Das Webinar@Lunchtime beginnt in Kürze.

Verwenden Sie die Widgets am Bildschirmrand für folgende Aktionen:

Stellen Sie Fragen
während des Webinars mit dem Q&A widget.

Geben Sie uns Feedback
mit dem Survey widget.

Technische Hilfe
ist über das Help widget verfügbar.

Minimieren oder Maximieren
eines Widgets

Größe eines Widgets ändern
durch Ziehen an der unteren rechten Ecke.

Webinar@Lunchtime

Macro News für SAS Dinosaurier

THE POWER TO KNOW.

Inhaltsübersicht

1. Makrovariablen
2. Makro (System-)Optionen
3. Makros und Makrofunktionen
4. Sonstiges

1. Makrovariablen

%put Anweisung

Call Symputx: Lokale oder globale Makrovariable

Proc SQL und into:

Automatische Makrovariablen

Schreibgeschützte Makrovariablen erstellen

Neue Schreibweise möglich für %put

%put: Makrovariablenwert im Log ausgeben

Der Name der Makrovariablen erscheint im Log ohne ihn separat angeben zu müssen.

```
%put   &=macvar ;
```

```
34          %let umsatz=1234;
35
36          %put &=umsatz;
UMSATZ=1234
37
38          %put &umsatz;
1234
```

Call Symput**X**

Makrovariablen im Data Step

Makrovariablen zur Laufzeit im Data Step erstellen

CALL SYMPUTX()

Im Gegensatz zur Call symput kann call symputX als dritten Parameter mitgeben, ob die Makrovariable lokal ‘L‘ oder global ‘G‘ erstellt werden soll.

Beispiel:

call symputx (“Makrovar“, “statischerWert“, **‘L‘**) ;

Achtung Falle: Wenn es keine lokale Symboltabelle gibt, dann öffnet call symputx auch keine, sondern schreibt in die bestehende Symboltabelle (global), auch wenn ‘L‘ angegeben wurde!

Call Symputx: Achtung Falle!

Syntaxbeispiele

```
data _null_ ;
  set sashelp.class;
  where name='Alfred';
  call symputx('mvname', name, 'L');
  run;
  %put _user_ ;
```

Wenn z. B. der EG gerade gestartet wurde, dann ist nur die globale Symboltabelle vorhanden; die Option ‚L' hat daher keine Wirkung!

In diesem Makro wird eine lokale Symboltabelle für die Makrovariable **datei=** erstellt, daher hat die Option 'L' jetzt eine Wirkung;

```
%macro test(datei=sashelp.class);
  data test_neu ;
  set &datei;
  where name='Carol';
  call symputx('mvname2',name,'L');
  run;
  %put _user_ ;
%mend test;
 %test
```

Proc SQL

Viele Makrovariablen mit jeweils einem Wert erstellen ohne die Anzahl zu wissen

Syntax: Select Spalte into :mv1-

```
Proc SQL;
Select name into :Mv1 –
from sashelp.class;
```

Die Datei hat 19 Zeilen, also werden die Makrovariablen mv1, mv2, ….mv19 erstellt.

Die obere Grenze kann offen bleiben, es muss jedoch mit einem Zahlenwert (hier: MV1) begonnen werden.

Auswahl neuer Automatischer Makro Variablen der letzten Jahre

Makrovariable	Wert
sysnobs	Anzahl **eingelesener** Zeilen
sqlobs	Anzahl **ausgegebener** Zeilen (bei Proc SQL)
sysprocessmode	Ausgabe des Modus der **aktuellen Sitzung** (SAS Batch Mode, SAS DMS Session) oder **Servertyp Namens**, z. B. Metadata Server, SAS OLAP Server, SAS Stored Process Server, SAS Workspace Server
sysdatastepphase	Ausgabe der **aktuellen Phase des Data Steps**: z. B. Compilation, Execution
sysprinttolist	Beinhaltete den **Pfad der LIST Datei**, die in der Proc Printto gesetzt wurde.
sysprinttolog	Beinhaltet den **Pfad der Log Datei**, die in der Proc Printto gesetzt wurde.
syshostname	Gibt den Host **Name** eines **Computers** aus
syswarningtext	Beinhaltet den Text der letzten **Warnungs-Meldung**, die im Log ausgegeben wurde
Syserrortext	Beinhaltet den Text der letzten **ERROR-Meldung**, die im Log ausgegeben wurde

Die automatische Makrovariable: &sqlobs

```
libname test 'c:\workshop\winsas\pg1'; *Hier liegen SAS Dateien;
%macro demo_sqlobs;
proc sql ;
    select "&libname.." || memname
       into :dsn1-
        from sashelp.vtable
           where libname="%upcase(&libname)";
   quit;
%put _user_ ;

   %do i=1 %to &sqlobs;    /*Die Schleife läuft so oft durch,
                             wie Zeilen im vorherigen Proc SQL
                           Schritt ermittelt wurden: &Sqlobs; */
      %put &&dsn&i
   %end;
%mend demo_sqlobs;
```

```
%demo_sqlobs
```

Beispiel für: Sysdatastepphase und Sysprocessmode

```
data test;
   call symputx('a',"&sysdatastepphase");
   set sashelp.class;
   welche_phase=symget("sysdatastepphase");
   call symputx('mvsysprocessmode',"&sysprocessmode");
run;
  %put &=a   &=mvsysprocessmode;
```

	Name	Sex	Age	Height	Weight	welche_phase
1	Alfred	M	14	69	112.5	EXECUTION
2	Alice	F	13	56.5	84	EXECUTION
3	Barbara	F	13	65.3	98	EXECUTION

```
41          %put &=a   &=mvsysprocessmode;
A=COMPILATION  MVSYSPROCESSMODE=SAS Workspace Server
```

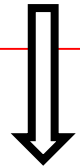
Makrovariablen schreibgeschützt erstellen

```
%put _writable_ ;
%put _readonly_ ;
```

Neue Option bei %global und %local: readonly

Normalerweise kann jede Makrovariable geändert bzw. gelöscht werden. Wenn Sie das nicht wollen, dass dies die Anwender machen, können Sie es mit der Option readonly verhindern.

```
%global/readonly mv_nichtloeschbar=1;
%global mv_kannManLoeschen;
%let mv_nichtloeschbar=2;
%symdel mv_nichtloeschbar;
```



Meldung im Log:

```
ERROR: The variable MV_NICHTLOESCHBAR was declared READONLY and cannot be modified or re-declared.
37          %symdel mv_nichtloeschbar;
ERROR: The variable MV_NICHTLOESCHBAR was declared READONLY and cannot be deleted.
```

Makro: (System-)Optionen

In (…) Verarbeitung: minoperator

%macro und secure Option

Im Makrokontext mit IN (…) programmieren

Damit die IN(…) Abfrage funktionieren, muss sie “eingeschaltet“ werden:

1. Bei der Makroerstellung

Bsp:

```
%macro test /minoperator ;
   irgendeine Syntax;
%if country in AU CA DE %then…… ;
%mend test;
```

Achtung: **Keine Klammer**, nur bei der Verneinung!

2. Als Systemoption setzen, so dass sie für die gesamte SAS Sitzung verfügbar ist:

```
Options minoperator;
```

SECURE Option bei %macro

Syntax nicht einsehbar

Schutz „geistigen Eigentums“

Verschlüsselt das dauerhaft erstellte Makro und deaktiviert die %COPY Anweisung sowie die Optionen Symbolgen , mprint und mlogic.

```
%macro calc(stats,vars)/store secure;
   proc means data=orion.order_fact;
      var quantity;
   run;
%mend calc;
```

Makros und Makrofunktionen

1. Verschachtelungstiefe + Name des aufrufenden Makros:%Sysmexecdepth und %sysmexecname
2. %sysmacexec und %sysmacexist
3. %sysmstoreclear und %sysmacdelete
4. Dosubl Funktion (als Alternative zu Call Execute)
5. %tslit Makro

Makrofunktionen

Verschachtelungstiefe: %sysmexecdepth und %sysmexecname

%SYSMEXECDEPTH

Gibt die Verschachtelungstiefe bei verschachtelten Makros zurück (eine Zahl).

%SYSMEXECNAME

Gibt den Namen des Makros zurück, der bei einem verschachtelten Level ein Makro aufruft.

%sysmexecdepth und %sysmexecname

Syntaxbeispiel

```
%macro Eins_outer;
%put das ist das Makro Eins_outer mit Level %sysmexecdepth();
%zwei_inner
%mend Eins_outer;

%macro Zwei_inner;
%put Makro zwei_inner mit der Verschachtelungstiefe  %sysmexecdepth
      aufgerufen von Makro  %sysmexecname(%sysmexecdepth-1);
%mend Zwei_inner;
%Eins_outer
```

Meldung im Log:

```
das ist das Makro Eins_outer mit Level1()
Makro zwei_inner mit der Verschachtelungstiefe2        aufgerufen von MakroEINS_OUTER
```

Makrofunktionen

Makrofunktionen: %SYSMACEXEC und %SYSMACEXIST

%SYSMACEXEC

Gibt an, ob ein Makro momentan ausgeführt wird

%SYSMACEXIST

Gibt an, ob eine Makrodefinition im WORK.SASMACR Katalog ist.

Makroanweisungen

%sysmstoreclear und %sysmacdelete

%SYSMSTORECLEAR:

Die permanent gespeicherten Makros locken die SASMSTORE Bibliothek während einer SAS Sitzung.

Um diese Sperrung aufzuheben, kann man folgende Anweisung abschicken:

```
%sysmstoreclear;
```

%SYSMACDELETE:

Löscht eine Makrodefinition aus dem WORK.SASMACR Katalog.

Syntax: %sysmacdelete makroname ;

Dosubl Funktion

Dosubl Funkion als Alternative zu Call Execute

Die DOBSUBL Funktion ermöglicht die SOFORTIGE Ausführung von SAS Code, nachdem ein String übergeben wurde.

Das passiert bei CALL Execute(Argument):

Wenn das Argument aufgelöst wird zu:	Das Folgende passiert:
Einem Makroaufruf	Das Makro wird sofort ausgeführt während der DATA Step die Ausführung pausiert.
SAS Code, oder wenn die Makroausführung SAS Code generiert	Der SAS Code wird ausgeführt NACH dem Data Step, der den Call EXECUTE Aufruf beinhaltet.

Fehlerhaftes Makro mit Call Execute

```
libname orion 'c:\workshop\winsas\mc2_2015';
%macro orders(year);
   data orders;
      set orion.order_fact end=final;
      where order_type=3 and year(order_date)=&year;
      Number+1;
      if final=1 then call symputx('num', Number, 'L');
   run;
   %if &num le 20 %then %do;
      proc print data=orders;
                  footnote "&num Internet Orders &year";
      run;
   %end;
%mend orders;
data _null_;
 do Year=2007 to 2011;
 call execute(cats('%orders(', Year, ')' ));
 end;
run;
```

Das Makro beinhaltet Data Step Syntax.
Call Execute ruft zwar das Makro mit jeweils unterschiedlichen Parametern auf, diese können jedoch nicht durchlaufen, solange der data _null_ Step noch nicht fertig ist.
Daher ist die Makrovar **num** nicht bekannt!

Die erzeugte Data Step Syntax im Makro %orders kann erst nach Beendigung des data _null_ Steps laufen

```
libname orion 'c:\workshop\winsas\mc2_2015';
%macro orders(year);
   data orders;
      set orion.order_fact end=final;
      where order_type=3 and year(order_date)=&year;
      Number+1;
      if final=1 then call symputx('num', Number, 'L');
   run;
   %if &num le 20 %then %do;
      proc print data=orders;
                  footnote "&num Internet Orders &year";
      run;
   %end;
%mend orders;
data _null_;
do Year=2007 to 2011;
*call execute(cats('%nrstr(%orders(', Year, '))' ));
rc=dosubl(cats('%orders(', Year, ')' ));
end;
run;
```

2 Lösungsansätze:

1. **call execute** mit verzögerter Ausführung (%nrstr), damit der data _null_ Step erst laufen kann.

2. **dosubl-Funktion:** pausiert den data _null_ Step, so dass der Makroaufruf und die damit enthaltene Data Step Syntax sofort durchlaufen kann

%TSLIT Makro

Überschreibt die Anforderung für doppelte Anführungszeichen und setzt den Eingabewert in einfache Anführungszeichen.

Die Unterscheidung zwischen einfachen und doppelten Anführungszeichen ist v.a. bei Datenbanken wichtig.

Syntaxbeispiel (s. auch nächste Folie):

```
proc fedsql;
    CREATE TABLE tab1(var1 CHAR(10));
    INSERT INTO tab1 VALUES(%tslit(&SYSHOSTNAME));
```

%TSLIT

```
proc fedsql;
    CREATE TABLE tab1(var1 CHAR(10));
    INSERT INTO tab1 VALUES(%tslit(&SYSHOSTNAME));

--- The following will produce an error because it ---
--- thinks it is looking for a column name.      ---

    INSERT INTO tab1 VALUES("&SYSHOSTNAME");
    SELECT * FROM tab1;
    DROP TABLE tab1;
quit;
```

Log Meldung:

```
8          INSERT INTO tab1 VALUES("SYSHOSTNAME");
ERROR: Syntax error or access violation
```

Sonstiges

1. %if in open code
2. %format Funktion
3. Fehlermeldungen und Beschreibung in der neuen Hilfe 9.4 „SAS Macro Language“

Sonstiges

%if in open code

%if in open code funktioniert, ohne dass ein äußeres Makro oder Sonstiges benötigt wird.

```
%if &sysday=Friday %then %do;
%put heute ist Freitag;
%end;
%else %do;
%put heute ist nicht Freitag;
%end;
```

Übrigens: Wissen Sie, wie man ein Format bei Makroavariablen mitgeben kann?

%FORMAT Funktion:

```
%let x=1111;                          Log
%put %format(&x,dollar11.);           $1,111
%put %format(&x,roman.);              MCXI
%put %format(&x,worddate.);           January 16, 1963
%put %format(Macro,$3.);              Mac
```

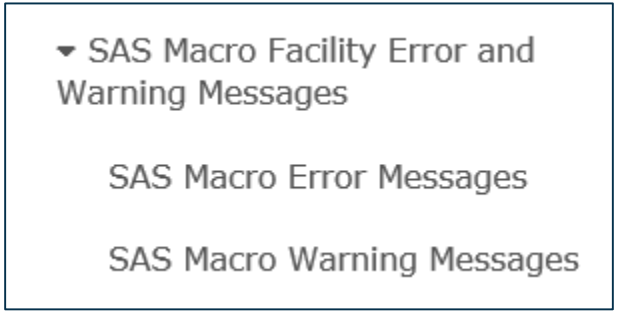
SAS 9.4 Macro language

Und zum Schluss: Wussten Sie, dass

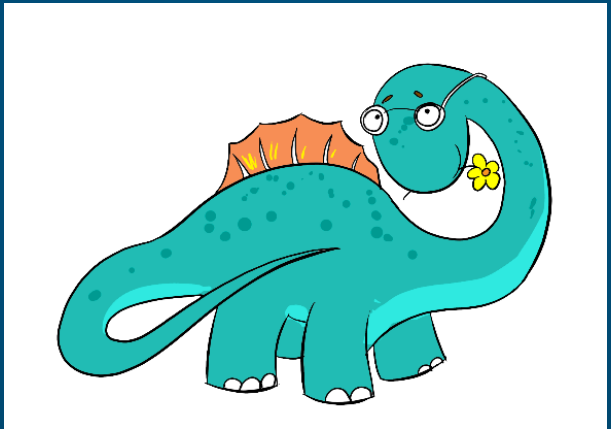
es in der Hilfe zur SAS Makro Sprache 9.4 im Internet jetzt die **Fehlermeldungen und ihre Beschreibungen** gibt?

Unter folgendem Link finden Sie Informationen dazu:

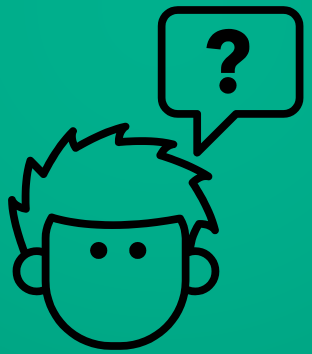
https://go.documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=mcrolref&docsetTarget=n1t4y4l0pye5fvn19bho9qc77cyu.htm&locale=en



Webinar@Lunchtime 2019



Fragen?



Weitere Informationen und Kurse zu diesem Thema…

SAS® Macro Language 1: Essentials

11. – 13. Dezember in Köln

13. – 15. Januar in Wien

20. – 22. Januar in Heidelberg

SAS® Makrosprache 2: Praxis für Fortgeschrittene

09. – 10. Dezember in Heidelberg

06. – 07. Februar in Heidelberg und als Connected Class

Strategien für die Entwicklung von Makro-basierten Anwendungen

16. – 18. Dezember in Heidelberg

Vielen Dank für Ihre Teilnahme!

SAS Education Newsletter

Der SAS Education Guide beinhaltet aktuelle Themen, wie neue Kurse und Services, Aktionen oder Neuerungen im Zertifizierungsprogramm. Sie erhalten außerdem hilfreiche Tipps von unseren Experten, die Ihnen das Arbeiten mit SAS erleichtern.

Jetzt abonnieren

Welche Newsletterversion interessiert Sie?

SAS® Education Guide Allgemeine Themen im Überblick

SAS® Education Guide Data Integration + Administration

SAS® Education Guide SAS® Programmierung + SAS® Enterprise Guide™

SAS® Education Guide Analytics + Reporting

Folien zum Download unter www.sas.de/lunchtime

Besten Dank für Ihr Feedback ☺

sas.com

THE POWER TO KNOW.