Filling your SAS® Efficiency Toolbox:
Creating a Stored Process to Interact with Your Shared SAS® Server Using the X and SYSTASK Commands
SAS® Enterprise Guide® is a great interface for businesses running SAS® in a shared server environment. However, interacting with the shared server outside of SAS can require costly third-party software and knowledge of specific server programming languages. Using stored procedures we can:

- Copy or Move Files or Folders
- Change File or Folder Permissions
- Send specific commands to the server

Objectives

- Mastering X and Systask commands
- Leveraging the Server Log to print results
- Creating a Stored Procedure to make the code accessible to all users

Conclusion

- Use SAS® functions to talk to the server/system.
- Utilize the log to output results to the results tab.
- Combine these with SAS® Enterprise Guide® Prompts to replicate the process.

References


Contact Information

- Chad Stegeman, chad.stegeman@gmail.com
- Cody Murray, murra582@umn.edu
X and SYSTASK Commands

With the X and SYSTASK COMMAND functions users can pass the client-server commands directly through SAS®. These commands change depending on the SAS® environment used.

X Function

- The X COMMAND works with either PC or UNIX SAS.
- Using the X COMMAND with Windows will require some change to the code covered on this poster, but it is possible to make the code work. The commands will change from UNIX code to DOS command code. The command is passed to the operating environment and executed. *Only one command can be entered per X COMMAND in the PC environment.*
- The UNIX version of the X COMMAND processes commands differently than most functions. The X COMMAND prompts SAS® to start a shell to execute the commands that you specified. Each command can be processed differently depending on whether you entered one command or more than one command.

SYSTASK Command

- The SYSTASK command works very similar to the X COMMAND with it having two different versions depending on what kind of SAS® being used.
- For the PC and UNIX versions of SAS® the SYSTASK runs commands as asynchronous tasks, this means that tasks are executed independently of all other tasks that are currently running. Asynchronous tasks run in the background, so you can perform additional tasks while the asynchronous task is still running.
- If there is an error on PC SAS® the command will cause the program to end. Additionally, there will output written to the SAS® log with Windows SYSTASK.
- In the UNIX environment, SYSTASK will output to the SAS® log and as expected converting PC SAS® programs with the SYSTASK command will most likely result in errors.
Filling your SAS® Efficiency Toolbox:
Creating a Stored Process to Interact with Your Shared SAS® Server
Using the X and SYSTASK Commands

Chad Stegeman and Cody Murray, Medica Health Plans

X Function Syntax

Windows Syntax:
X <'command '>;
no argument
open a DOS command window.
command
specifies the command that you want to execute.

UNIX Syntax:
X UNIX-command
X 'cmd1;cmd2....<;cmd-n>

SYSTASK Command Syntax

Windows Syntax:
SYSTASK COMMAND "operating system command"

<WAIT | NOWAIT>
<TASKNAME=taskname>
<MNAME= name-variable>
<STATUS= status-variable>
<SHELL="shell-command">;
SYSTASK LIST <_ALL_ | taskname> <STATE> <STATVAR>;
SYSTASK KILL taskname <taskname...>;

UNIX Syntax:
SYSTASK COMMAND "operating-environment-command"

<WAIT | NOWAIT>
<TASKNAME=taskname>
<MNAME= name-variable>
<STATUS= status-variable>
<SHELL="shell-command">;
<CLEANUP>;
SYSTASK LIST <_ALL_ | taskname> <STATE> <STATVAR>;
SYSTASK KILL taskname <taskname...>;
Printing Results with the Server Log

Syntax and coding will be slightly different depending on the platform you are running SAS®. The example below uses SAS® in an AIX UNIX server environment:

1. Go to the desired directory
2. Use the “ls” command to display the contents of the directory.
3. The “>” symbol writes the output to the destination text file
4. Import the text file into a SAS® dataset. You may need to edit the import process
5. Clean up dataset and separate permissions for end user
6. Use “Proc Print” to display the server log in the results.

```sas
/* Displays the home folder contents after more */
\%LET home = "cd &Folder; ";
\%LET ls = "ls -go &AMC_Dest./Source.txt" ;
RUN:

DATA WORK.Source_01;
  set WORK.source_01;
  LastModified = strip(modifyDate/); Filetime_MBA = round(modifyDate-190000000,0.001/);
  OwnerPerm = strip(permissions,1,4); GroupPerm = strip(permissions,9,9);
  WorldPerm = strip(permissions,15,15);
  drop hardest modifyDate/;
RUN:

PROC PRINT data=Source_01 node label;
  title "Contents of the Source: &Folder,";
  var Filetime_MBA LastModified Filesize_MBA OwnerPerm GroupPerm WorldPerm;
  label LastModified = "Last Modified Date Modified" Filetime_MBA = "File size in MBA" OwnerPerm = "Owner Permissions" GroupPerm = "Group Permissions" WorldPerm = "World Permissions";
  footnote "Permissions are in Read Write Execute format.";
  footnote "The '0' in Owner indicates if the file is a folder (directory) or not";
RUN:
```

1
2
3
4
5
6

Chad Stegeman and Cody Murray, Medica Health Plans
Utilizing the Stored Procedure

Combining SAS® functions, prompts and log output. The example below uses SAS® in an AIX UNIX server environment:

1. Use Choice Prompt, file prompt and folder prompt
2. Use system macro to convert file prompt choice to a folder view
3. Run your command
4. Print your Server Log (slide 5)
5. Clean up server logs

---

```
/*trim folder name out of address*/
&let s11 = $trim($sysfunc(findc("$MC_File","/")));
&let s12 = $sysval(4211 + 2);
&let folder = $trim($sysfunc($trim($sysfunc(findc("$MC_File","/")),'/')+$trim($sysfunc(findc("$MC_File","/")),'/')));
&put &folder;
/*use prompts to move/copy folder*/
x = "cd ";
sys task "&MC Choice. &MC File. &MC Dest. ";wait status=Status Opt;
/* Displays the Home folder contents after move */
x = "cd &Folder. ";
x = "ls -go > &MC Dest./Source.txt";
RUN;
/* Displays the destination folder contents after move */
x = "cd &MC Dest. ";
x = "ls -go > &MC Dest./Target.txt";
RUN;
/*Deletes log files that are used for Results*/
filename dlt1 ("$MC Dest./Source.txt");
filename dlt2 ("$MC Dest./Target.txt");
data null;
rc = fdelete('dlt1');
rc = fdelete('dlt2');
run;
filename dlt1 clear;
filename dlt2 clear;
```
Combining SAS® functions, prompts and log output. The example below uses SAS® in an AIX UNIX server environment:

1. This is an example of what the end user would see in SAS®. *Combination of the Prompts from slide 6*
2. Final results tab of the stored procedure shown in step 1