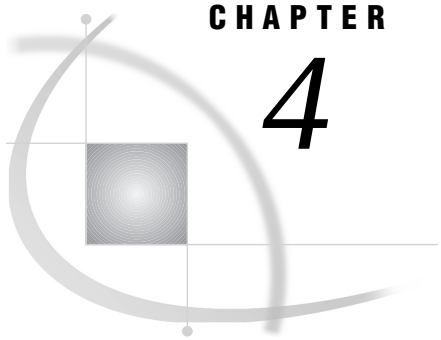


CHAPTER

4

Examples That Use Compute Services

Example 1. Compute Services: Using Remote Applications from a Local Host

Purpose

Some applications may not be available on every host at your computing site. In addition, the applications that are available on a remote host may perform some tasks better than the applications that are available on your local host. From a local host, you can use compute services to run applications that are available on a remote host.

Program

Here are two examples: the first example uses an application that is available only on the remote host; the second example uses an application that performs a sorting task more efficiently than the applications that are available on the local host.

Example 1.1: SAS/STAT Software

This example assumes that SAS/STAT is licensed only on the remote host. The program runs statistical procedures remotely by using SAS/STAT.

```
rsubmit;
      /*************************************/
      /* The output from GLM is returned   */
      /* to the local SAS listing.         */
      /*************************************/
   proc glm data=main.employee
      outstat=results;
      model sex=income;
   run;
      /*************************************/
      /* Use GLM's output data set RESULTS */
      /* to create macro variables F_STAT  */
      /* and PROB, which contain the       */
      /* F-statistic PROB>F respectively.  */
      /*************************************/
   data _null_; set results
      (where=(_type_= 'SS1'));
      call symput('f_stat',f);
      call symput('prob',prob);
   run;

      /*************************************/
      /* Create macro variables that       */
      /* contain the two statistics of     */
      /* interest in the local session.    */
      /*************************************/
   %sysrput f_statistic=&f_stat;
   %sysrput probability=&prob;
endrsubmit;
```

Example 1.2: Sorting

In this example, the remote host has access to a fast sorting utility, so that the data is sorted on the remote host before it is transferred to the local host.

```
rsubmit;
      /*************************************/
      /* Indicate to the remote host that  */
      /* the HOST sort utility should be   */
      /* used with PROC SORT. Ask SORT to  */
      /* subset out only those observations */
      /* of interest.                      */
      /*************************************/
   options sortpgm=host;
   proc sort data=tsolib.inventory
      out=out_of_stock;
      where status='Out-of-Stock';
      by orderdt stockid ;
   run;
```

```
      /************************************/
      /* Output results; local side will    */
      /* receive the listing from PRINT.    */
      /************************************/
   title 'Inventory That Is Currently Out-
         of-Stock';
   title2 'by Reorder Date';
   proc print data=out_of_stock;
      by orderdt;
   run;
endrsubmit;
```

Example 2. Compute Services: Administration Tasks for Remote Data Sets

Purpose

While working on a local host, you can use compute services to perform administration tasks on remote data sets.

Program

This example administers password protection to the TASKLIST data set and backs up a data set that is named CURRENT.

```
rsubmit;
   proc datasets lib=tsolib;
      /************************************/
      /* Add password SESAME to remote      */
      /* data set TASKLIST.                 */
      /************************************/
   modify tasklist (alter=sesame);
   run;

      /************************************/
      /* Maintain a week's worth of backup  */
      /* copies of data set CURRENT.        */
      /************************************/
   age current backup1 - backup7;
   run;
   quit;
endrsubmit;
```

Example 3. Compute Services: Remote Graphics Processing

Purpose

If you have SAS/GRAPH software installed on both your local host and remote host, you can submit graphics programs from your SAS session on a local host to a SAS session on a remote host, have the procedure execute on the remote host, and display the graphics output on your local host (or on a device attached to your local host). The link is especially useful when you want to generate graphics on your local host by using a large database on the remote host.

The GRLINK driver is a special driver available with SAS/CONNECT. You must always use the GRLINK driver on the remote host when using the link to display remote host graphics on your local host.

If you frequently use the link for remote graphics processing, consider specifying the GRLINK device driver in a script file (if you use a script file with the SIGNON command). You do this by including the driver specification for the remote host in the TYPE statement that invokes the remote SAS System. In the following example, if you are using TSO by means of a TCP/IP connection, change the TYPE statement in the script file to the following:

```
type
"sas options(remote=tcp device=grlink)";
```

By doing this every time you use the SIGNON command, you automatically specify the GRLINK driver on the remote host.

Note: This has already been specified in the sample scripts supplied with your SAS software. △

Program

Use the RSUBMIT command to submit your SAS statements. This includes any LIBNAME statement that is needed by the remote host. When the SAS/GRAPH procedure runs on the remote host, the output is displayed on your local host or on an attached device (based on the driver that you specified in your local session). If you did not specify a remote host driver name in step 2, you are prompted by the remote host to do it.

❶
```
goptions device=hplj;
```

❷
```
rsubmit;

   proc sort data=master.bg_reserve out=tmp;
      by origin rental_type;
   run;

   proc summary data=tmp vardef=n noprint;
      by origin rental_type;
      output out=temp_rental;
   run;
```

❸

```
   goptions dev=grlink ftitle=centx
   ftext=simplex htitle=2;

   title 'Rental Types by Franchise';
   pattern value=solid color=blue;

   axis1 label=('Franchise')
   order=
   ('ATLANTA' 'CHICAGO' 'LOS ANGELES'
      'NEW YORK' 'TORONTO')
   width=3;
   axis2 label=none width=3;
   axis3 label=none
   order=0 to 1000 by 100 width=3;
   proc gchart data=temp_rental;
      label rental_type='00'x;
      label origin='00'x;
      hbar rental_type /  frame
      sumvar= _freq_
      maxis=axis2
      raxis=axis3
      minor=0
      nostats
      group=origin
      gaxis=axis1
      discrete;
   run;
   quit;

endrsubmit;
```

❶ Specify an EGA graphics adapter to display the remote host graph on your local host. You can specify the name of the graphics driver for your local host display or its attached hardcopy device. To get a complete list of values for the DEVICE= option, run the GDEVICE procedure in your local SAS session.

❷ Remote submit procedures to preprocess data and graphics procedures to the remote SAS session.

❸ Specify GRLINK device driver so that commands to draw the graph will be sent over the network to the local session.

When using the link to display remote host graphs, you can use any graphics procedure on the remote host (including the GREPLAY procedure) and any graphics device driver on the local host.

The GRLINK remote host driver uses the attributes of the driver that is specified in the local host session when selecting default colors, character sizes, and other attributes. For example, if you specify DEVICE=EGAL in your local session, the GRLINK driver uses the default colors of the EGAL driver, but if you specify the printer driver DEVICE=FX85 in your local session, the GRLINK driver uses only black as a foreground color.

Note the following reminders when using the link for graphics:

- Do not specify GOPTIONS NODISPLAY in the program that you submit to the remote host. The option is not supported with the GRLINK driver on the remote host.

- Do not specify DEVICE=GRLINK in your local host SAS session. The GRLINK driver can only be specified on the remote host. In your local host SAS session, you can specify only a device driver that is available with SAS/GRAPH on that host.
- You can use hardware options, such as NOCHARACTERS, only on the local host side. That is, you cannot use hardware options that are not available with your particular local host hardware configuration even though they are supported on the remote host.
- To use the CBACK= or the ROTATE= option, you must specify it in your local host program, not in the program that you are submitting to the remote host. If you use the CBACK= or ROTATE= option in the program submitted to the remote host, the option is accepted but has no effect.
- To use the GREPLAY procedure through the link, you must use the NOFS option in the PROC GREPLAY statement.
- Every time you generate graphics output on the local host, it is stored temporarily, while running the same SAS session, in a catalog called GSEG in the WORK library of your local host. Later, displays of the same graphics output can be generated from this catalog. Copy this catalog to a permanent location if you want to retain a copy after termination of your current SAS session.

You can also transfer catalog entries containing graphics output by using the UPLOAD and DOWNLOAD procedures, as described in “Example 11. DTS: Uploading a Catalog That Contains Graphics Output” on page 166.

Example 4. Compute Services: Asynchronous Remote Processing

Purpose

Enables you to submit processing to be performed on a remote host so that the local host can immediately continue processing, and then retrieve and merge the results upon completion of that process.

Program

This example submits a PROC SORT and a PROC PRINT statement to be executed asynchronously on a remote host. This remote process can be tested for completion by using the macro variable DONE.

```
rsubmit cwait=no cmacvar=done;
   proc sort data=permdata.standard(keep=fname
      lname major tgpa gender)
      out=honor_graduates(where=(tgpa>3.5));
      by gender;
   run;

   title 'Male and Female Honor Graduates';
   proc print;
      by gender;
   run;
endrsubmit;

%macro get_results_when_complete;
```

```
   %if &done=0 %then %do;
     %put Remote submit complete,
        issuing "rget" to get the results.;
     rget;
   %end;
   %else %do;
     %put Remote submit not complete.;
     %put Issue:
        "%nrstr(%%)get_results_when_complete"
        later.;
   %end;
%mend;
%get_results_when_complete;

   /* issue again if RSUBMIT not complete */

%get_results_when_complete;
```

Example 5. Compute Services: Output Delivery System

Purpose

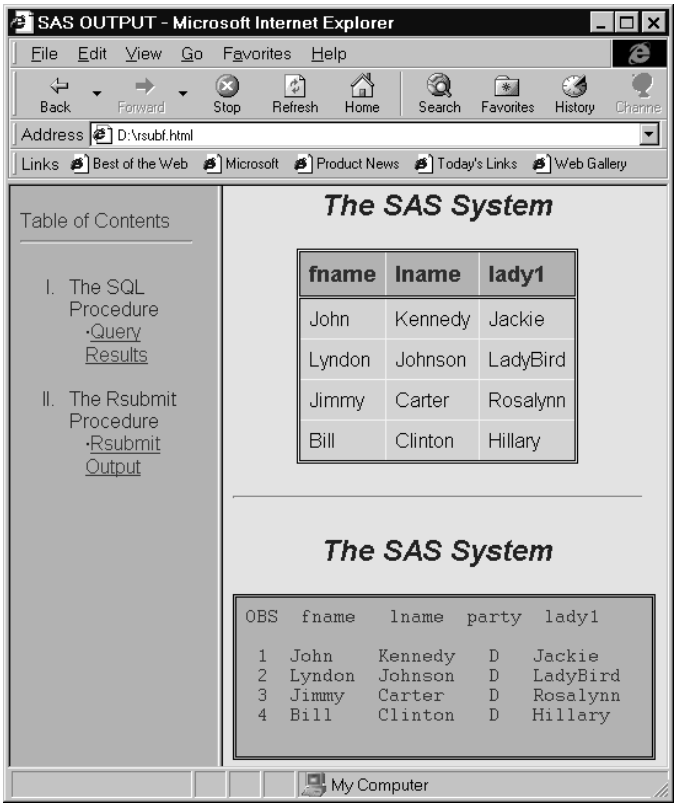
ODS allows you to format and change the appearance of a procedure's output. The output is converted into objects that can be stored in HTML or in a SAS data set. The output can then be manipulated and viewed in many different ways.

Program

This example creates a SAS data set and a SAS data view on a remote host, which contain information about U.S. Presidents and generates ODS output from them. The first half of this example creates HTML from the SAS data set and view. The second half uses ODS to create a SAS data set from the SAS data view.

```
rsubmit;

   data presidnt;
      length fname lname $8 party $1 lady1 $10;
      input fname lname party year_in lady1;
   datalines;
John Kennedy D 1961 Jackie
Lyndon Johnson D 1963 LadyBird
Richard Nixon R 1969 Pat
Gerald Ford R 1974 Betty
Jimmy Carter D 1977 Rosalynn
Ronald Reagan R 1981 Nancy
George Bush R 1989 Barbara
Bill Clinton D 1993 Hillary
   ;
   run;

   proc sql nocheck;
```

```
      create view democrat as
      select fname,lname,party,lady1
         from presidnt
         where party='D';
   quit;

endrsubmit;

   /* Use ODS to create HTML from the output */
ods html
   file="rsub.html"
   contents="rsubc.html"
   frame="rsubf.html";

   /* Remote SQL PassThru to DATA step view */
proc sql nocheck;
   connect to remote (server=rmthost);
   select fname,lname,lady1
      from connection to remote
      (select * from democrat);
quit;

   /* mix remote-submitted SQL with local SQL */
rsubmit;
   proc print data=democrat;
   run;
endrsubmit;

ods html close;

   /* Use ODS to create a SAS data set */
ods output output="rdata";

rsubmit;
   proc print data=democrat;
   run;
endrsubmit;
```



Example 6. Compute Services: Using MP CONNECT for Multi-Processing

Purpose

To perform two encapsulated tasks in parallel and to wait for the completion of both tasks before continuing the local SAS session execution.

Program

This example sorts two data sets asynchronously. After both sorts complete, the results are merged together.

```
/* Global SAS system option SASCMD starts an MP CONNECT remote session. */
option sascmd="sas";

/* Remote submit first task. */
rsubmit process=task1 wait=no;
libname mydata '/project/test1';

   /* Sort the first data set as one task. */
   /* SIGNON performed automatically by RSUBMIT. */
   proc sort data=mydata.part1;
     by x;
run;
endrsubmit;
```

```
/* Remote submit second task. */
/* SIGNON performed automatically by RSUBMIT. */
rsubmit process=task2 wait=no;
libname mydata '/project/test2';

   /* Sort the second data set as one task. */
   proc sort data=mydata.part2;
     by x;
run;
endrsubmit;

/* Wait for both tasks to complete. */
waitfor _all_ task1 task2;

*/ Merge the results and continue processing. */
libname mydata '/project/test2';
data work.sorted;
  merge mydata.part1 mydata.part2;
run;
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/CONNECT User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 537.

SAS/CONNECT User's Guide, Version 8

ISBN 1–58025–477–2

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999