

## Paper 106-27

## No Task Before Its Time: Schedule Your Jobs With Robot Code

Charles R. vanWynbergen, SunTrust Banks Inc., Atlanta, Georgia USA

### ABSTRACT

SAS® is a flexible, robust and highly portable language. These characteristics make SAS well suited for rapid report and application development in an ad hoc environment. Often these rapid solutions evolve into pseudo-production processes, which are manually executed on a monthly, weekly or daily basis. In a true production environment, jobs such as these could easily be scheduled for execution. However, in an ad hoc environment, scheduling facilities are not always available, and even when present, relatively few SAS users possess the required expertise or authority to utilize these tools. Commercial, shareware and freeware scheduling products are available for local platforms, but they offer an incomplete solution with limited options for customization.

This paper describes techniques that can be used to schedule and execute virtually any job, on any platform, at any time. These techniques take advantage of the multi-platform capabilities of SAS, and can be customized according to the special requirements of particular environments.

This paper is intended for intermediate SAS developers whose primary environment is Windows, but who execute jobs on other platforms. base SAS, SAS/Connect® and SAS Macros are used. The original code was developed on a Windows NT 4.0 PC, running SAS version 8.2.

### INTRODUCTION

SAS robot coding techniques offer a flexible, low-cost solution to the problem of scheduling jobs in a non-production environment.

There are three basic components of robot code:

- Robot driver code
  - Core scheduling mechanism
  - Executes jobs on an hourly basis
  - Schedule can be modified as required
- SAS "wrapper" code
  - Sign-on and sign-off statements
  - Standard remote-submit commands
- Automated logon scripts
  - Modified SAS remote connection scripts

### FEATURES

Robot code offers numerous features and benefits:

- Unattended job execution
  - Schedule a job for any hour, on any day, on almost any platform
- Improved resource management
  - Jobs can be executed at low-load times
- Quality Control support
  - Complete record of submitted jobs is preserved
  - HTML formatted summary reports can be automatically generated
- Target code can be written in *any* language

- Robot does not care if target job is in SAS, JCL, .exe, .bat, etc., or even operating system level commands.
- Optionally, for SAS code only, in-stream statements can be surrounded by wrapper code
  - This feature makes it very simple to adapt existing jobs for scheduling
- Flexible implementation
  - SAS based automation solution
    - Extensive in-house SAS expertise
  - Open source-code
    - Functionality not confined to black-box program
  - Fully customizable
    - SAS bells, whistles and PROCs can be integrated into process
  - Desirable price-point
    - If you own the most common SAS products, then you already have what you need to implement a Robot

### LIMITATIONS

Robot code is not a panacea-- certain restrictions apply:

- Target jobs must be self-contained and "production-ready"
  - No manual intervention is allowed
- SAS and SAS/CONNECT must be available on remote machine
  - Job owner must be authorized for SAS
- Platform on which robot code executes must be active when jobs are scheduled, and network connection must be available for remote submission
  - Desktop PC/ server is fairly reliable, but any device is acceptable as long as platform is stable
  - Best solution is a 100% dedicated SAS Robot PC
    - Since clock speed, per se, is not critical to robot code, obsolete machines can be adapted for this purpose quite nicely!

### PROCESS OVERVIEW

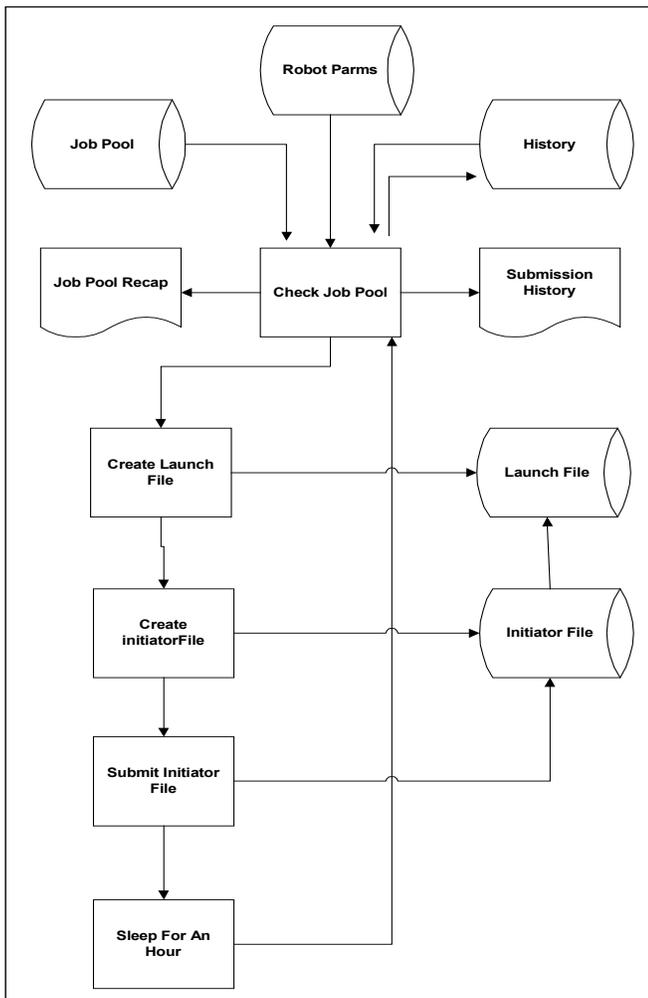
The robot process synthesizes several features of SAS in a single application. The key SAS components that make robot scheduling possible are the WAKEUP function and the CONNECT scripting options.

The robot process consists of a continuous cycle of five primary tasks:

- Check job pool
  - Robot driver checks the current job pool and identifies jobs that need to be executed in the current hour
- Create launch file
  - Robot outputs list of jobs that meet date and time criteria for execution to a serialized file

- Create initiator file
  - Robot creates an initiator file that points to the serialized launch file
- Launch initiator File
  - Robot executes the initiator file, which in turn submits the launch file, that contains the scheduled tasks
- Go to sleep
  - Robot sleeps for an hour, then wakes up and starts the cycle over again
  - The sleep parameter can be changed to any interval

See simplified process diagram below:



### Robot Cascade

The chain of files and executions can be visualized as a cascade. For example, a scheduled mainframe job would flow as follows:

Silo.bat=>

Launch20021470.bat=>

RobotTestMVSJCLcode.sas=>

Userid.testjobs(test1) -- which is the target job

## CONFIGURING AND RUNNING THE ROBOT

Implementing robot code involves four steps:

- Establish environment parameters
  - This is very simple
- Make target code "production-ready"
  - This can be simple or difficult, depending on current state of target code
- Schedule the job
  - This is simple, but tedious
- Run the robot code
  - This is really, really simple

### Establish Environment Parameters

- Set common libname and filename paths in the robotparms file
  - This only needs to be done once
- Set userid/password information in the logon scripts
  - This needs to be done initially and whenever your userid/password are modified on the target platform
  - Various escrow and encryption schemes may be used to improve security

### Make the Object Code Production-Ready

- Insure that the code is self-contained
  - Use ODS and/or PROC PRINTTO to route log and listings in SAS jobs.
  - Use other techniques to encapsulate target code
- Place a SAS "wrapper" around the object job
  - See appendix for various examples for UNIX, Mainframe and Windows jobs
  - Note: The "wrapper" is written in SAS, but the target job does not have to be written in SAS.

### Schedule the Job

- Open the job pool dataset and modify as necessary
  - Allocate the library
  - Click on SAS explorer and use the edit tool in SAS to access the job pool data set directly
  - Job information must be entered in the exact format
  - See Appendix for dataset specifications
- Alternately, create a spreadsheet and import into SAS
  - Job information must be entered very carefully
  - Take special care with quotation marks

### Run the Robot

- Run the robot code from inside of your PC SAS session
  - The robot code will launch the jobs in your job pool according to the schedule you specify
  - Robot code will run indefinitely, unless something interrupts the session, or the platform fails
- To halt the robot, click on the exclamation point

- Select "Halt Data Step Execution"
- Do not click "Cancel Submitted Statements" -- this can cause unpredictable results!

## ROOM FOR IMPROVEMENT

This paper outlines one solution in one particular environment, but it is not necessarily the best solution for every installation. The reader is encouraged to experiment with these techniques and share improvements with the SAS community.

Several areas that are ripe for enhancement can be readily identified:

- Simplified job pool entry
- Improved security
- Cleaner job spawning
- Return code intelligence
- Email notification
- Infrastructure recovery

## CONCLUSION

This paper describes how SAS robot coding techniques can be used to schedule and execute jobs on many platforms. These techniques offer an economical and highly customizable scheduling option for SAS users and developers.

## ACKNOWLEDGMENTS

Many thanks to Paul A. Bussard, whose insightful suggestions for improving a primitive scheduling tool many years ago provided the seed from which this paper finally blossomed.

Thanks to SunTrust Banks for fostering a spirit of innovation.

## DISCLAIMER

The author makes absolutely no warranty regarding the performance or reliability of the code described herein. The code is provided merely to demonstrate coding techniques that may be successful in some circumstances. Use the code at your own risk.

## CONTACT INFORMATION

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

Charles vanWynbergen

SunTrust Banks Inc.

PO Box 4418

Atlanta, GA 30302

404-827-6715

[charles.vanwynbergen@suntrust.com](mailto:charles.vanwynbergen@suntrust.com)

## APPENDIX

The appendix contains sample SAS source code for a "Robot Toolkit". The components include:

- Robot.sas
  - Sample robot engine
- RobotTestWindowsRaw.sas

- Sample Windows batch or non-SAS program
- RobotTestUNIXRaw.sas
  - Sample UNIX batch or non-SAS program
- RobotTestMVSSAS.sas
  - Sample mainframe SAS program
- RobotTestMVSJCL.sas
  - Sample mainframe batch program
- Tcpxautologon.scr
  - Sample UNIX logon script changes
- Tcptsoautologon.scr
  - Sample mainframe logon script changes
- There is no Windows logon script, since it is assumed you are running on that platform
- Robotparms.sas
  - Robot parameter file
- Robot.Schedule Contents
  - Proc Contents from the robot schedule dataset
  - These are the variables the robot code looks for to schedule jobs
- Robot.Schedule Job Specification Details
  - Sample value from the job specification variable in the job pool-- this line is parsed by the robot code and placed in the launch file
  - -sysin , -nosplash and -icon are required on each entry
  - The first part of the specification points to the job name to be scheduled -- this is always required
  - The last two parts point to log and list files, which are optional
- The sample code is fairly self-explanatory
  - The sign-on wrapper logs on to the appropriate platform
  - The target code executes after sign-on
  - The sign-off wrapper closes the session
  - Start off scheduling one job-- once you can reliably schedule a single job, it's easy to schedule any number of subsequent jobs.

That's all there is to it!

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 registered trademarks or trademarks of their respective companies.

## APPENDIX

## Robot.sas

```

options obs=max missing='.' errors=3 nodate notes symbolgen mprint mlogic macrogen source2 ls=132;
/* Start Infinite Loop*/
%macro outer;
%do %until (0);
/* Set Global Macro Variables*/
%global nowdate loglib printlib weblib filelib launlib connlib;
/* Include robotparms -- this file allocates librefs and filenames */
%include "h:\projects\robot\data\robotparms.sas";
/* Allocate Libraries And Files*/
filename outlog "&loglib.sr.log"; /* Allocate log file for report */
filename outprt "&printlib.sr.out"; /* Allocate out file for report */
filename outweb1 "&weblib.robotpool.html"; /* Allocate out file for job pool summary*/
filename outweb2 "&weblib.robothist.html"; /* Allocate out file for submitted job summary */
libname robotlib "&filelib"; /* Allocate libname for job and history data */
run;
/* Input Job Pool*/
data tasklist1;
set robotlib.schedule; /* This is the permanent dataset that contains the job pool information */
sasyear=year(today());
sasmonth=month(today());
sasday=day(today());
saswkday=weekday(today());
sashour=hour(datetime());
sasmin=minute(datetime());
runnow='N';
/* Determine if any job needs to be run now */
if runmonthly='Y' and whichdom=sasday and whattod=sashour then runnow='Y';
else if runweekly='Y' and whichdow=saswkday and whattod=sashour then runnow='Y';
else if rundaily='Y' and whattod=sashour then runnow='Y';
else if runhourly='Y' then runnow='Y';
call symput ('nowdate',compress(trim(sasyear||sasmonth||sasday||sashour||sasmin)));
run;
/* Start ODS to generate job pool recap */
ods listing; /* Cycle ODS to start fresh */
ods listing close;
ods html file=outweb1; /* Note: This does not have to be HTML output -- specify as desired */
run;
proc print data=tasklist1 label;
title1'Entire SASROBOT Job Pool';
title2;
title3;
title4;
title5;
var owner job runmonthly whichdom runweekly whichdow
rundaily whattod runhourly Desc;
run;
/* Create subset of records that need to be run now */
data tasklist2;
set tasklist1;
label
        runnow='Job Submitted'
        subdate='Date Submitted'
        subtime='Time Submitted';
if runnow='Y'; /* Only keep the records that are actually going to be executed */
format subdate date9. subtime time.;
subtime=time();
subdate=date();
run;
/* Create launch file listing jobs to be executed */
data tasklist3;
set tasklist2 end=eof;
file "&launlib.launch&nowdate..bat"; /* This is the launch file with actual files to be executed */
        put @1 "c:\Program Files\SAS Institute\SAS\V8\Sas.exe" 'job
        ;
if eof then do;
        file "&launlib.silo.bat"; /* This is the initiator file that points to the launch file */
        put "&launlib.launch&nowdate..bat";
end;
run; /* Note: two file layers needed for systask to work on this machine*/
/* Execute the initiator file */
systask command "&launlib.silo.bat" ;
run;

/* Update the job launch history */
proc append base=robotlib.subhist data=tasklist3 force;
run;
proc sort data=robotlib.subhist;
by descending subdate descending subtime;
run;
/* Start ODS to generate launch recap */
ods html file=outweb2; /* Note: This does not have to be HTML output -- specify as desired */
run;
proc print data=robotlib.subhist label noobs;

```

```

title1'History Of Submitted Jobs';
title2;
title3"System=&sysscp. SAS Version=&sysver. JobID=&sysjobid";
title4;
title5;
var owner job subdate subtime
runmonthly whichdom runweekly whichdow
rundaily whattod runhourly Desc;
run;
ods html close;
ods listing;
run;
/* Clear librefs */
libname_all_clear;
filename_all_clear;
run;
/* Sleep for an hour */
data_null_;
sashour=hour(datetime());
format sashplus z2.;
if sashour=23 then do;
    sashplus=0;
end;
else do;
    sashplus=sashour+1;
end;
sashfinl=sashplus||":00:00";
call symput('wakehour',sashfinl);
run;
data_null_;
slept=wakeup("&wakehour"t);
run;
%end;
%mend outer;
%outer; /*This actually executes the macro above*/
run;

```

### Robotparms.sas

```

/* You can point these libnames to wherever you deem appropriate */
/* Theoretically, they can all point to the same directory, but it makes testing more difficult*/
%let loglib=r:\projects\robot\log\; /* Allocate log from robot driver */
%let printlib=r:\projects\robot\output\; /* Allocate simple output from robot driver */
%let weblib=r:\home\; /* Allocate HTML summary reports */
%let filelib=r:\projects\robot\data\; /* Allocate robot work files */
%let launlib=r:\projects\robot\launch\; /* Allocate location of launch files */
%let conlib=r:\projects\robot\code\; /* Allocate SAS connect script location */

```

### RobotTestWindowsRaw.sas

```

/* Include robot parms -- this file allocates librefs and filenames */
%include "r:\projects\robot\data\robotparms.sas";
/* SIGNON TO REMOTE */
/* Since the ROBOT is running on Windows, there is no remote signon required*/
options obs=max missing='.' errors=3 nodate notes symbolgen mprint mlogic macrogen source2
        ls=132 ;
run;
/*****Target Job Goes Below This Line*****/
x 'r:\delrmdy.bat';
/*****Target Job Goes Above This Line*****/
run;
/* SIGNOFF FROM REMOTE */
/* Since the ROBOT is running on Windows, there is no remote signon required*/

```

### RobotTestUNIXRaw.sas

```

/* Include robot parms -- this file allocates librefs and filenames */
%include "r:\projects\robot\data\robotparms.sas";
/* SIGNON TO SERVER */
%let mybox=99.99.999.99;
options remote=mybox comamid=tcp;
signon "&conlib.tcpunixautologon.scr";
run;
options obs=max missing='.' errors=3 nodate notes symbolgen mprint mlogic macrogen source2
        ls=132 ;
rsubmit;
/*****Target Job Goes Below This Line*****/
/****Note: In this case we're just executing a UNIX Command *****/
x 'cp //export/home/userid/dead.letter //export/home/userid/dead.copy!';
/*****Target Job Goes Above This Line*****/
endrsubmit;
/* SIGNOFF FROM SERVER */
signoff;
run;

```

### RobotTestMVSSAS.sas

```

/* Include robot parms -- this file allocates librefs and filenames */
%include "r:\projects\robot\data\robotparms.sas";
/* SIGNON TO MAINFRAME */
%let rmtnode=atln1tcp;

```

```

options remote=rmtnode comamid=tcp;
filename tsoscr "&conlib.tcptsoautologon.scr";
signon tsoscr;
options obs=max missing='.' errors=3 nodate notes symbolgen mprint mlogic macrogen source2
ls=132 ;

rsubmit;
/*****Target Job Goes Below This Line*****/
x 'sas input(''userid.sas(test)'' )';
/*****Target Job Goes Above This Line*****/
endrsubmit;
/* SIGNOFF FROM MAINFRAME */
signoff;
run;

```

### RobotTestMVSJCL.sas

```

/* Include robot parms -- this file allocates librefs and filenames */
%include "r:\projects\robot\data\robotparms.sas";
/* SIGNON TO MAINFRAME */
%let rmtnode=atlnltcp;
options remote=rmtnode comamid=tcp;
filename tsoscr "&conlib.tcptsoautologon.scr";
signon tsoscr;
options obs=max missing='.' errors=3 nodate notes symbolgen mprint mlogic macrogen source2
ls=132 ;

rsubmit;
/*****Target Job Goes Below This Line*****/
x 'sub 'userid.testjobs(test1)''; /* This executes a JCL job */
/*****Target Job Goes Above This Line*****/
endrsubmit;
/* SIGNOFF FROM MAINFRAME */
signoff;
run;

```

### Tcpunixautologon.scr (Only the modifications to the standard SAS Connect script are shown)

```

/* input 'userid?'; */
type "rrrrrr" LF; /* <==Put your userid here */
waitfor 'Password', 30 seconds : nolog;
/* input nodisplay 'Password?'; */
type "rrrrrrrr" LF; /* <==Put your pw here */

```

### Tcptsoautologon.scr (Only the modifications to the standard SAS Connect script are shown)

```

waitfor 'IKJ56700A ENTER USERID -', 60 seconds : noinit;
/* input 'userid?'; */
type "rrrrrr" LF; /* <==Put your userid here */
waitfor 'IKJ56714A ENTER CURRENT PASSWORD', 60 seconds : nolog;
/* input nodisplay 'Password?'; */
type "rrrrrrrr" LF; /* <==Put your pw here */
waitfor 'IKJ56481I THE PROCEDURE NAME', 60 seconds : nolog;
type '$$PPUSR' LF;

```

### Robot.Schedule Contents

| #  | Variable   | Type | Len | Pos | Label             |
|----|------------|------|-----|-----|-------------------|
| 10 | Desc       | Char | 100 | 319 | Description       |
| 2  | job        | Char | 300 | 10  | Job Specification |
| 1  | owner      | Char | 10  | 0   | Job Owner         |
| 7  | rundaily   | Char | 1   | 315 | Run Daily         |
| 9  | runhourly  | Char | 1   | 318 | Run Hourly        |
| 3  | runmonthly | Char | 1   | 310 | Run Monthly       |
| 5  | runweekly  | Char | 1   | 313 | Run Weekly        |
| 8  | whattod    | Char | 2   | 316 | Time Of Day       |
| 4  | whichdom   | Char | 2   | 311 | Day Of Month      |
| 6  | whichdow   | Char | 1   | 314 | Day Of Week       |

### Robot.Schedule Job Specification Details (This gets parsed by the robot code and placed in the launch file)

```

-sysin s:\MCIF-Robot\testprogs\RobotTestMVSJCLCode.sas -print s:\MCIF-Robot\testprogs\rtmjc.lst -log s:\MCIF-Robot\testprogs\rtmjc.log -nosplash -icon

```