

How to Setup a SAS AutoCall Library

Clarence Wm. Jackson

Using a SAS AutoCall Library can be very helpful if you have many utility macros. This paper details how to setup an AutoCall library in the MVS and Windows environment.

One of the great features about SAS is the Macro Facility that is included with the Base SAS System, and the ability to include your own Macros within the SAS program via the AutoCall feature. Any valid SAS macro can be stored in the AutoCall library, which can do nothing but improve your productivity and the efficiency of your SAS programs. By using the AutoCall feature, you don't have to *%INCLUDE* or include the Macro definition within the program, but your macro is retrieved from the library when needed.

An AutoCall library is an aggregate storage location containing individual library members, each of which contains a macro definition. Each member should be named after the macro. To keep things simple, you should store only one macro in each member, and only macro code in the AutoCall library. By the AutoCall being in a separate storage location from your code, it can be used with any SAS program or session as needed, and by multiple SAS programs and users at the same time.

So those nice macros can now be used whenever you need them within a SAS program, and since the macros are stored independent of your SAS program, you can share the macros with other SAS programmers in your shop, as well as in other programs, without copying the macro. The benefits of having the AutoCall library are many, but using the example macro in the paper "How to Make Any SAS program a Macro" and a few others, I'll show you how to setup your own SAS AutoCall Library, and how to use it.

Setting SAS Up To Use AutoCall Libraries

To use the AutoCall macros, you'll need to set the following SAS System options using the *OPTIONS* at execution:

- *IMPLMAC* - allows SAS to recognize statement-style macros
- *MAUTOSOURCE* - enables the AutoCall facility
- *MRECALL* - causes the macro processor to search AutoCall libraries for members not found in a previous search
- *SASAUTOS=library-specification(s)* - specifies the AutoCall library or libraries operating system name. Refer back to this file using *FILENAME* later in your program.

The SAS System looks for AutoCall library members in libraries you specify with the *SASAUTOS=* system option by searching each opened library specified for the member with the same name as the macro you are calling. When it finds the member, it then compiles the source code within the member, executes any open code within the member once, and then executes the macro. The next time the macro is called in the same session or job, any open code that was executed is ignored. If the member is missing from your

library, you will get an *ERROR 180-322*, plus the other errors associated with missing macro definitions.

MVS AutoCall libraries uses the PDS (partitioned Data Set) storage method. The macros must be on the same file as members. For instance, say you have 3 macros that you want to use call AMAC, BMAC, and CMAC. Put the macros into separate PDS members call AMAC, BMAC, CMAC, on the PDS file (SAS.MACROS). The SAS program would include the following options in your EXEC SAS card, and the file reference to call the program in MVS batch JCL:

```
//S1 EXEC SAS,OPTIONS="MAUTOSOURCE SASAUTOS=SASMACS "  
//SASMACS DD DSN=SAS.MACROS,DISP=SHR
```

Or you can use the notation for all SAS programs:

```
FILENAME SASMACS "SAS.MACROS";  
OPTIONS SASAUTOS=SASMACS;
```

This notation works for all SAS programs regardless of the operating system. To setup a Personal Computer (all Windows versions, O/S 2, Mac, etc), all the macro members must have the ".SAS" extension, and be located within the same directory. So, for the 3 macros AMAC, BMAC, and CMAC to be use in the AutoCall facility, they must be stored as separate files with the names AMAC.SAS, BMAC.SAS, CMAC.SAS within a common directory (example C:\SASMACRO\). To call them, use the following notation:

```
FILENAME SASMACS " C:\SASMACRO";  
OPTIONS SASAUTOS=SASMACS;
```

You can add members to your macro library in what ever way you now write SAS code, from NotePad to Roscoe and TSO, as long as the member is included without special characters. I use NotePad on Windows 95, and Roscoe's "exp dsn=" on the mainframe, and sometimes TSO's editor. The key thing to remember is to name the member/file the same as the macro, and include the ".SAS" extension for PC files.

The AutoCall facility is the best way to store extensions to the SAS language in the form of utility routines and macro subsystems. You don't have to remember to retrieve key macros at DATA or PROC step boundaries with the *%INCLUDE* statements, and you don't have to include the macro definition and code within your program. You can use the macro in many programs just by calling it, and maintenance changes to the macro can be made in one place, and not in all the places that it's used. By the same token, any changes you make to the macro will be reflected in all the programs that call it, so if others are using the macro, let them know about any changes you make to it.

SAS Institute supplies libraries of AutoCall macros to each SAS site, and these can be used within your SAS programs. For more on these, either review the online help within the SAS System, or get a copy of the *SAS Guide to Macro Processing*.

The AutoCall Facility is a nice tool that can improve your personal productivity, and improve the SAS programs you write. If you are in a shop with other SAS programmers, it can be a way to share code and improve group productivity. It should be used with care, as there are unproductive ways to use the facility. You should only include fully tested and completely ready macros to the AutoCall library. Try to limit the content of the library to utility or reusable type code, as these types will pay back more than those one time or hardly used code. Develop your macros in your program, and move it to the AutoCall library when it's tested and ready. Document the macros, and share the documents with users of the library. Maintain control over who can update and delete members in the library, and treat those members like valuable resources. And have fun!

About the Author

Clarence Wm. Jackson currently works for the City of Dallas ECI as the Production Control Supervisor and Quality Assurance Analyst. You can reach him via the Internet at CJac@compuserve.com, cljacks@ci.dallas.tx.us, or via his homepage at <http://ourworld.compuserve.com/homepages/CJac>. Snail mail him using the City of Dallas ECI (Operations and Plans), 1500 Marilla St 4DS, Dallas, TX 75201.