

PROC UPLOAD and DOWNLOAD: An Easy Way to Export SAS Processing

Martin O'Connell
GEICO Direct, Washington DC

Abstract

File transfer using PROC UPLOAD and DOWNLOAD (available through the use of SAS/CONNECT[®]) represents a convenient way of transferring SAS[®] data and program libraries from platform to platform. By doing this, subsequent development can be transferred from restrictive environments to more user-friendly ones, such as the Windows PC. In addition, utility work such as job processing can be performed from a local SAS session using CONNECT. The use of SAS/AF[®] to automate these processes can provide a transparent way of developing on a client PC, while processing on a more powerful remote server.

This paper should be particularly useful to anyone using a high-end PC as a terminal for IBM[®] mainframe SAS program development. It discusses SAS/CONNECT in general, and gives specific details on transfers of SAS data and code libraries between mainframe and PC hosts. It also discusses the use of SAS/AF to automate processes when transfers are done routinely.

Introduction

The effective use of the SAS/CONNECT software to transfer some or all of SAS program and/or data libraries to other platforms can speed program development and testing immensely. In the process, relative strengths associated with certain platforms can be exploited, while exposure to relative weaknesses can be avoided or minimized during testing.

In the particular case when the developer has SAS/CONNECT licensed on both the PC and mainframe platforms and is developing mainframe applications, SAS/CONNECT offers a convenient, transparent, reliable transfer of SAS program and data files. During testing and development, it offers the ability to control when and where the execution of SAS programs occur. Used effectively, the raw power and storage depth of midrange or mainframes can be united with the convenient output processing of the PC to speed SAS program development. Further, processing and storage can be offloaded during testing thus saving processing and storage costs in cost-driven mainframe environments.

The power of SAS/CONNECT can also be coupled with the nice display and interactive qualities of SAS/AF, so that easy-to-build and functional user interfaces can be developed that automate much of these processes. In addition, utility functions (such as file allocation) can be performed through SAS, which, if incorporated into an appropriate SAS/AF application, would allow it to perform much of the utility work of such mainframe utilities as ISPF.

Overview of SAS/CONNECT

SAS/CONNECT is software that provides true client/server functionality between SAS sessions running on local and remote hosts. It can serve as a simple tool to facilitate file transfer, or as the backbone for distributed SAS systems. Latest releases of SAS/CONNECT supports access methods using the APPC, DECnet, NetBIOS, TCP/IP, ASYNC, EHLLAPI, 3270 and TELNET communication protocols making it adaptable to almost any configuration. Most of the time, the local host is Windows, while the remote one is either a UNIX, MVS or VMS running on either a midrange or mainframe.

With SAS/CONNECT, the user has access to most of the files, hardware and software on the local and remote hosts. All commands are issued from the local SAS program editor, with OUTPUT and LOG information served back to the local LOG and OUTPUT windows. It can also execute operating system commands, batch files, and perform batch submissions.

The two types of services offered by SAS/CONNECT are 1) Computing Services consisting of RSUBMIT/ENDRSUBMIT processing, and SQL Pass-Through, and 2) Library Services consisting of Remote Library Services (RLS), and file transfer using PROC UPLOAD and DOWNLOAD.

With RSUBMIT/ENDRSUBMIT processing, code can be parsed and directed to local and remote processor(s) as follows:

```
signon;
sas code for local processing;

RSUBMIT;
sascode for remote processing;
ENDRSUBMIT;

signoff;
more sas code for local processing;
```

With SQL Pass-Through, queries are generated in the local SAS session using PROC SQL and passed via a remote SAS session to a remote SQL processor, usually in conjunction with SAS/ACCESS[®]. This allows the user to submit SQL statements to a remote DBMS such as Oracle[®] or DB2 from a local SAS session. This type of processing is used frequently in data warehousing when SAS is used access the warehouse data.

Remote Library Services allows one to establish LIBNAMEs locally, which point to remote data libraries. This allows data and processing to reside on separate platforms. This can be thought of as implicit data transfer where data is transferred only as needed for processing. Use of RLS is limited to program-to-program communication protocols (APPC, DECnet, NetBIOS and TCP/IP). In general, experts recommend that SAS data reside as close to the processing as possible. Hence the use of RLS should be restricted only to cases where relatively small amounts of data are accessed.

File transfer using PROC UPLOAD and DOWLOAD is explicit data transfer where SAS data (or some subset of it), program or program libraries, or SAS catalogs are transferred before the next SAS statement is processed. The use of PROC UPLOAD and DOWNLOAD is very convenient for the transfer of moderately-sized SAS data files in that the data is transferred as SAS data, hence the user does not have create and manage transport data files. The transfer process takes place in the SAS environment meaning that all transfer details are written to the local SAS LOG.

Examples

The following examples assume the remote SAS session has been established. Example 1 illustrates the use of SQL pass-through and remote processing, while the remainder illustrates the use of PROC UPLOAD and DOWNLOAD to transfer SAS data and program libraries.

Example 1: Extract data from remote database

This example shows how a remote DBMS can be queried from a local SAS session. The query results are stored as permanent SAS data so that subsequent analyses access SAS data, not Informix data.

```
rsubmit;

libname mylib '/remote/sas/library/';

proc sql;

connect to informix(database='remote_db');
create table mylib.mytable as
select sas_var1 sas_var2
from connection to informix(
select A1.informix_col_1, A1.informix_col_2
from informix.table_1 A1)
as t1 (sas_var1 sas_var2);

quit;

endrsubmit;
```

Example 2: Download an entire SAS data library

This example shows how an entire SAS data library can be downloaded from a remote mainframe to a local PC.

```
libname pclib 'c:\saslibs\';

rsubmit;
libname mvslib 'main.frame.sas.library' disp=shr;
proc download inlib=mvslib outlib=pclib;
run;
endrsubmit;
```

Example 3: Download a subsetted SAS dataset

This example shows how to download a SAS dataset that has been subsetted using dataset options. In general, any valid options can be used to subset transferred data.

```
libname pclib 'c:\saslibs\';

rsubmit;
libname mvslib 'main.frame.sas.library' disp=shr;
proc download data=mvslib.data1 (keep=var1 var2 var3)
out=pclib.data1;
run;
endrsubmit;
```

Example 4: Upload and create a "small" SAS data library

This example shows the upload of a SAS library into a newly allocated dataset. The space allocated for the library will depend on the installation-set defaults.

```
libname pclib 'c:\saslibs\';

rsubmit;
libname mvslib 'main.frame.sas.library' disp=new;
proc upload inlib=pclib outlib=mvslib;
run;
endrsubmit;
```

Example 5: Upload and create a "large" SAS data library

This example shows how to dynamically create a larger dataset to upload to (one that exceeds default space allocation).

```
libname pclib 'c:\saslibs\';

rsubmit;
libname mvslib 'one.big.honker.of.a.sas.library'
disp=new blksize=27648 space=(27648,(1000,200));
proc upload inlib=pclib outlib=mvslib;
run;
endrsubmit;
```

Example 6: Upload a SAS program directory to a program library PDS

This example shows how program libraries can be uploaded "en masse" to a newly created mainframe program library partitioned dataset (PDS).

```
options noxwait;

/* directory that is to be uploaded */
%let source = c:\sas\;
filename inlib '&source';

/* target upload PDS */
%let target = 'my.sas.program.library';

/* push the source directory into a temp file */
x "dir &source > c:\dir.tmp";
filename indir 'c:\dir.tmp';

/* temp file to contain %included SAS code */
filename upload 'c:\source.tmp';

/* create SAS dataset containing the names of files to be
uploaded */
data dir;
infile indir;

input @1 test $char1. @;
if test not in ('.', ' ');

input @1 fil $char9.
ext $char3.;
if ext eq 'SAS';

filenam = compress(fil || '.' || ext);
run;

/* put rsubmit statement and PDS allocation statements in
%include file */
data _null_;
file upload;

put 'rsubmit;' /
'filename outlib &target disp=new lrecl=132
blksize=3960' /
'dsorg=po recfm=fb';
run;

/* generate a proc upload statement for each directory in
the
%include file. Note the mod option in the %include file
*/
data _null_;
file upload mod;
set dir;

infile = compress('inlib(' || filenam || ')');
outfile = compress('outlib(' || fil || ')');
put 'proc upload' /
'infile=' infile /
'outfile=' outfile;
put 'run;';
run;

/* put endrsubmit statement in the %include file */
```

```
data _null_;
  file upload mod;

  put 'endrsubmit';
run;

/* execute the SAS statements in the include file */
%include upload;

/* delete temp files */
x 'del c:\dir.tmp';
x 'del c:\source.tmp';
```

Example 7: Download a SAS program library PDS to a SAS program directory

This example shows how program libraries can be downloaded from a mainframe PDS to a PC directory. Note the use of PROC SOURCE, unique to SAS running under MVS.

```
/* target directory */
%let pcdir=c:\sascode;
filename pcdir "&pcdir";

x "md &pcdir";

rsubmit;

/* source directory */
%let proglib=my.main.frame.sas.code.library;

/* temp file to contain source code that is %included below
note: &out is temp space (not macro variable) */
filename out '&out' disp=new lrecl=80 blksize=8000 dsorg=ps
recfm=fb;

/* program library to be downloaded */
filename in "&proglib" disp=shr;

proc source indd=in outdd=out nodata noprint;
  before 'proc download';
  before 'infile=in(xxxxxxx)' 11 noblank;
  before 'outfile=pcdir(xxxxxxx.sas);' 15 noblank;
  before 'run';
run;

/* execute %include file */
%include out;
endrsubmit;
```

Interface Development

When transfers are to take place routinely, many times it is worth the time and effort to automate these processes so that the transfers can take place with minimal coding from the user. Examples 6 and 7 above represent relatively complex file transfers that would benefit from some kind of automation. In the minimal case, this can be accomplished by the use of the macro language. This enables the user to parameterize code so that later transfers can be carried out without recalling and changing the base code. Once developed and tested, the macros can be called for the different scenarios that the user encounters in the course of his or her daily work.

On a grander scale, it might be worth the development of a SAS/AF application that incorporates visuals into the process to give the user a point and click way of initiating transfers. My experience with this approach has been very positive. In one instance, an interface was needed to interact with a client-owned mainframe. This environment was a true charge-back mainframe environment, so my decision to offload data and/or development was based mostly on cost. My simple interface performed all of my transfers and performed most of my utility work at the push of a button. This enabled me to do most of my testing

and development work on my PC with only final testing done in the mainframe environment. In my current work, I have a SAS/AF system that interfaces with MVS and UNIX hosts, concurrently if needed. In addition to signon/signoff duties, this interface allows me to transfer SAS data, programs, and LOG and OUTPUT files from batch submissions. It also allows me to initiate and monitor batch SAS jobs. Both of these interfaces were developed quickly, with just a basic understanding of SAS/AF on my part. They also have performed very well in their reliability and stability.

Conclusions

With the high degree of portability of SAS systems, SAS developers quite often have the option of developing and testing on multiple platforms. SAS/CONNECT can be used to easily and quickly transfer data and program libraries from platform to platform while staying within a SAS environment. In situations where this is done constantly, it can be worth the additional effort to develop an interface so that the transfers are done with minimal effort on the users part.

References

SAS/CONNECT Software Usage and Reference, Version 6, 2nd Edition.

Acknowledgements

SAS, SAS/ACCESS, SAS/AF, and SAS/CONNECT are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

IBM is a registered trademark of International Business Machines Corporation. ® indicates USA registration.

Oracle is a registered trademark of Oracle Corporation. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Any further questions or comments are welcomed at:

Martin O'Connell
 GEICO Direct
 Corporate Actuarial Research Department
 1 GEICO Plaza
 Washington, DC 20076

Voice: (301) 986-2771
 e-mail: martin_o@juno.com