

Chrestos – Die grüne CRO

When the answer to your problems is just one CALL(EXECUTE) away

Main Author: Luis Enrique Osorio Puentes
luis.osorio@chrestos.de

Co-Author: Dr. Lara Suckut
lara.suckut@chrestos.de

Phuse EU Connect 13-Nov-2024 Strasbourg



What is CALL EXECUTE?

- CALL EXECUTE is a SAS function, it:
 - generates code** for subsequent **execution**
 - is used as **part of a data step**
 - combines** dataset **variable** values with traditional **SAS code** or **PROC SQL**
- CALL EXECUTE vs MACRO variables:
 - the use of dataset variable values **gives more control**
 - dataset values** can be used **to modify other data sets**

```
data callexe;
    ...
    call execute("..." || var || "...")
    ...
run;
```

Framework: how to invoke CALL EXECUTE?

● CALL EXECUTE(<argument>):

- the <argument> can consist of:
 - a **character string in quotation marks** to be executed as SAS code
 - a **dataset variable** without quotation marks

```
data callexe;
    ...
    call execute("..." || var || "...")
    ...
run;
```

● Note:

- The SAS code can be any valid SAS code (**data steps, macro invocations and PROC SQL**)
- ('...') lead to **delayed execution**
- ("...") lead to **immediate execution**

Power-up tools for CALL EXECUTE

- The possibility to use variables **together with metadata and dictionaries** increases its power

“A **dictionary** is a read-only view **table** that **contains information** about **libraries, datasets, macros**, and external files that are in use or available **in the current SAS sessions**.”

- **Dictionaries** allow for automatic generation of dynamic code to be executed
- **Metadata** allows to read variable specifications, e.g. formats or derivation rules of variables

Example 1: Implementation example with libraries

- Creation of a library "codelist" → information of all members in dictionary "MEMBERS"

	libname Library Name	memname Member Name	memtype Member Type	dbms_memtype DBMS Memb Type	engine Engine Name	index Indexes
139		SV	DATA		V9	no
140		TA	DATA		V9	no
141		TE	DATA		V9	no
142		TI	DATA		V9	no
143		TR	DATA		V9	no
144		TS	DATA		V9	no
145		TU	DATA		V9	no
146		TV	DATA		V9	no
147		VS	DATA		V9	no
148		XB	DATA		V9	no
149		XC	DATA		V9	no
150	CODELIST	ACCLASS	DATA		V9	no
151	CODELIST	ACN	DATA		V9	no
152	CODELIST	ACSPCAT	DATA		V9	no
153	CODELIST	ADTNQRS	DATA		V9	no
154	CODELIST	AESEV	DATA		V9	no
155	CODELIST	AGESPAN	DATA		V9	no
156	CODELIST	AGEU	DATA		V9	no
157	CODELIST	AJCCGRD	DATA		V9	no
158	CODELIST	ARMNULRS	DATA		V9	no

```

PROC SQL;

  create table codelist as select *
  from dictionary.members
  where libname="codelist";

END;

DATA _NULL_;
  set codelist;

  CALL EXECUTE("proc format library=codelist
  cntlin=" || memname || "run;" );

RUN;

```

Example 2: Compute mean of multiple parameters

Advantages of using CALL EXECUTE:

- Selection based on wildcards for unknown number of variables possible without knowing the origin dataset
- Calculation for any number of populations or other parameters, no need to repeat code

Common pitfalls (interactive examples)

```
data callexel;  
    var1="testing";  
call execute("data test;");  
...  
call execute("run;");  
run;
```

The log states:

... Code lines ...

NOTE: The data set WORK.CALLEXE1 has 1 observations and 1 variables.

NOTE: CALL EXECUTE generated line.

+ data test;

...

+ run;

Arguments that exceed 256 characters should be split to avoid warnings

```
data callexel;
  var1="testing";
  call execute("data test;");
  call execute('put "Testing "');
  call execute('"different use cases."');
  call execute(';');
  call execute("run;");

run;
```

The log states:

```
... Code lines ...
+ data test;
+ put "Testing "
+ "different use cases."
+ ;
+ run;
  Testing different use cases.
```

Don't confuse variables generated by CALL EXECUTE with variables in the DATA STEP

```
data callexel;
  var1="testing";
  call execute("data test;");
  call execute('var1="start";');
  call execute('put "Single quotation
               marks :'|'|var1'|'|";');
  call execute("run;");
run;
```

The log states:

```
... Code lines ...
+ data test;
+ var1="start";
+ put "Single quotation marks :testing";
+ run;
Single quotation marks :testing
```

SAS code WHITOUT macro triggers is processed in the order of entry

```
data callexel;  
  var1="testing";  
  call execute("data test;");  
  call execute('put "Single quotation  
    marks :'||var1||'";');  
  call execute("put 'Double quotation  
    marks :'||var1||'";");  
  call execute("run;");  
run;
```

The log states:

```
... Code lines ...  
+ data test;  
+ put "Single quotation marks :testing";  
+ put 'Double quotation marks :testing';  
+ run;  
Single quotation marks :testing  
Double quotation marks :testing
```

MACRO triggers

(`'...'`) execution during program execution

(`"..."`) execution during data step construction

```
data callexel;  
  
var1="testing";  
  
call execute("data test;");
```

call execute('%put Macro call without macro quoting in single quotation marks for '||var1||';');

call execute("%put Macro call without macro quoting in double quotation marks for "||var1||";");

```
call execute("run;");
```

```
run;
```

The log states:

... Code lines ...

Macro call without macro quoting in double quotation marks for "||var1||"

... Code lines ...

Macro call without macro quoting in single quotation marks for **testing**

+ data test;

...

MACRO triggers } %NRSTR allows to execute macros in the usual order of entry

```
data callexel;
```

```
var1="testing";
```

```
call execute("data test;");
```

```
call execute('%put Macro call without  
macro quoting in single quotation  
marks for '||var1||';');
```

```
call execute("%nrstr(%put Macro call  
without macro quoting in double  
quotation marks for "||var1||");");
```

```
call execute("run;");
```

```
run;
```

The log states:

... Code lines ...

Masked macro call with macro quoting in
double quotation marks for **testing**

Macro call without macro quoting in single
quotation marks for **testing**

+ data test;

...

Bibliography

- 1) PHUSE 2009, Paper CC01, Can I CALL EXECUTE here?, Vinod Kaippillil Mukundaprasad, Roche, Welwyn, UK
- 2) Paper 027-30 CALL EXECUTE: A Powerful Data Management Tool, Denis Michel, Johnson & Johnson Pharmaceutical Research and Development, L.L.C
- 3) SAS documentation for CALL EXECUTE,
https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/mcrolref/n1q1527d51eivsn1ob5hnz0yd1hx.htm
- 4) SAS documentation for Dictionary,
<https://documentation.sas.com/doc/en/lrcon/9.4/p13650yz8q04csn1m2xojl26ljsc.htm>



Thank you for your
attention!

