MWSUG 2019 - Paper PO-023

BSAS Macro: True Batch Processing From SAS EG Using Load Sharing Facility (LSF)

Adam Hendricks, General Dynamics Federal Health
Derek Grittmann, General Dynamics Federal Health

ABSTRACT

The Centers for Medicare and Medicaid (CMS) Chronic Conditions Warehouse (CCW) Virtual Data Research Center (VRDC) is a large multi-user SAS Grid environment where most of the users are limited to SAS EG with no command line interface. LSF is a powerful true batch tool that allows a user to launch dozens of SAS jobs and then log out of environment but only from the command line interface. It is possible to make LSF batch available to EG-only users with adequate security using custom macros, script modifications, and SAS invocation options tied to Linux/Unix group membership.

INTRODUCTION

Many SAS users may be used to submitting from Base/PC SAS in batch mode and were surprised to not have a similar functionality in Enterprise Guide (EG). This paper aims to demonstrate how to add in that functionality using XCMD on a Linux SAS Grid using Load Sharing Facility (LSF). While this paper doesn't cover it, similar functionality can be replicated on a single-server, non-Grid environment.

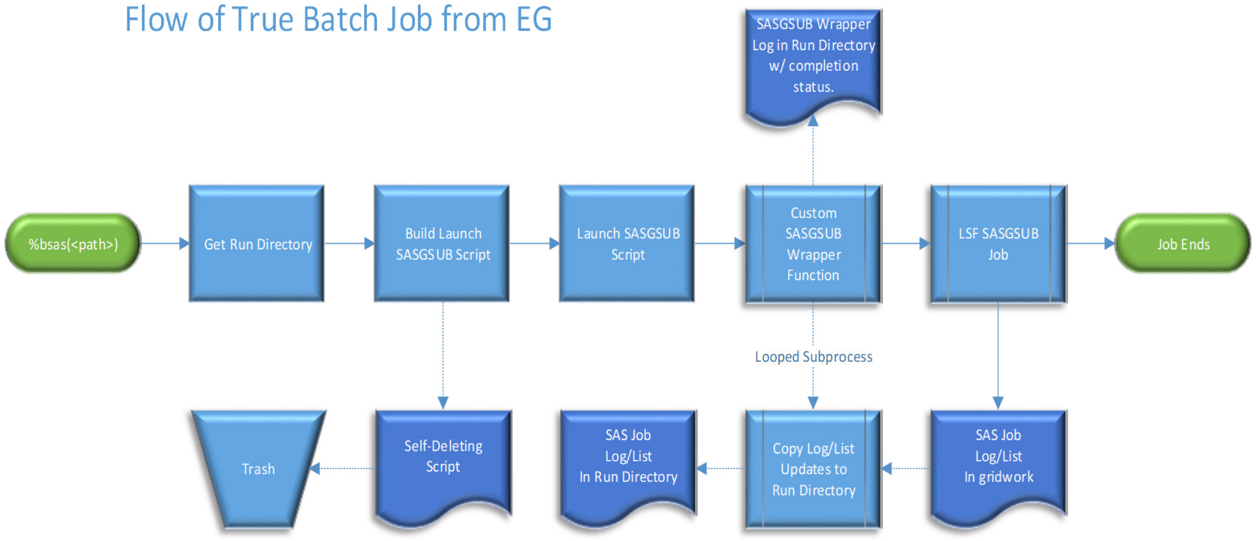
TRUE BATCH VS INTERACTIVE JOB ON LSF

What is a "True Batch Job"	SAS EG on LSF
➢ Not interactive.	❖ Interactive front end.
➢ Not child process of interactive process that launched it.	❖ Uses only one server grid as exec host by default.
➢ Completely independent of interactive process that launched it.	❖ Has background processes that are child processes of interactive session.
➢ I/O environment same as that of interactive process that launched it.	❖ Can leverage multiple servers on grid but only with complex SAS wrapper code. These are still not independent batch sessions. If the interactive parent session ends; so do the multiple grid job child sessions.

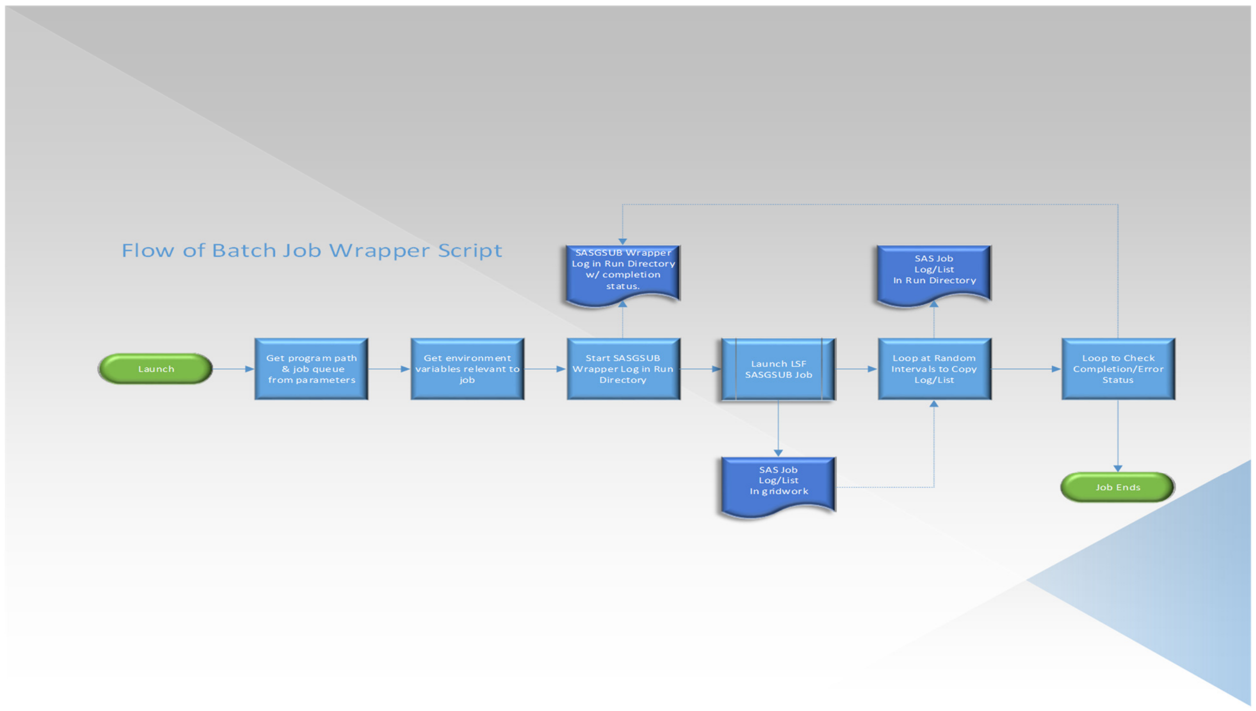
- ➢ SAS code is saved under file system available to EG.
- ➢ Batch job uses full Load Sharing Facility (LSF) grid capabilities.
- ➢ Batch job is completely independent of SAS EG session.
- ➢ Results of batch job are available in same directory as SAS program launched in batch.
- ➢ Depending on how EG environment is set up, no change in SAS code may be necessary to run job in batch.
- ➢ **Note: Any macro or script for true batch in EG will work identically on the command line.**

JOB FLOW

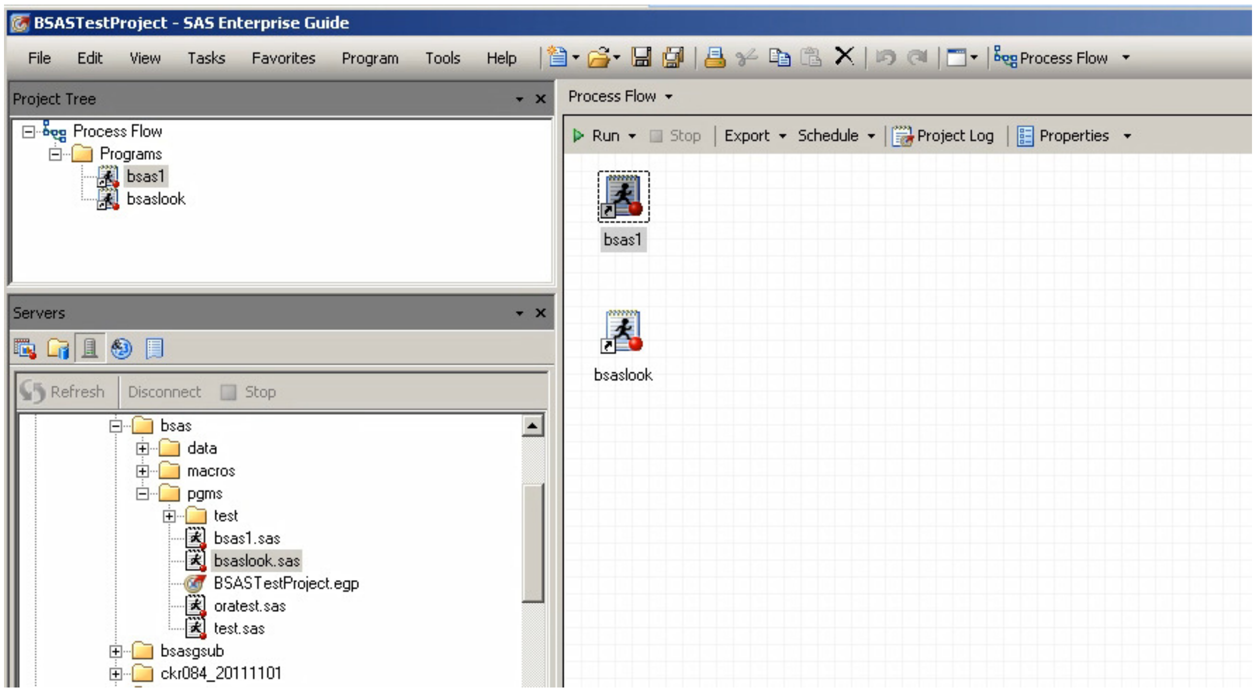
Flow of True Batch Job from EG

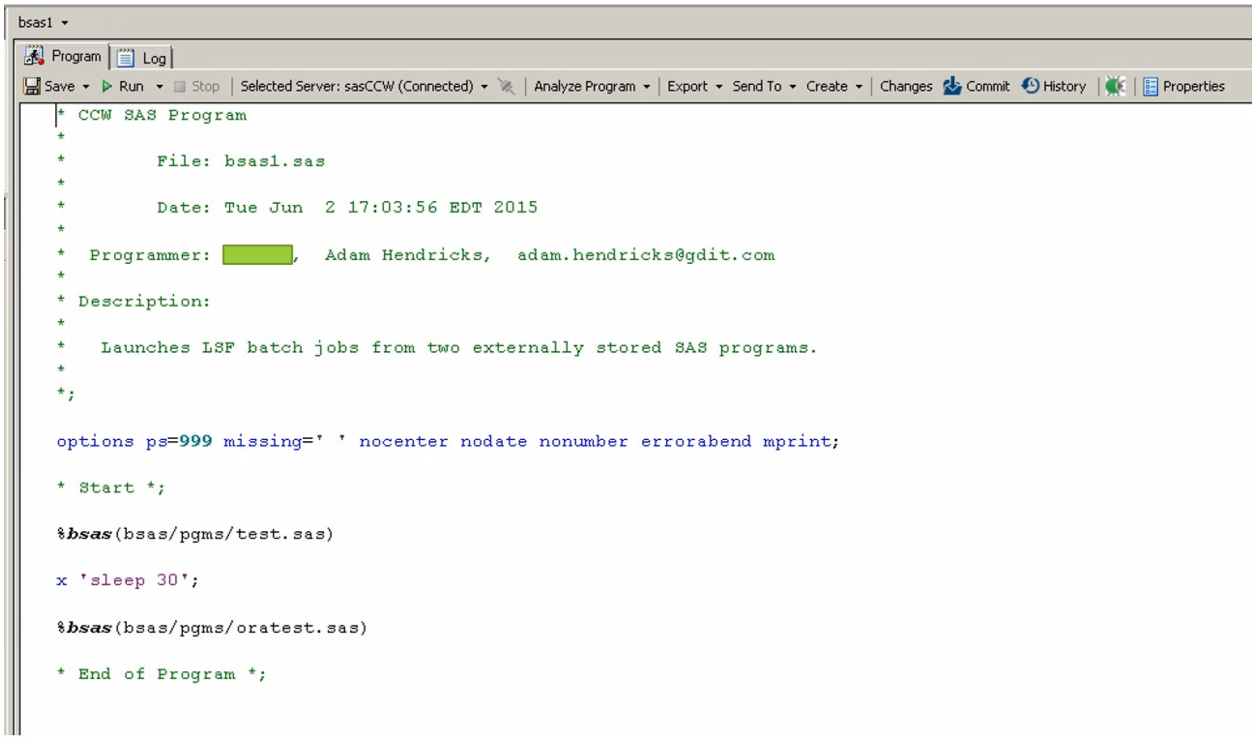


SASGSUB Wrapper Log in Run Directory w/ completion status.
%bsas(<path>)
Get Run Directory
Build Launch SASGSUB Script
Launch SASGSUB Script
Custom SASGSUB Wrapper Function
LSF SASGSUB Job
Job Ends
Looped Subprocess
Trash
Self-Deleting Script
SAS Job Log/List In Run Directory
Copy Log/List Updates to Run Directory
SAS Job Log/List In gridwork

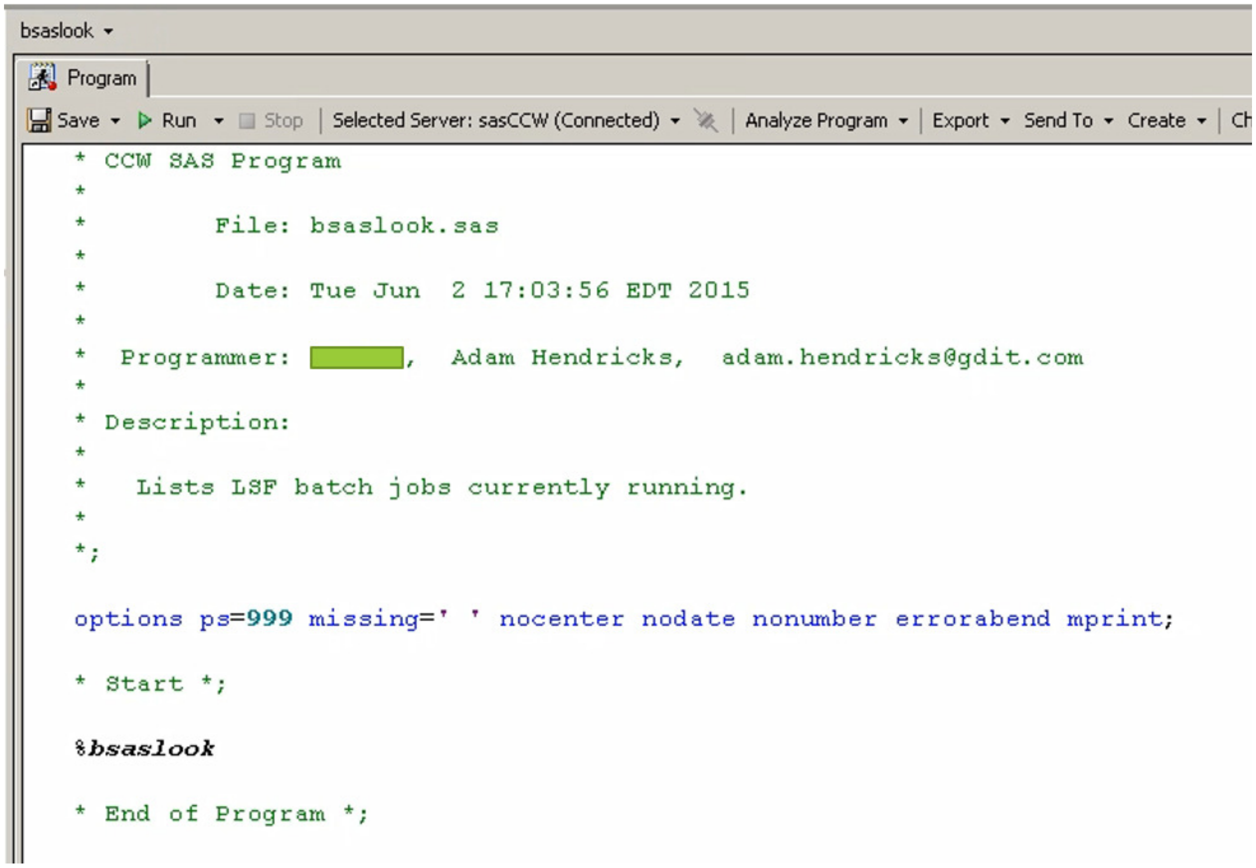


Flow of Batch Job Wrapper Script
SASGSUB Wrapper Log in Run Directory w/ completion status.
SAS Job Log/List In Run Directory
Launch
Get program path & job queue from parameters
Get environment variables relevant to job
Start SASGSUB Wrapper Log in Run Directory
Launch LSF SASGSUB Job
Loop at Random Intervals to Copy Log/List
Loop to Check Completion/Error Status
SAS Job Log/List In gridwork
Job Ends

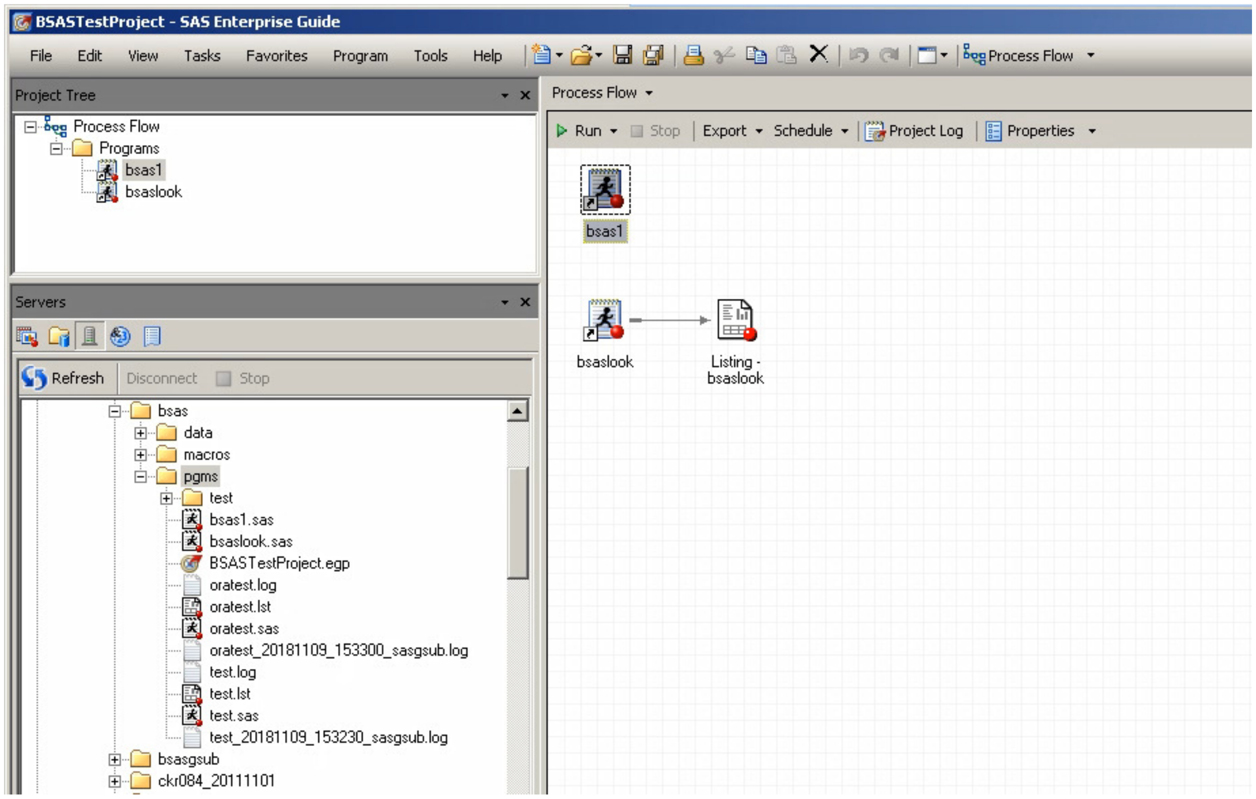


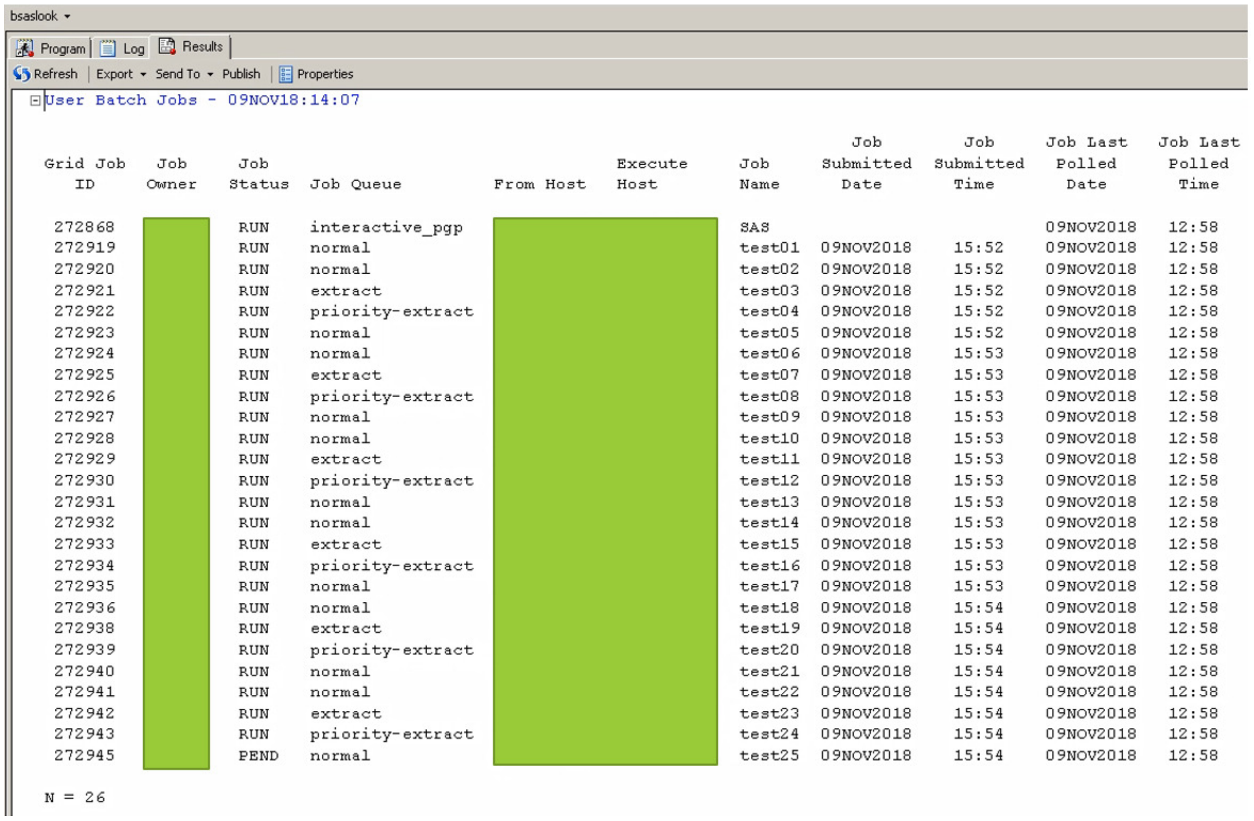

bsas1
Program Log
Save Run Stop Selected Server: sasCCW (Connected) Analyze Program Export Send To Create Changes Commit History Properties

```
* CCW SAS Program
*
*        File: bsas1.sas
*
*        Date: Tue Jun  2 17:03:56 EDT 2015
*
*  Programmer: [redacted],  Adam Hendricks,  adam.hendricks@gdit.com
*
* Description:
*
*   Launches LSF batch jobs from two externally stored SAS programs.
*
*;

options ps=999 missing=' ' nocenter nodate nonumber errorabend mprint;

* Start *;

%bsas(bsas/pgms/test.sas)

x 'sleep 30';

%bsas(bsas/pgms/oratest.sas)

* End of Program *;
```

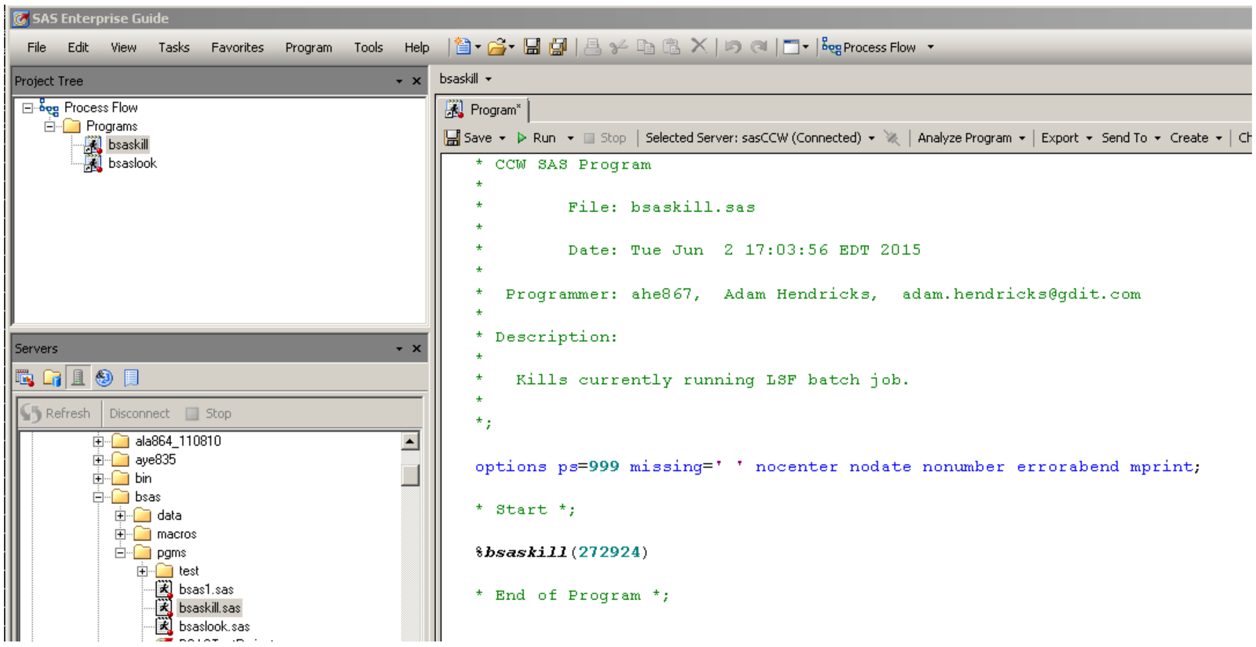


```
* CCW SAS Program
*
*         File: bsaslook.sas
*
*         Date: Tue Jun  2 17:03:56 EDT 2015
*
*  Programmer: [redacted],  Adam Hendricks,  adam.hendricks@gdit.com
*
* Description:
*
*   Lists LSF batch jobs currently running.
*
*;

options ps=999 missing=' ' nocenter nodate nonumber errorabend mprint;

* Start *;

%bsaslook

* End of Program *;
```





```
User Batch Jobs - 09NOV18:14:07

                                                                          Job        Job        Job Last   Job Last
 Grid Job    Job       Job                                   Execute  Job       Submitted  Submitted  Polled     Polled
    ID       Owner     Status  Job Queue          From Host   Host     Name      Date       Time       Date       Time

  272868   [redacted]  RUN    interactive_pgp    [redacted]           SAS                              09NOV2018  12:58
  272919   [redacted]  RUN    normal             [redacted]           test01    09NOV2018  15:52      09NOV2018  12:58
  272920   [redacted]  RUN    normal             [redacted]           test02    09NOV2018  15:52      09NOV2018  12:58
  272921   [redacted]  RUN    extract            [redacted]           test03    09NOV2018  15:52      09NOV2018  12:58
  272922   [redacted]  RUN    priority-extract   [redacted]           test04    09NOV2018  15:52      09NOV2018  12:58
  272923   [redacted]  RUN    normal             [redacted]           test05    09NOV2018  15:52      09NOV2018  12:58
  272924   [redacted]  RUN    normal             [redacted]           test06    09NOV2018  15:53      09NOV2018  12:58
  272925   [redacted]  RUN    extract            [redacted]           test07    09NOV2018  15:53      09NOV2018  12:58
  272926   [redacted]  RUN    priority-extract   [redacted]           test08    09NOV2018  15:53      09NOV2018  12:58
  272927   [redacted]  RUN    normal             [redacted]           test09    09NOV2018  15:53      09NOV2018  12:58
  272928   [redacted]  RUN    normal             [redacted]           test10    09NOV2018  15:53      09NOV2018  12:58
  272929   [redacted]  RUN    extract            [redacted]           test11    09NOV2018  15:53      09NOV2018  12:58
  272930   [redacted]  RUN    priority-extract   [redacted]           test12    09NOV2018  15:53      09NOV2018  12:58
  272931   [redacted]  RUN    normal             [redacted]           test13    09NOV2018  15:53      09NOV2018  12:58
  272932   [redacted]  RUN    normal             [redacted]           test14    09NOV2018  15:53      09NOV2018  12:58
  272933   [redacted]  RUN    extract            [redacted]           test15    09NOV2018  15:53      09NOV2018  12:58
  272934   [redacted]  RUN    priority-extract   [redacted]           test16    09NOV2018  15:53      09NOV2018  12:58
  272935   [redacted]  RUN    normal             [redacted]           test17    09NOV2018  15:53      09NOV2018  12:58
  272936   [redacted]  RUN    normal             [redacted]           test18    09NOV2018  15:54      09NOV2018  12:58
  272938   [redacted]  RUN    extract            [redacted]           test19    09NOV2018  15:54      09NOV2018  12:58
  272939   [redacted]  RUN    priority-extract   [redacted]           test20    09NOV2018  15:54      09NOV2018  12:58
  272940   [redacted]  RUN    normal             [redacted]           test21    09NOV2018  15:54      09NOV2018  12:58
  272941   [redacted]  RUN    normal             [redacted]           test22    09NOV2018  15:54      09NOV2018  12:58
  272942   [redacted]  RUN    extract            [redacted]           test23    09NOV2018  15:54      09NOV2018  12:58
  272943   [redacted]  RUN    priority-extract   [redacted]           test24    09NOV2018  15:54      09NOV2018  12:58
  272945   [redacted]  PEND   normal             [redacted]           test25    09NOV2018  15:54      09NOV2018  12:58

 N = 26
```



```
* CCW SAS Program
*
*        File: bsaskill.sas
*
*        Date: Tue Jun  2 17:03:56 EDT 2015
*
*  Programmer: ahe867,  Adam Hendricks,  adam.hendricks@gdit.com
*
* Description:
*
*   Kills currently running LSF batch job.
*
*;

options ps=999 missing=' ' nocenter nodate nonumber errorabend mprint;

* Start *;

%bsaskill(272924)

* End of Program *;
```

CONFIGURATION AND CODE EXAMPLES

Second grid metadata server with XCMD turned on or sasgrid & WorkspacesServer.sh script edited to give XCMD capability to users approved for true batch processing.

Example of WorkspaceServer.sh modification:

```
cmd="$SAS_COMMAND $USERMODS_OPTIONS"
arg='-noxcmd'
batch_group_found=`/usr/bin/groups $USER | grep 'sas_batch' | wc -l`

if [[ $batch_group_found -ne 0 ]]
then
  arg="-xcmd"
fi
cmd="$cmd $arg"
```

- Get Run Directory and SAS Code Path to Global Macrovariables
  - Call by system autoexec.sas

```
%macro pgmpath;
  %global pwd cwd pgm;
  %let pwd = %sysget(PWD);
  %let cwd = %sysget(LS_SUBCWD);

  %if %superq(cwd) =
  %then %let cwd = &pwd.;
  %else %let pwd = &cwd.;

  %put pwd=&pwd. cwd=&cwd.;

  proc sql noprint;
    select scan(xpath,-1,'/')
      into :pgm
      from sashelp.vextfl
     where upcase(xpath) like '%.SAS';
  quit;

  %let pgm = %trim(&pgm.);
  %put pgm=&pgm.;
  %put pgmpath=&pwd./&pgm.;
%mend;

%pgmpath
```

- Macros for Launching Batch Job

```
%macro bsas_sched(pgm=,waithr=0,waitmin=0);
%let uid = %sysget(LOGNAME);

data _null_;
  length dirname pgm $200 waitsec 8;
  dirname  = compress("&user_dirs./&uid./files");
  pgm      = compress(dirname||"/&pgm.");
  pgm      = strip(reverse(pgm));
  dirstrt  = index(pgm,'/') + 1;
  dirname  = strip(reverse(substr(pgm,dirstrt)));
  pgm      = strip(reverse(scan(pgm,1,'/')));

  * Timestamps *;
  ccyymmdd = put(today(),yymmddn8.);
  hhmmss   = compress(translate(put(time(),time.),' ',':'));
  ccyymmddhhmmss = compress(ccyymmdd||hhmmss);

  * Wait Time *;
  waithr   = max(&waithr.,0);
  waitmin  = &waitmin.;
  waitsec  = waithr*60*60 + waitmin*60;

  * Extension Name *;
  ext = scan(left(reverse(pgm)),1,'.');

  put ext=;

  pgmnm = scan(pgm,1,'.');

  put pgmnm=;

  if ext eq 'sas'
  then pgm = scan(pgm,1,'.');

  put pgm=;

  callfile = compress(dirname||"/bsasgsub_&uid._"||pgmnm||'_'||ccyymmddhhmmss||'.sh');

  call symput('dirname',strip(dirname));
  call symput('callfile',strip(callfile));
  call symput('pgm',pgm);
  call symput('waitsec',strip(left(waitsec)));

  put dirname=;
  put pgm=;
  put callfile=;
run;

filename bsasgsub "%trim(&callfile.)";

data _null_;
  file bsasgsub noprint;
  put '#!/bin/ksh';
  put '#';
  put ' ';
  put ". &profile_dir./sas_user_profile.sh > /dev/null";
  put ' ';
  put "cd %trim(&dirname.)";
  put ' ';

%if &waitsec gt 0
%then put "sleep &waitsec.";;

  put "sasgsub_launch.sh %trim(&pgm.) &";
  put ' ';
  put 'sleep 5';
  put ' ';
  put "/bin/rm -f %trim(&callfile.)";
  put ' ';
  put '#';
  put '# End of Script';
run;

filename bsasgsub clear;

x "chmod 740 %trim(&callfile.)";
x "nohup %trim(&callfile.) > /dev/null 2>&1 &";
%mend;

%macro bsas(pgm);
  %bsas_sched(pgm=&pgm.)
%mend;
```

- Macro to List LSF Batch Jobs Running or Pending

```
%macro bsaslook;
* bjobs command in wide tab-delimited format *;
filename bjobs pipe "bjobs -w | &tools_dir./mktab";

* Start *;
data __bjobs;
  length jobid 8 user $8 status $10 queue $20 from_host exec_host job_name $40 mon $3 dd
$2 time $5
         submit_date 4 submit_time 8 last_poll_date 4 last_poll_time runtime 8;

  format submit_date last_poll_date date9. submit_time last_poll_time time5. runtime
hhmm12.;

  label jobid          = 'Grid Job ID'
        user           = 'Job Owner'
        status         = 'Job Status'
        queue          = 'Job Queue'
        from_host      = 'From Host'
        exec_host      = 'Execute Host'
        job_name       = 'Job Name'
        submit_date    = 'Job Submitted Date'
        submit_time    = 'Job Submitted Time'
        last_poll_date = 'Job Last Polled Date'
        last_poll_time = 'Job Last Polled Time'
        runtime        = 'Job Run Time HH:MM';


  infile bjobs dlm='09'x dsd firstobs=2;
  input jobid user $ status $ queue $ from_host $ exec_host $ job_name $ mon $ dd $ time
$;

  if queue ne 'priority';

  from_host      = scan(from_host,1,'.');
  exec_host      = scan(exec_host,1,'.');
  submit_date    = input(compress(dd||mon||put(year(date()),4.)),date9.) ;
  submit_time    = input(time,time.);
  last_poll_date = date();
  last_poll_time = time();
  start          = dhms(submit_date,hour(submit_time),minute(submit_time),0);
  end            = dhms(last_poll_date,hour(last_poll_time),minute(last_poll_time),0);
  runtime        = end - start;

  drop mon dd time start end;
run;

proc sort data=__bjobs;
  by jobid;
run;

proc print data=__bjobs noobs n width=minimum label;
  title "User Batch Jobs - &sysdate.:&systime.";
run;

proc sql;
  drop table __bjobs;
quit;
%mend;
```

- Macro to Kill LSF Batch Job

```
%macro bsaskill(jobid);
%let uid = %sysget(LOGNAME);

data _null_;
  length dirname $200;
  dirname  = compress("&users_dir./&uid./files");
  jobid    = strip("&jobid.");
  ccyymmdd = put(today(),yymmddn8.);
  hhmmss   = compress(translate(put(time(),time.),' ',':'));
  ccyymmddhhmmss = compress(ccyymmdd||hhmmss);

  callfile =
compress(dirname||"/bsasgsub_kill_&uid._"||jobid||'_'||ccyymmddhhmmss||'.sh');

  call symput('dirname',strip(dirname));
  call symput('callfile',strip(callfile));

  put dirname=;
  put jobid=;
  put callfile=;
run;

filename bsasgsub "%trim(&callfile.)";

data _null_;
  file bsasgsub noprint;
  put '#!/bin/ksh';
  put '#';
  put ' ';
  put ". &profile_dir./sas_user_profile.sh > /dev/null";
  put ' ';
  put "bkill %trim(&jobid.) > /dev/null 2>&1";
  put 'sleep 5';
  put "/bin/rm -f %trim(&callfile.)";
  put ' ';
  put '#';
  put '# End of Script';
run;

filename bsasgsub clear;

x "chmod 740 %trim(&callfile.)";
x "%trim(&callfile.)";
%mend;
```

- Wrapper Script

```
#!/bin/ksh
#
# File: sasgsub_launch.sh
#
# Purpose: Wrapper Script for SASGSUB call.
#
# Parameters: saspgmpath - SAS Program Path
#

# Get SAS Program Path and Login ID
#
saspgmpath=$1                                              # <-- Input SAS program path,
literal or relative
gridqueue=$2                                               # <-- Grid Queue, NULL
(defaults to normal), normal, extract, or priority-extract.
wrapper_pid=$$                                             # <-- Unix PID of this
process.
sasgriduser=${LOGNAME}

if [[ ${gridqueue} == "" ]]
then
  gridqueue="normal"
fi

# Check if Current Location is Writetable
#
echo "test" > tmp$$.txt
status=$?

if (( $status != 0 ))
then
  echo "Error: Current Directory is Not Writable by User.  Exiting..."
  exit
else
  /bin/rm -f tmp$$.txt
fi

# Get SAS Program Dir Path
#
saspgmdir=`dirname ${saspgmpath}`                          # <-- Directory of input SAS
program

# Get server ID job on which this script was called.
#
NODE=`hostname`

# Give literal path if program path and start directory are same.
#
if [[ ${saspgmdir} == '.' ]]
then
  saspgmdir=`pwd`
fi

# Allow for omission of '.sas' extension in SAS program path.
#
if [ -f ${saspgmpath} ]
then
  echo " " > /dev/null  # Do Nothing
else
  saspgmpath="${saspgmpath}.sas"
fi

# Generate Run Parameter Strings
#
saspgm=`basename ${saspgmpath}`                            # <-- Input SAS program name
startdir=`pwd`                                             # <-- Directory where
sasgsub_launch.sh called.
```

```
dt_str=`date '+%Y%m%d_%H%M%S'`                              # <-- Date string for sasgsub
job log
sasprefix=`echo "${saspgm}" | cut -f 1 -d .`                # <-- Get prefix of SAS
program for log/lst names.
saspgm="${sasprefix}.sas"                                   # <-- Define standard SAS
program extension.
saslog="${sasprefix}.log"                                   # <-- Define SAS log name.
saslst="${sasprefix}.lst"                                   # <-- Define SAS log name.
sasgsublog="${startdir}/${sasprefix}_${dt_str}_sasgsub.log" # <-- SASGSUB job log
statuslog="${startdir}/${sasprefix}_${dt_str}_status.log"   # <-- SASGSUB temporary
status log

# Change to Start Directory
#
cd ${startdir}

# Start SASGSUB Job Log
#
echo "-- SASGSUB Launch --"         > ${sasgsublog}
echo " "                             >> ${sasgsublog}
echo "       This Log: ${sasgsublog}"  >> ${sasgsublog}
echo "Start Date/Time: `date`"         >> ${sasgsublog}
echo "           User: ${LOGNAME}"     >> ${sasgsublog}
echo "Start Directory: ${startdir}"    >> ${sasgsublog}
echo "     Start Host: ${NODE}"        >> ${sasgsublog}
echo "    Wrapper PID: ${wrapper_pid}" >> ${sasgsublog}
echo "     Grid Queue: ${gridqueue}"   >> ${sasgsublog}
echo " "                               >> ${sasgsublog}
echo "------------------------"        >> ${sasgsublog}

# Error Out if Bad Queue Entered
#
if [[ ${gridqueue} != "normal" && ${gridqueue} != "extract" && ${gridqueue} != "priority-
extract" && ${gridqueue} != "workbench" && ${gridqueue} != "normal_rerunnable" &&
${gridqu
eue} != "pgp" && ${gridqueue} != "support_users" ]]
then
  echo " "
>> ${sasgsublog}
  echo "ERROR: Invalid grid queue entered (${gridqueue}).  Job not submitted."
>> ${sasgsublog}
  echo " "
>> ${sasgsublog}
  echo "NOTE: Valid queue choices are NULL (defaults to normal), normal, extract, or
priority-extract." >> ${sasgsublog}
  echo " "
>> ${sasgsublog}
  echo "End Date/Time: `date`"
>> ${sasgsublog}
  echo " "
>> ${sasgsublog}
  echo "--- End of SASGSUB Log"
>> ${sasgsublog}
  exit
fi

# If input path exists, launch job, else error out.
#
if [ -f ${saspgmpath} ]
then
  sasgsub.sh -GRIDCONFIG ${sas_config_path}/sasgsub.cfg -GRIDSUBMITPGM ${saspgmpath} -
GRIDJOBOPTS queue=${gridqueue} -GRIDUSE
R ${sasgriduser} >> ${sasgsublog}
else
  echo " "                                                                          >>
${sasgsublog}
  echo "ERROR: Input path (${saspgmpath}) does not exist.  Exiting SASGSUB launch." >>
${sasgsublog}
  echo " "                                                                          >>
${sasgsublog}
```

```
  echo "End Date/Time: `date`"                                                    >>
${sasgsublog}
  echo " "                                                                        >>
${sasgsublog}
  echo "--- End of SASGSUB Log"                                                   >>
${sasgsublog}
  exit
fi

# Get Job ID and Directory from launch messages.
#
job_id=`grep 'Job ID:' ${sasgsublog} | cut -f 2 -d :`
job_id=`printf "%-10s" ${job_id}`
job_id=`echo $job_id | awk '{ print $1 }'`
job_dir=`grep 'Job directory:' ${sasgsublog} | cut -f 2 -d : | tr -d \"`
job_dir=`printf "%-90s" ${job_dir}`
job_dir=`echo $job_dir | awk '{ print $1 }'`
job_status="Submitted"

# If Job ID not retrieved, then exit with error.
#
if [[ ${job_id} == '' ]]
then
  echo " "                                                                        >>
${sasgsublog}
  echo "ERROR: Grid Job ID not ascertained.  Exiting SASGSUB launch."             >>
${sasgsublog}
  echo " "                                                                        >>
${sasgsublog}
  echo "End Date/Time: `date`"                                                    >>
${sasgsublog}
  echo " "                                                                        >>
${sasgsublog}
  echo "--- End of SASGSUB Log"                                                   >>
${sasgsublog}
  echo "sasgsub.sh -GRIDCONFIG ${sas_config_path}/sasgsub.cfg -GRIDSUBMITPGM
${saspgmpath} -GRIDJOBOPTS queue=${gridqueue} -G
RIDUSER ${sasgriduser}"  >> ${sasgsublog}
  exit
fi

# Append job specifics to SASGSUB job log
#
echo " "                                        >> ${sasgsublog}
echo "               Job ID: ${job_id}"          >> ${sasgsublog}
echo "        Job Directory: ${job_dir}"         >> ${sasgsublog}
echo "           Job Status: ${job_status}"      >> ${sasgsublog}
echo "      Start Directory: ${startdir}"        >> ${sasgsublog}
echo "SAS Program Directory: ${saspgmdir}"       >> ${sasgsublog}
echo "          SAS Program: ${saspgm}"          >> ${sasgsublog}
echo "              SAS Log: ${saslog}"          >> ${sasgsublog}
echo "             SAS List: ${saslst}"          >> ${sasgsublog}
echo " "                                        >> ${sasgsublog}

sleep 20   # <-- Sleep 20 seconds to give Running status time to come up.

acquired_server=0  # <-- Acquired Run Server ID Boolean Established as False
crash_counter=0    # <-- Crash Counter to Prevent False Crash Message

# Check job every minute for status. Close out loop when done.
#
while [[ ${job_status} != 'Finished' && ${job_status} != 'Failed' ]]
do
  sasgsub.sh -GRIDCONFIG ${sas_config_path}/sasgsub.cfg -GRIDGETSTATUS ${job_id} >
${statuslog}  # <-- Run job status. Output
 to job log.

  # Update job status variable from new status log.
  #
  submitted=`grep 'is Submitted:' ${statuslog} | wc -l`
  running=`grep 'is Running:' ${statuslog} | wc -l`
  finished=`grep 'is Finished:' ${statuslog} | wc -l`
  failed=`grep 'has Failed:' ${statuslog} | wc -l`

  /bin/rm -f ${statuslog} # <-- Delete temporary status log file.

  if (( ${running} == 1 ))
  then
    job_status="Running"
  elif (( ${finished} == 1 ))
  then
    job_status="Finished"
  elif (( ${failed} == 1 ))
  then
    job_status="Failed"
  elif (( ${submitted} == 1 ))
  then
    job_status="Submitted"
  fi

  #
  # Exits this script when orphaned from grid job.
  #
  job_running=`bjobs 2>&1 | grep ${job_id} | wc -l`

  if (( ${job_running} == 0 && ${running} == 1 ))
  then
      if (( ${crash_counter} > 0 ))  # Two cycles of fail test positive required for
Failed message.
      then
        echo "Error: Job appears to have crashed.  Exiting..." >> ${sasgsublog}
        echo " "                                               >> ${sasgsublog}
        job_status="Failed"
      fi

    (( crash_counter=${crash_counter}+1 ))
  elif (( ${job_running} == 0 && ${submitted} == 1 ))
  then
    echo "Error: Job appears to have been cancelled.  Exiting..." >> ${sasgsublog}
    echo " "                                                      >> ${sasgsublog}
    job_status="Failed"
  fi

  # Acquire ID of server on which job is running.
  #
  if (( ${running} == 1 && ${acquired_server} == 0 ))
  then
    run_server=`bjobs 2>&1 | grep ${job_id} | /sas/scripts/common_tools/mktab | cut -f 6
| cut -f 1 -d .`
    acquired_server=1   # <-- Acquired Run Server ID Boolean Set True

    echo " "                            >> ${sasgsublog}
    echo "Job Run Server: ${run_server}" >> ${sasgsublog}
    echo " "                            >> ${sasgsublog}
  fi

  # Left Just and Trim Job Status
  #
  job_status=`printf "%-20s" ${job_status}`
  job_status=`echo $job_status | awk '{ print $1 }'`

  # Convert Failed status caused by mere warnings to Finished
  #
  if [[ ${job_status} == 'Failed' ]]
  then
    /bin/cp ${job_dir}/${sasprefix}.l* ${startdir}  # <-- Move SAS Log/List files to
start directory.
    errors=`grep ERROR: ${startdir}/${saslog} | grep -v CCW | head -1 | wc -l`
    endofsaslog=`grep "NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-
2414 " ${startdir}/${saslog} | wc -l`

    if (( ${errors} == 0 )) && (( ${endofsaslog} == 1 ))
    then
      job_status="Finished"
    fi
  fi

  # Append current status to job log.
  #
  echo "`date` - Job Status: ${job_status}" >> ${sasgsublog}

  # Get Current SAS Log and List and Keep Running
  #
  if [[ ${job_status} != 'Finished' && ${job_status} != 'Failed' ]]
  then
    /bin/cp ${job_dir}/${sasprefix}.l* ${startdir}  # <-- Move SAS Log/List files to
start directory.
    sleep_sec=$(((RANDOM%30+120))) # <-- Random number between 120 and 150
    sleep ${sleep_sec}             # <-- Sleep randomly between 1 and 2 minutes
  fi
done

# Close Out Finished Job.  Delete Temp. Status Log and Copy Log/List to Current
Directory.
#
/bin/rm -f ${statuslog}                            # <-- Delete temporary status log
file.
/bin/cp ${job_dir}/${sasprefix}.l* ${startdir}     # <-- Move SAS Log/List files to start
directory.
chmod 660 ${startdir}/${sasprefix}.l*

#/bin/rm -rf ${job_dir}                            # <-- Delete SASGSUB job work
directory.

# Look Errors, Warnings, and Uninitialized messages in SAS log.
#
errors=`grep ERROR: ${startdir}/${saslog} | grep -v CCW | head -1`
warnings=`grep WARNING: ${startdir}/${saslog} | head -1`
uninitialized=`grep uninitial ${startdir}/${saslog} | head -1`

# Close out SASGSUB job log
#
echo " "                     >> ${sasgsublog}
echo "-- Results --"         >> ${sasgsublog}
echo " "                     >> ${sasgsublog}
ls -l ${startdir}/${saslog}   >> ${sasgsublog}  # <-- List new SAS log file to SASGSUB
log.

if [ -f ${startdir}/${saslst} ]
then
  ls -l ${startdir}/${saslst} >> ${sasgsublog}  # <-- List new SAS list file to SASGSUB
log.
fi

# Note that errors were found in SAS log.
#
if [[ ${errors} != '' ]]
then
  echo " "                                 >> ${sasgsublog}
  echo "Errors found in ${startdir}/${saslog}" >> ${sasgsublog}
fi

# Note that warnings were found in SAS log.
#
if [[ ${warnings} != '' ]]
then
  echo " "                                   >> ${sasgsublog}
  echo "Warnings found in ${startdir}/${saslog}" >> ${sasgsublog}
fi

# Note that uninitialized messages were found in SAS log.
#
if [[ ${uninitialized} != '' ]]
```

```
then
  echo " "                                                    >> ${sasgsublog}
  echo "Uninitialized messages found in ${startdir}/${saslog}" >> ${sasgsublog}
fi

# End of Job Log
#
echo " "                      >> ${sasgsublog}
echo "End Date/Time: `date`"  >> ${sasgsublog}  # <-- Append end date/time to job log.
echo " "                      >> ${sasgsublog}
echo "--- End of SASGSUB Log" >> ${sasgsublog}

#
# End of Script
```

- SASGSUB Log Example from Wrapper Script

```
-- CCW SASGSUB Launch --

       This Log: xxxxxx: gridtest/test25_20181028_173021_sasgsub.log
Start Date/Time: Sun Oct 28 17:30:21 EDT 2018
           User: xxxxxx
Start Directory: gridtest
     Start Host: sasgrid04
    Wrapper PID: 19580
     Grid Queue: normal

------------------------

SAS Grid Manager Client Utility Version 9.45 (build date: Oct  3 2017)
Copyright (C) 2009-2017, SAS Institute Inc., Cary, NC, USA. All Rights Reserved

Job <258221> is submitted to queue <normal>.
Job ID:         258221
Job directory: "/sas/grid-work/xxxxxx/SASGSUB-2018-10-28_17.30.22.096_test25"
Job log file:  "/sas/grid-work/xxxxxx/SASGSUB-2018-10-28_17.30.22.096_test25/test25.log"

               Job ID: 258221
        Job Directory: /sas/grid-work/xxxxxx/SASGSUB-2018-10-28_17.30.22.096_test25
           Job Status: Submitted
      Start Directory: gridtest
SAS Program Directory: gridtest
          SAS Program: test25.sas
              SAS Log: test25.log
             SAS List: test25.lst

Sun Oct 28 17:32:01 EDT 2018 - Job Status: Submitted
Sun Oct 28 17:35:35 EDT 2018 - Job Status: Submitted
Sun Oct 28 17:39:25 EDT 2018 - Job Status: Submitted
Sun Oct 28 17:42:11 EDT 2018 - Job Status: Submitted

Job Run Server: sasgrid13

Sun Oct 28 17:45:29 EDT 2018 - Job Status: Running
Sun Oct 28 17:49:05 EDT 2018 - Job Status: Running
Sun Oct 28 17:52:37 EDT 2018 - Job Status: Running
Sun Oct 28 17:56:07 EDT 2018 - Job Status: Finished

-- Results --

-rw-rw---- 1 xxxxxx sas_usr 73216 Oct 28 17:56 gridtest/test25.log
-rw-rw---- 1 xxxxxx sas_usr 11283 Oct 28 17:56 gridtest/test25.lst
```

```
Errors found in gridtest/test25.log

End Date/Time: Sun Oct 28 17:56:07 EDT 2018

--- End of SASGSUB Log
```

CONCLUSION

Whether your environment has SAS Grid or not, you too can have a batch submission capability within Enterprise Guide (EG).

CONTACT INFORMATION

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

Adam Hendricks
General Dynamics Federal Health
515-645-3015
adam.hendricks@gdit.com

Derek Grittmann
General Dynamics Federal Health
515-221-4059
derek.grittmann@gdit.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.