

Karl Quon (Cont')

Family: My wonderful wife Patricia, one “Princess in Training” daughter, and my spunky son.

Pets: Do golf clubs count?

Sports/Hobbies: Golf, running, home improvement and more golf.

What your ideal weekend would be:

Private jet to Augusta
Tee time at Augusta National
Private jet to Oahu
Seafood dinner by the ocean
Tee time at Kapalua the next morning
(Ok, I can wake up now)

Favourite Foods: Sushi, anything my wife makes me.

If I could be anything at all (besides a SAS consultant): I would be: A professional golfer.

When I’m not involved on SAS consulting projects, I like to: Watch my kids grow up, train for marathons, oh yeah, and golf.

Something I’ve learned out in the field doing consulting work for SAS that I feel would benefit other SAS users is:

Security and password protection. Having a userid and password in the connection string to a DBMS poses a potential security hole, especially if it is left open in a .sas program or autoexec.sas file.

One option is to embed this information in a precompiled macro. The catch here is that the macro is then vulnerable to run-time inspection using various SAS options. To protect against it, use a macro like the one below to detect for presence of these SAS options – mlogic, mprint, symbolgen.

The macro works like so:

- Cache the initial value of the SAS options mentioned above.
- Turn off the SAS options (e.g., nomlogic, nomprint, nosymbol) in the macro.
- Connect to the database.
- Reset the SAS options back to their initial value.

For example, use this macro in conjunction with explicit SQL Pass Through statements.

Step 1 – Precompile the macro.

```
%macro connectDB (database=undefined)
  / store des = 'Database Connection Macro';
```

```

*****;
* Initializations;
*****;

%local COMPNT;
%local MODULE;

%let COMPNT = UTILITY;
%let MODULE = CONNECTDB;

* following SAS options, if set by the user will present a security gap.;
* As a result, if any option is set, it will be disabled for the duration;
* of this macro and reenabled at the end of the macro (if applicable).;

* the MACROGEN option is not in sashelp.voption, disable it in any case.;
* Note: SAS Tech Support Note SN-001821 only recommends this option be;
* used in Version 5.;

options nomacrogen;

* determine state of MPRINT option;
select count(*) into :mprint
  from sashelp.voption
  where upcase (optname) = 'MPRINT'
     and upcase (setting) = 'MPRINT';

%if (&mprint EQ 1) %then %do;
  options nomprint;
%end;

* determine state of MLOGIC option;
select count(*) into :mlogic
  from sashelp.voption
  where upcase (optname) = 'MLOGIC'
     and upcase (setting) = 'MLOGIC';

%if (&mlogic EQ 1) %then %do;
  options nomlogic;
%end;

* determine state of SYMBOLGEN option;
select count(*) into :symbolgen
  from sashelp.voption
  where upcase (optname) = 'SYMBOLGEN'
     and upcase (setting) = 'SYMBOLGEN';

%if (&symbolgen EQ 1) %then %do;
  options nosymbolgen;
%end;

*****;
* Program Logic;
*****;

%global prod_db;

%local user;
%local password;
%local database;
%local schema;

* Locale: Pebble Beach Database;

```

```

%if (%lowcase (%trim (&database)) EQ pebble) %then %do;

    %let user      = kquon;      * db2 userid;
    %let password = golfing;    * db2 password;
    %let database  = pebble;    * db2 database;
    %let schema    = beach;     * db2 schema;

    * db connection string;
    %let db2Parm = user=&user using=&password database=&database schema=&schema;

    * establish connection to database;
    connect to DB2 as pebble (&db2Parm);

%end;

* Locale: Augusta National Database;
%else %if (%lowcase (%trim (&database)) EQ augusta) %then %do;

    %let user      = kaquon;     * db2 userid;
    %let password = golfing;    * db2 password;
    %let database  = augusta;   * db2 database;
    %let schema    = national;  * db2 schema;

    * db connection string;
    %let db2Parm = user=&user using=&password database=&database schema=&schema;

    * establish connection to database;
    connect to DB2 as augusta (&db2Parm);

%end;

%else %do;
    %put ERROR: [&COMPNT:&MODULE] Invalid database name.;
    %let sqlrc = 1;
%end;

* restore SAS options to their original values;
%if (&symbolgen EQ 1) %then %do;
    options symbolgen;
%end;

%if (&mlogic EQ 1) %then %do;
    options mlogic;
%end;

%if (&mprint EQ 1) %then %do;
    options mprint;
%end;

%mend connectDB;

```

Step 2 – Using the macro

```

* try and get the userid and password;
options mlogic;
options mprint;
options symbolgen;

proc sql;
%connect (database=pebble);
*<your sql statements here>;
disconnect from pebble;

```

```
quit;
```

Step 3 – Inspect SAS Log

Output of the SAS log (note: userid and password are not displayed).

```
MLOGIC(CONNECTDB): Beginning execution.
MLOGIC(CONNECTDB): Parameter DATABASE has value td_aml
MLOGIC(CONNECTDB): %LOCAL COMPNT
MLOGIC(CONNECTDB): %LOCAL MODULE
MLOGIC(CONNECTDB): %LET (variable name is COMPNT)
MLOGIC(CONNECTDB): %LET (variable name is MODULE)
MPRINT(CONNECTDB): options nomacrogen;
MPRINT(CONNECTDB): select count(*) into :mprint from sashelp.voption where upcase
(optname) = 'MPRINT' and upcase (setting) = 'MPRINT';
SYMBOLGEN: Macro variable MPRINT resolves to 1
MLOGIC(CONNECTDB): %IF condition (&mprint EQ 1) is TRUE
MPRINT(CONNECTDB): options nomprint;
SYMBOLGEN: Macro variable MLOGIC resolves to 1
MLOGIC(CONNECTDB): %IF condition (&mlogic EQ 1) is TRUE
SYMBOLGEN: Macro variable SYMBOLGEN resolves to 1
SYMBOLGEN: Macro variable MLOGIC resolves to 1
SYMBOLGEN: Macro variable MPRINT resolves to 1
MLOGIC(CONNECTDB): %IF condition (&mprint EQ 1) is TRUE
MLOGIC(CONNECTDB): Ending execution.
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.55 seconds
      cpu time           0.02 seconds
```

Exercise for the reader: modify this macro for Implicit DBMS Libraries.