A Programmer’s Guide To The SAS Macro Facility

Macro Essentials

- %global <macro-variable>;
- %local <macro-variable>;
- %let <macro-variable> = <value> ;
- **%macro; %mend;**
- **proc sql**; select <dataset-columns>,... into: <macro-variable>" from <sas-dataset>; **quit**;
- **data** _null_; call symput(<macro-variable>, <macro-variable-value>); **run**
- options mprint symbolgen mlogic sasautos=(<macros location>);
- %put _all_;
- Always encase macro variables in DOUBLE QUOTES "<macro-variable>" for a string, SAS will not resolve macro variables for a character value in single quotes '<macro-variable>'

Getting Started with the Macro Facility

•The macro facility is a tool for extending and customizing SAS and for reducing the amount of text you must enter to do common tasks. The macro facility enables you to assign a name to character strings or groups of SAS programming statements. From that point on, you can work with the names rather than with the text itself.

•When you use a macro facility name in a SAS program or from a command prompt, the macro facility generates SAS statements and commands as needed. The rest of SAS receives those statements and uses them in the same way it uses the ones you enter in the standard manner.

•The macro facility has two components:

- the macro processor is the portion of SAS that does the work.
- the macro language is the syntax that you use to communicate with the macro processor.

- When SAS compiles program text, two delimiters trigger macro processor activity:
  - **&**<name> refers to a macro variable. The form **&**<name> is called a macro variable reference. A macro variables value is always character. The value can be a variable name, a number, or any text string to substitute into a program.
  - **%**<name> refers to a macro. The form **%**<name> is called a macro call. A macro call can have multiple code statements. Multiple data steps , procedures and conditional logic can be called in the same macro call. Conditional logic coding, `%do`, `%end`, and `%if-%then / %else` code must be contained in a macro call.

- When a macro variable is created in a macro call, it is defaulted at `%local` , unless it is set as `%global`, meaning that if the macro variable is `%local` , the macro facility will only recognize it in the macro call where it was created, whereas if it is set to `%global` it can be used throughout the session.

- The text substitution produced by the macro processor is completed before the program text is compiled and executed.

- The macro facility uses statements and functions that resemble those that you use in the DATA step. An important difference, however, is that macro language elements can only trigger text substitution and are not present during program or command execution.

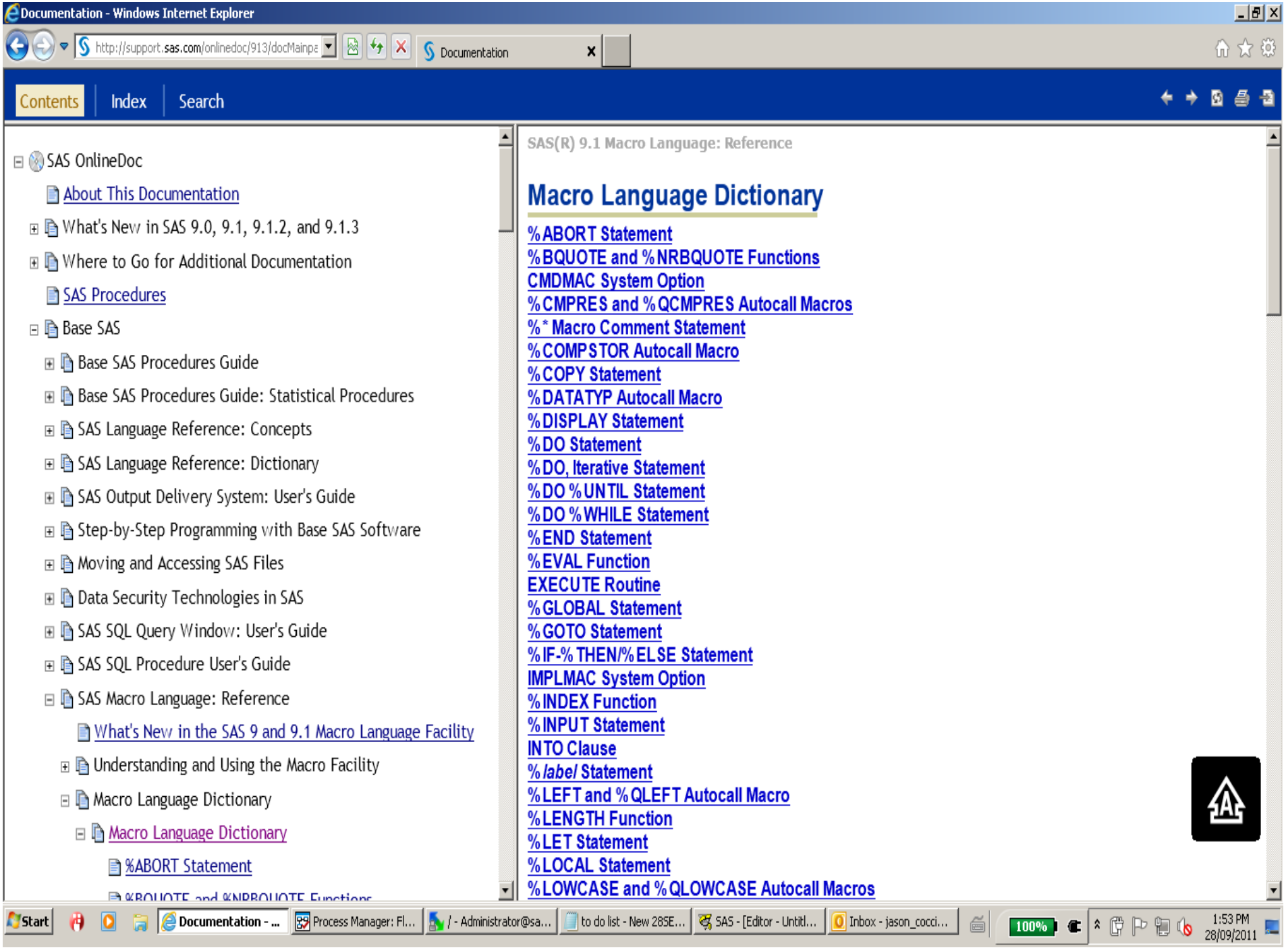
- Note: Three SAS statements begin with a % that are not part of the macro facility. These elements are the %INCLUDE, %LIST, and %RUN statements. These statements are documented in your Base SAS documentation.

Advantages to Macro Processing

- Program Automation (Run the same code for multiple queries, run times, reports etc…..)
- Promotes Standardized Programming Techniques.
- Code Integrity (changes to one variable instead of multiple pieces of code)
- Increased Portability. (Moving Code from one system to the other is easily managed by changing macro variables instead of hard coded locations etc….
- Simplified Coding (less repeat coding decreases the chance for errors)
- Alternative to Arrays (more efficient, easier to use)
- Yes/No Options – Exclude code using conditional logic.

Macro Literature/References

- SAS Online Documentation (http://support.sas.com/onlinedoc/913/docMai)
- Google It (www.google.com, www.groups.google.com)
- The Little SAS Book - Chapter #: Writing Flexible Code with the SAS Macro Facility
- SAS Training and Bookstore http://support.sas.com/learn/


Documentation - Windows Internet Explorer
http://support.sas.com/onlinedoc/913/docMainpa
Documentation
Contents
Index
Search
SAS OnlineDoc
About This Documentation
What's New in SAS 9.0, 9.1, 9.1.2, and 9.1.3
Where to Go for Additional Documentation
SAS Procedures
Base SAS
Base SAS Procedures Guide
Base SAS Procedures Guide: Statistical Procedures
SAS Language Reference: Concepts
SAS Language Reference: Dictionary
SAS Output Delivery System: User's Guide
Step-by-Step Programming with Base SAS Software
Moving and Accessing SAS Files
Data Security Technologies in SAS
SAS SQL Query Window: User's Guide
SAS SQL Procedure User's Guide
SAS Macro Language: Reference
What's New in the SAS 9 and 9.1 Macro Language Facility
Understanding and Using the Macro Facility
Macro Language Dictionary
Macro Language Dictionary
%ABORT Statement
SAS(R) 9.1 Macro Language: Reference
Macro Language Dictionary
%ABORT Statement
%BQUOTE and %NRBQUOTE Functions
CMDMAC System Option
%CMPRES and %QCMPRES Autocall Macros
%* Macro Comment Statement
%COMPSTOR Autocall Macro
%COPY Statement
%DATATYP Autocall Macro
%DISPLAY Statement
%DO Statement
%DO, Iterative Statement
%DO %UNTIL Statement
%DO %WHILE Statement
%END Statement
%EVAL Function
EXECUTE Routine
%GLOBAL Statement
%GOTO Statement
%IF-%THEN/%ELSE Statement
IMPLMAC System Option
%INDEX Function
%INPUT Statement
INTO Clause
%label Statement
%LEFT and %QLEFT Autocall Macro
%LENGTH Function
%LET Statement
%LOCAL Statement
%LOWCASE and %QLOWCASE Autocall Macros
Start
Documentation - ...
Process Manager: Fl...
/ - Administrator@sa...
to do list - New 28SE...
SAS - [Editor - Untitl...
Inbox - jason_cocci...
100%
1:53 PM
28/09/2011

Let's Look at Some Code

```
%let region = Africa;

data work.dset;
  set SASHELP.Shoes;
  where Region = "&region.";
run;
```

Resolves to:

```
data work.dset;
  set SASHELP.Shoes;
  where Region = "Africa";
run;
```

```
%macro get_regional_data(region);
data work.&region._dset;
  set SASHELP.Shoes;
  where Region = "&region.";
run;
%mend get_regional_data;

%get_regional_data(region=Africa);
```

Resolves to:

```
data work.Africa_dset;
  set SASHELP.Shoes;
  where Region = "Africa";
run;

%get_regional_data(region=Pacific);
```

Resolves to:

```
data work.Pacific_dset;
  set SASHELP.Shoes;
  where Region = "Pacific";
run;
```

SAS Coding Examples

Refer to base SAS programs.

- SAS_Macro_Processing_Examples.sas

Using Prompts With SAS Enterprise Guide

- Prompts are a way to automate you SAS EG projects.
- Creating Prompts uses the macro facility to pass macro variables created by prompts
- Refer to EG project
  - Prompts Example.egp

Thank You

Jason Coccimiglio

jcoccimiglio@da-global.com