

Tired of CALL EXECUTE? Try DOSUBL

Jueru Fan, PPD, Morrisville, NC

ABSTRACT

DOSUBL was first introduced as a function in SAS® V9.3. It enables the immediate execution of SAS® code after a text string is passed. Macro variables that are created or updated during the execution of the submitted code are exported back to the calling environment. With this feature it can be treated as a powerful alternative to traditional CALL EXECUTE when creating macros. This paper demonstrates the principal difference between DOSUBL and CALL EXECUTE with a real-world application. With a thorough understanding of the discussion in this paper a reader should be able to apply the technique more generally.

INTRODUCTION

DOSUBL was first introduced as a new function in SAS® V9.3. It enables the immediate execution of SAS® code after a text string is passed. Macro variables that are created or updated during the execution of the submitted code are exported back to the calling environment. This execution timing difference makes DOSUBL more advanced than CALL EXECUTE under certain circumstances when users need to use the macro variables created during the execution.

This paper will first use a simple example to explain this difference. Then a real-world application from the author's daily work will be introduced. Originally CALL EXECUTE was used in this application. With DOSUBL added in SAS® V9.3, the the author will demonstrate how one part of the application can be modified with DOSUBL, and why CALL EXECUTE cannot function in the same way.

DOSUBL SYNTAX

```
data _null_;  
    rc=DOSUBL(x); ← 1  
run;
```

The DOSUBL function enables the immediate execution of SAS® code after a text string is passed. Macro variables that are created or updated during the execution of the submitted code are exported back to the calling environment.

DOSUBL returns a value of zero if SAS® code was able to execute, and returns a nonzero value if SAS® code was not able to execute X at arrow 1 specifies a text string.

Let's look at an example.

```
data _null_; ← 1  
    length b $1;  
    a="data _null_; call symputx('execute',_n_); run;"; ← 2  
    CALL EXECUTE(a);  
    b=symget("execute");  
    put (b) (=);  
run;
```

SAS® log:

NOTE: Invalid argument to function SYMGET at line 5 column 9.

b=

b= a=data _null_; call symputx('execute',_n_); run; _ERROR_=1 _N_=1

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

The purpose of the example above is to execute a data step in which a macro variable `&execute` was created, and to resolve this macro variable in the same data step which has the `CALL EXECUTE`. The error note in SAS® log is of no surprise to us as `CALL EXECUTE` only executes when the current data step ends. In this example, data step at arrow 2 only executes after data step at arrow 1 ends.

Now let's replace `CALL EXECUTE` with `DOSUBL` in this example.

```
data a2;
  length b $1;
  a="data a1; call symputx('DOSUBL',_n_); run;";
  rc=DOSUBL(a);
  b=symget("DOSUBL");
  put (b) (=);
run;
```

SAS® log:

NOTE: The data set WORK.A1 has 1 observations and 0 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

b=1

NOTE: The data set WORK.A2 has 1 observations and 3 variables.

NOTE: DATA statement used (Total process time):

real time	0.51 seconds
cpu time	0.01 seconds

After revising the example, a reader may notice that two actual data sets `A1` and `A2` are created. The purpose is to better demonstrate the order of execution. As we can see, data set `A1` is created before data set `A2`, and macro variable `&DOSUBL` has already been created before data step `A2` ends. This is consistent with the feature of `DOSUBL`, it immediately executes and, directly afterwards, returns the results to the calling environment.

A REAL-WORLD APPLICATION DISCUSSION

It is common to process data files from various sources. At times it is not American Standard Code for Information Interchange (ASCII) compliant. If a character is defined in the standard ASCII table, it then needs to be identified. Therefore the author created a macro called `%CheckNonAscii` to find which non-ASCII characters there were in data, and positions of them. For demonstration purposes, certain simplifications and modifications have been made to this macro. The macro code is as following:

```
libname test "D:\_TD7452_*****_TEST"; ← 1

%macro CheckNonAscii(dsnin=, var=, dsnout=);
data temp;
  set test.&dsnin;

  length ascii_string $96 not_ascii_char $1;
  do x = 32 to 127;
    ascii_string=trim(left(ascii_string))||byte(x); ← 2
  end;
```

```

not_ascii_pos = verify(&var,ascii_string);
if not_ascii_pos>=1 then not_ascii_char = substr(&var, not_ascii_pos, 1);
if not missing(not_ascii_char);
run;

Data dsn; ← 3
    If 0 then set temp(drop = _all_) nobs = totobs;
    Obs = totobs;
    output;
    stop;

Run;

data _null_;
    set dsn;
    if obs>0 then do;
        CALL EXECUTE("data &dsnout; set temp; run;"); ← 4
    end;
run;

%mend CheckNonAscii;

```

Let's do a brief walkthrough of this macro.

Arrow 1: a temporary SAS® library `TEST` is created to mimic the author's routine working library. Data sets to be checked for non-ASCII will be created in this library.

- Arrow 2: in ASCII collating system, values between 32 and 127 specifies standard and printing ASCII characters. These characters are allowable in the author's work. `BYTE` function was used to create these characters.
- Arrow 3: as it is possible that a working data set does not contain any non-ASCII characters, data set `TEMP` can be missing. This data step created a temporary data set `DSN` to count the number of observation in data set `TEMP`. Only if `TEMP` is not missing an output data set will be created in the next data step. Note that discussion focused on this part will be formed in later section.
- Arrow 4: note double quotations are used in the `CALL EXECUTE` routine to make sure `&dsnout` to be resolved. As the difference of single quotation and double quotation in a `CALL EXECUTE` routine is beyond the scope of this paper, it will not be discussed here.

Then we create two data sets in `TEST` library:

```

data test.a1 test.a2;
    a = byte(128);
    output test.a1;

    a = byte(128);
    output test.a2;

run;

```

As we can see, a non-ASCII value has been added as value of the variable `a` in both data sets. These 2 data sets will be used as source data sets for the `%CheckNonAscii` macro.

SINGLE MACRO CALL WITH CALL EXECUTE VS. DOSUBL

First let's start with single macro call by feeding the macro with data set `A1`. `CALL EXECUTE` and `DOSUBL` will be used separately to call the macro, and SAS® logs will follow.

```

/*CALL EXECUTE example*/
data _null_;
    text='%nrstr(%CheckNonAscii(DSNIN=a1,VAR=a,DSNOUT=b1))';
    CALL EXECUTE(text);
run;

```

SAS® log:

NOTE: DATA statement used (Total process time): < 1

real time	0.01 seconds
cpu time	0.01 seconds

NOTE: CALL EXECUTE generated line. < 2

```

1  + data temp;  set test.a1;  length ascii_string $96 not_ascii_char $1; do x = 32 to
127;  ascii_string=trim(left(ascii_string))||byte(x); end; not_ascii_pos =
verify(a,ascii_string); if not_ascii_pos>=1 then not_ascii_char = substr(a, not_ascii_pos, 1
2  +); if not missing(not_ascii_char); run;

```

NOTE: There were 1 observations read from the data set TEST.A1.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

```

2  +                               Data dsn;  If 0 then set temp(drop = _all_)
nobs = totobs;  Obs = totobs;  output;  stop; Run;

```

NOTE: The data set WORK.DSN has 1 observations and 1 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

```

2  +
data _null_;  set dsn;  if obs>0 then do;      call execute("data b1; set temp; run;");
end
3  +; run;

```

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

NOTE: CALL EXECUTE generated line.

```

1  + data b1; set temp; run;

```

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.B1 has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

/*DOSUBL example*/

```

data _null_;
    text='%CheckNonAscii(DSNIN=a1,VAR=a,DSNOUT=b1)';
    rc=DOSUBL(text);
run;

```

SAS® log:

NOTE: There were 1 observations read from the data set TEST.A1.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

NOTE: The data set WORK.DSN has 1 observations and 1 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

NOTE: CALL EXECUTE generated line.

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.B1 has 1 observations and 5 variables. ← 4

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

NOTE: DATA statement used (Total process time): ← 5

real time	0.40 seconds
cpu time	0.04 seconds

Apparently both codes performed as expected and managed to serve our purpose. Data set B1 is created with non-ASCII character and position. What's interesting is the order of execution.

If we look at arrows 1, 2 and 3 we will know that the main data step which we used to trigger the CALL EXECUTE has been executed first. Data set B1 was created at the very last.

Then by looking at arrows 4 and 5 we will see that DOSUBL indeed ran within the data step. Evidence was that data set B1 was created first at arrow 1, and main data step ended afterwards at arrow 5.

As we mentioned before there would be discussion for the data step creating data set DSN. Just as the author, the reader may be curious - what if a macro variable containing the number of observation is created in this data step instead of an actual SAS® data set variable? Let's apply this modification and see what will happen.

```
%macro CheckNonAscii(dsnin=, var=, dsnout=);

data temp;
    set test.&dsnin;

    length ascii_string $96 not_ascii_char $1;
    do x = 32 to 127;
        ascii_string=trim(left(ascii_string))||byte(x);
    end;

    not_ascii_pos = verify(&var,ascii_string);
    not_ascii_char = substr(&var, not_ascii_pos, 1);
    if not missing(not_ascii_char);
run;
```

```

Data _null_;
  If 0 then set temp(drop = _all_) nobs = totobs;
  call symputx("obs",totobs); ← 1
  stop;
Run;

data _null_;
  if &obs>0 then do;
    CALL EXECUTE("data &dsnout; set temp; run;");
  end;
run;

%mend CheckNonAscii;

```

At arrow 1, a macro variable called `&obs` was created. It served the same purpose as variable `obs` in previous version. In order to make this macro execute successfully, this macro variable `&obs` should be resolved correctly at compilation. For this reason, we expect `CALL EXECUTE` to fail and `DOSUBL` to succeed. Now let's see the result.

```

/*CALL EXECUTE call*/
data _null_;
  text='%CheckNonAscii (DSNIN=a1,VAR=a,DSNOUT=b1)';
  CALL EXECUTE(text);
run;

/*DOSUBL call*/
data _null_;
  text='%CheckNonAscii (DSNIN=a1,VAR=a,DSNOUT=b1)';
  rc=DOSUBL(text);
run;

```

In order to review the result in mode depth, `OPTION MPRINT` and `SYMBOLGEN` are turned on. Here is the SAS® log from `CALL EXECUTE` macro call.

SAS® Log:

```

MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to a1
MPRINT(CHECKNONASCII): set test.a1;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII): not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): Data _null_;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): call symputx("obs",totobs);
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
MPRINT(CHECKNONASCII): data _null_;
MPRINT(CHECKNONASCII): set dsn;
SYMBOLGEN: Macro variable OBS resolves to ← 1
SYMBOLGEN: Some characters in the above value which were subject to macro quoting have been

```

unquoted for printing.

```

MPRINT(CHECKNONASCII): if >0 then do; ← 2
SYMBOLGEN: Macro variable DSNOUT resolves to b1
MPRINT(CHECKNONASCII): CALL EXECUTE("data b1; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
NOTE: DATA statement used (Total process time):
      real time      0.01 seconds
      cpu time       0.01 seconds

```

NOTE: CALL EXECUTE generated line.

```

1  + data temp;  set test.a1; length ascii_string $96 not_ascii_char $1; do x = 32 to
127;  ascii_string=trim(left(ascii_string))||byte(x); end; not_ascii_pos =
verify(a,ascii_string); not_ascii_char = substr(a, not_ascii_pos, 1); if not missing
2  +(not_ascii_char); run;

```

NOTE: There were 1 observations read from the data set TEST.A1.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

```

      real time      0.00 seconds
      cpu time       0.00 seconds

```

```

2  +          Data_null_;  If 0 then set temp(drop = _all_) nobs = totobs;
call symputx("obs",totobs);  stop; Run;

```

NOTE: DATA statement used (Total process time):

```

      real time      0.00 seconds
      cpu time       0.00 seconds

```

NOTE: Line generated by the CALL EXECUTE routine.

```

2  +
data_null_;  set dsn;  if >0 then do;  CALL EXECUTE("data b1; set temp; run;

-
-
---

22
180
161
2  !+");  end; run;

```

ERROR 22-322: Syntax error, expecting one of the following: a name, a quoted string, ← 3
a numeric constant, a datetime constant, a missing value, INPUT, PUT.

ERROR 180-322: Statement is not valid or it is used out of proper order.

ERROR 161-185: No matching DO/SELECT statement.

NOTE: The SAS System stopped processing this step because of errors.

NOTE: DATA statement used (Total process time):

```

      real time      0.01 seconds
      cpu time       0.00 seconds

```

With this lengthy, even somehow redundant log attached, a reader can clearly see that macro variable `&obs` was not resolved at arrow 1 at the compilation stage. Moreover, a number was expected in the statement at arrow 2 to make

this statement either true or false. Now due to the fact that &obs was not resolved, this statement was apparently incorrect. Not surprisingly an error occurred at arrow 3 when the SAS® code was actually running.

Now let's examine the log from DOSUBL macro call.

SAS® Log:

```
MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to a1
MPRINT(CHECKNONASCII): set test.a1;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII): not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;
NOTE: There were 1 observations read from the data set TEST.A1.
NOTE: The data set WORK.TEMP has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time      0.00 seconds
      cpu time       0.00 seconds

MPRINT(CHECKNONASCII): Data _null_;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): call symputx("obs",totobs);
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
NOTE: DATA statement used (Total process time): ← 1
      real time      0.00 seconds
      cpu time       0.00 seconds

MPRINT(CHECKNONASCII): data _null_;
MPRINT(CHECKNONASCII): set dsn;
SYMBOLGEN: Macro variable OBS resolves to 1 ←2
MPRINT(CHECKNONASCII): if 1>0 then do; ← 3
SYMBOLGEN: Macro variable DSNOUT resolves to b1
MPRINT(CHECKNONASCII): CALL EXECUTE("data b1; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): data b1;
MPRINT(CHECKNONASCII): set temp;
MPRINT(CHECKNONASCII): run;
NOTE: There were 1 observations read from the data set WORK.DSN.
NOTE: DATA statement used (Total process time):
      real time      0.01 seconds
      cpu time       0.03 seconds

NOTE: CALL EXECUTE generated line.
NOTE: There were 1 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.B1 has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time      0.00 seconds
      cpu time       0.00 seconds

NOTE: DATA statement used (Total process time):
      real time      0.45 seconds
```

cpu time 0.07 seconds

At arrow 1 a reader can tell that the data step already executed before the next data step with macro resolution involved. So &obs resolved to the number of observation correctly at arrow 2 and the statement was complete at arrow 3.

The results from CALL EXECUTE and DOSUBL were exactly as we expected. Due to the feature of DOSUBL, it was more versatile when dealing with macro within which immediate macro value return is required. Respectively, CALL EXECUTE could not solely function correctly in such circumstance since it actually executes after the current data step ends. If the reader takes a further look into this example, he may realize that a second run of the exactly same CALL EXECUTE macro call will be successful, assuming the macro variable &obs not being restored. Yes this is correct. Here is the SAS® log:

```
MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to a1
MPRINT(CHECKNONASCII): set test.a1;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII): not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): Data _null_;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): call symputx("obs",totobs);
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
MPRINT(CHECKNONASCII): data _null_;
MPRINT(CHECKNONASCII): set dsn;
SYMBOLGEN: Macro variable OBS resolves to 1 ← 1
MPRINT(CHECKNONASCII): if 1>0 then do; ← 2
SYMBOLGEN: Macro variable DSNOUT resolves to b1
MPRINT(CHECKNONASCII): CALL EXECUTE("data b1; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
NOTE: DATA statement used (Total process time):
      real time      0.01 seconds
      cpu time       0.01 seconds
```

NOTE: CALL EXECUTE generated line.

```
1  + data temp; set test.a1; length ascii_string $96 not_ascii_char $1; do x = 32 to
127; ascii_string=trim(left(ascii_string))||byte(x); end; not_ascii_pos =
verify(a,ascii_string); not_ascii_char = substr(a, not_ascii_pos, 1); if not missing
2  +(not_ascii_char); run;
```

NOTE: There were 1 observations read from the data set TEST.A1.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

```
      real time      0.01 seconds
      cpu time       0.01 seconds
```

```
2  + Data _null_; If 0 then set temp(drop = _all_) nobs = totobs;
call symputx("obs",totobs); stop; Run;
```

NOTE: DATA statement used (Total process time):

```

real time      0.01 seconds
cpu time       0.01 seconds

```

```

2  +
data _null_;  set dsn;  if 1>0 then do;    CALL EXECUTE("data b1; set temp; run;");
end; run;

```

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

```

real time      0.01 seconds
cpu time       0.01 seconds

```

NOTE: CALL EXECUTE generated line.

```

1  + data b1; set temp; run;

```

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.B1 has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

```

real time      0.00 seconds
cpu time       0.00 seconds

```

Since the macro variable was already created at the first run, this second run of `CALL EXECUTE` managed to compile the code as desired at arrow 1 and 2. The reader may question, 'why this macro variable was still created given that the previous run errored out?' Note that the error occurred after the data step which assigned value to the macro variable. Therefore there was no issue with the macro variable creation process.

Another question is, if `CALL EXECUTE` cannot solely function correctly in such circumstance, is there a workaround available? The solution is a `%nrstr` function. By using `%nrstr` the macro call will be masked and therefore the compilation will be delayed. In this case the macro variable `&obs` can be resolved first to make the if-then statement correct. Please see the following SAS® code and log:

```

/*CALL EXECUTE call with %nrstr*/
data _null_;
  text='%CheckNonAscii (DSNIN=a1,VAR=a,DSNOUT=b1)';
  CALL EXECUTE('%nrstr(` || text || `)');
run;

```

SAS® Log:

NOTE: DATA statement used (Total process time):

```

real time      0.01 seconds
cpu time       0.00 seconds

```

NOTE: CALL EXECUTE generated line.

```

1  + %CheckNonAscii(dsnin=a1,var=a,dsnout=b1) ← 1

```

MPRINT(CHECKNONASCII): data temp;

SYMBOLGEN: Macro variable DSNIN resolves to a1

MPRINT(CHECKNONASCII): set test.a1;

MPRINT(CHECKNONASCII): length ascii_string \$96 not_ascii_char \$1;

MPRINT(CHECKNONASCII): do x = 32 to 127;

MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);

MPRINT(CHECKNONASCII): end;

MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);

MPRINT(CHECKNONASCII): not_ascii_char = substr(a, not_ascii_pos, 1);

MPRINT(CHECKNONASCII): if not missing(not_ascii_char);

MPRINT(CHECKNONASCII): run;

NOTE: There were 1 observations read from the data set TEST.A1.
NOTE: The data set WORK.TEMP has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

```
MPRINT(CHECKNONASCII): Data _null_;  
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;  
MPRINT(CHECKNONASCII): call symputx("obs",totobs);  
MPRINT(CHECKNONASCII): stop;  
MPRINT(CHECKNONASCII): Run;
```

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

```
MPRINT(CHECKNONASCII): data _null_;  
MPRINT(CHECKNONASCII): set dsn;  
SYMBOLGEN: Macro variable OBS resolves to 1  
MPRINT(CHECKNONASCII): if 1>0 then do;  
SYMBOLGEN: Macro variable DSNOUT resolves to b1  
MPRINT(CHECKNONASCII): call execute("data b1; set temp; run;");  
MPRINT(CHECKNONASCII): end;  
MPRINT(CHECKNONASCII): run;
```

```
MPRINT(CHECKNONASCII): data b1;  
MPRINT(CHECKNONASCII): set temp;  
MPRINT(CHECKNONASCII): run;
```

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

NOTE: CALL EXECUTE generated line.

```
1 + data b1; set temp; run;
```

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.B1 has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

From arrow 1 we can see that just as expected, the compilation has been delayed and therefore the code was correctly generated with &obs resolved. In this case, CALL EXECUTE with %nrstr works the same way as DOSUBL.

MULTIPLE MACRO CALLS WITH CALL EXECUTE VS. DOSUBL

Remember 2 data sets were created? Intuitively we want to test the performance of CALL EXECUTE and DOSUBL with multiple macro calls. Also, this scenario is closer to our real-world work in which a macro is rarely used to deal with single data set.

First we created a lookup table for all data sets and variables:

```
proc sort data = SAShelp.vcolumn(where=(libname="TEST"))  
    out = dsnlist(keep=libname memname name) nodupkey tagsort;  
    by memname ;  
run;
```

In this case we know the data sets are named as A1 and A2. There is a clear pattern in naming. Therefore we can use the following code to make a simple multiple macro call.

```
/*CALL EXECUTE call*/
data _null_;
  set dsnlist;
  length call $32767 ;
  call=compress('%CheckNonAscii')||
    '(DSNIN='||memname||',VAR='||name||
    ',DSNOUT='||tranwrd(memname,'A','B')||')';
  call execute(call);
run;

/*DOSUBL call*/
data _null_;
  set dsnlist;
  length call $32767 ;
  call=compress('%CheckNonAscii')||
    '(DSNIN="||memname||',VAR="||name||',DSNIN='||
    tranwrd(memname,'A','B')||')';
  rc=DOSUBL(call);
run;
```

In this example we wanted the output data sets to be named as B1 and B2. Note that macro %CheckNonAscii was changed back to its original version. We expect both data steps to run successfully as there is only 1 level of compilation for macro %CheckNonAscii and no macro variable resolution is needed. It turned out that the results were as expected. The logs are omitted here as they are too lengthy. Please refer to SAS® log 1 and 2 for complete log in appendix section at the end of this paper.

In real-world scenario, however, most data set names do not follow any pattern. Hence, to output meaningful data set names, the code must be made more dynamic. For this reason, another macro call needs to be created based on this lookup table to trigger the %CheckNonAscii macro:

```
%macro call(libname=, dsn=);

proc contents data = &libname..&dsn out = list(where=(type=2) keep = name type)
  noprint ; run;

data call;
  set list;
  length call $32767 ;
  call='%CheckNonAscii'||
    "(DSNIN=&dsn,VAR="||strip(name)||',
    DSNOUT=out_'||"&dsn"||'_'||strip(name)||')'; < 1
  keep call;
run;

data _null_ ;
  set call;
  CALL EXECUTE(call);
run;

%mend call;
```

In this case, the author wanted to name the output data sets based on the name of input data sets at arrow 1. &DSN is the parameter of the macro %CALL. Parameters of macro %CheckNonAscii will be resolved from &DSN. With previous discussions the reader might intuitively think CALL EXECUTE is expected to fail as macro variable resolution is needed within the call. However the truth is that both CALL EXECUTE and DOSUBL will succeed. It's not surprising for DOSUBL, but why for CALL EXECUTE? If a reader is familiar with CALL EXECUTE, he will know that if the argument of CALL EXECUTE resolves to a macro invocation, the macro executes immediately and DATA step execution pauses while the macro executes. This means &DSN has already been resolved when compilation reaches arrow 1. The following codes are used to trigger this nested macro call, using CALL EXECUTE and DOSUBL respectively:

```
/*CALL EXECUTE*/
data _null_;
    set dsncat;
    length text $32767 ;
    text = '%call'||"(libname='||libname||',dsn='||memname||')';";
    CALL EXECUTE(text);
run;

/*DOSUBL*/
data _null_;
    set dsncat;
    length text $32767 ;
    text = '%call'||"(libname='||libname||',dsn='||memname||')';";
    rc=DOSUBL(text);
run;
```

Both codes executed without any issue. This result is with no surprise, as we discussed before. The complete logs can be found as SAS® log 3 and 4 in the appendix section.

CONCLUSION

In this paper several scenarios are presented to illustrate the difference between CALL EXECUTE and DOSUBL. By reading this paper readers are expected to understand the execution timing difference of CALL EXECUTE and DOSUBL.

REFERENCES

Rick Langston (2013), "Submitting SAS Code On The Side".

<https://support.SAS.com/resources/papers/proceedings13/032-2013.pdf>

SAS® Online Documents

ACKNOWLEDGEMENT

The author would like to thank Ken Borowiak, Ajay Gupta, Matthew Lesko, Kyle Thompson, Thomas Souers and John Cohen for their review and precious comments. Also he would like to thank his wife Xinming Zhang for her encouragement to submit this paper.

SAS® and all other SAS® Institute Inc. product or service names are registered trademarks or trademarks of SAS® Institute Inc. in the USA and other countries. ® indicates USA registration.

DISCLAIMER

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of PPD.

Contact information

Comments, questions and additions are welcome. Contact the author at:

Jueru Fan

PPD

3900 Paramount Parkway

Morrisville, NC 27560

Phone: (919)-456-6450

Email: Jueru.Fan2@ppdi.com

APPENDIX

SAS log 1:

```
MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to A1
MPRINT(CHECKNONASCII): set test.A1;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII): if not_ascii_pos>=1 then not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): Data dsn;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): Obs = totobs;
MPRINT(CHECKNONASCII): output;
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
MPRINT(CHECKNONASCII): data _null_;
MPRINT(CHECKNONASCII): set dsn;
MPRINT(CHECKNONASCII): if obs>0 then do;
SYMBOLGEN: Macro variable DSNOUT resolves to B1
MPRINT(CHECKNONASCII): call execute("data B1; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to A2
MPRINT(CHECKNONASCII): set test.A2;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII): if not_ascii_pos>=1 then not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): Data dsn;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): Obs = totobs;
MPRINT(CHECKNONASCII): output;
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
MPRINT(CHECKNONASCII): data _null_;
MPRINT(CHECKNONASCII): set dsn;
MPRINT(CHECKNONASCII): if obs>0 then do;
SYMBOLGEN: Macro variable DSNOUT resolves to B2
MPRINT(CHECKNONASCII): call execute("data B2; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
NOTE: There were 2 observations read from the data set WORK.DSNLIST.
NOTE: DATA statement used (Total process time):
      real time      0.04 seconds
      cpu time       0.04 seconds
```

NOTE: CALL EXECUTE generated line.

```
1 + data temp; set test.A1; length ascii_string $96 not_ascii_char $1; do x = 32 to 127; ascii_string=trim(left(ascii_string)||byte(x)); end; not_ascii_pos = verify(a,ascii_string); if not_ascii_pos>=1 then not_ascii_char = substr(a, not_ascii_pos, 1 2 +); if not missing(not_ascii_char); run;
```

NOTE: There were 1 observations read from the data set TEST.A1.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

```
2 + Data dsn; If 0 then set temp(drop = _all_) nobs = totobs; Obs = totobs; output; stop; Run;
```

NOTE: The data set WORK.DSN has 1 observations and 1 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

```
2 + data _null_; set dsn; if obs>0 then do; call execute("data B1; set temp; run;"); end 3 +; run;
```

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

NOTE: CALL EXECUTE generated line.

```
1 + data B1; set temp; run;
```

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.B1 has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

```
4 + data temp; set test.A2; length ascii_string $96 not_ascii_char $1; do x = 32 to 127; ascii_string=trim(left(ascii_string)||byte(x)); end; not_ascii_pos = verify(a,ascii_string); if not_ascii_pos>=1 then not_ascii_char = substr(a, not_ascii_pos, 1 5 +); if not missing(not_ascii_char); run;
```

NOTE: There were 1 observations read from the data set TEST.A2.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

```
5 + Data dsn; If 0 then set temp(drop = _all_) nobs = totobs; Obs = totobs; output; stop; Run;
```

NOTE: The data set WORK.DSN has 1 observations and 1 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

```

5  +
data _null_ ; set dsn; if obs>0 then do; call execute("data B2; set temp; run;");
end
6  +; run;

```

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

```

real time      0.01 seconds
cpu time       0.01 seconds

```

NOTE: CALL EXECUTE generated line.

```

1  + data B2; set temp; run;

```

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.B2 has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

```

real time      0.00 seconds
cpu time       0.00 seconds

```

SAS log 2:

```

MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to A1
MPRINT(CHECKNONASCII): set test.A1;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII): if not_ascii_pos>=1 then not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;

```

NOTE: There were 1 observations read from the data set TEST.A1.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

```

real time      0.01 seconds
cpu time       0.00 seconds

```

```

MPRINT(CHECKNONASCII): Data dsn;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): Obs = totobs;
MPRINT(CHECKNONASCII): output;
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;

```

NOTE: The data set WORK.DSN has 1 observations and 1 variables.

NOTE: DATA statement used (Total process time):

```

real time      0.00 seconds
cpu time       0.00 seconds

```

```

MPRINT(CHECKNONASCII): data _null_ ;
MPRINT(CHECKNONASCII): set dsn;
MPRINT(CHECKNONASCII): if obs>0 then do;
SYMBOLGEN: Macro variable DSNOUT resolves to B1
MPRINT(CHECKNONASCII): call execute("data B1; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): data B1;

```

```
MPRINT(CHECKNONASCII): set temp;
MPRINT(CHECKNONASCII): run;
NOTE: There were 1 observations read from the data set WORK.DSN.
NOTE: DATA statement used (Total process time):
```

```
    real time      0.01 seconds
    cpu time       0.01 seconds
```

```
NOTE: CALL EXECUTE generated line.
NOTE: There were 1 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.B1 has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):
```

```
    real time      0.00 seconds
    cpu time       0.00 seconds
```

```
MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to A2
MPRINT(CHECKNONASCII): set test.A2;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII):  ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII):  not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII):  if not_ascii_pos>=1 then not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII):  if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;
NOTE: There were 1 observations read from the data set TEST.A2.
NOTE: The data set WORK.TEMP has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):
```

```
    real time      0.01 seconds
    cpu time       0.00 seconds
```

```
MPRINT(CHECKNONASCII): Data dsn;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): Obs = totobs;
MPRINT(CHECKNONASCII): output;
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
NOTE: The data set WORK.DSN has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
```

```
    real time      0.01 seconds
    cpu time       0.01 seconds
```

```
MPRINT(CHECKNONASCII): data _null_;
MPRINT(CHECKNONASCII): set dsn;
MPRINT(CHECKNONASCII): if obs>0 then do;
SYMBOLGEN: Macro variable DSNOUT resolves to B2
MPRINT(CHECKNONASCII): call execute("data B2; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): data B2;
MPRINT(CHECKNONASCII): set temp;
MPRINT(CHECKNONASCII): run;
NOTE: There were 1 observations read from the data set WORK.DSN.
NOTE: DATA statement used (Total process time):
```

```
    real time      0.00 seconds
    cpu time       0.00 seconds
```

```
NOTE: CALL EXECUTE generated line.
```

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.B2 has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

NOTE: There were 2 observations read from the data set WORK.DSNLIST.

NOTE: DATA statement used (Total process time):

real time	1.21 seconds
cpu time	0.14 seconds

SAS log 3:

SYMBOLGEN: Macro variable LIBNAME resolves to TEST

SYMBOLGEN: Macro variable DSN resolves to A1

MPRINT(CALL): proc contents data = TEST.A1 out = list(where=(type=2) keep = name type) noprint

;

MPRINT(CALL): run;

MPRINT(CALL): data call;

MPRINT(CALL): set list;

MPRINT(CALL): length call \$32767 ;

SYMBOLGEN: Macro variable DSN resolves to A1

SYMBOLGEN: Macro variable DSN resolves to A1

MPRINT(CALL): call = '%CheckNonAscii' ||'(DSNIN=A1,VAR="||strip(name)||",

DSNOUT=out_'||"A1"||'_'||strip(name)||')';

MPRINT(CALL): keep call;

MPRINT(CALL): run;

MPRINT(CALL): data _null_ ;

MPRINT(CALL): set call;

MPRINT(CALL): call execute(call);

MPRINT(CALL): run;

SYMBOLGEN: Macro variable LIBNAME resolves to TEST

SYMBOLGEN: Macro variable DSN resolves to A2

MPRINT(CALL): proc contents data = TEST.A2 out = list(where=(type=2) keep = name type) noprint

;

MPRINT(CALL): run;

MPRINT(CALL): data call;

MPRINT(CALL): set list;

MPRINT(CALL): length call \$32767 ;

SYMBOLGEN: Macro variable DSN resolves to A2

SYMBOLGEN: Macro variable DSN resolves to A2

MPRINT(CALL): call = '%CheckNonAscii' ||'(DSNIN=A2,VAR="||strip(name)||",

DSNOUT=out_'||"A2"||'_'||strip(name)||')';

MPRINT(CALL): keep call;

MPRINT(CALL): run;

MPRINT(CALL): data _null_ ;

MPRINT(CALL): set call;

MPRINT(CALL): call execute(call);

MPRINT(CALL): run;

NOTE: There were 2 observations read from the data set WORK.DSNLIST.

NOTE: DATA statement used (Total process time):

real time	0.03 seconds
cpu time	0.03 seconds

NOTE: CALL EXECUTE generated line.

1 + proc contents data = TEST.A1 out = list(where=(type=2) keep = name type) noprint ; run;

NOTE: The data set WORK.LIST has 1 observations and 2 variables.

NOTE: PROCEDURE CONTENTS used (Total process time):

real time	0.01 seconds
cpu time	0.00 seconds

```
1 +  
data call; set list; length call $32767 ; call =  
'%CheckNonAscii' || "(DSNIN=A1,VAR=||strip(name)||", DSNOUT=out_'||"A1"||'_||strip(name)||')";  
2 + keep call; run;
```

NOTE: There were 1 observations read from the data set WORK.LIST.

NOTE: The data set WORK.CALL has 1 observations and 1 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.00 seconds

```
2 + data _null_ ; set call; call execute(call); run;
```

```
MPRINT(CHECKNONASCII): data temp;  
SYMBOLGEN: Macro variable DSNIN resolves to A1  
MPRINT(CHECKNONASCII): set test.A1;  
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;  
MPRINT(CHECKNONASCII): do x = 32 to 127;  
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);  
MPRINT(CHECKNONASCII): end;  
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);  
MPRINT(CHECKNONASCII): not_ascii_char = substr(a, not_ascii_pos, 1);  
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);  
MPRINT(CHECKNONASCII): run;  
MPRINT(CHECKNONASCII): Data _null_ ;  
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;  
MPRINT(CHECKNONASCII): call symputx("obs",totobs);  
MPRINT(CHECKNONASCII): stop;  
MPRINT(CHECKNONASCII): Run;  
MPRINT(CHECKNONASCII): data _null_ ;  
MPRINT(CHECKNONASCII): set dsn;  
SYMBOLGEN: Macro variable OBS resolves to 1  
MPRINT(CHECKNONASCII): if 1>0 then do;  
SYMBOLGEN: Macro variable DSNOUT resolves to out_A1_a  
MPRINT(CHECKNONASCII): call execute("data out_A1_a; set temp; run;");  
MPRINT(CHECKNONASCII): end;  
MPRINT(CHECKNONASCII): run;
```

NOTE: There were 1 observations read from the data set WORK.CALL.

NOTE: DATA statement used (Total process time):

real time	0.03 seconds
cpu time	0.03 seconds

NOTE: CALL EXECUTE generated line.

```
2 + ;  
1 + data temp; set test.A1; length ascii_string $96 not_ascii_char $1; do x = 32 to  
127; ascii_string=trim(left(ascii_string))||byte(x); end; not_ascii_pos =  
verify(a,ascii_string); not_ascii_char = substr(a, not_ascii_pos, 1); if not missing  
2 +(not_ascii_char); run;
```

NOTE: There were 1 observations read from the data set TEST.A1.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

```
2 + Data _null_ ; If 0 then set temp(drop = _all_) nobs = totobs;
call symputx("obs",totobs); stop; Run;
```

NOTE: DATA statement used (Total process time):

```
real time    0.01 seconds
cpu time     0.01 seconds
```

```
2 +
data _null_ ; set dsn; if 1>0 then do; call execute("data out_A1_a; set temp;
run;"); end; run;
```

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

```
real time    0.00 seconds
cpu time     0.00 seconds
```

NOTE: CALL EXECUTE generated line.

```
2 +
;
1 + data out_A1_a; set temp; run;
```

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.OUT_A1_A has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

```
real time    0.00 seconds
cpu time     0.00 seconds
```

```
3 + proc contents data = TEST.A2 out = list(where=(type=2) keep = name type) noprint ; run;
```

NOTE: The data set WORK.LIST has 1 observations and 2 variables.

NOTE: PROCEDURE CONTENTS used (Total process time):

```
real time    0.00 seconds
cpu time     0.00 seconds
```

```
3 +
data call; set list; length call $32767 ; call =
'%CheckNonAscii' || "(DSNIN=A2,VAR=||strip(name)||", DSNOUT=out_'||A2"||_'||strip(name)||');";
4 + keep call; run;
```

NOTE: There were 1 observations read from the data set WORK.LIST.

NOTE: The data set WORK.CALL has 1 observations and 1 variables.

NOTE: DATA statement used (Total process time):

```
real time    0.00 seconds
cpu time     0.00 seconds
```

```
4 + data _null_ ; set call; call execute(call); run;
```

MPRINT(CHECKNONASCII): data temp;

SYMBOLGEN: Macro variable DSNIN resolves to A2

MPRINT(CHECKNONASCII): set test.A2;

MPRINT(CHECKNONASCII): length ascii_string \$96 not_ascii_char \$1;

MPRINT(CHECKNONASCII): do x = 32 to 127;

MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);

MPRINT(CHECKNONASCII): end;

MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);

MPRINT(CHECKNONASCII): not_ascii_char = substr(a, not_ascii_pos, 1);

MPRINT(CHECKNONASCII): if not missing(not_ascii_char);

```

MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): Data _null_;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): call symputx("obs",totobs);
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
MPRINT(CHECKNONASCII): data _null_;
MPRINT(CHECKNONASCII): set dsn;
SYMBOLGEN: Macro variable OBS resolves to 1
MPRINT(CHECKNONASCII): if 1>0 then do;
SYMBOLGEN: Macro variable DSNOUT resolves to out_A2_a
MPRINT(CHECKNONASCII): call execute("data out_A2_a; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
NOTE: There were 1 observations read from the data set WORK.CALL.
NOTE: DATA statement used (Total process time):
    real time      0.03 seconds
    cpu time       0.03 seconds

```

NOTE: CALL EXECUTE generated line.

```

4  +                               ;
1  + data temp;  set test.A2; length ascii_string $96 not_ascii_char $1; do x = 32 to
127;  ascii_string=trim(left(ascii_string))||byte(x); end; not_ascii_pos =
verify(a,ascii_string); not_ascii_char = substr(a, not_ascii_pos, 1); if not missing
2  +(not_ascii_char); run;

```

NOTE: There were 1 observations read from the data set TEST.A2.
NOTE: The data set WORK.TEMP has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):

```

    real time      0.01 seconds
    cpu time       0.01 seconds

```

```

2  +          Data _null_;  If 0 then set temp(drop = _all_) nobs = totobs;
    call symputx("obs",totobs);  stop; Run;

```

NOTE: DATA statement used (Total process time):

```

    real time      0.00 seconds
    cpu time       0.00 seconds

```

```

2  +
data _null_;  set dsn;  if 1>0 then do;      call execute("data out_A2_a; set temp;
run;");  end; run;

```

NOTE: There were 1 observations read from the data set WORK.DSN.
NOTE: DATA statement used (Total process time):

```

    real time      0.00 seconds
    cpu time       0.00 seconds

```

NOTE: CALL EXECUTE generated line.

```

2  +
;
1  + data out_A2_a; set temp; run;

```

NOTE: There were 1 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.OUT_A2_A has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):

```

    real time      0.00 seconds

```

cpu time 0.00 seconds

SAS log 4:

```
SYMBOLGEN: Macro variable LIBNAME resolves to TEST
SYMBOLGEN: Macro variable DSN resolves to A1
MPRINT(CALL): proc contents data = TEST.A1 out = list(where=(type=2) keep = name type) noprint
;
MPRINT(CALL): run;
NOTE: The data set WORK.LIST has 1 observations and 2 variables.
NOTE: PROCEDURE CONTENTS used (Total process time):
      real time        0.00 seconds
      cpu time        0.00 seconds

MPRINT(CALL): data call;
MPRINT(CALL): set list;
MPRINT(CALL): length call $32767 ;
SYMBOLGEN: Macro variable DSN resolves to A1
SYMBOLGEN: Macro variable DSN resolves to A1
MPRINT(CALL): call = '%CheckNonAscii' ||'(DSNIN=A1,VAR="||strip(name)||",
DSNOUT=out_' ||"A1"||"_' ||strip(name)||')';
MPRINT(CALL): keep call;
MPRINT(CALL): run;
NOTE: There were 1 observations read from the data set WORK.LIST.
NOTE: The data set WORK.CALL has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time        0.01 seconds
      cpu time        0.00 seconds

MPRINT(CALL): data _null_ ;
MPRINT(CALL): set call;
MPRINT(CALL): call execute(call);
MPRINT(CALL): run;
MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to A1
MPRINT(CHECKNONASCII): set test.A1;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII): not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): Data _null_ ;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): call symputx("obs",totobs);
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
MPRINT(CHECKNONASCII): data _null_ ;
MPRINT(CHECKNONASCII): set dsn;
SYMBOLGEN: Macro variable OBS resolves to 1
MPRINT(CHECKNONASCII): if 1>0 then do;
SYMBOLGEN: Macro variable DSNOUT resolves to out_A1_a
MPRINT(CHECKNONASCII): call execute("data out_A1_a; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
MPRINT(CALL): ;
```

NOTE: There were 1 observations read from the data set WORK.CALL.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.00 seconds

NOTE: CALL EXECUTE generated line.

MPRINT(CALL): 96 not_ascii_char \$1

NOTE: There were 1 observations read from the data set TEST.A1.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

MPRINT(CALL): data out_A1_a;

MPRINT(CALL): set temp;

MPRINT(CALL): run;

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

NOTE: CALL EXECUTE generated line.

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.OUT_A1_A has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

SYMBOLGEN: Macro variable LIBNAME resolves to TEST

SYMBOLGEN: Macro variable DSN resolves to A2

MPRINT(CALL): proc contents data = TEST.A2 out = list(where=(type=2) keep = name type) noprint
;

MPRINT(CALL): run;

NOTE: The data set WORK.LIST has 1 observations and 2 variables.

NOTE: PROCEDURE CONTENTS used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

MPRINT(CALL): data call;

MPRINT(CALL): set list;

MPRINT(CALL): length call \$32767 ;

SYMBOLGEN: Macro variable DSN resolves to A2

SYMBOLGEN: Macro variable DSN resolves to A2

MPRINT(CALL): call = '%CheckNonAscii' || '{(DSNIN=A2,VAR="||strip(name)||",
DSNOUT=out_' || "A2" || " _ ||strip(name)||"}';

MPRINT(CALL): keep call;

MPRINT(CALL): run;

NOTE: There were 1 observations read from the data set WORK.LIST.

NOTE: The data set WORK.CALL has 1 observations and 1 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

MPRINT(CALL): data _null_;

```

MPRINT(CALL): set call;
MPRINT(CALL): call execute(call);
MPRINT(CALL): run;
MPRINT(CHECKNONASCII): data temp;
SYMBOLGEN: Macro variable DSNIN resolves to A2
MPRINT(CHECKNONASCII): set test.A2;
MPRINT(CHECKNONASCII): length ascii_string $96 not_ascii_char $1;
MPRINT(CHECKNONASCII): do x = 32 to 127;
MPRINT(CHECKNONASCII): ascii_string=trim(left(ascii_string))||byte(x);
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): not_ascii_pos = verify(a,ascii_string);
MPRINT(CHECKNONASCII): not_ascii_char = substr(a, not_ascii_pos, 1);
MPRINT(CHECKNONASCII): if not missing(not_ascii_char);
MPRINT(CHECKNONASCII): run;
MPRINT(CHECKNONASCII): Data _null_;
MPRINT(CHECKNONASCII): If 0 then set temp(drop = _all_) nobs = totobs;
MPRINT(CHECKNONASCII): call symputx("obs",totobs);
MPRINT(CHECKNONASCII): stop;
MPRINT(CHECKNONASCII): Run;
MPRINT(CHECKNONASCII): data _null_;
MPRINT(CHECKNONASCII): set dsn;
SYMBOLGEN: Macro variable OBS resolves to 1
MPRINT(CHECKNONASCII): if 1>0 then do;
SYMBOLGEN: Macro variable DSNOUT resolves to out_A2_a
MPRINT(CHECKNONASCII): call execute("data out_A2_a; set temp; run;");
MPRINT(CHECKNONASCII): end;
MPRINT(CHECKNONASCII): run;
MPRINT(CALL): ;

```

NOTE: There were 1 observations read from the data set WORK.CALL.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

NOTE: CALL EXECUTE generated line.

```
MPRINT(CALL): 96 not_ascii_char $1
```

NOTE: There were 1 observations read from the data set TEST.A2.

NOTE: The data set WORK.TEMP has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

```
MPRINT(CALL): data out_A2_a;
```

```
MPRINT(CALL): set temp;
```

```
MPRINT(CALL): run;
```

NOTE: There were 1 observations read from the data set WORK.DSN.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

NOTE: CALL EXECUTE generated line.

NOTE: There were 1 observations read from the data set WORK.TEMP.

NOTE: The data set WORK.OUT_A2_A has 1 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	0.01 seconds
-----------	--------------

cpu time 0.01 seconds

NOTE: There were 2 observations read from the data set WORK.DSNLIST.

NOTE: DATA statement used (Total process time):

real time 1.29 seconds

cpu time 0.25 seconds