

A Get the data without SAS/Connect?

And the PC SAS user does HOW LITTLE? With a shoe string budget?

Zeke Torres, 1016inc.com, Chicago, IL

ABSTRACT

Distribute SAS (almost any data) not only to your own machine but various users you may support throughout your organization even if SAS/Connect is available or not. SAS/Connect was available as in this case, however the client specifically requested to stop giving PC SAS users access to SAS/Connect and the Host CPU. The main intent was to keep the ad-hoc SAS report(s) submitted by PC SAS users from draining the CPU resources but not put constraints on their ability to produce reports their departments required. I was able to come up with a solution for my client by using simple tools available with PC SAS and FTP.

INTRODUCTION

We want our user to get data (SAS) to work with on their PC. Minimal input from the user. Connect to our HOST (Server/Mainframe) pick up specific data, leave a footprint, exit the system. Once the data is on the users PC they can use it without connecting to the host for that data. We will provide the user with the ability to enter their password/id and if appropriate what kind of file or data they will be picking up.

As the administrator of this site we also want to maintain this 'process' for multiple users and we want the core of the code this process uses under our supervision as much as possible. We want the user to constantly obtain from the host the most current code this process works with.

A friendly FYI is that with the more knowledge the user has of SAS and FTP the more they can either customize this process to suit their needs or even jeopardize the process if they experiment and interfere with other users. Make sure to establish the appropriate read/write functions at your site or situation.

SECURITY AND PROTOCOL

Review the password and userid process with your site administrator. This client provided users with tailored id's and passwords, which allowed security to direct them only to the files we wanted and days or times that were applicable for their use. This presentation will make every attempt to caution you on potential security risks but this will be the only warning documented in this abstract. As an additional warning and precaution you should not deploy this code or similar concept in your organization or site without consulting with your manager and or security administrator first. *See WARNING

PC AND HOST INFORMATION

In this example the HOST is an IBM os/390 MVS mainframe. PC is operating on a WIN9# machine. FTP has been used via the DOS or command prompt on the PC. SAS on the PC is version 6.12 and higher. SAS on the HOST is not required for this example or your site except if you need to have SAS datasets created there.

SPACE AND MEMORY REQUIREMENTS

Consider space requirements on either HOST or PC. Some users have limited hard drive space. Sending large datasets can cause problems if the users machine is running low on space. If you send data to the HOST from the users pc consider what and how much your transferring. Finally even if the user has the data on their PC the RAM or memory may not be enough to sufficiently

process their SAS code or programs along with the DATA. We determined what users demographics and usage levels were documented and tailored.

PC SAS AND FTP

1. User will enter:
 - Id and Password
 - What they want delivered
2. User starts the SAS code, which will initiate the process.
 - Sets up the information needed by FTP to connect the user.
 - Construct the commands for FTP to complete the required tasks.
 - Initiate the 1st of "7" FTP commands using the constructed command file.
3. This Initial FTP obtains the core code from the host.
 - The host supplies new code to the user's computer.
 - We maintain this 'core code' on the host.
 - With this code in this example we will:
 - Construct more FTP commands/requests tailored for the user.
 - Construct SAS code to be executed on the user SAS Session (PC).
 - Construct clean up code where applicable on either the PC or host.
 - Send TO or FROM the host the files we just constructed.
4. The code that was picked up from the host is executed on the PC SAS Session and will then initiate more code. This code includes additional FTP steps and SAS code.
5. The process completes when all the codes or FTP commands are done.

SAS CODE USED

LIST OF CODE FOUND IN THIS EXAMPLE and where this code will be (USER PC and/or HOST).

USER PC	HOST
MAINSAS.TXT	-
FTP0.INI	-
FTP1.INI = FTPX.INI	SASFTP0.SAS
-	SASFTP1.SAS
DONE.SAS	SASFTP2.SAS

- MAINSAS.TXT and FTP0.INI need to be sent to the user and properly configured by you. These will only be on the users PC.
- FTP1.INI, FTPX.INI, and SASFTP0.SAS should start out as identical copies of each other. This code will initiate the 1" FTP and get SASFTP1.SAS.
- SASFTP1.SAS directs the users PC SAS session to correctly copy, rename and or send/receive the applicable code or data you set up for the users. This is with more FTP syntax created by this code.
- DONE.SAS and SASFTP2.SAS are identical. DONE.SAS will be created as a copy of SASFTP2.SAS by SASFTP1.SAS upon execution. Its important as a clean up but its up to you how far you'll take this 'clean up' idea.

To correctly initiate this process with a user you'll need to distribute the following files: MAINSAS.TXT, FTP0.INI, FTP1.INI, FTPX.INI. To be safe I send DONE.SAS also. You must ensure that the correct copies of SASFTP0, SASFTP1, SASFTP2 are on the HOST and accessible to the code set up for the users.

THE CODE

Lets review the elements of this process and what each accomplishes. Once they begin to work together the process becomes a cycle and with each execution the user obtains new or revised code from the host. This allows you to adapt this process as you go and deploy it with minimal effort.

MAINSAS.TXT

```
%LET MYID=herehere;
%LET MYPW=password;
%LET TYPE=ALL;
%include 'c:\xasdata\ftp0.ini';
**MAINSAS.TXT - END
```

This code is located on the Users machine. In MAINSAS.TXT we assign values to the USERID, PASSWORD and an additional variable called "type". The purpose of this code is to populate within the program the information the HOST will need to allow the user to connect. The variable called 'type' will determine what kind of information the user will pick up. Monthly, Daily, Weekly, you decide.

FTP0.INI

```
options noxwait nonotes;
x 'copy c:\xasdata\ftpx.ini c:\xasdata\ftp1.ini';
%include 'c:\xasdata\ftp1.ini';
**FTP0.INI - END
```

This code is located on the Users machine. Here we turn off options that are required to avoid the SAS "x" command window to wait for replies or input from the user. The NONOTES option will keep the SAS log from collecting information like the user id and password.

An 'x' command copies the file called FTPX.INI into FTP1.INI. This allows us to update the code. FTPX.INI is from the last time this code was executed. Notice how we invoke that same SAS code in the next line with %INCLUDE.

BRIEF RECAP

The user will basically encounter only MAINSAS.TXT. They will execute that code and the rest will(should) be transparent to them. Unless the user is savvier with SAS skills they would not know about FTP0.INI or any of the other code you're about to review.

CAUTION: It is important, critical to the process is that the user maintain the file or directory path you have set up for their machine via the code you are designing. It is critical that they avoid deleting folders or renaming them. And if you should ever get phone calls with problems, this is generally the main problem I've encountered.

FTP1.INI (FTPX.INI OR SASFTP0.SAS)

NOTE: The first 3 numeric digits in this document are line numbers intended to facilitate this abstract they have no purpose in the actual code.

```
001 OPTIONS NONOTES COMPRESS=YES;
002 x 'del c:\xasdata\ftpx.ini';
003 FILENAME OUTPARMA 'c:\xasdata\ftpparma.dat';
004
005 %LET XTARGET=TODAY();
006
007 DATA YOURPW;
008   LENGTH YOURPW $8.;
009   LENGTH YOURID $8.;
```

```
010   YOURPW=&MYPW;
011   YOURID=&MYID;
012   LENGTH DATAAT $3.;
013   LENGTH TESTTYPE $3.;
014   YOURID=UPCASE(YOURID);
015   IF YOURID IN : ('e') THEN DATAAT='BAA';
016   IF YOURID IN : ('USER5487') THEN DATAAT='BAA';
017   IF YOURID NOT IN : ('e','USER5487') THEN DATAAT='ALL';
018   TESTTYPE=&"type";
019   IF YOURID IN : ('e','USER5487') & TESTTYPE IN : ('a')
020   THEN DATAAT='ALL';
021   DATAAT=UPCASE(DATAAT);
022   XDATE=&XTARGET; FORMAT XDATE yymmdd10.;
023   XTIME=&XTARGET; FORMAT XTIME MMSS5.;
024   MFILE=YOURID!!".D"!!(PUT(XDATE,yymmdd10.));
025   PUT MFILE;
026   PUT TESTTYPE;
027   MFILE=COMPRESS(MFILE);
028   MFILE=UPCASE(MFILE);
029   RUN;
030
031   DATA _NULL_;
032   SET YOURPW;
033   FILE OUTPARMA;
034   PUT @1 YOURID $;
035   PUT @1 YOURPW $;
036   PUT @1 "get 'hlqfr.sasftp1' c:\xasdata\sasparm1.sas";
037   PUT @1 "get 'hlqfr.sasftp0' c:\xasdata\ftpx.ini";
038   PUT @1 "bye";
039   RUN;
040
041   x 'ftp -s:c:\xasdata\ftpparma.dat 333.12.345.67';
042   %include "c:\xasdata\sasparm1.sas";
043   %include "c:\xasdata\done.sas";
044   x 'del c:\xasdata\done.sas';
```

Line 001 - Set up options. Here NONOTES is repeated and Compress=yes. This is optional to you. I don't mind the redundancy.

Line 002 - We use the 'x' to delete the FTPX file we just used and this clears the way for a fresh one we'll copy in a few steps. We want to make sure that once this gets executed it will not get re-executed unless its 'new' code obtained from the host.

Line 003 - We are making the file or data for the ftp instructions, which are going to initiate all of this. This is for the first set of instructions sent to the host.

Line 005 - We want the date from the users machine. It is not critical but I set up a file, which gets placed on the host, so I know when and who picked up data from the host.

Line 007 thru 029 - The dataset "YOURPW" is created. Here is where we begin to determine who we are dealing with and what they will obtain via the parms they entered in 'mainsas' program. This also starts the process of constructing the required FTP syntax which will be used later.

Line 031 thru 039 - This is a simple null step. We are basically using the YOURPW dataset to construct the output file, which is intended to be FTP syntax.

- A. put the userid.
- B. put the password.
- C. put a GET statement.
- D. put a GET statement.
- E. put a BYE statement.

Essentially we have just entered the text/syntax we would have typed in a prompted FTP session. With this step we are putting this into a file, which we will execute shortly.

Line 041 - We will execute the FTP session via 'X' in SAS and use the file which has the syntax we just constructed. Remember this FTP process has to complete before the SAS session continues with additional SAS code or the rest of the program(s). At this point the FTP process is working to "GET" code, data, etc.

Line 042 - We now initiate SASPARM1.SAS. But this is actually from the HOST. We just copied it with the FTP process we initiated in *Line 041*. With these steps we have control of the users machine for a brief moment. Only when SAS is finished with that file/code SAS will we go to *Line 043*.

Line 043 - Initiates the DONE.SAS code. Again it's the same concept as SASPARM1.SAS only its devoted to clean up. I chose to have this code perform file deletes or clean up work. Remember that SAS will not go to *Line 044* in this program until it completes the SAS code in DONE.SAS.

Line 044 - Clean up the clean up file? Redundant? Maybe, but why leave files where they are not needed. More importantly, if the code is left behind it can start to become outdated as you make additional modifications to your process. By removing or deleting files once they are done in the loop you will ensure that the code is 'fresh' and updated via the entire process completing.

RECAP OF FTP1.INI

FTP1.INI will basically take the information collected by the previous sets of code and incorporate it into the initiating FTP syntax which will GET from the host more instructions, code or data to facilitate the next FTP process which will then actually get the target code or data you intend to distribute.

Remember FTP1.INI is actually a copy of FTPX.INI which we obtained the last time this process was executed and completed. It will obtain SASPARM1.SAS from the HOST and execute it in the USERS PC SAS session. Hence with the correct design and appropriate code along with good intentions you can do just about anything with the users machine, files, data, etc. *See Security and Protocol.

SASPARM1.SAS -

```
001 FILENAME OUTPARM1 "c:\xasdata\ftpparmx.dat";
002 FILENAME OUTPARM2 "c:\xasdata\ftpparmy.dat";
003 FILENAME OUTPARM3 "c:\xasdata\ftpparmz.dat";
004
005 DATA _NULL_;
006 SET YOURPW;
007 DATEX=TODAY();
008 DATEY=DATETIME();
009 FILE OUTPARM2 LRECL=400; /* ftp parm y */
010 PUT @1 "-----";
011 PUT @1 " SYSDATE/PCDATE=" DATEX YYMMDD10.;
012 PUT @1 " USERID=" YOURID;
013 PUT @1 " TESTTYPE=" TESTTYPE;
014 PUT @1 " DEFAULT TYPE=" DATAAT;
015 PUT @1 " DATE/TIME=" DATEY DATETIME16.;
016 PUT @1 " STAMP=" MFILE;
017 PUT @1 " FTP ATTEMPTED";
018 RUN;
```

SASPARM1. SAS - Continues

Line 001 thru 003 - There is nothing exotic about these lines. We are simply defining the files where we will store either the syntax or data for either the FTP commands or data, which tells us who/what/when they connected.

Line 005 thru 018 - We are using NULL to create data going to one of the 3 files we just allocated. This is the file which contains who/what/when tried this. We are simply creating a receipt or

stamp we will append to a file we keep on the host. I use this to track when the last time someone used this process and what 'basic' information our site needed.

In my original abstract I mentioned we wanted to centralize reporting and know what code users were submitting. This kind of file is important because you now know when the last time the user picked up data and what they requested. Certainly you know what is available since you designed it. If you wish, a file like this could be left at the end of the process. You could use this process to send copies of user SAS Logs from PC SAS to the HOST and see what users are doing.

For any file you put, send, append etc. be sure to keep the LRECL in the code consistent with the LRECL on the host. Your append or what you decide to do with this part of the process is important since the FTP syntax might fail if the file structures are different and you did not code for this variation. You might also have difficulty isolating this problem since what messages you get from FTP might be limited based on your site setup.

SASPARM1. SAS - CONTINUED

```
019
020 DATA _NULL_;
021 SET YOURPW;
022 FILE OUTPARM1; /* ftp parm x */
023 WHERE YOURID IN ('easy1','user5487') AND DATAAT IN
('BAA');
024 PUT @1 YOURID $;
025 PUT @1 YOURPW $;
026 PUT @1 "bi"; /* bi stands for BINARY;
027 PUT @1 "get 'hlqfr.fonxport.sasdumpc' c:\xasdata\sasdata1.dat";
028 PUT @1 "bye";
029 RUN;
030
031 DATA _NULL_;
032 SET YOURPW;
033 FILE OUTPARM1 mod; /* ftp parm x */
034 WHERE YOURID IN ('easy1','user5487') &
DATAAT='ALL';
035 PUT @1 YOURID $;
036 PUT @1 YOURPW $;
037 PUT @1 "bi";
038 PUT @1 "get 'hlqfr.fonxport.sasdumpa' c:\xasdata\sasdata1.dat";
039 PUT @1 "bye";
040 RUN;
041
042 DATA _NULL_;
043 SET YOURPW;
044 FILE OUTPARM1 mod; /* ftp parm x */
045 WHERE YOURID IN ('user4444','user3333','user3322');
046 PUT @1 YOURID $;
047 PUT @1 YOURPW $;
048 PUT @1 "bi";
049 PUT @1 "get 'hlqfr.fonxport.sasdumpe' c:\xasdata\sasdata1.dat";
050 PUT @1 "get 'hlqfr.fonxport.sasdumpf' c:\xasdata\sasdata2.dat";
051 PUT @1 "bye";
052 RUN;
053
```

SASPARM1. SAS - Continues

Line 020 thru 053 - This part where you are capable of making this process do some really cool things. I will explain the concept more than the presented detail since this is going to become specific to your site application. First the NULL steps. Where we set the YOURPW dataset we originally started to work with. I am now checking which user I have and what file they have

requested. It is at this point where your organization of HOST files and this SAS code must be correctly set per user etc. Lets remember that each one of these steps is exploiting the FTP syntax via the SAS PUT process within the NULL statement and the "X" command prompt in PC SAS. In more simple terms: I've used PC SAS code to make FTP code to get what the user wants via SAS Code copied from the HOST.

So we have checked the USER id for predetermined users allowed to use this code on the HOST as 'authorized'. We let the code check if the value of the next variable DATAAT would allow the NULL step to populate our FTP PARM file. Where all things 'match' the NULL step will appropriately construct the FTP syntax.

In this very rigid example I've explicitly determined two types of users. All others would have this process crash upon them since they are not defined in this step or any of the other steps. My example does not go into the process of coding a nice good bye for the unauthorized user.

SASPARM1. SAS – CONTINUED

```
054 DATA _NULL_;
055 SET YOURPW;
056 FILE OUTPARM3;
057 PUT @1 YOURID $;
058 PUT @1 YOURPW $;
059 PUT @1 "append c:\xasdata\ftpparmy.dat
'hqfr.fortport.xport.ftpuse"; ** this line wraps around in this document;
060 PUT @1 "get 'hqfr.sasftp2' c:\xasdata\done.sas";
061 PUT @1 "bye";
062 RUN;
063
064 X 'ftp -s:c:\xasdata\ftpparmx.dat 333.12.345.67';
065 X 'ftp -s:c:\xasdata\ftpparmz.dat 333.12.345.67';
066
```

SASPARM1. SAS – Continues

Line 054 - 063 - This data null step will create the FTP parameters to send over our who, what, when file and get the code which will carry the final clean up code. *****NOTE: Line 059 Wraps in the abstract and should not be on 2 lines in your code.**

Here you can also take advantage of the FTP features presented. You can send files from the users machine to the host. Say the SAS Logs on the users PC or the SAS code or a SAS dataset, Lib, catalog, transport files you name it. Here we are taking our file and appending it to the HOST file, which is the only read/write file available to the user in this process.

Line 064 - 065 - We are now using the "X" command feature in PC SAS to execute FTP and the syntax we just constructed. What this will do is initiate that communications medium and 'send or get' our files. Notice how we do this in 2 steps for our current example. The first will get information. The second will get the clean up SAS code and deliver our time, stamp file.

SASPARM1. SAS – CONTINUED

```
067 DATA _NULL_; FILE OUTPARM1; PUT @1 "-"; RUN;
068 DATA _NULL_; FILE OUTPARM2; PUT @1 "-"; RUN;
069 DATA _NULL_; FILE OUTPARM3; PUT @1 "-"; RUN;
070 DATA _NULL_; FILE OUTPARMA; PUT @1 "-"; RUN;
071
072 PROC DATASETS LIB=WORK; DELETE YOURPW;
073 RUN;
074 LIBNAME SASDATA1 XPORT 'c:\xasdata\sasdata1.dat';
075 LIBNAME SASDATA2 XPORT 'c:\xasdata\sasdata2.dat';
076 LIBNAME YOURLIB1 'c:\xasdata\';
077 LIBNAME YOURLIB2 'c:\xasdata\m\';
078
```

```
079 PROC COPY IN=SASDATA1 OUT=YOURLIB1; RUN;
080 PROC COPY IN=SASDATA2 OUT=YOURLIB2; RUN;
081
082 FILENAME OUTPARM1 "c:\xasdata\sasdata1.dat";
083 FILENAME OUTPARM2 "c:\xasdata\sasdata2.dat";
084
085 DATA _NULL_; FILE OUTPARM1; PUT @1 "-"; RUN;
086 DATA _NULL_; FILE OUTPARM2; PUT @1 "-"; RUN;
087
088 PROC CONTENTS DATA=YOURLIB1._ALL_;
089 TITLE1 "THIS IS FOR THE LIB ON YOUR PC LIB 1";
090 RUN;
091
092 PROC CONTENTS DATA=YOURLIB2._ALL_;
093 TITLE1 "THIS IS FOR THE LIB ON YOUR PC LIB 2";
094 RUN;
095 TITLE1 " "; TITLE2 " ";
    SASPARM1. SAS – END
```

Line 067 - 072 - I initiate Null steps to "put" information (1 byte) back into the original parm files. I am making an attempt to clean out the files from values they held for the FTP process. Now that the process is completed I want to start the clean up process. Proc datasets is used to delete the YOURPW dataset.

Line 074 - 095 - For this example we have just used, our user has picked up data from our host. We are using COPY to transfer the data to and from our host. At the users machine we want to have the data ready for them. We initiate the LIBNAME statements.

The transport file is delivered into a full SAS dataset. We use PROC COPY. The users SAS session will do the work. The files will now be ready for the user.

We now want to cleanup the transport file. They can be space wasters so we want to ensure they are gone. FILENAME/Null step suffices for now.

A PROC CONTENTS on the data for the user. This works great for the first few times, but I suggest you consider if your user actually needs this or not, especially with really ugly datasets with hundreds of variables.

Remember, this code is NOT executed yet. It is simply going to be sent to the USERS PC for the actual execution or processing. This is the code you should ensure accomplishes what you want.

1. Tailor data to specific users
2. Control which user obtains which files
3. Track which users are using files and at what frequency

If we go to the rest of the example we'll see how the cycle is complete but now its time to wrap things up.

DONE. SAS

```
001 FILENAME OUTPARM0 "c:\xasdata\sasparm1.sas";
002 DATA _NULL_; FILE OUTPARM0; PUT @1 "-"; RUN;
003
004 OPTIONS NOXWAIT;
005 x 'del c:\xasdata\sasparm1.sas';
006 x 'del c:\xasdata\ftpparma.dat';
007 x 'del c:\xasdata\ftpparmx.dat';
008 x 'del c:\xasdata\ftpparmy.dat';
009 x 'del c:\xasdata\ftpparmz.dat';
010 x 'del c:\xasdata\sasdata1.dat';
011 x 'del c:\xasdata\sasdata2.dat';
    DONE. SAS – END
```

DONE.SAS just cleans up the elements we used, created and makes sure we don't leave much behind.

CHECKLIST

There has been lots of information presented here. I want to provide you with a checklist of things you'll need to remember, do or set up. In no particular order.

1. Determine if security at your site has a problem with this concept. If so, try to work with them on a solution.
2. Establish that all of the users can connect to your host via FTP and can access 'a directory' or set of files you can administer. Some sites keep this kind of access on a user by user basis.
3. Ensure the users are working on similar operating systems. Example Win 9x or that you won't have a stray MAC user. Crazy example... but possible.
4. Determine how you want to distribute the code for the first time and train the users or roll this out. I opted for a user by user basis. Helping them unzip a single file I sent them and making sure the correct folders and path's were set up on their machines.
5. Ensure you users can get a 'new' setup file. I found a few users who wanted to experiment and trampled on the code or process. I made sure to provide them with the means to obtain a new zip file and make sure they can get back on track. Don't think this won't happen to you.
6. Have each user run this once with you on the phone. I set up test files on the host side for users to play with. It works well when you are rolling this out and when you do "go live" you know that everyone is up and running.
7. Don't just test on your PC. Test on another PC and then another with some other version of windows etc. And ensure that it works with the versions of SAS found at your site. The code here should. It was designed to suffice that requirement. However with newer versions of SAS or modifications of configuration files, site standards, security, who knows. You need to see how much of this is possible where you are.
8. Is your PC on? Is the host on?
9. Is your user id allowed access to the host? Remember different users, ids, security levels, passwords etc.
10. Is your password valid? Hey passwords do and will expire, well they should. So your user or code will need to accommodate this fact of life. In our example the user just crashes.
11. Are you using the correct IP address or DNS name? Your site might use numbers for IP addresses (333.222.333.22) or a name like "ipsite". But if you don't know which one is for your 'host' you'll need to find out to get this to work. This example assumes you're an admin or someone with the applicable authority and responsibility to have knowledge of this information. Chances are if you are not immediately aware of this information you really should work with someone who has the applicable authority to help you set this up following your sites standards.
12. Are the folders/directories or paths in your code set up correctly? Ok, in my example I have specific folders and directory paths set up which are applicable to the site and user PC's. An important item is that the user side is designed to have 1 standard for all users. It makes it easy to work with users and deploy more code or more complex code as the need surfaces. Finally, it makes it easy to collect information from the user since we can manipulate where we store data or files we need collected on the host.
13. Did you set up the files correctly on the users PC? Ok, so you sent the files to the user and they opened them. Make sure they unzipped correctly or were placed in the correct drive or folder. Make sure that they can ftp to the host and have the correct authority to read or write to the host machine or files. This is just as important.
14. Did you set up the files correctly on the HOST? This is essential. Remember the user is dependant on the host files or code being ready to get picked up when they start this on their machine. If the code is not synchronized it will result in either a quick failure or a prolonged process of trouble, which will lead to failure. If you ensure that the files stored on the host are set up with the correct read/write and allocated correctly then this should not be a problem.

15. Space, the final frontier. You want to collect the SAS log of everyone to know what they run. Be careful. Remember, many times users are getting 'errors' in their everyday code. We're talking lots of errors. This means if you decide to track the user via the SAS log your in for a big file to look at. I urge you to use this concept carefully. Not only could you lock up the user's PC but you could find the host space getting used up quickly if you do not take this into consideration. The SAS log's will be kept on the users machine until it is sent to you with this code. Then it will only get deleted via the instructions you send to the user. If you do not make sure this process works well you will have an angry user calling you.

BASIC FTP INFORMATION

What is FTP? The FTP (File Transfer Protocol) utility program is commonly used for copying files to and from other computers. These computers may be at the same site or at different sites thousands of miles apart. FTP is a general protocol that works on UNIX systems as well as a variety of other (non-UNIX) systems.

The local machine refers to the machine you are initially logged into, the one on which you type the ftp command. The remote/Host machine is the other one, the one that is the argument of the ftp command. Most operating systems and communication programs now include some form of an FTP utility program, but the commands differ slightly between them. The following explanations and alphabetical list of commands refers to the common FTP utility program as provided on a UNIX machine. Check the documentation for your own machine to determine the comparable commands.

Getting Started To connect your local machine to the remote machine, type: `ftp hostname`

where HOSTNAME is the full machine name of the remote machine or host, e.g., yoursite.com. If the name of the machine is unknown, you may type `ftp host I P number`

where host I P number is the net address of the remote machine, e.g., 333.44.55.331. In either case, this command is similar to logging onto the remote machine/host. If the remote machine has been reached successfully, FTP responds by asking for a loginname and password. When you enter your own loginname and password for the remote machine, it returns the prompt ftp>

Common FTP Commands

?	To request help or information about the FTP commands
ASCII	To set the mode of file transfer to ASCII (this is the default and transmits seven bits per character)
binary	To set the mode of file transfer to binary (the binary mode transmits all eight bits per byte and thus provides less chance of a transmission error and must be used to transmit files other than ASCII files)
bi	Same as binary command
bye	To exit the FTP environment (same as quit)
cd	To change directory on the remote machine
close	To terminate a connection with another computer
delete	To delete (remove) a file in the current remote directory (same as rm in UNIX)
get	To copy one file from the remote machine to the local machine
get ABC DEF	Copies file ABC in the current remote directory to (or on top of) a file named DEF in your current local directory.
get ABC	Copies file ABC in the current remote directory to (or on top of) a file with the same name, ABC, in your current local directory.
help	To request a list of all available FTP commands
lcd	To change directory on your local machine (same as UNIX cd)
ls	To list the names of the files in the current remote directory
mkdir	To make a new directory within the current remote directory
mget	To copy multiple files from the remote machine to the local machine; you are prompted for a y/n answer before transferring each file
mget *	Copies all the files in the current remote directory to your current local directory, using the same filenames. Notice the use of the wild card character, *.
input	To copy multiple files from the local machine to the remote machine; you are prompted for a y/n answer before transferring each file
open	To open a connection with another computer
put	To copy one file from the local machine to the remote machine

pwd To find out the pathname of the current directory on the remote
machine
quit To exit the FTP environment (same as bye)
rmdir To remove (delete) a directory in the current remote directory

CONCLUSION

This example is functional because it exploits the features of PCSAS and the 'X' command along with FTP. By combining these into a functional SAS application we can provide a user with a flexible method of obtaining data from a remote host. As the administrators of this process we can manage it easily by simply adding code that will create a LOOP in the overall process and constantly provide the users PC (client) with new code or fresh code which has the modifications your site would require over time. Once the appropriate security and design is worked out you'll find this method or concept functional to help reduce user running PCSAS via SAS Connect to your host.

REFERENCES

PAPERS REFERENCED

FROM SUGI 27, PAPER 6-27

<http://www2.sas.com/proceedings/sugi27/p006-27.pdf>

Reading From Alternate Sources: What To Do When The Input Is Not a Flat File

Author

Michael Davis,
Bassett Consulting Services
North Haven, Connecticut

WEB SITES REFERENCED

<http://www.sas.com>

ACKNOWLEDGMENTS

The author would like to thank the following individuals for their invaluable assistance in the preparation and technical review of this paper.

Dana Laskowski
Transamerica
Hoffman Estates IL

Sally Ernst
SAS Consultant
Kansas City KS

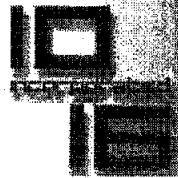
Chris Williams
Allstate Financial
Prospect Heights IL

Steve First
System Seminar Consultants
Madison WI

AUTHOR'S CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the author at:



Ezequiel 'Zeke' Torres
1016Inc.com
Chicago, Illinois 60630
Work Phone: (773)467-8492
Fax: (773)467-8492
Email: info@1016inc.com
Web: www.1016inc.com

FINAL CONSIDERATIONS

Please note that the code supplied in this paper is designed only to illustrate the concepts being discussed and may need to be modified to work in other applications and/or environments. The author of this paper does not support modified code.

WARNING

This code and the concept and/or your ability to execute it does not indicate you are complying with the security protocols or standards your site, office, place of work have established. The author makes no warranty, assurances implied or otherwise that this method is in anyway 'secure' or complies with your site standards or protocols. The code, information and concept is presented 'as is'. The use and/or manipulation of USER ID's, Passwords and similar confidential user information can be considered ILLEGAL and against policy or standards at your location and in some cases by public agencies, communications companies, telephone, internet service providers and/or law enforcement agencies. You are responsible for taking that into consideration before applying this model in any way.

Use the information, concept presented here at your own risk.