Jedi SAS Tricks – Roll Your Own Function

The other day, a former student here in the US emailed me about reading in UTC (Coordinated Universal Time) datetime data from server logs. To make future calculations and comparisons easier, he wanted to translate the UTC datetime to a local datetime. He said the INTNX() function worked great, until he had to account for daylight savings time (DST). And he was wondering "Does SAS have an interval calculating function that takes DST into account?" I couldn't find one, so I had to answer no. I hate that! I thought "Surely, the power of the SAS can bend this data to my will!" And so off I went to tackle the SAS Function Compiler procedure, affectionately known as PROC FCMP.

Introduced in SAS 9.1, PROC FCMP lets you to create custom SAS functions and CALL routines. PROC FCMP syntax is very much like DATA step, and you can leverage most features of Base SAS when defining your routines. The custom functions and CALL routines created are used in subsequent DATA steps or SAS procedures just as you would any standard SAS function or CALL routine.

First I searched for some starter code, and found this excellent tidbit "Sample 24735: Compute daylight saving time". Next, I thought I remembered that Canada and the US shared DST, but because the US had legislated recent changes, I checked Wikipedia. It says we still do – and I read it on the Internet, so it must be true! So I determined I'd write a function what would convert DST to local time, valid for Canada and US time zones.

The first step was writing a data step to adjust the data time values and test it.

```
data TEST;
   /* Set values for testing */
   DT='26JUN12:17:00:00'dt;
   FORMAT DT DATETIME. NEWDATE DATETIME.;
   ZONE=-5;
   year=year((datepart(dt)));
   day1=mdy(1,1,year);
   /* Year <=2006 DST starts 1st SUN in April, ends last SUN in October */
   if year <= 2006 then do;
      dst_month_start=intnx('month',day1,3);
      dst_beg=intnx('week.1',dst_month_start,(weekday(dst_month_start) ne 1));
      dst_month_end=intnx('month',day1,9);
      dst_end=intnx('week.1',dst_month_end,(weekday(dst_month_end) in (6,7))+4);
   end;
   /* Year >2006 DST starts 2nd SUN in March, ends 1st SUN in November. */
   else do;
      dst_month_start=intnx('month',day1,2);
      dst_beg=intnx('week.1',dst_month_start, (weekday(dst_month_start) in
(2,3,4,5,6,7))+1);
      dst_month_end=intnx('month',day1,10);
      dst_end=intnx('week.1',dst_month_end,(weekday(dst_month_end) ne 1));
   end;
   NEWDATE=dt+(zone*60*60);
   if dst_beg <= datepart(dt)<= dst_end then NEWDATE=NEWDATE+(3600);
   putlog _all_;
run;
```

This worked like a charm, so now it was time to convert my data step code into a function with PROC FCMP. However, it was a pain to look up the correct "minus number" for my time zone while I was testing. I'd rather be able to use the text time zone name - 'Eastern' for example. So, in my function I added a new character variable Z, and a SELECT group to do the conversion for the ZONE variable. The function will take 2 parameters: DT (a datetime value) and Z (the name of the time zone). Here is my code:

```
proc fcmp outlib=sasuser.myfunc.UTC2LOCAL;
   function UTC2LOC(dt,z $);
      Z=UPCASE(Z);
      select (z);
         when ('NEWFOUNDLAND') ZONE=-3.5;
         when ('ATLANTIC') ZONE=-4;
         when ('EASTERN') ZONE=-5;
         when ('CENTRAL') ZONE=-6;
         when ('SASKATCHEWAN') ZONE=-6;
         when ('MOUNTAIN') ZONE=-7;
         when ('PACIFIC') ZONE=-8;
         when ('ARIZONA') ZONE=-7;
         otherwise do;
            zone=0;
            PUT 'Valid zones: NEWFOUNDLAND, ALLANTIC, EASTERN, CENTRAL, '
                'MOUNTAIN, PACIFIC, ARIZONA or SASKATCHEWAN.';
         end;
      end;
      year=year((datepart(dt)));
      day1=mdy(1,1,year);
      /* Year <=2006 DST starts 1st SUN in April, ends last SUN in October */
      if year <= 2006 then do;
         dst_month_start=intnx('month',day1,3);
         dst_beg=intnx('week.1',dst_month_start
                       ,(weekday(dst_month_start) ne 1));
         dst_month_end=intnx('month',day1,9);
         dst_end=intnx('week.1',dst_month_end
                       ,(weekday(dst_month_end) in (6,7))+4);
      end;
      /* Year >2006 DST starts 2nd SUN in March, ends 1st SUN in November. */
      else do;
         dst_month_start=intnx('month',day1,2);
         dst_beg=intnx('week.1',dst_month_start
                       ,(weekday(dst_month_start) in (2,3,4,5,6,7))+1);
         dst_month_end=intnx('month',day1,10);
         dst_end=intnx('week.1',dst_month_end,(weekday(dst_month_end) ne 1));
      end;
      NEWDATE=dt+(zone*3600);
      if dst_beg<=datepart(dt)<=dst_end and Z NOT IN ('ARIZONA','SASKATCHEWAN')
         then NEWDATE=NEWDATE+(3600);
      return(NEWDATE);
   endsub;
run;
```

And now to test my handiwork! This data step will use the new function to convert a UTC datetime value to local time:

```
data TEST;
   do UTC='31JAN12:17:00:00'dt
         ,'31MAR12:17:00:00'dt
```

```
         ,'30JUN12:17:00:00'dt
         ,'30NOV12:17:00:00'dt;
      Newfoundland=UTC2LOC(UTC,'Newfoundland');
      Atlantic=UTC2LOC(UTC,'atlantic');
      Eastern=UTC2LOC(UTC,'EASTERN');
      Central=UTC2LOC(UTC,'Central');
      Saskatchewan=UTC2LOC(UTC,'SASKATCHEWAN');
      Mountain=UTC2LOC(UTC,'Mountain');
      Arizona=UTC2LOC(UTC,'Arizona');
      Alaska=UTC2LOC(UTC,'Alaska');
      Hawaii=UTC2LOC(UTC,'Hawaii');
      output;
   end;
   FORMAT UTC -- HAWAII datetime.;
run;

PROC PRINT data=test noobs;
   format UTC -- HAWAII tod.;
RUN;
```

Output:

UTC	Newfoundland	Atlantic	Eastern	Central	Saskatchewan	Mountain	Arizona	Alaska	Hawaii
17:00:00	13:30:00	13:00:00	12:00:00	11:00:00	11:00:00	10:00:00	10:00:00	08:00:00	06:00:00
17:00:00	14:30:00	14:00:00	13:00:00	12:00:00	11:00:00	11:00:00	10:00:00	09:00:00	06:00:00
17:00:00	14:30:00	14:00:00	13:00:00	12:00:00	11:00:00	11:00:00	10:00:00	09:00:00	06:00:00
17:00:00	13:30:00	13:00:00	12:00:00	11:00:00	11:00:00	10:00:00	10:00:00	08:00:00	06:00:00

It works just like I had hoped! I wonder – will it work in PROC SQL? Let's see:

```
proc sql;
select UTC
      ,UTC2LOC(UTC,'EASTERN') format=datetime. as Eastern
   from test
;
quit;
```

Output:

UTC	Eastern
31JAN12:17:00:00	31JAN12:12:00:00
31MAR12:17:00:00	31MAR12:13:00:00
30JUN12:17:00:00	30JUN12:13:00:00
30NOV12:17:00:00	30NOV12:12:00:00

Beautiful - I *like* this PROC FCMP!!