Batch processing and sysparm

A step towards scheduling
SAS modes of execution

• Foreground PC/SAS or SAS Display Manager
• Client server SAS/Connect or Enterprise Guide
• Background MVS JCL or Unix shell script
• Background SAS on MS Windows
PC/SAS native options

• Right-click on program.sas and Batch submit
• Batch submit from the SAS Viewer
• Windows command shell
Batch submit from MS Explorer
Batch submit from SAS Viewer
Batch submit from command shell

```bash
D:\Docs\farreran\My_presentations\SASBatch_sysparm\FinalCode> dir *.sas /b
slow_ds_build.sas

D:\Docs\farreran\My_presentations\SASBatch_sysparm\FinalCode> ..\scripts\ses91 slow_ds_build.sas
```
Why bother?

• Majority of SAS users have repeatable tasks
• Every SAS user has access to a sas executable
• All platforms have background processing
• Operational groups do not know SAS
• We move on but our code may stay
Operational requirements

• Consistency
• Minimum intervention
• Process level control
• Isolation from other activity
• Graceful failure
• Parallel processing
Existing parallel processing options

- Enterprise Guide Process Designer
- Multiple sessions of EG or PC/SAS
- MP Connect
- %systask coding
- SAS DI Studio
- Scripting
Principles of SAS scripting

• SAS business logic is read-only
• SAS code is portable and self contained
• Use parameters to control processing
• Use environment settings at initialisation
• Keep logs and listings
Combining existing techniques

- SAS macro functions
- Operating system scripting
- Process monitoring
- -sysparm startup option
SAS startup options

• Default options are defined in sasv9.cfg
• Options can be overridden at session startup
• -sysparm must be the last option
• Sas.exe –sysin annual.sas -sysparm “2007”
Enhancing simple reports

• Some basic code:

```sas
%let year = 2004 ;

proc print noobs
  data = sashelp.orsales ;
  where year=&year and product_line='Sports' ;
run ;
```
Enhancing simple reports

• Some basic macro code:

```plaintext
%macro orion_print(year) ;
proc print noobs
  data = sashelp.orsales ;
  where
    year=&year and
    product_line='Sports' ;
run ;
%mend orion_print ;

%orion_print(2004) ;
```
Enhancing simple reports

• Some read-only macro code:

```
%macro orion_print(year) ;
proc print noobs
  data = sashelp.orsales ;
  where
    year=&year and
    product_line='Sports' ;
run ;
%mend orion_print ;

%orion_print(&sysparm.) ;
```
Enhancing simple reports

• Some read-only macro code with basic check:

```
%macro orion_print(year) ;
proc print noobs
  data = sashelp.orsales ;
  where
  %if "&year" ^= "" %then %do ;
  year=&year and
  %end ;
  product_line='Sports' ;
run ;
%mend orion_print ;

%orion_print(&sysparm.) ;
```

Check for non-blank &year
Putting it together

• Open cmd window in SAS code location
• >"c:\Program Files\SAS\SAS 9.1\sas.exe" macro_orion_print -sysparm '2004'
• Notice that macro_orion_print.log and macro_orion_print.lst have been created
Enhancements

- More source code control
- Automatic macro locations
- Return code trapping
- Logging
- Multiple parameter passing
Enhancing simple reports

• Production ready code:

```plaintext
   options mautosource
     sasautos=('&MLIB','!sasroot/sasautos');

   %orion_print(&sysparm.);
```
echo off
"C:\SAS\SAS_Home\SAS 9.1\sas_batch.exe" %*
if errorlevel 2 if not errorlevel 3 goto err
if errorlevel 1 if not errorlevel 2 goto warn
if errorlevel 0 if not errorlevel 1 goto ok
:warn
echo SAS exited with a Warning. Check %1.log
goto end
:err
echo SAS exited with a Error. Check %1.log
echo Return code = %ERRORLEVEL%
goto end
:ok
echo Seems OK.
echo Return code = %ERRORLEVEL%
echo Parm1 = %1
echo SASlog is %1.log
:end
echo End of sas91.bat
Adding more controls

- Add more parameters to basic code:

```plaintext
%let year = 2004;
%let pline = Sports;

proc print noobs
  data = sashelp.orsales;
  where year=&year and product_line="&pline";
run;
```
Adding more controls

• Add more parameters to macro:

```sas
%macro orion_print2(year,pline) ;
proc print noobs
  data = sashelp.orsales ;
  where year=&year and product_line="&pline" ;
run ;
%mend orion_print 2;

%multi_sysparm ;  ← Creates multiple macro variables

%orion_print2(year,pline) ;
```
Multi sysparm macro

- Accepts multiple name value pairs e.g –sysparm ‘name1=value1,name2=value2’
- &name1 &name2 are %global
Sysparm splitting example

%let sysparm = name1=value1,name2=value2,name3=value3 ;

data _null_;
  length sysparm express param value $ 200;
  sysparm = symget('sysparm');
  do i=1 to 50 until(express = '');
    express = left(scan(sysparm, i, ',')); /* name=value */
    param   = left(upcase(scan(express, 1, '='))); /* name */
    value   = left(scan(express, 2, '='));
    valid   = not verify(substr(param, 1, 1),'ABCDEFGHIJKLMNOPQRSTUVWXYZ_')
      and     not verify(trim(param),'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')
      and     length(param) <= 32 ; /* Ensure valid V8 macrovar name */
    if valid then call symput(param, trim(left(value)));
  end;
run;

%put _user_ ;

Gives

GLOBAL NAME2 value2
GLOBAL NAME3 value3
GLOBAL NAME1 value1
Putting it together – Part 2

- Open cmd window in SAS code location
- `>..\scripts\sas91.bat macro_orion_print2 -sysparm "year=2001,pline='Sports'"`
Other additional information

Google search string for SUGI and SGF papers: sysparm site:www2.sas.com filetype:pdf

Peter Crawford’s very robust techniques for isolating PC-SAS projects
Bernard Tremblay’s reference to Rick Aster’s split sysparm code
http://listserv.uga.edu/cgi-bin/wa?A2=ind0009D&L=sas-l&P=R11619
How SAS startup configurations are decided
Elaborate system that uses SYSTASK for controlling process flow
Comprehensive Unix based production system
Using SYSPARM and other automatic variables
Very comprehensive and complicated script
http://www.sascommunity.org/wiki/Batch-submission_WSH_Script