Paper 92-28

An Automated MS PowerPoint Presentation Using SAS ®

Rick Allen, CitiCapital, Irving, TX

ABSTRACT

Monthly reports are a fact of life for many SAS programmers. In many cases the reports must be in the form of MS Excel spreadsheets or PowerPoint slides. Creating these reports each month consumes valuable programming resources. This paper presents a method to conserve these resources by automating the process of building a set of MS PowerPoint slides from information contained in SAS data sets.

INTRODUCTION

Each month our department produces an overview report for Senior Management on each of CitiCapital's four main commercial lending divisions. The final reports are prepared in MS PowerPoint to provide a consistent "look and feel" across the divisions. Additionally, it provides a simple method for the user to extract or print individual pages from the report for their own purpose.

The majority of the reports consist of MS Excel charts and tables that are automatically included via links within PowerPoint. The source data for the Excel files is, in turn, contained within .CSV files that are built using SAS programs and data sets running on UNIX. The SAS programs are automatically scheduled to run at the appropriate time each month, when all prerequisites are met.

The entire process requires no manual intervention other than some "fine tuning" of the Excel or PowerPoint files to refresh the links, to add monthly commentary or to insert manually entered (offline) data.

THE CRONTAB SCHEDULER

The UNIX operating system contains a rudimentary built-in scheduling function known as Crontab. Crontab is not a "full function" scheduling package, however, for this project it suits the purpose just fine and it can be managed entirely by the SAS development staff.

A "how-to" on the subject of Crontab is beyond the scope of this paper. Simply put, Crontab will allow you to schedule a UNIX command (in our case, initiate a batch file that starts SAS and runs the main scheduling routine) at any particular minute, hour, day, day of week, or month. Every UNIX user can create a personal Crontab file.

Here is the Crontab entry in my file that runs our SAS-to-Power Point automation:

```
15,45  7-18  1-12  *  1-5   /…path…/schedule.bat
```

In this Crontab command I am telling UNIX to run my batch file (monthly_schedule.bat) when the time and date match ALL of these criteria (the corresponding Crontab operand is in parentheses):

Exactly 15 and 45 minutes past the hour (30)
Any hour between 7 AM and 6 PM (7-18)
Any day of the month from the 1st to the 12th (1-12)
Any month of the year (*)
Monday through Friday only (1-5)

SCHEDULING OVERVIEW

Our monthly SAS jobs require information from several different production financial systems such as Credit Application, Finance and Accounting. At various times during the first 10 days of each month, new SAS (or text) files with the latest information from all of these systems become available on our UNIX server. The SAS Monthly Scheduler runs once every hour during these two weeks and performs these functions:

- Check to see which of this month's files are now available
- Check to see which SAS job steps this scheduler has already submitted
- Check to see which SAS job steps have completed
- Run any SAS jobs which meet all of its prerequisites:
    1. Production files available
    2. Previous dependent SAS job(s) have run
    3. This SAS job has not already been started
- Send email(s) to the development staff when any job starts or completes
- Add an entry to the monthly activity log file for each automation event

THE MAIN SCHEDULER ROUTINE

11 separate SAS jobs make up our complete monthly schedule. The scheduler will submit the jobs individually when that job's criteria have been met. Here is a sample of the SAS code to test for all required prerequisites and then submit a job:

```
%include '..path../monthly_triggers.sas';

%macro startjob(sasfil,msg);

  proc printto; run;

  proc printto new log =
     "/….path…./&sasfil..log"; run;

  %include "/….path…./&sasfil..sas";

  proc printto; run;

%mend startjob;


%macro xrun_3a; /*one of these per job step*/

/* run step3a if step2b complete, xpress
credit file is available and step3a not
already started */

(continued on next page)


%if &step2b_completed and &xcredit and not
 &step3a_started %then %do;
     %startjob(step_3a,
                Step 3a(Xcredit reporting));
%end;

%mend xrun_3a;

%xrun_3a;
```

IMPORTANT! The scheduler assigns a unique log dataset for each job step. Check the SAS code in each of your jobs to ensure that there are no "DM 'clear log'" commands or PROC PRINTTO; RUN; statements. These will clear or reset the log file. PROC PRINTTO; RUN; should be replaced with PROC PRINTTO PRINT=PRINT; RUN;

MONTHLY TRIGGERS (FILE AVAILABILITY CHECKING)

In most cases the SAS "If exist" function is used to determine which files are available and which SAS job steps have started or completed. All steps can be tracked in this method because the monthly scheduler creates a small SAS file when any job starts or completes. Here is a sample of the SAS code used to determine if a new monthly file has been created:

```
%global cdb;

/* note: &lm is one of our system variables;
   it is the current cycle year/month (yymm)
*/

if exist ("cdbtrd.cdb&lm", 'data')
    then call symput ('cdb',1);
    else call symput ('cdb',0);
```

Note that the file name contains the year and month of the current production cycle and is therefore a new name each month. "If exist" will always determine if a new monthly file is available.

Similarly, a given SAS job step can be determined to have "started" or "completed" by the existence of a small dummy file (see SET SWITCH MACRO pg. 3).

```
if exist ('steps.started_step1', 'data')
     then call symput ('step1_started',1);
     else call symput ('step1_started',0);

  if exist ('steps.completed_step1', 'data')
     then call symput ('step1_completed',1);
     else call symput ('step1_completed',0);
```

Some monthly files are "flat" files (not SAS format) and as a result the "if exist" function cannot be used. In these cases I redirect a UNIX directory list command ("LS") to a dummy file and then read that file with SAS until I find the new file name, as follows:

```
%global gainloss;

x 'cd /..path..';
x "ls -lat gnls* > /…path…./gainloss.dir";

filename gldir '/…path…/gainloss.dir';

data _null_;
   call symput('gainloss',0);
   infile gldir;
   input security $ units id $ group $ size
         month $ day timeyear $ filename $;
   if size ge 50000 and filename = "gnls&lm"
   then do;
        call symput('gainloss',1); stop;
   end;
run;
```

Another complication, some SAS files are the same name each month, they are simply refreshed with new data. In this case "If exist" will not suffice to tell us if this month's file is available. For these files I use the **%sysfunc** and **attrn** functions to find out more information about the file such as the most recent modification date. As an additional precaution I also "sanity check" the number of variables and number of observations to ensure we have a valid file.

IMPORTANT! Do not use the attrn CRDTE function to obtain the modification date; use MODTE instead! CRDTE will **not** always reflect the latest date the file was changed, for example if the file is refreshed via a UNIX 'mv' or 'cp' command to copy the file to the directory.

Here is a sample of the SAS code to check for these types of files (see SAS Macro Language: Reference Version 8, pp. 252-253 for details on the "obsnvars" macro example):

```
libname xcr '…path…';

%macro obsnvars(ds);

   %global dset nvars nobs crdte xcredit;

   %let dset = &ds;
   %let dsid = %sysfunc(open(&dset));

   %if &dsid %then %do;

       %let nvars =
            %sysfunc(attrn(&dsid,NVARS));
       %let nobs  =
            %sysfunc(attrn(&dsid,NOBS));
       %let crdte =
            %sysfunc(datepart(%sysfunc
                     (attrn(&dsid,MODTE))));

       %let rc = %sysfunc(close(&dsid));
   %end;

   %else %do;
      %let nvars = 0;
      %let nobs  = 0;
      %let crdte = 0;
   %end;

%mend obsnvars;

%obsnvars(xcr.transp);

/* set a global indicating the file is
available if created in the same month as
today's date, or if it was created on the
last day of last month ONLY if on a Sunday.
This is allowed since the files are not
available for update on Sundays. */

data _null_;
   tday = date();

   plus1 = intnx('day',&crdte,1);
   if month(plus1) ne month(&crdte)
      then lastdom=1;
   else lastdom=0;

   if &nobs ge 1000 and &nvars ge 50 and
   (month(&crdte) eq month(tday) or
   lastdom eq 1 and weekday(&crdte) eq 1)
       then call symput('xcredit',1);

   else call symput('xcredit',0);
run;
```

THE JOB (STEP) CODE

The code to run each of our 11 SAS jobs is "%included" by the startjob macro (see MAIN SCHEDULER ROUTINE, pg.1). The SAS job code sets a "started" switch, then %include's the individual SAS programs that make up the job step, and then sets a "completed" switch after the programs have run.

```
%include '/….path…/set_switch.sas';

%setswitch(Step3,Started);

%let lib =/….path…/;
%include "&lib.my_program_name_1.sas";
%include "&lib.my_program_name_2.sas";
* etc….;

%setswitch(Step3,Completed);
```

THE SET SWITCH MACRO

The set switch macro receives a parameter of "started" or "completed" from the SAS job code (above). A small file is created to represent the "started" and "completed" functions. These are the files that the scheduler's "If exist" function used to determine the step status (see MONTHLY TRIGGERS, pg. 2).

The set switch macro can also receive a "reset" parameter from the "reset at end of month" routine, which instructs it to delete both the started and completed files for that step (see RESET AT END-OF-MONTH, pg. 4).

The set switch macro assists the SAS developer in tracking the status of the SAS automation system by automatically sending emails when any job step starts or completes, **with the SAS return code** (&syscc). The macro also records an entry in a global event file.

```
libname steps '/….path…./checklists';

%macro setswitch(stepname,status);

%if &status eq Started %then
   %let syscc=0;     /* reset RC to zero */

%if &status ne Reset %then %do;
   %if &syscc le 4 %then %do;
      data steps.&status._&stepname;
          x=1; /* create a dummy file */
      run;
   %end;

   filename mymail email
   to=('rick.allen@citigroup.com')
   subject="&stepname:  &status"  with RC:
            &syscc";

   data _null_; file mymail; put ' '; run;
%end;

%else %do;       /* process this RESET */

proc datasets lib=steps nolist;
    delete Started_&stepname;
    delete Completed_&stepname;
run; quit;
%end;

data x123(keep=logdate logtime event);
    logdate=put(today(),weekdate17.);
    logtime=put(time(),hhmm.);

    length event $30.;
    event="&stepname:  &status";
run;
```

```
proc append force data=x123
        base=steps.event_log;
run

%if &status eq Completed %then %do;
   proc printto; run;  /* release log */
%end;
```

.CSV CREATION

The majority of the monthly PowerPoint report consists of MS Excel charts and tables. The source data for the Excel files is contained within .CSV files that are created within SAS using PROC EXPORT. Here is a hypothetical sample of the dataset created by SAS and used to create a vintage report by Quarter:

Qtr	Age	pctc
2001Q1	0	0.08
2001Q1	1	0.25
2001Q1	2	1.23
…		
2001Q2	0	0.1
2001Q2	1	0.31
2001Q2	2	1.01
…		
2001Q3	0	0.28
2001Q3	1	0.39
2001Q3	2	1.12
…		

The dataset is then transposed and exported with the following code to create the .CSV file:

```
proc transpose data=temp0 out=temp1;
   var pct;
   id age;
   by Qtr;
run;

proc export data=temp1
   outfile= "/…path…/excel_q29.csv"
   dbms=csv replace;
run;
```

FTP FROM SAS/UNIX TO WINDOWS SERVER

The .CSV files are automatically FTP'd to a Windows 2000 server with a macro call. Here is a **portion** of the automated FTP macro code:

```
%macro ftp_copy
   (from=,to=,fileref=ftp_file)

%let to=indriskftp/&to;

%let
ftp_settings=~/ftp_settings_group_drive.sas;

%include "&ftp_settings"; /* user + pswd */

filename &fileref ftp "&to"
    host="…host name…"
    user="&ftp_user"   pass="&ftp_pass";

%let ftp_pass=;

data _null_;
    infile "&from" lrecl=32767;
    file &fileref noprint lrecl=32767;
    input;
    put _infile_;
run;

%mend ftp_copy;
```

Here is the ftp_copy macro call that immediately follows the PROC EXPORT that creates our .CSV file:

```
%ftp_copy(from=…unix path…/excel_q29.csv,
          to=…windows path…/excel_q29.csv)
```

Here is a portion of what our transferred .CSV file looks like when opened in MS Excel:

Qtr	_0	_1	_2	_3
2001Q1	0.08	0.25	1.23	1.84
2001Q2	0.10	0.31	1.01	1.67
2001Q3	0.28	0.39	1.12	1.73
2001Q4	0.14	0.33	0.98	1.72
2002Q1	0.21	0.44		

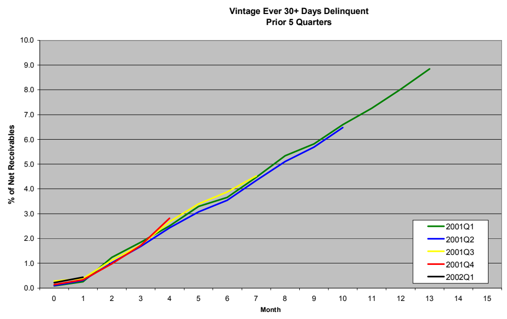
MS EXCEL LINKS

The .CSV files are not edited or copied once they are FTP'd to the Windows 2000 server. Instead, completely separate Excel files are used to construct the charts. The .CSV files are "linked" to the Excel chart files automatically. Each cell on the data sheets in the Excel chart file contains a reference such as this:

```
%=IF('G:\..path...\[excel_q29.csv]excel_q29'!
C2="",NA(),'G:\..path..\[excel_q29.csv]excel_
q29'!C2)
```

IMPORTANT! The IF statement is used to fill in cells where data is not yet available. The cells are filled with the Excel "NA()" function value. This technique prevents the Excel charting methodology from plotting zeros for these unfilled cells.

Here is our finished chart in MS Excel:



POWERPOINT LINKS

The final link between our SAS programs and the printed monthly report is contained within PowerPoint. The charts and tables on the PowerPoint slides are dynamically linked from the Excel files. Here are the steps to follow to create these dynamic links:

1. In Excel, click on the sheet tab or on the border outside the Excel chart that you want to use
2. Choose Edit | Copy
3. Switch to PowerPoint
4. Choose Edit | Paste Special (**not Paste**)
5. Choose the 'Paste Link' option
6. For charts, choose 'Microsoft Excel Chart Object' (this should be the only option)
7. For tables, choose 'Microsoft Excel Worksheet Object'
8. Click 'OK'
9. Resize the pasted object accordingly. Try not to overlap the upper left corner of any pasted frame.

POWERPOINT WARNING MESSAGES

When you include links into your PowerPoint file, the following message will appear whenever you **open** the .PPT file:

**The presentation ______ contains links.
Do you want to update now?**

Clicking 'OK' will refresh the linked information with any changes to the underlying data (this may take awhile). Clicking 'Cancel' is recommended if you know there have been no changes to the data since the previous refresh. However, to ensure you have the most current data, click 'OK'.

Note that an individual PowerPoint object can always be refreshed on demand by right-clicking on the object and choosing "Update Link".

If the Excel files that are linked to by PowerPoint have their own set of links to .CSV data files (as described in the MS EXCEL LINKS), **you will receive a second message** (from Excel, within your PowerPoint session) if you clicked 'OK' above:

The workbook you opened contains automatic links to information in another workbook. Do you want to update this workbook with the changes made to the other workbook?

When you are working in PowerPoint, **always click 'No'** in response to this question! Clicking 'Yes' will not work since Excel cannot update the links in the workbook unless you have opened all of the "linked to" .CSV files also. Excel links should only be refreshed in a separate Excel window using these steps:

1. Open the Excel file
2. Choose Edit | Links
3. Highlight **all** of the link file names in the list
4. Click 'Open Source' (**not 'Update Now'**)
5. Choose File | Save

RESET AT END-OF-MONTH

Since the automation system is always finished by the 13th of each month, a "reset" job is scheduled via Crontab to run on the 26th, 28th, and 30th. Any of these 3 days will suffice, the "reset" job only has to run one time before the next month begins. The reset is scheduled multiple times in the event the Unix server is unavailable that day due to weekend maintenance.

The following code sends a "reset" function to the set switch routine for every job step; this deletes all of the "started" and "completed" indicator files. At this point the scheduler is ready for the 1st day of the following month, to begin the automation all over again.

```
%include '/…path…./set_switch.sas';

%setswitch(Step_1,Reset);
%setswitch(Step_2a,Reset);
%setswitch(Step_2b,Reset);
%setswitch(Step_3a,Reset);

*….etc ;

%setswitch(Step_Perf5,Reset);
```

CONCLUSION

The attachment on page 6 diagrams each of the component pieces of the SAS automation system and how everything fits together.

Hopefully this system, or a component piece, will provide a technique that you can deploy in your environment to automate some repetitive SAS reporting tasks and free up some of your valuable programming resources.

REFERENCES

SAS Institute Inc. (199), *SAS Macro Language: Reference, Version 8*, Cary, NC, 252-253. Copyright © 1999, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. Reproduced with permission of SAS Institute Inc., Cary, NC.

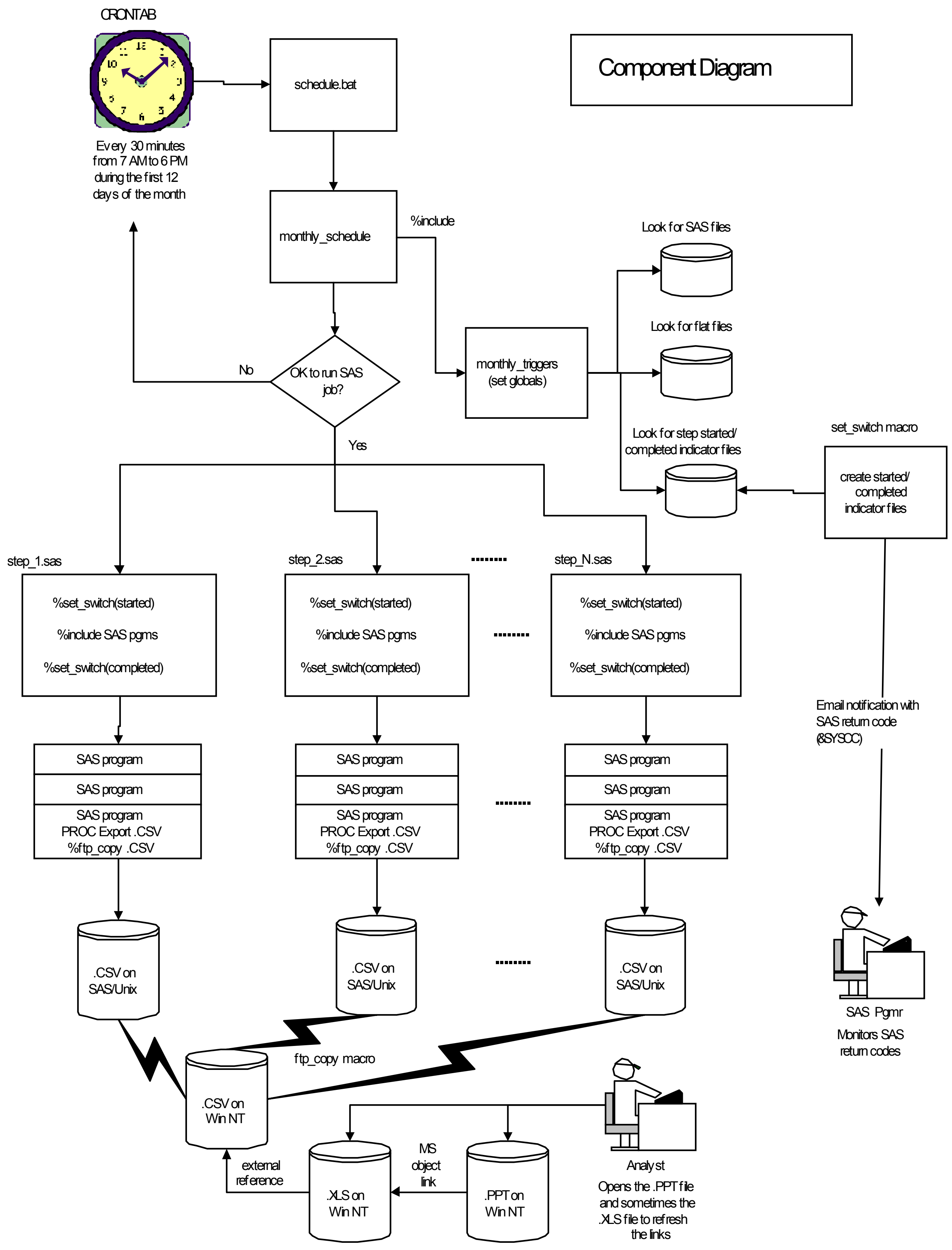
ACKNOWLEDGMENTS

The author would like to thank Dave Devoll for his development of the automated FTP macro. He would like to thank Barry Wolfe for the SYSCC technique. He also would like to thank Mark Prater for reviewing this paper as well as Jeff Weisman and Cathy Bradley for their support and encouragement.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the author at:

Rick Allen
CitiCapital
290 E. Carpenter Freeway, MS H03-145
Irving, TX 75062
Work Phone: 972-652-7308
Fax: 972-652-6397
Email: rick.allen@citigroup.com


CRONTAB
Every 30 minutes from 7 AM to 6 PM during the first 12 days of the month
schedule.bat
Component Diagram
monthly_schedule
%include
OK to run SAS job?
No
Yes
monthly_triggers (set globals)
Look for SAS files
Look for flat files
Look for step started/ completed indicator files
set_switch macro
create started/ completed indicator files
step_1.sas
step_2.sas
step_N.sas
%set_switch(started)
%include SAS pgms
%set_switch(completed)
Email notification with SAS return code (&SYSCC)
SAS program
SAS program
SAS program
PROC Export .CSV
%ftp_copy .CSV
.CSV on SAS/Unix
ftp_copy macro
.CSV on Win NT
external reference
.XLS on Win NT
MS object link
.PPT on Win NT
Analyst
Opens the .PPT file and sometimes the .XLS file to refresh the links
SAS Pgmr
Monitors SAS return codes