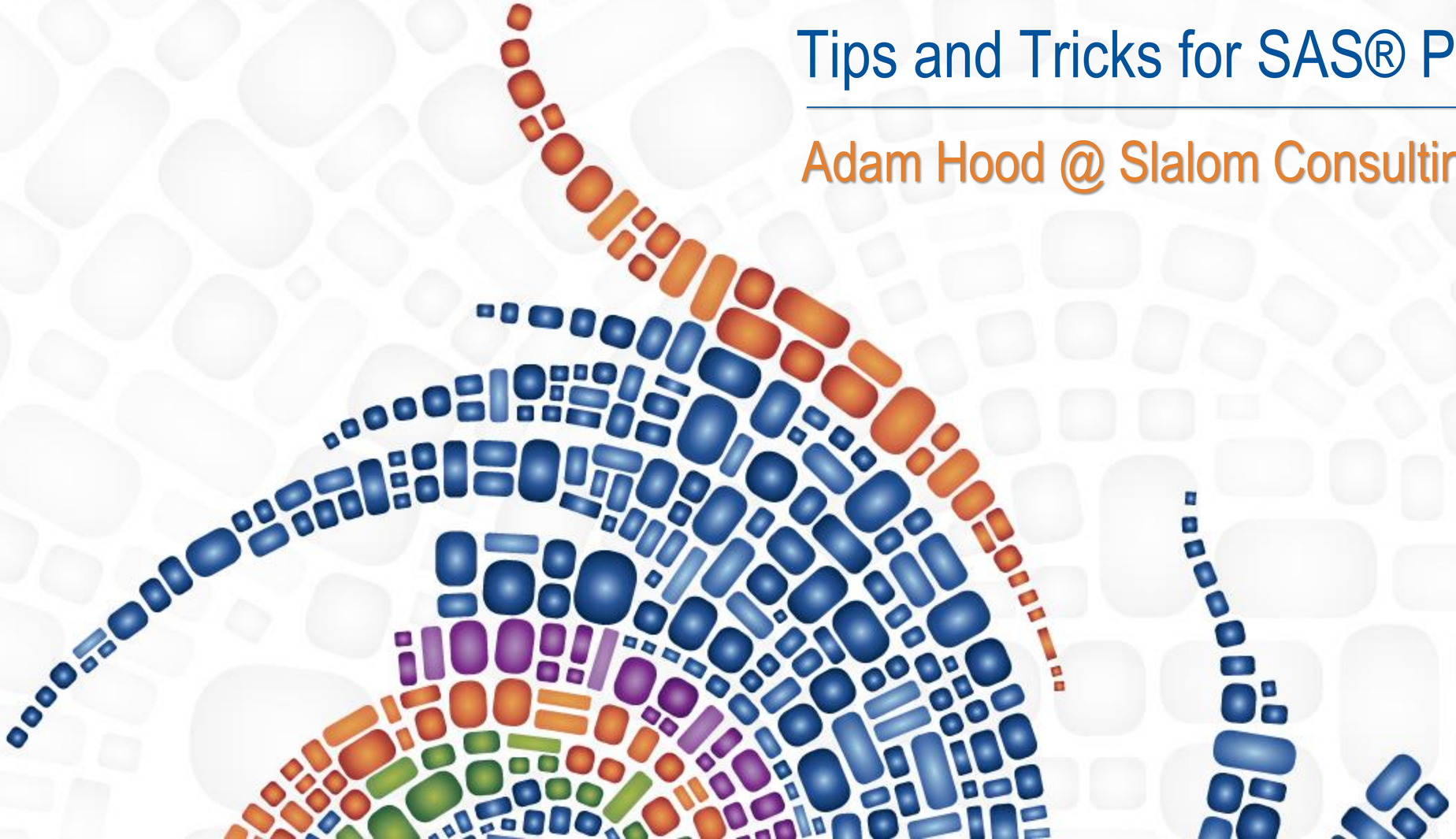
SAS® GLOBAL FORUM 2015

The Journey Is Yours

Tips and Tricks for SAS® Program Automation

Adam Hood @ Slalom Consulting



SAS GLOBAL FORUM

Tips and Tricks for SAS® Program Automation

Adam Hood
Slalom Consulting

Abstract

Automating SAS programs can be difficult and result in overcoming unexpected obstacles. Often automation forces us to account for edge case scenarios that we would normally assume won't be encountered.
The goal of this poster is to help programmers overcome the 'gotcha's associated with building and troubleshooting automated programs. We will touch on 3 areas: logging, basic error handling, and system options.

Logging Tip #1

Use PROC PRINTTO to put all logs in a designated directory. This will allow any team member to quickly find and troubleshoot problems in processes. All too often teams spend time trying to find the appropriate logs for a given process.
To further enhance your log files, add a timestamp for easy sorting and a naming convention for specific process identification.

```
proc format ;
        picture myfmt low-high = '%0Y%0m%0d%0H%0M%0S'
        (datatype = datetime) ;
run ;

%let DTstamp =%sysfunc(datetime(),myfmt.);

proc printto log="/home/ahood/logs/myprocess_&DTstamp..log" new;
```

Logging Tip #2

Often there are multiple stakeholders for a given automated job. Emails are a great way to let others know when jobs finish. However, if you want a more self-service approach or you want to build dependencies with other systems, job run logs stored in a database may be the answer. This allows other systems to access and report on job status.
An important part of successfully implementing a busy job run log is to limit logging to inserts. Updates and deletes should be avoided if possible. Best practice would be to create a database view that filtered the data to the last status per process per day (or process run).
The DTstamp macro variable from tip #1 makes a great process identifier.

Table: job_run_log

*	process_id	process_name	status	note	rcrd_create_ts
1	20150305105033	gather_stats	started	(null)	2015-03-05 10:51:22
2	20150305105314	gather_stats	complete	Completed successfully	2015-03-05 10:53:55
3	20150305115351	load_test	started		2015-03-05 11:54:26
4	20150305132512	load_test	complete	Completed successfully	2015-03-05 13:26:01

```
CREATE TABLE job_run_log
        (  process_id bigint NOT NULL, process_name VARCHAR(100),
        status VARCHAR(50), note VARCHAR(255),
        rcrd_create_ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY (process_id, rcrd_create_ts)    )
```

View: job_run_log_v

process_id	process_name	status	note	rcrd_create_ts
20150305105033	gather_stats	complete	Completed successfully	2015-03-05 10:53:55
20150305115351	load_test	complete	Completed successfully	2015-03-05 13:26:01

```
CREATE VIEW job_run_log_v as
select a.* from test.job_run_log a
        join (
                select process_id, max(rcrd_create_ts) as last_ts
                from test.job_run_log
                group by 1
        ) b on a.process_id = b.process_id and a.rcrd_create_ts = b.last_ts
;
```

Logging Tip #3

Often we need to understand when something happened at the second level. This is especially true for long running jobs.
To aid in this effort, we add timestamps to the SAS log for key events or interactions.

```
%macro timestamp_it();
/* Note that the format statement can be moved to the settings files for
the job */

proc format;
picture DTstamp
        other='%Y-%0m-%0d %0H:%0M:%0S' (datatype=datetime);
run;

%put NOTE: %sysfunc(datetime(),DTstamp.);
%mend;

%timestamp_it;
```

Error Handling Tip #1

There are occasions where missing data can cause a domino effect of downstream errors. We find it helpful to check the rows returned by critical queries so an error can be thrown immediately instead of downstream. Here is an example macro for just that.

```
%macro check_sql;
%if &sqlobs.=0 %then %do;
        %put ERROR: SQL Query returned no rows.;
%end;
%else %do;
        %put NOTE: SQL Checked.;
%end;

%mend check_sql;
```

Error Handling Tip #2

Locked datasets can cause problems if there are potentially multiple processes accessing them. We have removed this by locking a dataset first, accessing or updating, and then unlocking. Here are examples of how this works from PharmaSUG2005.

```
%macro trylock(member=,timeout=10,retry=1);

%local starttime;
%let starttime = %sysfunc(datetime());

%do %until(&syslckrc <= 0
        or %sysevalf(%sysfunc(datetime()) > (&starttime + &timeout)));
        %put trying open ...;
        data _null_;
                dsid = 0;
                do until (dsid > 0 or datetime() > (&starttime + &timeout));
                dsid = open("&member");
                if (dsid = 0) then rc = sleep(&retry);
                end;
                if (dsid > 0) then rc = close(dsid);
        run;
        %put trying lock ...;
        lock &member;
        %put syslckrc=&syslckrc;
%end;
%mend trylock;
```

System Options Tips

Debugging is critical when working with automated programs. Our approach is to turn on the debugging options in the settings file for the job. This produces larger logs, but we overcome this with an aggressive log retention policy.
There are 4 options that we use to help debug programs.

MPRINT – If you use macros in any significant way, this will help you debug. MPRINT will print the macro as it is included in the code.

MLOGIC – This option is different from MPRINT in that it displays the logic that is being executed. Therefore, if you have for loops or conditional statements, MLOGIC will help understand the execution.
SYMBOLGEN – Have you ever had to include %PUT statements just to know what the macro variables resolve to? SYMBOLGEN does that for you. It outputs the value of macro variables as they are used.
FULLSTIMER – Briefly, FULLSTIMER outputs performance statistics for each step in a SAS job. This helps debug long running queries, etc.

References

SAS Community Tips – Adding a date and time stamp to the SAS Log - http://www.sascommunity.org/wiki/Tips:Adding_a_date_and_time_stamp_message_to_the_SAS_Log
The Big Introduction from the Smallest Macro - http://www2.sas.com/proceedings/sugi31/038-31.pdf
Automating your SAS jobs, Paul Anderson, SGF 2009 - http://support.sas.com/resources/papers/proceedings09/184-2009.pdf
Play Nice with Others: Waiting for a Lock on SAS Table, John Leveille and Jimmy Hunnings, PharmaSUG 2005 - http://www.lexjansen.com/pharmasug/2005/posters/po33.pdf

SAS® GLOBAL FORUM

April 26-29
Dallas, TX