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 suglxmlpl
Default SAS file set to: SUGLXMLPL.SAS
Jobs are sent to queue: SYSS$BATCH by default.

$ create code
%include defaults; * Standard options, librefs, etc.;
proc contents data=jdsdb.tuition;
<CTRL-Z> Exit

$ set def saswrk 1 go to directory pointed to by logical SASWRK

$ bsas
Submitting SAS job SUGLXMLPL.SAS to queue: SYSS$BATCH
Job SAS_SMITH_JD_155031 (queue SYSS$BATCH, entry 1435)
started on SYSS$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: SUGLXMLPL.SAS
Jobs are sent to queue: SYSS$BATCH by default.

Job SAS_SMITH_JD_155031_2 (queue SYSS$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 tiog ps=63 nostimer noovp;
ERROR: LIBNAME JDSDB NOT DEFINED.
NOTE: SAS INSTITUTE INC., SAS CIRCLE, BOX 8000, CARY, N. C.

$ setsas suglxmpl 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

$ io

III. DMS Commands submitted

Here is what BSAS constructs and submits for
batch execution:

$ set def SYSUSERDEVICE:[CADPC.SUITH_JD.SASWRK]
$ set process /name=SUGIXMPL.SAS
$ sas SUGIXMPL.SAS

$ 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 locals 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 appropriate parameters. BSAS can construct 2 typical tape
$! mount statements. It also submits a second job to unprotect the .LOG file
$! from the first job.

$!
$!

IV. VMS DCL Code with Internal Documentation

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

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!

$!
$!
$ on control_c then goto interrupt2
$ if fstrlnnm("saswrk") .eqs. ** then goto nossawrk ! issue a message, exit
$ if pi .nes. '-S' .and. pi .nes. '-B' then goto bad_mode
$ mode := setsas
$ if pi .eqs. '-B' then mode := bsas
gosub shift
gosub parse_params
$!
! Begin main
$!
$ if file .eqs. ** then goto nofileerr
$ 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 $type(tape_num) .eqs. "INTEGER" then goto invtapenum
$ if $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, directory or type specs off of file.
$! It's ok if 'file' doesn't exist yet
$! check for valid queue names:
$ len = $length(queue)
$ queue = f$edlt(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 = "BATCH"
$ if queue .nes. "SYS$BATCH" .and. queue .nes. "LARGE_BATCH" then goto qerr
$ swrkdlr = fstrlnnm("saswrk")
$!
! from this point down to EXIT: SETSAS and BSAS diverge.
$!
$ if mode .eqs. "BSAS" then goto do_bsas
$!
$ assign/nolog 'swrkdlr' name 'LOG' !
$ assign/nolog 'swrkdlr' name 'SAS' !
$ assign/nolog 'swrkdlr' name 'LIS'
$ write sys$output "Default SAS file set to: 'file'" !
$ write sys$output "Jobs are sent to queue: 'queue' by default."
$ if $type(tape_num) .eqs. "INTEGER" then -
$ swrkdir = fstrlnnm("saswrk")
$!
! 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 interrupt
$ 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 $ set $ as""file"
$ close outfile
$ on control_c then goto interrupt2
$ submit both jobs:
$ submit 'fname' /name='"jobname' /cputime=3:00 /delete /noprint -
/log=saswrk:'logname' /parameters='file' /queue='queue'
$ submit 'swrkdir'jobfile' /name='"jobname' /noldent /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$locateC·r,p1) + 1 | strip off r if necessary
$ if len .gt. 1 then pl = 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
$ done: close in
$ return
$ error messages
$ nofile:
$ write sys$error -
"Error: 'mode' requires a parameter for the .SAS file or a file"
$ write sys$error -
"Error: named SETSASMEM.DAT in directory 'f$trnlnm("SASWRK")'"
$ goto exit