Observations®
THE TECHNICAL JOURNAL
for SAS Software Users
www.sas.com/obs

Submitting a Batch Job on a Remote MVS Host from Your Local SAS Session

Peter Kellberg

Peter Kellberg is head of internal SAS Support in Statistics Denmark. His areas of expertise include base SAS software, including macro programming, SAS/AF software, client/server technology using SAS/CONNECT and SAS/SHARE software, and SAS/GRAPH software (maps and device driver aspects). He has been a SAS software user for more than 10 years.

Abstract

This article describes how it is possible to submit MVS batch jobs from your local SAS session. In other words, you don't have become an ISPF/PDF expert. The products used are SAS/CONNECT and SAS/AF software.

Contents

Introduction

From the PROGRAM window/Enhanced Editor you have possibly submitted quite a few SAS jobs. While the job is running you can't do anything else in that SAS session. You can of course switch to another task in Windows, let's say editing a letter. Or perhaps you start another SAS session. The nature of batch jobs is different. Once submitted you are free to do anything else (on the computer).

Batch jobs are convenient when you have to run large jobs that take considerable amounts of time (lets say, over night) or in other ways that are demanding such as disk capacity, memory, usage of tapes and so on. If this is not the case but your data reside on the MVS platform, you simply use the compute service (remote submit).

What is a Batch Job?

Think of a MVS batch job as a letter in an envelope. The letter could be the program that is to be executed, while instructions on the envelope tell the computer about the programming language in which the letter is written, what data is going to be used, and all of the other special requirements concerning the job. The instructions on the envelope have to be written in a special language called JCL (Job Control Language). When your letter is ready to be mailed you send it to the mainframe. In a figurative sense the letter then dumps through the mailbox. A little helper in the mainfame picks up your letter and checks the envelope in order to see if he is able to fulfill your request. If so, the job is executed. If not, the letter will be sent back to you with instructions on what's wrong.

An example of JCL:

```
1 //PKTEST JOB    123456
2 //          EXEC   SAS
3 //SYSIN     DD     *
4 libname master 'tso.pk.master';
5 data master.select;
6   set master.prod;
7   where region='2';
8   totexp=sum(of exp1-exp12);
```

Recognizable here is the SAS program starting in line 4. The first 2 lines are the JCL that gives the name of the Job, “PKTEST” and the account number (use of CPU is not free - yet), “123456”. The program to be run is a SAS program. Note that JCL statements begin with “//” and must be written in uppercase. Note also that the syntax may differ slightly from the syntax at your site.

Of course many other programs can be run this way (COBOL, PL/1, FORTRAN).

Batch jobs often reside as members in partitioned data sets. One way to edit and submit them is to use ISPF/PDF. If this sounds like gibberish to you - stay tuned!. You don’t have to get familiar with this developing environment (not very much). It would be nice if you could submit a batch job in the same way that you are used to by simply pressing a button.

The Goals

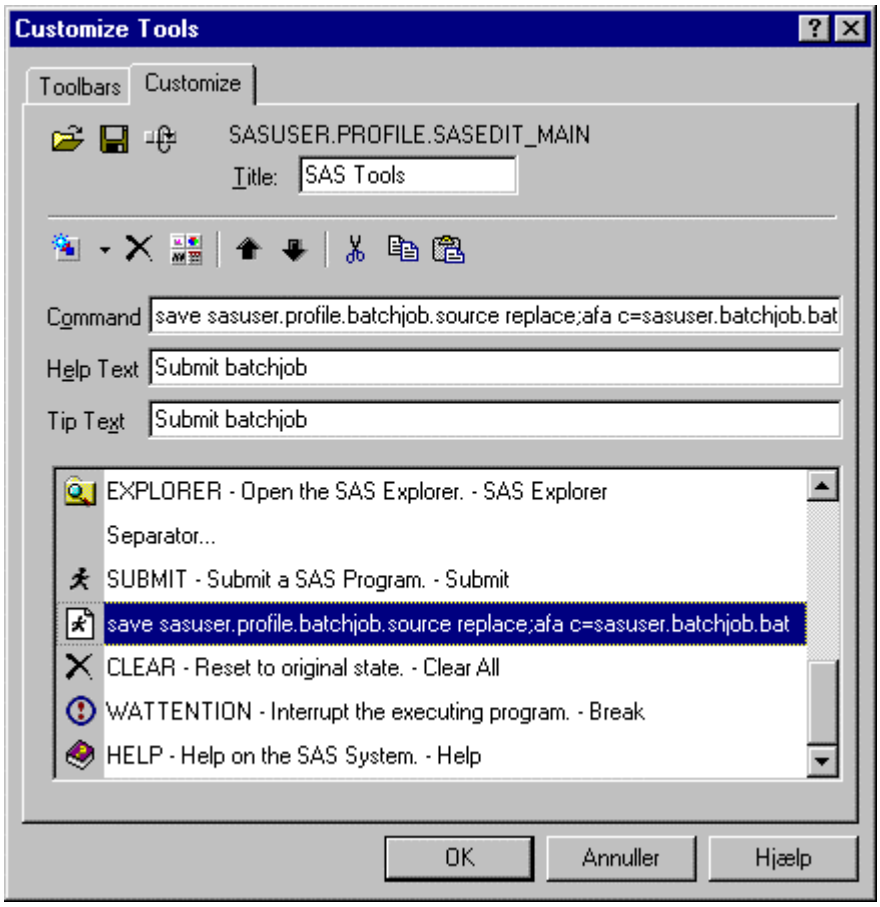
We want to supply the SAS user with an extra tool button that submits the batch job residing in the Enhanced Editor window.

The functionality of the button is to:

1 Get hold of the batch job from the Enhanced Editor window.
2 Invoke an SCL program that
   - Checks for/establishes a link between the PC and the mainframe.
   - Uploads the batch job to a special program called The Internal Reader.

Implementation of the Tool Button

From the Enhanced Editor Window select “Tools, Customize...” from the menu and select the “Customize” tab. Create an extra tool button next to the submit button like this:



The button should have these properties:

Properties of the new button	
Command	save sasuser.profile.batchjob.source replace; afa c=sasuser.batchjob.batchjob.scl
Help Text	Submit batchjob
Tip Text	Submit batchjob

The new toolbar should now look like this:

When you press the new button these commands are executed:

- **save sasuser.profile.batchjob.source replace**

  Save the program in the SOURCE entry BATCHJOB in the catalog SASUSER.PROFILE. Replace if the entry exists already. We’re doing this because we need to be able to get hold of the batch job when the text is to be uploaded. You may ask: Couldn’t we have used the FILE command instead? Doing so will change the filename in the Enhanced Editor window which will not only confuse the user, but it will also mean that the target file for saving future changes will NOT be the file that the user opened. Therefore, this method is not recommended.

I have chosen the catalog SASUSER.PROFILE but you could choose another catalog in another SAS library.

The entryname BATCHJOB in this example is simply straightforward but it is not likely that the user has an entry with that name in the SASUSER.PROFILE catalog. You are welcome to choose a more "odd" name.

- **afa c=sasuser.batchjob.batchjob.scl**

  This invokes the SCL program BATCHOB that resides in the catalog SASUSER.BATCHJOB. Use the AFA command instead of AF to ensure that this will work even though another AF application is running.

  In an installation where SAS is residing on a server, for example, the SAS System is installed at the local workstation as a "Client-installation", it's more convenient to place the application catalog in one of the folders included in the SASHELP library. This means that you place the file BATCHJOB.SC2 in one of the SASHELP folders of the SAS products you have licensed.

  Alternatively, you could create a new folder and place the catalog in there, but this would require changes to the SAS Configuration file.

  The SASUSER libref should then be replaced by the SASHELP libref instead.

Implementation of the SCL Program

As mentioned earlier the SCL program should perform the following tasks:

- Check for/establish a link between the PC and the mainframe.
- Upload the batch job to a special program called The Internal Reader.

This example is based on a EHLLAPI connection which implies usage of a script file. To specify a default script file you could enter the following in the CONFIG.SAS file:

```
set rlink c:\sas\connect\saslink\tso.scr  /* SIGNON and SIGNOFF to MVS */
```

The remote host name in this example is A.

To check for and establish a link is something you could benefit from not only in this context, so we will think of this as a method. Create an entry METHODS.SCL in the SASUSER.BATCHJOB catalog with the following code:

```
*********************************************
* Method for checking/establishing link to MVS *
* from a non FRAME environment                *
*********************************************;
mainbat: method OK_arg 8 / resident;

   /* set appropriate options */

   rc=optsetc('remote','A');
   rc=optsetc('comamid','ehllapi');

   if rlink('A') then   /* Link exists        */
      OK_arg=1;
   else do;            /* Link doesn't exist */
     submit continue;  /* Establish link     */
       signon;
     endsubmit;
     OK_arg=rlink('A'); /* set returncode: 0=failed 1=success */
   end;
endmethod;
```

Then create an entry BATCHJOB.SCL in the SASUSER.BATCHJOB catalog with the following code:

```
************************************************************
*      This program submits a batchjob via Internal Reader      *
************************************************************;
init:
   call method('methods.scl','mainbat',returncode);

   if returncode=0 then do; /* unsuccessfull SIGNON */
      put 'ERROR: Unable to establish connection';
      put 'Start your whatever 3270 emulator first. Then try again!';
      return;
   end;
   else do; /* successfull SIGNON */
   ******************************************************
   *      Clear the PREVIEW-Window and copy in the batchjob *
   *      from entry SASUSER.PROFILE.BATCHJOB.SOURCE.       *
   *                                                        *
   *      Then save batchjob as a flat textfile.            *
   ******************************************************;
      rc=preview('clear');
      rc=preview('copy','sasuser.profile.batchjob.source');
      rc=filename('pgmbin','c:\temp\batchjob.txt');
      rc=preview('file','pgmbin');
      rc=preview('clear');

      /* Upload file into Internal Reader */
      submit continue remote;
         options nonotes nosource;
         filename intread SYSOUT=A pgm=intrdr recfm=fb lrecl=80;
         proc upload infile='c:\temp\batchjob.txt'
               outfile=intread
               status=no;
         run;
         options notes source;
      endsubmit;

      rc=fdelete('pgmbin');        /* delete the file */
      rc=filename('pgmbin',' ');   /* clear filref    */

      put 'NOTE: Batchjob was submitted!';
   end;
return;
```

There isn't really much "hocus pocus" in this program:

1 Your method MAINBAT in METHODS.SCL is called returning a return code. If this return code is zero then something went wrong. Otherwise, let's do the Job!

2 Earlier in this article it was discussed why we had to save the batch job in a catalog entry. We have a problem here because we want the job in a flat file (OUTFILE in PROC UPLOAD demands INFILE). The PREVIEW window does the job, and you can manipulate it from SCL with a series of PREVIEW functions. And you don't need to see it at all in this application. First of all we make sure that the PREVIEW window is empty. Then it is filled with the batch job from a catalog entry. Then it is saved to an external file in a temporary folder, then made empty again.

Another workaround here could be to fill an SCL list with the contents of the SOURCE entry (FILLIST function) and then save the SCL list as a flat file (SAVELIST), like the following:

```
joblist=makelist();
rc=fillist('catalog(stripcc)','sasuser.profile.batchjob.source',
           joblist);
rc=savelist('file(stripcc)','d:\temp\batchjob.txt',joblist);
rc=dellist(joblist);
```

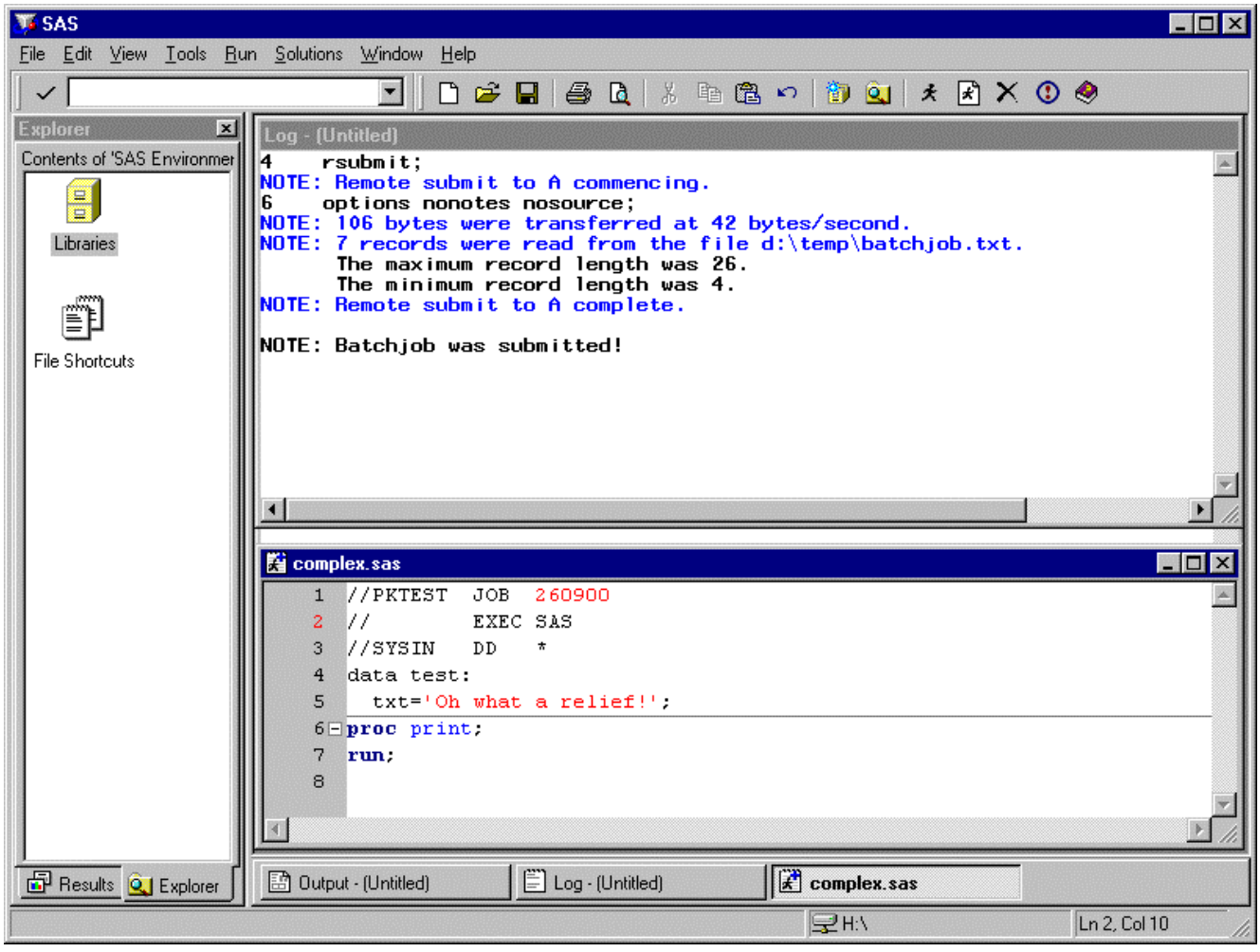
3 When the flat file is created we are ready to upload the file to the Internal Reader. Notice the special FILENAME statement that gets hold of the Internal Reader.

The Internal Reader is a special program that can be run under MVS. The input for the Internal Reader is text files containing one or more batch jobs. The Internal Reader then submits the batch job as if you have done it yourself. You could say that the Internal Reader acts like a spawner program for batch jobs. There is a very convenient way to get hold of the Internal Reader as you will see later.

Also notice that both the FILENAME statement and PROC UPLOAD are executed on the remote MVS host (`submit continue REMOTE;`)

An Example

A SAS user has opened a very complex batch job in the Enhanced Editor window:



When the batch job button is pressed the batch job is submitted. Nice and easy.
Notice that the program could be a COBOL, PL/I, FORTRAN etc. program as well.

Summary

If you are not an experienced user of ISPF/PDF you can use your local SAS session instead to edit and submit your batch jobs. You only have to frequent ISPF/PDF when you want to see the results from your efforts.

To do so you need only an extra tool button which saves the batch job and then invokes an SCL program that establishes a link to the remote MVS host, then uploads the batch job to the Internal Reader.

In this way you can submit batch jobs that contain SAS code but also COBOL, PL/I, FORTRAN etc programs as well. However, the great advantage comes when you are using SAS code because you can establish a test environment on the local platform. When the functionality is OK, simply change libname and filename statements into MVS specific syntax and supply, in the beginning of the program, the necessary JCL lines. Push the button and off your batch job goes.

References

SAS Institute Inc. (1994), *SAS/CONNECT Software: Usage and Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1996), *SAS Companion for the MVS Environment, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1994), *SAS Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.