A VAX/VMS Command Procedure for Submitting SAS Jobs in Batch Mode

John D. Smith
James Barbour
University of Colorado

I. Abstract

A VAX/VMS command procedure is presented which helps SAS users submit jobs more easily and more reliably.

When the procedure is called as SETSAS it remembers the name of the file containing your SAS code from session to session, the queue to which the job is to be submitted, and the details of any tape mounts that the job requires. It sets up logicals CODE, LOG and LIS which point to the corresponding .SAS, .LOG and .LIS files.

When called as BSAS, the command procedure submits a SAS batch job, using the same parameters that SETSAS does. When no parameters are given, both SETSAS and BSAS retrieve the parameters from a file that is written by SETSAS.

II. Sample Sessions

Two sample sessions are shown below to demonstrate the use of SETSAS and BSAS and the resulting messages. What the user enters is indicated by italics.

```
 PLEASE SELECT SYSTEM: vaxc

 Username: SMITH_JD
 Password:
                Academic Computing Services, VAXC, VAX/VMS V4.5

 $ setsas sugixmpl

Default SAS file set to: SUGIXMPL.SAS
Jobs are sent to queue: SYS$BATCH by default.

 $ create code

%include defaults;  * Standard options, librefs, etc.;
proc contents data=jdsdb.tuition;

<CNTL-Z> Exit

 $ set def saswrk  ! go to directory pointed to by logical SASWRK

 $ bsas
Submitting SAS job SUGIXMPL.SAS to queue: SYS$BATCH
Job SAS_SMITH_JD_155031 (queue SYS$BATCH, entry 1435)
  started on SYS$BATCH_VAXC

 $ logoff

   SMITH_JD      logged out at 18-JAN-1987 15:51:39.02

 PLEASE SELECT SYSTEM: vaxc
 Username: SMITH_JD
 Password:
                Academic Computing Services, VAXC, VAX/VMS V4.5

 $ setsas

Default SAS file set to: SUGIXMPL.SAS
Jobs are sent to queue: SYS$BATCH by default.

Job SAS_SMITH_JD_155031_2
  (queue SYS$BATCH_VAXC, entry 1436) completed
```

```
 $ search log "error","Note"

NOTE: VMS Version of SAS Release 5.03 at UNIVERSITY OF COLORADO
NOTE: LICENSED CPUID MODEL = 11/750, SERIAL = BT06493.
    16+options pdisk idisk errors=3 tlog ps=63 nostimer noovp;
ERROR: LIBNAME JDSDB NOT DEFINED.
NOTE: SAS INSTITUTE INC., SAS CIRCLE, BOX 8000, CARY, N. C.

 $ setsas sugixmpl large/q x=123456/t

Default SAS file set to: SUGIXMPL.SAS
Jobs are sent to queue: LARGE_BATCH by default.
TAPE_ID = 123456, TAPE_TYPE = X

 $ lo
```

III. DMS Commands submitted

Here is what BSAS constructs and submits for batch execution:

```
$ set def SYS$USERDEVICE:[CADPC.SMITH_JD.SASWRK]
$! *** This is where mount statement would go. **
$ set process /name=SUGIXMPL.SAS
$ sas SUGIXMPL.SAS
```

IV. VMS DCL Code with Internal Documentation

The following VMS DCL Code includes internal documentation that explains how to use the command procedure as well as how to install it.

```
$! BSET_SAS.com / R+T John Smith and Jim Barbour  Wed 10-01-1986
$!
$! To run BSET_SAS.com, enter the commands as follows:
$!
$!   BSAS   [ current [ large/q  |  x=123456/t ] ]
$!   SETSAS [ current [ large/q  |  x=123456/t ] ]
$!
$! BSET_SAS provides help in running SAS in batch mode.  Invoked as SETSAS,
$! it sets up logicals for editing and typing the SAS source code, log and
$! output: CODE for current.SAS, LIS for current.LIS, and LOG for
$! current.LOG.  It also stores the name of the current file and other
$! parameters in saswrk:setsasmem.dat.  Thus, by entering SETSAS by itself,
$! you are reminded of what you were working on last. Invoked as BSAS, it
$! checks for the existence of current.SAS, submits it for batch processing,
$! specifying the apprpriate parameters.  BSAS can construct 2 typical tape
$! mount statements. It also submits a second job to unprotect the .LOG file
$! from the first job.
$!
$! Both modes use the following parametere:
$!
$!   none       = get parameters from saswrk:setsasmem.dat
$!   filename   = filename.sas is the SAS code, logical CODE;
$!                required if any parameters are given.
$!   large/Q    = large$batch queue, with a CPU limit of 3 hours.
$!   X=123456/T = mount tape 123456 with logical name IXTAPE for XCOPY.
$!   P=123456/T = mount tape 123456 with logical name PS123456 -- PSF.
$!
$! Once these parameters are set, BSET_SAS can retrieve them from
$! saswrk:setsasmem.dat when you enter either command with no parameters.
$!
$! If any parameters are given, filename is required as the first.
$!
$! BSET_SAS requires an extra parameter (which is normally hidden from the
$! user) to indicate which task it is to accomplish.  The following symbols
$! must be created in the login process:
$!  BSAS     :== @BSET_SAS -b
$!  SETSAS   :== @BSET_SAS -s
$!
$! This file will be sent to anyone who requests it through BITNET from
$! SMITH_JD@COLORADO.
$!
```

```
$ on control_c then goto interupt2
$ if f$trnlnm("saswrk") .eqs. "" then goto nosaswrk ! issue a message, exit
$ if p1 .nes. "-S" .and. p1 .nes. "-B" then goto bad_mode
$ mode :== setsas
$ if p1 .eqs. "-B" then mode :== bsas
$ gosub shift
$ gosub parse_params
$!
$! Begin main
$!
$ if file .eqs. "" then goto nofilerr
$ if tape .eqs. "" then goto tape_ok
$ tape_type = f$element(0,"=",tape)
$ if tape_type .eqs. "" then goto notapetype
$ if tape_type .nes. "P" .and. tape_type .nes. "X" then goto invtapetype
$ tape_num = f$element(1,"=",tape)
$ if tape_num .eqs. "" then goto notapenum
$ if f$type(tape_num) .nes. "INTEGER" then goto invtapenum
$ if f$length(tape_num) .ne. 6 then goto invtapelen
$ tape_ok: ! tape specifications are OK or no tape mount is requested
$  if tape .eqs. "" then tape_type = ""
$  name = f$parse(file,,,"name")
$  file = name + ".SAS" ! strips any node,dirctory or type specs off of file.
$!  it's ok if ''file' doesn't exist yet
$! check for valid queue names:
$ len = f$length(queue)
$  queue = f$edit(queue,"upcase")
$  if queue .eqs. f$extract(0,len,"LARGE_BATCH") then queue = "LARGE_BATCH"
$  if queue .eqs. f$extract(0,len,"SYS$BATCH") then queue = "SYS$BATCH"
$  if queue .eqs. f$extract(0,len,"BATCH") then queue = "SYS$BATCH"
$  if queue .nes. "SYS$BATCH" .and. queue .nes. "LARGE_BATCH" then goto qerr
$  swrkdir = f$trnlnm("saswrk")
$!
$! from this point down to EXIT:, SETSAS and BSAS diverge.
$!
$ if mode .eqs. "BSAS" then goto do_bsas
$!
$ assign/nolog 'swrkdir''name'.log LOG
$ assign/nolog 'swrkdir''name'.sas CODE
$ assign/nolog 'swrkdir''name'.lis LIS
$ write sys$output "Default SAS file set to: ''file'"
$ write sys$output "Jobs are sent to queue: ''queue' by default."
$ if f$type(tape_num) .eqs. "INTEGER" then -
   write sys$output "TAPE_ID = ''tape_num', TAPE_TYPE = ''tape_type'"
$ if f$search("saswrk:setsasmem.dat") .nes. "" then -
   delete/noconf saswrk:setsasmem.dat;0
$ open /write outfile saswrk:setsasmem.dat
$ on control_c then goto interupt
$  write outfile file
$  write outfile queue
$  write outfile tape
$  close outfile
$ on control_c then goto interupt2
$ goto exit
$!
$ do_bsas:
$ if f$search("''file'") .eqs. "" then goto filnotfou
$! set up tape mount instructions
$   tape_line = "$! *** This is where mount etatement would go. ***"
$   if tape_type .eqs. "P" then tape_line = -
"$ magtape ''tape_num' /foreign /record=380 /block=3800 /nowrite ps''tape_num'"
$   if tape_type .eqs. "X" then tape_line = -
  "$ magtape ''tape_num' /foreign ixtape"
$  fname = "BAT_" + f$user() + ".SAS" - "[" - "]"
$  logname = "BAT_" + f$user() + ".LOG" - "[" - "]"
$  idate = f$time()
$  date = f$extract(12,2,idate) + f$extract(15,2,idate) + f$extract(18,2,idate)
$  jobname = "SAS_" + f$user() + "_" + date - "[" - "]"
$  jobfile = "SAS_" + f$user() + "_" + date + ".JOB" - "[" - "]"
$  write sys$output " Submitting SAS job ''file' to queue: ''queue'"
$  write sys$output ""
$  protection_flag = f$environment( " protection " )
$! following section writes a small batch job to UN-protect the system log
$! so we can all read each others' logs.
```

```
$  open/write outfile 'swrkdir''jobfile'
$  on control_c then goto interupt
$  write outfile "$ ! set noon"
$  write outfile "$ synch ''jobname'" ! wait till main SAS job finishes
$  write outfile "$ set prot=(''protection_flag') ''swrkdir'''logname'"
$  write outfile "$ exit"
$  close outfile
$! write the DCL for the SAS job:
$  open/write outfile 'fname'
$  write outfile "$ set def ''swrkdir'"
$  write outfile tape_line
$  write outfile "$ set process /name=''file'"
$  write outfile "$ sas ''file'"
$   close outfile
$  on control_c then goto interupt2
$! submit both jobs:
$  submit 'fname' /name='jobname' /cputime=3:00 /delete /noprint -
    /log=saswrk:'logname' /parameters='file' /queue='queue'
$  submit 'swrkdir''jobfile' /name='jobname'_2 /noident /delete -
    /notify /queue='queue' /nolog
$ exit: exit
$!
$! begin subroutines:
$!
$ parse_params:
$! Set up defaults
$ file = ""
$ queue = "SYS$BATCH"
$ tape = ""
$ if p4 .nes. "" then goto too_many_parms
$ if p1 .eqs. "" then goto readfile
$ len= f$locate("/",p1) + 1  ! strip off "/" if necessary
$ if len .gt. 1 then p1 = f$extract(0,len,p1)
$ file = f$parse(p1,,,"name")
$ gosub shift
$! Return point for shift.
$ loop: if p1 .eqs. "" then return
$  p1 = f$edit(p1,"upcase")
$  ptype = f$element(1,"/",p1)
$  if ptype .eqs. "" then goto invparmform
$  p1 = f$element(0,"/",p1)
$  if p1 .eqs. "" then goto invparmform
$  len = f$length(ptype)
$! check against each alternative qualifier type:
$ if ptype .nes. f$extract(0,len,"QUEUE") then goto check1
$  queue = p1
$  gosub shift
$  goto loop
$ check1:if ptype .nes. f$extract(0,len,"TAPE") then goto dont_know_parm
$  tape = p1
$  gosub shift
$  goto loop
$ shift: p1 = p2
$  p2 = p3
$  p3 = p4
$  p4 = p5
$  p5 = p6
$  p6 = p7
$  return
$ readfile: if f$search("saswrk:setsasmem.dat") .eqs. "" then goto nofile
$  open/read in saswrk:setsasmem.dat
$  read/end=done in file
$  read/end=done in queue
$  read/end=done in tape
$  dons:  close in
$  return
$!
$! error messages
$!
$ nofile:
$ write sys$error -
  "Error: ''mode' requires a parameter for the .SAS file or a file"
$ write sys$error -
  "       named SETSASMEM.DAT in directory ''f$trnlnm(""SASWRK"")'"
$ goto exit
```

```
$ nosaswrk:  write sys$error -
   "Create a logical SASWRK pointing to the directory ''mode' should work in."
$  goto exit
$ interupt: close outfile
$ interupt2:  write sys$error -
   "''mode' aborted by user."
$  close/nolog in
$  goto exit
$ nofilerr:  write sys$error -
   "A filename is required - ''mode' filename [... optional parameters]"
$  goto exit
$ invparmform:
$ write sys$error -
   "Optional parameters require a slash as a delimiter between the parameter"
$ write sys$error -
   " and the parameter type: e.g., queue/Q, not ""''p1'"""
$  write sys$error "\''p1'\"
$  gosub shift
$  goto loop
$ dont_know_parm: write sys$error -
  "Unknown parameter type.  Ignored parameter \''p1'/''ptype'\"
$  gosub shift
$  goto loop
$ qerr: write sys$error -
   "Valid queues are: SYS$BATCH and LARGE$BATCH, not \''queue'\"
$  goto exit
$ notapetype: write sys$error -
   "No tape type given - \''tape'\."
$  write sys$error "Use [X or P]=tape_num/T (X for IXTAPE or P for PSF)"
$  goto exit
$ invtapetype: write sys$error -
   "Invalid tape type.  Must either be P or X, not \''tape_type'\.
$ write sss$error "Use [X or P]=tape_num/T (X for IXTAPE or P for PSF)."
$  goto exit
$ notapenum: write sys$error -
   "No tape number given - \''tape'\"
$  write sys$error "Use [X or P]=tape_num/T (X for IXTAPE or P for PSF)."
$  goto exit
$ invtapelen: write sys$error -
   "TAPE_NUM numbers must be exactly 6 digits, not \''tape_num'\."
$  write sys$error "Use [X or P]=tape_num/T (X for IXTAPE or P for PSF)."
$  goto exit
$ invtapenum: write sys$error -
   "TAPE_NUM must contain only digits. \''tape_num'\."
$  write sys$error "Use [X or P]=tape_num/T (X for IXTAPE or P for PSF)."
$  goto exit
$ too_many_parms: write sys$error -
   "Too many parameters -- only file, queue and tape # can be specified."
   goto exit
$ bad_mode: write sys$error -
   "1st parameter must be -S or -B, not ''p1'.  Symbols SETSAS and BSAS"
$ write sys$error -
   "provide the 1st parameter automatically."
   goto exit
$ filnotfou: write sys$error "Error opening ''file' as input."
$  write sys$error "File not found."
$  goto exit
```