submit">
</TD>
</TR>

<INPUT TYPE=HIDDEN NAME=_program
value=asprgs.test.sas>
<INPUT TYPE=HIDDEN NAME=_service VALUE=cmo5-2dr>
<INPUT TYPE=HIDDEN NAME=_debug value=2>
</FORM>
```

There are buttons to reset the form values and submit the values. The hidden inputs tags "_program", "_service" and "_debug" are very important. These tags are not displayed on the Web page. The input tag "_program" is the SAS program that will be run when submitted. The SAS program **test.sas** in the directory associated with **asprgs** will be submitted. Recall that **asprgs** was defined by the allocate file statement in the Application Dispatcher service **cmo5-2dr**. The input tag "_service" is the Application Dispatcher service. The input tag "_debug" supports debugging during the testing period. The _debug will be discussed later in detail.

## SAS PROGRAM

The SAS program looks much like a regular SAS program, but there are a few differences. The differences will be discussed here with a simple example. A complete program is included in Appendix 3.

```
libname pdata v6 '/adp5/web/2001/pop';
%global state evar;

%let st2 =%substr(&state,2,2);
ods html body = _webout (dynamic) rs=none;

%macro printrpt;

proc print data = pdata.tal01_&st2 label;
  var array cell descrip tally;
  format tally comma15.;
  by array;
  label descrip="Description";
  where type in ("T","H") and evar="&evar";
  title "ACS 2001 Tally Report for
         Edit=&evar and
         State=%sysfunc(fipnamel(&st2))";
run ;
%mend printrpt ;

%printrpt

ods html close ;
ods listing ;
```

This program uses ODS to create HTML output. The body option on the ODS statement sends the output to the "_webout" destination. The "_webout" is a predefined file reference that sends output back to the Web server and onto the Web browser. The HTML form sends the global macro variables **"state"** and **"evar"**. By using the "%substring" function, the "state" is converted from three to two digits **"st2"** to match the state code in the input dataset. The macro variable "**st2**" is used in selecting the state code on the input dataset and in the title statement. The WHERE statement in the PROC PRINT procedure resolves the macro variable "**evar**" to restrict the printed report to the **evar** selected.

## DEBUGGING

Debugging is important in any programming project especially during the testing period. You can use several debug values in the input tag "_debug" on your HTML page. Here are some of the values you can use:
"_debug=0" suppresses all debugging information

"_debug=1" echoes all fields provided to the Application Server.

"_debug=2" prints elapsed time at the end of the results.

"_debug=128" returns the SAS log as part of the results.

You can add the individual option values and use the total as the "_debug" parameter value to request multiple debugging options. For example, "_debug"=131 returns the SAS log, elapsed time and all fields provided to the Application Dispatcher.

## DOWNLOADING THE RESULTS TO A SPREADSHEET

Some of the users may want to download their report from the browser to a spreadsheet such as Lotus 123® or Microsoft Excel® for further analysis. SAS provides a tool to create a button on the HTML page to accomplish this need. The necessary instruction will be given to the reader to download the report when that button is clicked. Refer to Appendix 3 for detailed code.

## CONCLUSION

During our edit process for 2001 data, we produced the reports dynamically using SAS/IntrNet as explained in this paper. Unlike the traditional listing reports, the new reports are displayed on the Web browser in a user-friendly format and they take only a few pages when printed. Therefore, the analysts who review the data are pleased with this product that fulfills their longtime request.

Since SAS Institute introduced SAS/IntrNet software, SAS programmers all over the world have been creating new applications for their users to view reports or extract data from a

Web browser. SAS/IntrNet Application Dispatcher has proved to be a useful tool for turning traditional reports into easily accessed Web pages.

## APPENDICES

Appendix 1: The detailed HTML code

Appendix 2: The detailed SAS code for creating dynamic Web pages and downloading to a spreadsheet.

## REFERENCES

SAS Institutes Instructor Based Training Course Notes:
*SAS Web Tools: Static and Dynamic Solutions Using SAS/IntrNet*
*SAS Web Tools: Advanced Dynamic Solutions Using SAS/IntrNet*

SAS Institutes:
*SAS/IntrNet Software: A roadmap*

SUGI 26 Paper 5-26:
*The Basics of Dynamic SAS/IntrNet Applications by Roderick A. Rose*

SUGI 27 Paper 182-27:
*Connecting SAS The Basics of Dynamic SAS/IntrNet Applications by Roderick A. Rose*

## ACKNOWLEDGMENTS

I would like to thank the following people:

1. The DP Team Leader Barbara Diskin for encouraging me to write this paper and reviewing it.
2. The DP Team members John Stiller, Tony Tang and Jonah Turner for their valuable contribution to this project.
3. The instructor Warren Repole from SAS Institute for the course *SAS Web Tools: Static and Dynamic Solutions Using SAS/IntrNet* for answering questions on the subject.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Arumugam Sutha
US Census Bureau
DSD/CMO, Room 1657
Washington DC 20233
Work Phone: (301)-763-5400
Email: arumugam.sutha@census.gov

# APPENDIX 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head><title>ACS Housing Tally Report</title></head>
<body><br><br>
<center>
   <h1>2001 ACS Edit Analysis Reports</h1>
   <h2 class = "title2">Housing Tally Report</h2>
</center>
<hr><br><center>
   <form name = "tallyParam" method=get action="../../sasscripts/broker.exe">
   <table cellpadding = 5 width = 600>
     <tr>
         <td colspan = 2><b>Please select the parameters for the tally report</b></td>
      </tr>
      <tr>
         <td width = "10%">Edit</td>
         <td>
            <select name="evar">
                      <option value="TVAC" select>Vacancy
                      <option value="TACR">Lot Size
                      <option value="TMH">Mobile Home
            </select>
         </td>
      </tr>
      <tr>
         <td>
            State
         </td>
         <td>
            <select name=state>
               <option value="001" selected>Alabama
               <option value="002">Alaska
               <option value="055">Wisconsin
               <option value="056">Wyoming
       </select>
         </td>
               </tr>
               <tr>
         <td colspan = 2>
                              <input class="LargeButtons" type=reset value="Clear Form">
                              <input class="LargeButtons" type=submit value="Submit the Values"
name="submit">
         </td>
      </tr>
   </table>
      <input type=hidden name=_program value=drprgs01.testhtalrptcsv.sas>
               <input type=hidden name=_service value=cmo5-2dr>
               <input type=hidden name=_debug value=2>
   </form></center>
   <hr><br><center>
   <form>
      <INPUT class="LargeButtons" TYPE="Button" VALUE="Go Back to Report Selection"
               onClick="parent.location='indexcsv.html'">
   </form></center><br>
</body>
</html>
```

Only three edit variables are listed here.

Only four states are listed here.

# APPENDIX 2

```
/**     ACS Edit Analysis reports Using SAS/IntrNet          **/
/**     Tally Report Selection                                                  **/

libname hdata v6 '/adp5/edtreps_web/2001/hse';
%global state evar  ;

%let fnote = <form> <INPUT TYPE="Button" VALUE="Back"
onClick="parent.location='javascript:history.back()'"> </form> ;
%let st2 = %substr(&state,2,2) ;
%let report = TALLY ;
options missing=' ' mprint;
ods listing close ;
ods html body = _webout (dynamic title='ACS REPORTS' no_top_matter no_bottom_matter)
         style = statdoc rs = none  ;
%let st=&st2;
%let var=&evar ;
%macro printrpt ;
proc print data = hdata.tal01_&st2 noobs split='*' ;
var array cell descrip tally ;
format tally comma15. ;
by array;
label descrip="Description" ;
where type in ( "T", "H" ) and array =: "&evar" ;
title "ACS 2001 Housing Tally Report for Edit=&evar and State=%sysfunc(fipnamel(&st2)) " ;
footnote3 &fnote;
run ;
%mend printrpt ;
%printrpt
ods html close ;
ods listing ;
/* Create a session  */
%let rc=%sysfunc(appsrv_session(create)) ;

/* Create data set saved with the session */
data save.csvdata ;
   set hdata.tal01_&st2  ;
   keep array cell descrip tally ;
   format tally comma15. ;
   by array;
   label descrip="Description" ;
   where type in ( "T", "H" ) and array =: "&evar" ;
run ;
/* Generate CSV link and terminate output stream   */
data _null_ ;
   file _webout ;
   csvtag='<A HREF="'||trim(symget('_thissession'))||'&_program='||"&_pgmlib..csvroutes.sas"||
          '&csvfile='||"&report-&evar-&state"||'">Download as a Spreadsheet (CSV) File</A>';
   put csvtag ;
run ;
```

/* This is a separate SAS program and it should be in the directory referenced by pgmlib */

```
/* csvroutes.sas                                */
/* Expected parameters */
%global csvfile ;
/* Send CSV output   */
```

%ds2csv(data=save.csvdata,conttype=y,contdisp=y, savefile=**&csvfile**..csv,
csvfref=_webout,runmode=s,openmode=replace) ;