

LPUT and RPUT and Macros .. Oh My!

James R. Cowles

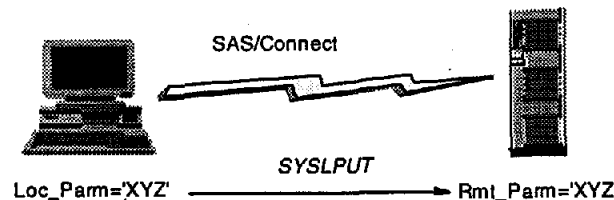
Abstract

In the use of SAS/Connect, occasions arise in which it is necessary to send macro parameters from a local host to a remote host, or *vice versa*, and have those macro parameter values resolved on the host to which they are sent. The exchange and resolution of macro values between hosts during a SAS/Connect session is accomplished by the use of two very useful but little-known SAS macros, SYSLPUT and SYSRPUT. SYSLPUT sends macro parameters from a local host to a remote host; SYSRPUT, the reverse. This paper illustrates the usage of both SYSLPUT and SYSRPUT in macro-parameter interchange and resolution.

On occasion, the necessity arises to use SAS/Connect to establish a conversation across a network between two platforms, and to run a macro on the remote host which takes as parameters values which must be passed in from the local host, and *vice versa*. The question then arises as to how to create on one host values that will be passed to the other host and resolved as macro parameters. The SAS System incorporates two very useful but little-known macro utilities, SYSLPUT and SYSRPUT, that can be used to pass macro parameters from the local to the remote host and *vice versa*, respectively. This paper gives short examples of the way these macro utilities work, and a couple of caveats about subtle differences in the way the two utilities are invoked.

The naming convention for these utilities is slightly confusing. In each case, the letter following the "SYS" prefix refers to the platform from which the macro invocation is *launched, not the target platform*. So in the case of SYSLPUT, the "L" indicates that SYSLPUT invocations are launched from the local platform because SYSLPUT is used to transfer macro parameter values from the local machine to the remote host. Similarly, the "R" in SYSRPUT indicates that SYSRPUT invocations transfer macro parameter values from the remote host to the local machine.

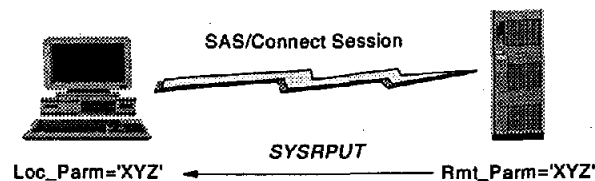
SYSLPUT



Page 3 of

LPUT and RPUT and Macros .. Oh My!

SYSRPUT



Page 4 of

A couple of examples will illustrate how SYSLPUT and SYSRPUT are invoked, and the possible uses of both in actual SAS macros. The following macro uses SYSLPUT to define the parameters for doing a PROC DOWNLOAD of an entry of type SLIST from a remote to a local host.

```
%macro getlist(LIB,CAT,LISTNAME,DIR);

    %syslput(LIB,&LIB);
    %syslput(CAT,&CAT);
    %syslput(LISTNAME,&LISTNAME);
    %syslput(REMDIR,&DIR);

    rsubmit;

    proc download incat=&LIB.&CAT
                  outcat=work_lib.work_cat;
    select &LISTNAME..slist;
    run;
    quit;

endrsubmit;

%mend getlist;
```

Obviously, I am presupposing a lot in the macro as written, the most important presupposition being that a SAS/Connect conversation has been established in the first place, along with all the appropriate librefs to enable the PROC DOWNLOAD call to function properly. But aside from such technical details, the important point to note is the collection of four calls to SYSLPUT following the %MACRO statement. These SYSLPUT calls pass the values of LIB, CAT, LISTNAME, and DIR to the *remote* host from which the selected SLIST entry will be extracted when the PROC DOWNLOAD executes. Without a way to resolve these macro parameters on the *remote* host, PROC DOWNLOAD would fail because it would have no way of knowing from which catalog in which library to extract which entry of type SLIST.

Similarly, suppose I am working on an application in which it is important to know whether or not the operating system I am running locally is the same operating system being run on the remote host I am accessing. Knowing this could be important because SAS/Connect Remote Library Services (RLS) gives no visibility of entries contained in catalogs that are resident on remote machines whose operating system is different from that of the machine I am using to access the remote catalog. If the respective operating systems are different, I cannot use RLS and must physically download the contents of the remote catalog to my local machine, using SAS/Connect Data Transfer Services (DTS), in order to see the catalog's contents. Since the network is a limited resource subject to contention, it might be important to make this determination ahead of time, so I would know whether I would have to use DTS, or whether I could get by using RLS to access remote catalog entries. Consequently, I might write the following macro to compare operating systems beforehand.

```
%macro getos(HOST,COMAMID,LOGSCRPT,VERBOSE=NO_VERBOSE);

    options remote=&HOST comamid=&COMAMID;
    filename rlink "&LOGSCRPT";
    signon &HOST;

    /* SYSRPUT has to be NRSTRd because of ;
    /* the possibility of embedded blanks ;

    rsubmit;
    %nrstr(%sysrput remoteos=&sysscp);
    endrsubmit;

    %if "&VERBOSE"="VERBOSE" %then
    %do;
```

```

%put OSNAME-Remote=&remoteos;
%put SYSSCP-Local=&sysscp;
%end;

%signoff &HOST;

%mend getos;

```

Again, aside from the assumptions I am making about the technical details of the network, the important point to note is the call to SYSRPUT in the RSUBMIT block. This call uses the pre-defined system macro variable SYSSCP, which always contains the name of the operating system, to assign a value to the *local* macro variable REMOTEOS. Therefore, the variable REMOTEOS will contain the text string denoting the name of the *remote* host operating system. This value can then be made available to whatever application needs it in the local system. For example, an SCL program could use a SYMGET call to pass REMOTEOS into a frames application.

A couple of points should be made about the foregoing macro invocations of SYSRPUT and SYSLPUT. First of all, note that the SYSRPUT call is issued as part of an RSUBMIT block. This mode of invocation is necessary because SYSRPUT has to run on the *remote* system in order to retrieve *remote* parameter values and pass them back to the local host. For reasons that will be clear subsequently in this paper, the SYSLPUT macro utility invocation must *not* appear in an RSUBMIT block. Finally, the SYSRPUT call is wrapped up in an %NRSTR call instead of being issued by itself. This is necessitated by the fact that the text string being retrieved in this particular case is the name of the remote-host operating system, and therefore there is a possibility that REMOTEOS may contain embedded blanks, e.g., "Sun Solaris 2.5". The %NRSTR call ensures that the entire text string, including embedded blanks, will be retrieved from the remote system. In other applications of SYSRPUT, in particular, those retrieving numeric values, it would not be necessary to enclose the SYSRPUT invocation in %NRSTR call.

Finally, there is an important difference in the availability of SYSLPUT and SYSRPUT. SYSRPUT is available as part of the default "out of the box" repertoire of SAS macro utilities, just like macro utilities SCAN, SUBSTR, etc. However, SYSLPUT is not that readily available, since, unlike SYSRPUT, SCAN, SUBSTR, *et al.*, SYSLPUT is not part of the pre-compiled suite of SAS macro utilities. Depending on the version of SAS you are running, the SAS macro language source for SYSLPUT may or may not be included as an ASCII text file on your system's SAS installation. If the macro source for SYSLPUT is not available on your system, you one of three alternatives: (1) call the macro specialists at SAS technical support and have them fax the macro source to you, or (2) have SAS technical support e-mail you the SYSLPUT source, or (3) keep reading this paper. The reason alternative (3) is a realistic option is because the source code for SYSLPUT is extremely short:

```

%macro syslput(macvar,macval,remote=);
  options nosource nonotes;
  %let STR = %str(rsubmit &REMOTE; options nosource;)
             %nrstr(%let) %str(&MACVAR=&MACVAL
             options source; endrsubmit;);
  &STR;
  options notes source;
%mend syslput;

```

Without entering into all the technical arcana of the SYSLPUT macro, it should be pointed out that the SYSLPUT macro, since it already has an RSUBMIT call embedded in the part where the *local* macro variable is set equal to the *remote* parameter value, does not need to be submitted in an RSUBMIT block, and in fact must not be. This is the reason for the major difference between the mode of invocation of SYSLPUT and its SYSRPUT counterpart.