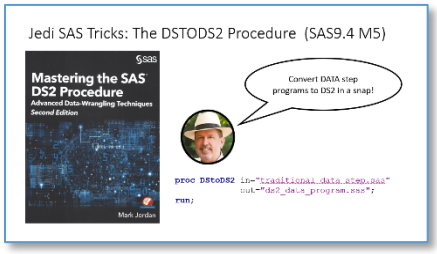
Jedi SAS Tricks - The DStoDS2 Procedure

By SAS Jedi on SAS Learning Post



You have the infrastructure, and you *know* that DATA step would run so much better in DS2, but it's complicated and you don't know how to get started. Well, if you have SAS 9.4M5, rejoice! The DStoDS2 procedure is here, and it's a humdinger! (No, Chris, not a Hemedinger… ;-) It's a sample of the new things in the second edition of "Mastering the SAS DS2 Procedure - Advanced Data Wrangling Techniques"

Here are a few important caveats we should be aware of before diving in:

1. PROC DStoDS2 can't translate all possible DATA step syntax, but it does support a huge subset. Any lines of code lines in the traditional DATA step that cannot be translated will appear as comments in the output. Any comments in the original DATA step program are removed.
2. PROC DStoDS2 translates only one DATA step at a time. The DATA step to be translated must be saved to its own text file which will serve as input.
3. In the traditional DATA step, character variable lengths are measured in *bytes*, but in DS2 fixed-length character variable lengths are specified in *characters*. If your data has multi-byte encoding, adjust any LENGTH statements in the original DATA step before running PROC DSTODS2 to specify the variable length in *characters*, so that the DS2 DATA program will specify the correct lengths upon conversion.
4. The resulting DS2 program might not be syntactically complete. You will have to add a RUN statement and any commented sections will have to be converted by hand. Perhaps a few other tweaks will be necessary before it executes and produce the desired results.

Having said all of that, PROC DStoDS2 make converting traditional DATA step code to DS2 a snap!

Let's try it out on a sample program that as a lot of typical DATA step features:

```
data audi   (drop=make count)
     bmw    (drop=make count)
     counts (keep=make count);
   if _n_=1 then do;
      put 'And so it begins!';
   end;
   set cars;
   by Make;
   if first.make then Count=0;
   Count+1;
   select (Make);
      when ("Audi") output audi;
      when ("BMW")  output bmw;
      otherwise put Make= Model= ' - how did YOU get in here?';
   end;
   if last.make then output counts;
run;
```

We'll save that to a file named **`DStoDS2_data_step.sas`**. Next, we'll write and execute a PROC DStoDS2 step to convert the DATA step program to DS2 and save the new program as **`DStoDS2_data_step_converted.sas`**:

Jedi SAS Tricks - The DStoDS2 Procedure

By SAS Jedi on SAS Learning Post

```
proc dstods2 in="&path/DStoDS2_data_step.sas"
             out="&path/DStoDS2_data_step_converted.sas";
run;
```

A quick look at the output shows it's not too pretty, but it seems functional enough. Because I'm working in SAS Studio, a quick click on the Format Code button makes it more legible. For the purposes of the blog, I'll format it a little by hand:

```
data AUDI(DROP=(MAKE COUNT) )
     BMW(DROP=(MAKE COUNT) )
     COUNTS(KEEP=(MAKE COUNT));
   method run();
   if _N_=1.0 then
      do;
         put 'And so it begins!';
      end;
   set CARS;
   by MAKE;

   if FIRST.MAKE then
      COUNT=0.0;
   COUNT + 1.0;

   select (MAKE);
      when ('Audi') output AUDI;
      when ('BMW') output BMW;
      otherwise put MAKE=MODEL=' - how did YOU get in here?';
   end;

   if LAST.MAKE then
      output COUNTS;
   ;
_return:
   ;
end;
enddata;
```

That's not too bad, actually. I have to add the PROC DS2 statement and the RUN and QUIT, obviously. I'll remove that unused _return: label. And for efficiency's sake, I'll convert the IF _N_ =1 DO group into an INIT method:

Jedi SAS Tricks - The DStoDS2 Procedure

By SAS Jedi on SAS Learning Post

```
proc ds2;
data AUDI_DS2(DROP=(MAKE COUNT))
     BMW_DS2(DROP=(MAKE COUNT))
     COUNTS_DS2(KEEP=(MAKE COUNT));
   method init();
         put 'And so it begins!';
   end;
   method run();
      set CARS;
      by MAKE;
      if FIRST.MAKE then COUNT=0.0;
      COUNT + 1.0;
      select (MAKE);
         when ('Audi') output AUDI_DS2;
         when ('BMW') output BMW_DS2;
         otherwise put MAKE=MODEL=' - how did YOU get in here?';
      end;
      if LAST.MAKE then output COUNTS_DS2;
   end;
enddata;
run;
quit;
```

Now, that seemed almost too easy, didn't it? I'll just run the original DATA step program, run my shiny new DS2 program to make sure they are producing the same output. When I executed the program, I received a warning in the log:

```
WARNING: No DECLARE for assigned-to variable count; creating it as a global variable
         of type double.
```

So back I'll go to add a declaration for the COUNT variable, and this time it runs without a warning. I'm a suspicious guy, so I'll sort the outputs from both processes into the same order and run PROC COMPARE to ensure it's all working as expected – and sure enough, I get that lovely to see report "No unequal values were found. All values compared are exactly equal."

So that's the scoop – PROC DStoDS2 can ingest some pretty complex DATA step code and do 99% of the work needed to convert it to DS2. Remember Jedi Programming Rule #1? If not, here you go: "Lazy programmers are great programmers."

So lazy programmers of the world, rejoice - the DStoDS2 procedure has arrived! And, if you're too lazy to copy the code in this post by hand, you can download the ZIP file from this link.

Until next time, may the SAS be with you!
Mark