Paper AD05
SAS High-speed Automated Reporting Queue (SHARQ)

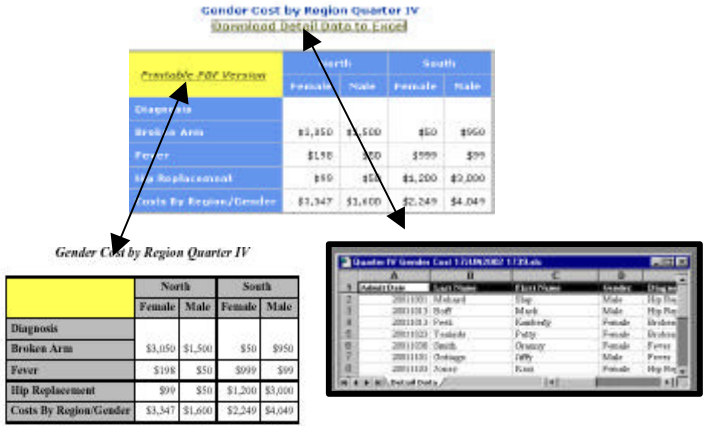
Eric Puhlman, BlueCross BlueShield of Tennessee, Chattanooga, Tennessee
Kasi Peek, BlueCross BlueShield of Tennessee, Chattanooga, Tennessee

Abstract

Take a bite out of time with SHARQ – Completely automate standard reports using PC SAS 8.2. BlueCross BlueShield of Tennessee (BCBST) experienced substantial downtime when manually processing standard SAS reports. To reduce this downtime, all manual intervention was eliminated through the development of an automatic SAS reporting system called SHARQ. SHARQ extracts data from multiple sources, manipulates extracted data, and outputs final report information in multiple formats – all without manual intervention. Task scheduling, batch mode processing, and macro programs developed and/or customized as "tools" will be presented. These tools provide automated date routines for program execution, error code processing with automatic program termination and email notification to developer(s), and automated final report output in multiple formats (e.g. MS Excel/Word via DDE, HTML using ODS). Finally, SAS environment configuration issues that help standardize program coding will be addressed.

Introduction

SAS environment configuration, SAS reporting code, Macro programs, and Windows NT scheduler are integrated to create SHARQ. This paper will provide instructions, programming code, and examples to produce SHARQ output (see Figure 1). At the end of this paper, a sample program that demonstrates report automation from start to finish is provided. SHARQ is a reliable way to automate reporting using PC SAS V8.2 in the Windows NT environment. SHARQ code and concepts can be applied to other operating systems (e.g., Windows 2000); however, this paper presents automation code/examples in the Windows NT environment.



	North		South	
	Female	Male	Female	Male
Diagnosis				
Broken Arm	$3,050	$1,500	$50	$950
Fever	$198	$50	$999	$99
Hip Replacement	$99	$50	$1,200	$3,000
Costs By Region/Gender	$3,347	$1,600	$2,249	$4,049

Figure 1 Sample Output from SHARQ

SAS Environment Configuration

Storage configuration, a group level autoexec, and a userauto macro (along with other macro programs) are combined to develop the production/test environment configuration. The presented concepts can be modified and/or incorporated when customizing the SAS environment.

Storage Configuration

The production environment requires assigned space to store automated standard reports and related information. However, automated standard reports must be created and tested outside the production environment; therefore, separate space is required to create and test SAS code. Also, space should be available for related testing information during development.

The production environment directory should have disk space available for documentation, SAS examples**,** shared macros, datasets, formats, SAS production programs, SAS reports, and web reports. Information stored in each of these directory folders include:

- Documentation – user documentation for standard reports,
- SAS Examples – SAS code examples that can be used by other developers,
- Prodmacs – stored compiled macros set to READ-ONLY,
- Datasets – permanent datasets from production jobs,
- Formats – formats for data commonly used in production jobs,
- SAS Production Programs – production standard report code and related bat files,
- SAS Reports – archived copies of production report output, and
- Web Reports - web output for transfer to web server (if you do not have specific write capabilities to the Web).

The test environment directory should provide disk space for datasets, Mymacs, SAS production programs, SAS reports, and web reports. Information stored in each of these directory folders include:

- Datasets – permanent datasets generated from development jobs,
- Mymacs – user macros created and/or tested during development,
- SAS Production Programs - testing standard report code and related BAT files,
- SAS Reports – test emulation of production SAS report storage, and

- Web Reports - test emulation of production web output for transfer to web server (if you do not have specific write capabilities to the Web).

Essentially, a mirror image of the production environment is created for the test environment. Separate disk drives for production/test environments are recommended; but, a single disk drive, with separate paths differentiating the production environment from the test environment, may be sufficient.

Group Level Autoexec

Autoexec files are processed immediately after the SAS System initializes but before it processes any source statements[1]. Therefore, the autoexec file should include filenames, libnames, and option statements necessary to invoke an automated SAS environment.

In the autoexec, a directory is required for use as an autocall library for user created macros. In addition, libraries are required for production/global macros and common formats

Local macros should be stored as *name.sas* and the respective fileref should be listed in the SASAUTOS. Production/global macros, which are stored compiled macros in a user-defined directory, become available when MSTORED and SASMSTORE options are specified. This library should be READ ONLY and all users will have to exit SAS when a new production/global macro is added. Common formats should also be stored in a user-defined library. These formats become available when the FMTSEARCH option is specified.

The following is sample SAS code for a group autoexec:

```
filename mymacs 'drive:\directory file\folder\mymacs';

libname prodmacs 'drive:\directory file\ folder \Shared Macros';

libname cmnfmts 'drive:\directory file\ folder \formats';


OPTIONS MSTORED SASMSTORE=prodmacs
FMTSEARCH=(cmnfmts);

OPTIONS SASAUTOS=(mymacs,
                  '!sasroot\connect\sasmacro',
                  '!sasroot\sasapps\macros', . . .
                  '!sasroot\usage\sasmacro');
```

Note: The standard group-level autoexec should be distributed to all developers.

Userauto Macro

To customize SHARQ, the following code is added to the bottom of the AUTOEXEC file. The following userauto macro allows developers to switch between the test and production environments.

```
%MACRO USERAUTO;
    FILENAME userauto '!sasroot\userauto.sas';
    %IF not &sysfilrc %THEN %DO;
    %INCLUDE '!sasroot\userauto.sas';
    %END;
%MEND;

%USERAUTO;
```

The userauto.sas file is located in the same directory as the autoexec.sas - C:\Program Files\SAS Institute\Sas\V8. Issuing the **%TEST** macro in the userauto.sas file sets the programming environment to the test environment when SAS starts. The following code invokes the test environment macro:

```
%TEST(group=group_folder,
        webdir=group_web_output_directory);
```

- Group - parameter helps set the fileref DATA
- Webdir - parameter sets the web output directory

A reference to the fileref DATA points to datasets stored in the datasets directory in your test environment, (e.g., Test Drive:\Directory\datasets). If permanent datasets are not required, then temporary datasets can be used as an alternative.

Reference to **&web** directs output to the directory set up for the test environment web output (e.g., Test Drive:\ Directory\web_reports)**.** The statement: **filename htmout "&web\report1\report1.htm"** indicates that a folder has been created for a new report being tested named Test Drive:\ Directory\web_output\report1, and web output, while testing, will be stored in this directory.

Issuing the **%PROD** macro in the userauto.sas sets the programming environment to the production environment when SAS starts. A reference to the fileref DATA points to datasets stored in the datasets directory in the production environment, (e.g., Production Drive:\Directory\datasets).

Reference to **&web** directs output to the directory set up for the production environment web output (e.g., Production Drive:\Directory\web_reports). The statement: **filename htmout "&web\report1\report1.htm"** indicates that a folder has been created for a new report named Production Drive:\ Directory\web_output\report1, and web output will be stored in this directory.

This approach allows developers to seamlessly transition test programs to production programs, especially when modifications and/or changes are needed in the future.

SAS Reporting Code

This section discusses automating common manual code conversions in data extracts, data manipulation and data output.

Data Extracts

Often data are extracted from multiple data sources such as text files, MS Excel files, MS Access databases, Sybase or DB2. Each source must be configured and coded without the use of manual data imports. For instance, to automatically read in a MS Access database, PROC IMPORT or ODBC technology can be used. To use ODBC technology the SAS/ACCESS must be available. ODBC technology is a very dynamic option as it interacts with many types of data sources such as Sybase, Oracle, DB2, Text, AS/400 and many more. Other files such as MS Excel can be imported automatically using Dynamic Data Exchange (DDE), PROC IMPORT or PROC DBLOAD database connection statements. In addition, SAS/ CONNECT can be used to connect SAS to any TSO host via the TCP access method using full screen 3270 telnet processing. When using SAS/CONNECT, scripts may require modifications to account for local "flavors" of MVS/TSO environments. The following are SAS code examples using SAS/Access, DDE, and SAS/CONNECT.

SAS /ACCESS – Sample extracting data from Sybase

```
proc sql noprint;
connect to odbc as getdss1(dsn='SYBPDSS1' uid=XXXXXXX
pwd=XXXXXXX);
create table cmemp as
select *
from connection to getdss1(
select  HCS_REF_ID as authno,
        HCS_ID as empid,
        HCS_MCTR_REAS as rsn,
        HCS_MCTR_CPLX as cplx,
        HCS_ACTIV_DT as initdt
from HCS_ACTIVITY
where HCS_MCTR_REAS in ('AP','CO','PI','PE')
  and HCS_ACTIV_DT >= &start
  and HCS_ACTIV_DT <= &end);
%put &sqlxmsg;
disconnect from getdss1;
quit;
```

DDE – Sample extracting data from network

```
OPTIONS NOXWAIT NOXSYNC;
data _null_;
call system('c:\progra~1\micros~1\office\excel.exe');
run;
DATA _NULL_; rc=SLEEP(6); RUN;
FILENAME ddecmds DDE 'excel|system';
DATA _NULL_;
 FILE ddecmds;
 PUT '[Open("Drive:\PopMgt\UM Report Data.xls")]';
run;
DATA _NULL_; rc=SLEEP(5); RUN;
filename bcemplye
 dde 'Excel|BlueCare!R2C2:R201C3';
data bcemplye;
 infile bcemplye dlm='09'x notab dsd missover;
 informat name $50. empid $8.;
 input name empid;
run;
```

SAS/CONNECT – Sample extracting data from IBM mainframe

```
%signon(MF,file);  /* References mainframe signon script*/
rsubmit;
proc sql noprint;
connect to db2(ssid=dsn);
create table          ext as
select *
from connection to db2(
select  a.mbr_no,
        a.admsn_cmr_cd as outcome,
        a.appl_rvw_dt_2 as rev_dt,
        b.frst_nme as fname,
        b.mid_initl_nme as mi,
        b.lst_nme as lname,
        b.rec_seq_no as seq
from da_cmr a, da_mbr b
where (a.mbr_no = b.mbr_no)
  and (a.admsn_cmr_cd <> 'VD')
  and (a.appl_rvw_dt_2 between &start and &end));
%put &sqlxmsg;
quit;
```

Logic, requiring limiting date fields, must use automatically calculated dates. This is done through stored and compiled macros or creating automatic macros variables with calculated date(s) prior to data extract(s). An example of automated date code using automatic macro variables is provided:

```
data dates(keep=begday endday);
startmo = (month(today()))- 1;
if startmo = 0 then startmo = 12;
startdy = '01';
if startmo = 12 then startyr = (year(today())) - 1;
 else startyr = year(today());
begday =
"'"||trim(left(startmo))||'/'||startdy||'/'||trim(left(startyr))||"'";
endmo = month(today());
enddy = '01';
endyr = year(today());
eday =
(input(trim(left(endmo))||'/'||enddy||'/'||trim(left(endyr)),mmddyy10
.) - 1);
endday = "'"||put(eday,mmddyy10.)||"'";
run;

data _null_;
 set dates;
       call symput('start',begday);
       call symput('end',endday);
run;
```

Note: All automated date routine macros are designed to output the macro variables &start and &end for subsequent use in data extraction. This is another way to ensure seamless transition between test and production environments as well as change the reporting period.

There are many ways to extract data; therefore a developer must ensure all parameters of extracts are completed without *any* need for manual changes. Also, ensure maintenance/

updates on external data sources are completed prior to scheduling automated code.

Data Manipulation

Data are manipulated to summarize, reduce, or otherwise transform raw data into meaningful and useful information[2] and stored in a SAS dataset. For the most part, data manipulation programming is seamless when developing automated code. However, there are coding procedures necessary to remove manual intervention when manipulating data.

Since data are extracted from multiple sources (e.g., MS Excel, DB2, and Sybase) and combined to create datasets, these datasets must have capability to be transitioned between local hosts and remote hosts for further manipulation. For example, if data from the DB2 subsytem (remote host) are extracted and/or massaged to merge with other data from Sybase (local host), then DB2 data must be transitioned to the local host or the Sybase data must be transitioned to the remote host before these datasets can be merged/concatenated.

The primary method to transition data between the local host and remote host is with PROC Upload and PROC Download. PROC Upload transitions data from the local host to the remote host while PROC Download transitions data from the remote host to the local host. Both procedures require SAS/Connect. The following are SAS code examples of each procedure.

PROC Upload

```
%signon(MF,file);
rsubmit;

proc upload data=perm.lcldata out=work.rmtdata;
run;
```

PROC Download

```
%signon(MF,file);
rsubmit;
options db2ssid=dsn;

proc sql noprint;
connect to db2(ssid=dsn);
create table work.rmtdata as
select *
from connection to db2(
select  frst_nme as fname,
        lst_nme as lname,
        mid_initl_nme as mi
from da_mbr);
%put &sqlxmsg;
quit;

proc download data=rmtdata out=perm.lcldata;
run;
endrsubmit;
```

After transitioning data between remote and local hosts, additional SAS procedures/statements can be used for newly created data. However, data processing will occur on the host to which data are transitioned.

Other coding procedures to consider involve column formats. When extracting data from other applications/systems, sometimes SAS is forced to assign default formats to columns. These columns often require modification before data can be manipulated. For example, if a date column on a MS Excel spreadsheet is formatted CCYYMMDD, then SAS will read this column as numeric 8 or character 8 when using DDE (these are the only formats available to assign). Once this column is in a SAS dataset, it must be converted to a SAS date format.

There are data that appear incapable of manipulating automatically even after every "work-around" has been attempted; but solutions are often available through networking with other SAS programmers or through SAS technical support.

Data Output

Once data are extracted and manipulated automatically, data are ready to output for presentation. Most end users prefer output in formats they are accustomed to using (e.g., MS Excel or HTML). In addition, some end users prefer summary level output while other end users prefer detail level output – often from common data sources. This section discusses automating output to HTML, PDF, and MS Excel.

Automating output to HTML is made easy with SAS V8.2 Output Delivery System (ODS). ODS opens, manages, or closes the HTML destination. If the destination is open, you can create HTML output[3]. When creating an HTML file, SAS dataset(s) can be manipulated for presentation with procedures such as PROC Print, PROC Tabulate, PROC Report, PROC Means and many more. SAS code, creating a HTML file using PROC tabulate, is provided in "Create a html file using PROC Tabulate" section of the Sample Program.

Similarly to HTML, PDF files are created automatically using ODS. Datasets sent to HTML can be sent to PDF which provides a user friendly printable file. SAS code, creating a PDF file using PROC Tabulate, is provided in "Create a PDF file to link printable version of the TABULATE procedure to the html file" section of the Sample Program.

SAS can automatically send output to MS Excel in various ways. When end users prefer customized report layouts, a macro[4,5], using DDE, is the primary method for creating customized MS Excel output. MS Excel formats are programmed in SAS and include formats such as right header/center header/left headers, align left/center /right, bold, italic and many other formats. Programming code for this macro is provided in Macro 3 of the Macro Program section.

Although HTML, PDF and MS Excel are the only outputs discussed, many other output files can be automated using ODS, PROC Export, PROC DBLoad and DDE.

Macro Programs

Production macros increase a developer's capability to standardize processes and monitor quality of programs. Macros included in this paper allow programmers to call the production/test environment, export SAS data to MS Excel, ensure dataset(s) are populated, and check for programmatic errors.

The following macros provide a base structure for automating SAS programs. To implement automation of the sample code provided at the end of this section, the following macros should be copied and submitted in PC SAS v8.2.

Note: Macro code will have to be modified for your environment.

Macro 1

/*Environment Configuration Macros*/

```
%macro test(group=MIM,
            webdir=F:\MIM\web_output);
option nomprint nosymbolgen nomlogic nomacrogen;
libname data "F:\&group\datasets";
%global web;
%if %upcase (&group) ^= MIM and &webdir = F:\MIM\web_output
%then
%let web=F:\&GROUP\web reports;
%else %do;
        %global web;
        %let web=&webdir;
%end;
%mend;
```

Specify the directories for your test datasets and test web output.

```
%macro prod(group=MIM,
            webdir=\\webserver\MIM\web_output);
options nomprint nosymbolgen nomlogic nomacrogen;
  libname data "U:\Share\&group\datasets";
%if %upcase (&group) ^= MIM and &webdir =\\webserver\MIM\web
reports %then
  %let web=F:;
%else %do;
        %global web;
        %let web=&webdir;
      %end;
%mend;
```

Specify the directories for your production datasets and production web output. See the section 'Userauto.sas' or simply invoke one or the other in a sas session.

Macro 2

/* Macro to Count Obs in a dataset */

```
%macro numobs(dsn);
%global num;
 data _null_;
  if 0 then set &dsn nobs=count;
  call symput('num',left(put(count,8.)));
  stop; run;
%mend numobs;
```

Macro 3

/* Macro to output data to Excel*/

```
/*****************************************************************
**              EXPORT DATA TO AN EXCEL FILE             **
*****************************************************************/
OPTIONS noMPRINT noMLOGIC noMACROGEN noSYMBOLGEN;
%macro out_xl (noparm,sasin=,savedir=f:\,xlsrpt=,
savefile=Sas2excl Report,datestmp=Y,timestmp=Y,shtname=,
colhead=LABEL,lfthdr=,ctrhdr=Report Results,rgthdr=,
lftftr=,ctrftr=Page %nrstr(&p),rgtftr=%nrstr(&d),
grid=Y,specfmt=,savecopy=,openexcl=Y,closexcl=Y );
%* Check macro parameters.       *;
%if &noparm >= and  &sasin= %then %do;
data _null_;
 put @2 '84'x 80*'83'x  @82 '86'x;
 put @2 '82'x @07 '%OUT_XL.sas, parent example presented at SUGI
17. ' @82'82'x;
 put @2 '82'x @07 "Send data via DDE(Dynamic Data Exchange) from
SAS to EXCEL." @82'82'x;
 put @2 '82'x @07 "This version has been modified to fit in this paper."
@82'82'x;
 put @2 '8a'x  17*'83'x @14 '8b'x 67*'83'x @82 '8c'x;
 run;
%goto exitout;
%end;
%if &shtname=  %then %let SHTNAME=%upcase(&sasin);
%* Open file to contain Excel commands. *;
filename cmdexcel dde 'excel|system';
%* Start Excel *;
%* Use the system registry to open excel instead of specifying a  *;
%* specific amount of time to put SAS to sleep. More efficient.   *;
%if %upcase(&openexcl) eq Y %then %do;
options noxwait noxsync;
data _null_;
  length fid rc start stop time 8.;
  fid=fopen('cmdexcl','s');
  if (fid le 0) then
    do;
      rc=system('start excel');
      start=datetime();
      stop=start+10;
      do while (fid le 0);
        fid=fopen('cmdexcl','s');
        time=datetime();
        if (time ge stop) then fid=1;
      end;
    end;
  rc=close(fid);
run;
%end;
%* Open exsisting file or new file with 1 sheet *;
%if &xlsrpt= %then
%do;
     data _null_;
       file cmdexcel;
       %let nw=%str(%'[New(1)]%');
       put %unquote(&nw);
     run;
%end;
%else %do;
   data _null_;
     file cmdexcel;
     %let filepath=%str(%'[open("&xlsrpt")]%');
     put %unquote(&filepath);
     %let sheet=%str(%'[Workbook.insert(1)]%');
```

```
     put %unquote(&sheet);
    run;
%end;
%* Save the file to its new name. *;
%local svcopy;
%if &xlsrpt= %then
%do;
    %let h=%sysfunc(hour(%sysfunc(time())),z2.);
    %let tm=%sysfunc(time());
    %let m=%sysfunc(minute(&tm),z2.);
   %if %upcase(&datestmp)=Y and %upcase(&timestmp)=N %then
   %do;
    %let str=&SAVEDIR\%nrbquote(&savefile) &sysdate9;
    %let linkit=%nrbquote(&savefile) &sysdate9..xls;
    %if &savecopy ne %then %let svcopy=&savecopy&linkit;
   %end;
   %if %upcase(&datestmp)=N and %upcase(&timestmp)=Y %then
   %do;
    %let str=&SAVEDIR\%nrbquote(&savefile) &h.&m;
    %let linkit=%nrbquote(&savefile) &h.&m..xls;
    %if &savecopy ne %then %let svcopy=&savecopy&linkit;
   %end;
%if %upcase(&datestmp)=Y and %upcase(&timestmp)=Y %then
   %do;
    %let str=&SAVEDIR\%nrbquote(&savefile) &sysdate9 &h.&m;
    %let linkit=%nrbquote(&savefile) &sysdate9 &h.&m..xls;
    %if &savecopy ne %then %let svcopy=&savecopy&linkit;
   %end;
%if %upcase(&datestmp)=N and %upcase(&timestmp)=N %then
   %do;
    %let str=&SAVEDIR\%nrbquote(&savefile);
    %let linkit=%nrbquote(&savefile).xls;
    %if &savecopy ne %then %let svcopy=&savecopy&linkit;
   %end;
%end;
%else
  %do;
     %let str=&xlsrpt;
/* Specify SAVEDIR= where a savecopy will be */
/* performed. It is needed to calculate linkit */
     %let dirlen=%sysfunc(length(&SAVEDIR));
     %let rptlen=%sysfunc(length(&xlsrpt));
     %let
linkit=%sysfunc(substr(%str(&rptname),%eval(&dirlen+1),%eval(&rptlen-
&dirlen))).xls;
     %if &savecopy ne %then %let svcopy=&savecopy&linkit;
   %end;
%* Create var &rptname in case more sheets will be added  *;
%* to the same report and for  html output if needed.            *;
%* Create macro variable &linkrpt to use in HTML href.       *;
%global rptname; %global linkrpt;
data _null_;
file cmdexcel;
put '[error(false)]';
put "[save.as(%bquote("&str"))]";
call symput('rptname',"&str");
call symput('linkrpt',"&linkit");
run;
%* Rename the worksheet and move it to the last position *;
 /* Insert a Macro sheet */
    data _null_;
    file cmdexcel;
      %let sheet=%str(%'[Workbook.next()]%');
      put %unquote(&sheet);
      %let sheet=%str(%'[Workbook.insert(3)]%');
      put %unquote(&sheet);
      put '[error(false)]';
    run;
  /* Define a DDE-triplet fileref pointing to the Macro Sheet */
    filename xlmacro dde 'excel|macro1!r1c1:r100c1' notab lrecl=200;
  /* Write a Macro on the Macro Sheet and run it */
    data _null_;
      file xlmacro;
      %let sheet=%str(%'=Workbook.name("Sheet1","&shtname")%');
      put %unquote(&sheet);
          %let
shtmove=%str(%'=Workbook.move("&shtname","&linkrpt")%');
      put %unquote(&shtmove);
      put '=halt(true)';
      put '!dde_flush';
      file cmdexcel;
      put '[run("macro1!r1c1:r100c1")]';
      put '[error(false)]';
    run;
 /* Select the sheet for processing */
  data _null_;
     file cmdexcel;
     %let sheet=%str(%'[Workbook.select("&shtname")]%');
     put %unquote(&sheet);
  run;
%* Info about SAS dataset. Include varnum and sort by  *;
%* varnum in order for the cols to come out in excel in the  *;
%* order they appear in the dataset. *;
proc contents data=&sasin noprint
out=conts(keep=nobs name label length varnum type);
run;
%numobs(conts);   /* Calling another macro */
%if &num=%eval(0) %then
%do;
    data _null_;
      put 'Parameter error: SASIN= ' "&sasin";
      put 'DATASET missing or empty';
    run;
%* If the conts dataset is empty, there is no data. The sheet is produced
and the program ends;
       %goto cleanup;
%end;
proc sort data=conts; by varnum; run;
data conts;
set conts end=eof;
if label eq '' then
label=NAME;
call symput('col'||left(_n_),trim(NAME));
call symput('lab'||left(_n_),trim(LABEL));
call symput('lablen'||left(_n_),length(LABEL));
call symput('len'||left(_n_),length);
if NAME='PRVTIN' then type=2;
call symput('type'||left(_n_),type);
if eof then do;
call symput('columns',trim(left(_n_)));
call symput('rows',trim(left(nobs)));
end;
run;
%*  Number of data rows exceeds DDE to Excel limit     *;
%if &rows > %eval(16384) %then
%do;
 filename excel1 dde "excel|&shtname!r1c1:r8c5";
 data _null_;
    file excel1 notab;
    put "Dataset is to large to transfer using SAS2EXCL macro/DDE.";
        put "Limit is 16384 rows of data";
        file cmdexcel;
         %let ColA=%str(%'[column.width("54")]%');
       put %unquote(&ColA);
run;
 %goto cleanup;
%end;
%* Link to Excel spreadsheet   *;
```

```
%if %eval(&columns) > %eval(20) %then
  %do;
    %let first=%sysfunc(int(%eval(&columns/2)));
    %let more=%eval(&first+1);
    filename excel1 dde "excel|&shtname!r1c1:r1c&first" notab;
    filename excel2 dde "excel|&shtname!r1c&more:r1c&columns" notab;
  %end;
%else %do;
  filename excel1 dde "excel|&shtname!r1c1:r1c&columns" notab;
  %let more=%eval(0);
  %let first=&columns;
%end;
%* send column headings to spreadsheet *;
data _null_;
file excel1;
hextab='09'x;
%if %upcase(&colhead) eq NAME %then %do;
put %do i=1 %to &first;
"&&col&i" hextab
%end;
%end;
%else %if %upcase(&colhead) eq LABEL %then %do;
put %do i=1 %to &first;
"&&lab&i" hextab
%end;
%end;
run;
%if %eval(&more) > 0 %then
%do;
    data _null_;
    file excel2;
    hextab='09'x;
    %if %upcase(&colhead) eq NAME %then %do;
    put %do i=&more %to &columns;
    "&&col&i" hextab
    %end;
    %end;
    %else %if %upcase(&colhead) eq LABEL %then %do;
    put %do i=&more %to &columns;
    "&&lab&i" hextab
    %end;
    %end;
    run;
%end;
%* If variable type is numeric then format that column number *;
%* If variable type is character then format that column text *;
data _null_;
  file cmdexcel;
  %do i=1 %to &columns;
   %if &&type&i=1 %then
    %do;
        %let selcol=C&i;
        %let colpath=%str(%'[select("&selcol")]%');
        put %unquote(&colpath);
        put '[format.number("General")]';
    %end;
       %if &&type&i=2 %then
    %do;
        %let selcol=C&i;
        %let colpath=%str(%'[select("&selcol")]%');
        put %unquote(&colpath);
        put '[format.number("@")]';
    %end;
 %end;
 run;
%* Link to spreadsheet to send data beginning 2nd row *;
filename excel dde
"excel|&shtname.!r2c1:R%eval(&rows+1)C&columns." notab lrecl=350;
%* now send data to Excel spreadsheet    *;
data _null_;
set &sasin;
hextab='09'x;
file excel;
put %do i=1 %to &columns;
&&col&i hextab
%end;
run;
%* Adjust column width to variable length or label length *;
%let calc=%eval(0);
data _null_;
  file cmdexcel;
  %do i=1 %to &columns;
    %let sel2=r1c&i:R%eval(&rows+1)C&i;
    %let path2=%str(%'[select("&sel2")]%');
    %let varlen=%eval(&&len&i+3);
    %if %eval(&&lablen&i) > %eval(&varlen) %then
      %let colwide=%eval(&&lablen&i+2);
    %else %let colwide=%eval(&varlen+2);
    %if &colwide > 35 %then %let colwide=36;
    %let wide=%str(%'[column.width("&colwide")]%');
       put %unquote(&path2);
       put %unquote(&wide);
%* Calculate the total for all the columns   *;
    %if %eval(&&lablen&i) > %eval(&varlen) %then
      %let calc=%eval(%eval(&&lablen&i)+%eval(&calc));
    %else %let calc=%eval(%eval(&varlen)+%eval(&calc));
  %end;
%let all=r2c1:R%eval(&rows+1)C&columns;
%let allp=%str(%'[select("&all")]%');
put %unquote(&allp);
put '[format.font("Times New
Roman",10,FALSE,FALSE,FALSE,FALSE,0,FALSE,FALSE)]';
run;
%* Format Header and Footer   *;
%let head=%nrstr(&L &10) &lfthdr %nrstr(&C &B &12) &ctrhdr
%nrstr(&R &10) &rgthdr;
%let foot=%nrstr(&L &8) &lftftr %nrstr(&C &8) &ctrftr %nrstr(&R &8)
&rgtftr;
%if %eval(&calc) > 95 %then
  %let orient=2;
%else %let orient=1;
%if %eval(&calc) > 125 and < 175 %then %let scale=70;
%else %if %eval(&calc) >= 175 and <  210 %then %let scale=55;
%else %if %eval(&calc) >= 210 %then %let scale=45;
%else %let scale=100;
%let paper=1;
%let sel3=r1c1:r1c&columns;
%let path3=%str(%'[select("&sel3")]%');
%if &grid=Y %then %let gridline=TRUE;
%else %let gridline=FALSE;
%let path4=%str(%'[page.setup("&head","&foot")]%');
%let
path5=%str(%'[page.setup(,,.25,.25,1,1,FALSE,&gridline,TRUE,FALSE,
&orient,&paper,&scale)]%');
data _null_;
 file cmdexcel;
put %unquote(&path3);
put '[format.font("Times New
Roman",10,TRUE,FALSE,FALSE,FALSE,0,FALSE,FALSE)]';
put %unquote(&path4);
put %unquote(&path5);
put '[SET.PRINT.TITLES("R1")]';
run;
%* If additional formatting needs required, the name of the user     *;
%* created macro is called with the &SPECFMT parameter.             *;
%if &specfmt ne %then
%do;
%&specfmt;
```

```
%end;
%else;
%* Close Excel *;
%CLEANUP:
/*  Make the Macro sheet the first sheet. Select it, delete it.  */
/*  Causes the next sheet, the first sheet on the left, to be the active
sheet.   */
data _null_;
file xlmacro;
%let shtmove2=%str(%'=Workbook.move("Macro1","&linkrpt",1)%');
 put %unquote(&shtmove2);
file cmdexcel;
 %let sheet2=%str(%'[Workbook.select("Macro1")]%');
 put %unquote(&sheet2);
 put '[workbook.delete("Macro1")]';
run;
proc datasets library=work; delete conts; run; quit;
%* Save copy of file to another directory if requested     *;
%if &savecopy ne %then
%do;
  data _null_;
   file cmdexcel;
     put "[save.as(%bquote("&svcopy"))]";
   run;
%end;
data _null_;
file cmdexcel;
%if %upcase(&closexcl) = Y %then %do;
  put '[error(false)]';
  put "[save.as(%bquote("&str"))]";
  put '[quit()]';
%end;
%else %do;
  put '[error(false)]';
  put "[save.as(%bquote("&str"))]";
%end;
run;
filename excel clear;
filename excel1 clear;
filename excel2 clear;
filename cmdexcel clear;
filename xlmacro clear;
%exitout:
%mend out_xl;
```

Macro 4
/* Error Checking*/

```
%macro errcheck(dsn,obs);
%if &sysparm= %then %let sysparm=Kasi Peek;
%if &dsn= %then %let dsn=&syslast;
%let db=no;
%let sql=no;
%let zero=no;
%let nods=no;
options nomprint nosymbolgen nomlogic nomacrogen;
%let contact=%scan(&sysparm,1) %scan(&sysparm,2);
filename rpt "U:\IntraDept_Shares\IM_Share\Im Health\SAS Production
Jobs\sharq_rpt_&sysjobid..sas";
/* Start Creating Email to send contact */
   data _null_;
     file rpt ;
       put %unquote(%str(%')filename errmail email "&contact"  %str(%'));
       put 'cc=("Kasi Peek" )';
put %unquote(%str(%')subject="SHARQ Automated Error Message" ;
%str(%'));
       put "data _null_ ;";
       put "file errmail;";
put "put 'S.H.A.R.Q. (SAS High-Speed Automated Reporting Queue) ';";
       put "put 'Automated Error Message' ;";
       put "put;";
       put "put '%bquote(&sysprocessname)';";
       put "put 'Run Time:  &sysdate9 &systime';";
       put "put;";
  run;
/* What dataset are we checking?*/
   data _null_;
    file rpt mod ls=80 flowover ;
       put "put 'CHECKING DATASET: %trim(&dsn)';";
       put "put ;";
  run;
/* Capture DB Error Messages */
%check(sysdbrc);
%if &chkvar=yes %then
%do;
  %if %trim(&varval) ^= 0 %then %do;
    data _null_;
      file rpt mod ls=80 flowover;
       put "put 'DB Error Messages:';";
       put 'put "SYSDBRC= &sysdbrc";';
       put 'put "SYSDBMSG=  &sysdbmsg";';
       put "put;";
     run;
        %let db=yes;
  %end;
%end;
/* Capture SQL Error Messages */
%check(sqlxrc);
%if &chkvar=yes %then
%do;
  %if %trim(&varval) ^= 0 %then %do;
    data _null_;
         file rpt mod ls=80 flowover;
       put "put 'SQL Error Messages:';";
       put 'put "SQLxRC= &sqlxrc";';
       put 'put "SQLxMSG= &sqlxmsg";';
       put "put;";
   run;
  %let sql=yes;
  %end;
%end;
/* Message regarding 0 obs in a dataset */
   /* check if dataset exists */
   %if %sysfunc(exist(&dsn)) % then
      %do;
             %if &obs ^= 0 %then
            %do;
/* 0 Observations is not OK */
      %numobs(&dsn)
      %if &num = 0 %then
         %do;
             data _null_;
               file rpt mod ls=80 flowover;
               put "put 'Critical Error: ';";
      put "put 'Dataset %trim(&dsn) has &num observations.';";
               put "put 'Program execution terminated.';";
             run;
    %let zero=yes;
     %end; %end; %else; %end;
    %else
        %do;
              data _null_;
                file rpt mod ls=80 flowover;
                 put "put 'Critical Error: ';";
                 put "put 'Dataset %trim(&dsn) does not exist.';";
                 put "put 'Program execution terminated.';";
              run;
 %let nods=yes;
```

```
      %end;
/* If Errors detected, send email. Delete when finished. */
%if &db=yes or &sql=yes or &zero=yes or &nods=yes %then
%do;
    data _null_;
      file rpt mod ls=80 flowover;
      put "run;";
    run;
%include "U:\IntraDept_Shares\IM_Share\IM Health\Sas Production
Jobs\sharq_rpt_&sysjobid..sas";
      %let filecmd=del %str(%")U:\IntraDept_Shares\IM_Share\IM
Health\Sas Production Jobs\sharq_rpt_&sysjobid..sas%str(%");
          data _null_;
             call system("&filecmd");
          run;
          quit;
      Endsas;
%end;
/* No Errors Detected. Delete the external file that was started to send
the email. */
%let filecmd=del %str(%")U:\IntraDept_Shares\IM_Share\IM Health\Sas
Production Jobs\sharq_rpt_&sysjobid..sas%str(%");
data _null_;
call system("&filecmd");
run; quit;
%mend;
```

Sample Program

If the previous macros have been modified for your environment and submitted, the following sample program will demonstrate automation.

```
/* Test Environment */
/* Create directory or use another */
%test(webdir=c:\sample);
/* Sample data for demonstration*/
data sample;
input date 1-8 lname $10-20 fname $21-32 gender $34-36 diag $38-48
costs 50-56 region $58-77; format costs dollar8.;
cards;
20011212 Klutz      William      M   Broken Arm  900     South
20020102 Klutz      Susan        F   Broken Leg  950     South
20020201 Klutz      Billy        M   Broken Arm  300     South
20011130 Bowens     Cathy        F   Hip Replace 99      North
20011023 Tenkids    Patty        F   Broken Arm  3000    North
20011215 Little     Tommy        M   Broken Arm  1500    North
20011201 Little     Susie        F   Fever       99      North
20011123 Little     Sally        F   Fever       99      North
20011013 Boff       Mack         M   Hip Replace 3000    South
20011101 Gottago    Jiffy        M   Fever       50      North
20011105 Wake       Staya        F   Broken Arm  50      South
20011001 Mehard     Slap         M   Hip Replace 50      North
20011202 Jones      Billy Bob    M   Fever       99      South
20011103 Jones      Kasi         F   Hip Replace 1200    South
20011013 Peek       Kimberly     F   Broken Arm  50      North
20011221 Stewart    Jeffrey      M   Broken Arm  50      South
20011030 Smith      Granny       F   Fever       999     South
;
/* Formats like this could be stored in a catalog/location */
/* for all developers to use */
proc format;
 value $gender M='Male'
               F='Female';
run;
/* Invoke automatic date routine */
/* Contact author for date routine code or use another */
%*quarter(quarter=4,system=hp);
%let start=20011001; %let end=20011231;
/* Extract data needed for report */
proc sort data=sample; by date; run;
data output;
 set sample;
 where date ge &start and date le &end;
 label date='Admit Date'
       lname='Last Name'
       fname='First Name'
       gender='Gender'
       diag='Diagnosis';
format gender $gender.;
run;
/* output data to Excel */
%out_xl(sasin=output, shtname=Detail Data,
        ctrhdr=Gender Cost by Region,
        savedir=&web, savefile=Gender Cost);
ods listing close;
/* A macro variable &linkrpt is created in the %out_xl */
/* macro so the excel file can be linked on the html page */
filename htmout "&web\Gender_Report.htm";
Title "<a>Gender Cost by Region</a>";
Title3 "<a href=""&linkrpt"">Download Detail Data to Excel</a>";
/* Create an html file using a proc tabulate */
ods html
file=htmout style=sasweb;
options nodate nonumber missing='';
proc tabulate data=output format=dollar12.;
 class region diag gender;
 table diag (all='Costs By Region/Gender'), (region='')*(gender='
')*(costs='' )*sum=''
   / box={label="Printable PDF Version"
    style=[font_style=italic
    background=light yellow
    font_weight=bold
    nobreakspace=off
    url="Gender_Report.pdf"]};
 var costs; keylabel sum='';
run;
ods html close;
%*Create PDF file to link printable version of the ;
%*  TABULATE procedure to the html file          ;
ods pdf file="&web\Gender_Report.pdf" notoc;
Title "Gender Cost by Region";
proc tabulate data=output format=dollar12.;
 class region diag gender;
 table diag (all='Costs By Region/Gender'),
(region='')*(gender='' )*(costs='' )*sum=''
   / box={label="" style=[font_style=italic  background=light yellow
          font_weight=bold  nobreakspace=off]};
 var costs;  keylabel sum=''; run; ods pdf close;  ods listing;
```

Windows NT Scheduler

Create a Batch File

The final step in automating SAS programs is scheduling batch file programs to invoke a SAS session and run the program. A batch file is created in NOTEPAD and is scheduled to invoke in the Task Scheduler. This file will initiate a SAS session and execute the SAS program.

A batch file requires 4 main commands

1.) "C:\Program Files\SAS Institute\Sas\V8\SAS.EXE" (Starts a Sas Session),

2.) -SYSIN "Weekly\Employee Production Report.sas (Indicates which program to run),

3.) -CONFIG "C:\Program Files\SAS Institute\Sas\V8\ SasV8.cfg (Indicates config file directory path), and

4.) -AUTOEXEC "C:\Program Files\SAS Institute\Sas\V8\autoexec.sas" (Indicates autoexec directory path).

The respective batch file is saved with a .bat extension (see Figure 2).

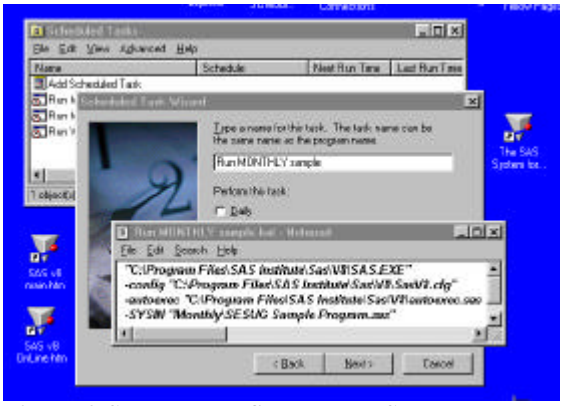


Figure 2 Sample .bat file and NT Scheduler

Schedule Batch Job

Windows NT Task Scheduler is used to schedule batch files. To schedule batch files in Windows NT, open Explorer then open Scheduled Tasks and select "Add Scheduled Task" - the Scheduled Task Wizard will provide guidance throughout the remainder of the process.

When the scheduled job runs, a SAS log is automatically created and stored in the directory specified as the 'Start in:' directory. *(To view properties of a scheduled job, right click the job and choose properties.)*

Conclusion

Creating an automated tool like SHARQ saves substantial time compared to manually submitting SAS programs. In addition, programmers can maximize development opportunities, as time resubmitting standard reports is no longer required. Furthermore, data resources can be used during low utilization periods - maximizing processing time.

Acknowledgements

Jeffrey C. Jones, MBA, BCBST – IM Health Informatics
Sheila D. Keith, BCBST – Medical Information Management
Kimberly E. Stewart, BCBST – IM Health Informatics
Robert E. Williams, BCBST – MCCAD

About the Authors

Eric Puhlman is currently employed as Information Project Analyst at BCBST with six (6) years of SAS programming experience. Eric is a team lead and is responsible for selecting appropriate clinical reporting methods for the design and analysis of studies, projects, and programs which focus on the impact cost and utilization of health care services on the economic, financial, and quality aspects of all lines of business.

Kasi Peek is currently employed as an IM Developer/Analyst at BCBST with six (6) years of SAS programming experience. She is responsible for the analysis, design and implementation of reporting methods, project management and user support in the development, enhancement, maintenance and integration of application software systems. Kasi served as a SESUG Section Co-Chair from 1999 through 2001.

Contact Authors

Eric Puhlman, Medical Information Management
BlueCross BlueShield, Tennessee
Work phone: 423-763-7077 e-mail: Eric_puhlman@bcbst.com

Kasi Peek, IM Health Informatics
BlueCross, BlueShield, Tennessee
Work phone: 423-763-3492 e-mail: Kasi_peek@bcbst.com

References

(1) SAS Institute Inc., SAS OnlineDoc *SAS Companion for the OpenVMS Operating Environment* (Cary, NC: SAS Institute Inc., 2000), Autoexec Files.

(2) SAS Institute Inc., SAS Fundamentals: A Programming Approach (Cary, NC: SAS Institute Inc., 1995), 5.

(3) SAS Institute Inc., SAS OnlineDoc *The ODS Statements* (Cary, NC: SAS Institute Inc., 2000), ODS HTML Statement.

(4) Koen Vyverman, "Using Dynamic Data Exchange to Export Your SAS ® Data to MS Excel — Against All ODS,,Part I —." Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference, paper 11 (2001).

(5) Peter Ruzza, SAS Technical Support (Cary NC: SAS Institute Inc., 2002), Dynamic Data Exchange.