Paper BB-05-2014

File Finder: Conditionally Processing Files from a Directory Using SAS®

Katrina Drager, OptumInsight, Eden Prairie, MN

ABSTRACT

File processing for incoming files could be a lot easier and less labor intensive. What is needed is automated processing of the files needed without manually looking for files and running code separately for each import. This code is a method to set people free from looking for files. This program is generic and runs off a configuration file, which provides some basic information about the files for the program to execute import process code for each file individually based on the existence of the file. It is setup to be date agnostic and relatively free of the errors experienced when importing files from sources which change filenames because it is driven by a keyword (At least one item in the filename should be consistent and not redundant between source files found in the directory). It can be executed daily, weekly, monthly, or whenever needed once the files anticipated have been setup in the configuration file and a few simple steps are followed to make sure that directories are regularly cleaned out so they are ready for new files. This program will even notify people what files were in the directory and processing status of each feed via email.

This code uses the concepts listed below in an integrated way to solve a real life problem; People just want their data imported so they can get to work!

Concepts Used:

- Macro Language
- Unix PIPE
- PROC SQL
- Conditional Processing
- Email coded into SAS® Program
- PROC EXPORT

INTRODUCTION

This paper is geared toward individuals involved in data warehousing and those who deal with incoming file feeds which their SAS® processes depend upon.

This process aids in solving the problem of identifying if a file has been delivered to a directory and then calling the appropriate code to process that file. This process will also notify a user or programmer via e-mail what files were found in the directory and which processes were started for the identified files.

In this implementation, SAS® is running on a UNIX Server, which is Grid Enabled. This process should also work on a Non-Grid Unix Server. It could be used partially with modifications in a windows environment using PC SAS.

In this situation, the files are coming to a known location on the Unix server but the exact file name isn't known for certain. It was anticipated that the file names may change but certain keywords were expected to be a constant for each file.

There are multiple file feeds expected to the same directory from multiple sources and we wanted to process those which existed with the right import program when the file is deposited in the directory.

The following are requirements for this code to work:

- Each file has its own layout and its own import program
- Each file may or may not conform to a naming convention
- This process is generic, requiring only updates to the configuration file in order for it be used for several import processes
- In order for the process to work the input file location where the feeds are found should be emptied by the import programs or another process. For example move the file to a new location once it is processed or delete the file once it is processed

Below you will see the basic logical flow:

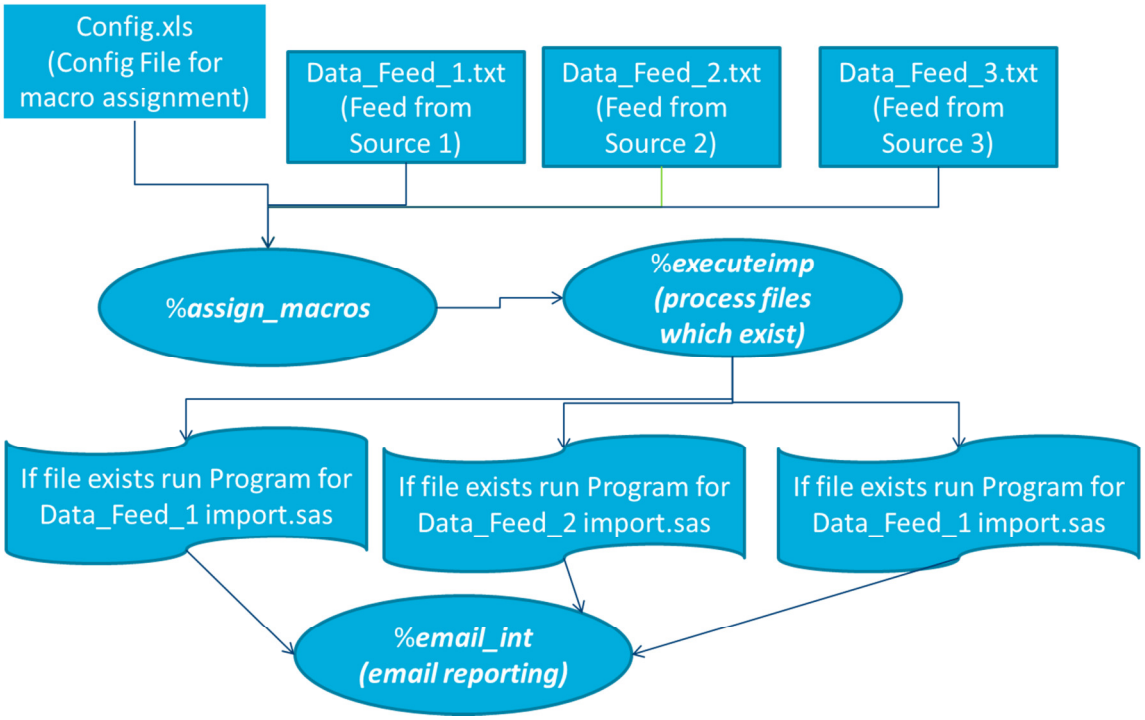

Figure 1. Basic Logical Flow

The organization of the following paper is as follows:

- Step 1 – Import “Configuration” File And Email File
- Step 2 - Identifying Files in Directory
- Step 3 - Assigning Macros for Processing Files
- Step 4 - Running Import Programs for Files which exist
- Step 5 - Email Report of File Status

STEP 1 – IMPORT "CONFIGURATION" FILE AND EMAIL FILE

This process as show in this paper requires a configuration file. The configuration file contains all the required information to locate and process the files. The process itself is driven off a keyword, which is contained in the configuration file. When the keyword from the configuration file is found in the filename, it is matched to its import process and processed. This keyword must be within the file name.

This filename is then assigned to a macro so that it can be processed with the right import program and is controlled by the configuration File.

Using a keyword to identify the filename is a way to avoid errors if the source changes filenames. (This will not eliminate all issues – if the source doesn't include the keyword!)

Example keyword: US

Example File Names:

DataFeed_Us_052014.txt - This will process

DataFeedFromUs.txt - This will process

Data_feed_from_us_May2014.txt – This will process

Data_feed_from_u_s_May2014.txt – This will **not** process

CONFIG.XLS

Below is Figure 2 which contains the column names and example data which should be contained in the config.xls file. The following Figure 3 contains a brief file layout for the config.xls file. The file delimiter in this case is actually used in the import program itself and is tied to the keyword within the import program.

Columns and Sample Data within the config.xls:

ID	File_Keyword	Import_Program_Name	File_Delimiter
1	US	s100_Import_Files_US.sas	'09'x
2	COMPANY	s100_Import_Files_Company.sas	'|'
3	EVENT	s100_Import_Files_event.sas	','
…	…	…	…

Figure 2. Example Columns and Data for config.xls

Description of the data elements contained in config.xls:

Column Name	Description of the data for the column
ID	Used for the do loop (the max number is counted from this file for the do loop). (For example N=1 means ID=1 will process)
File_Keyword	**Used to identify the file within the directory.**
Import_Program_Name	**Name of the program to execute for the file if found.**
File_Delimiter	**The file delimiter for the file expected.**

Figure 3. Description of Config.xls File Columns

EMAIL.XLS

Below is Figure 4 which contains the column names and example data which should be contained in the email.xls file. The following Figure 5 contains a brief file layout for the email.xls file. This file is used in Step 5.

user	e_mail_type	e_mail_address
drager	to	katrina.drager@optum.com
other	to	Other_email@uhc.com

Figure 4. Example Columns and Data for Email.xls

Column Name	Description of the data for the column
user	The user column allows the program to identify the &from_email. address by matching to &sysuserid. (sas macro).
e_mail_type	This will allow macros to be assigned for email from and email to in our email macro.
e_mail_address	The e_mail_type column indicates we want to email this list of people and they will be assigned to the &to_email. macro

Figure 5. Description of Email.xls File Columns

CODE FOR STEP 1 – IMPORT CONFIG FILE AND EMAIL FILE

```
***Once you have created and placed your config file into your code directory.
Import it as the first step in your program;

/*************** Config File *****************/
 PROC IMPORT DATAFILE="&basepath./code/config.xls“ OUT=config  DBMS=xls
REPLACE; DATAROW=2; GETNAMES=YES; SHEET=Config; RUN;

 /*************** Email File – optional *****************/
 PROC IMPORT DATAFILE ="&basepath./code/email.xls“ out=email DBMS=xls REPLACE;
DATAROW=2; GETNAMES=YES; SHEET=Email; RUN;

/*** Count number of file feeds
in the config file for
the do loops below  ****/
 PROC SQL NOPRINT; SELCT COUNT(*)
 INTO :OBSCOUNT FROM config; QUIT;
```

Note the addition of the email file which allows you to send an email to a list of people regarding the status of the files processed / not processed.

STEP 2 - IDENTIFYING FILES IN DIRECTORY

Now that you have a Configuration file setup and your process knows what it should look for regarding files for processing. Next, you need to figure out what files exist in your directory for processing. Further information on how to read files from a directory can be found in “Smokin’ With Unix Pipes” by LeBouton and Rice.

This code identifies files in the directory and makes them into a dataset which the process will then read into macro variables to be used for processing.

```
/************************************************/
/*** Identify Files: what files are in the Landing Zone: "input"  ****/
/*** requires no spaces in the file names *****/
/** &BasePath. is assigned above by a macro ***/
/*** Note: Unix includes the year only when the file is greater than 6 months
old *****/
/*** -lt  lists files in order (most recent is first)***/

filename lsinfo pipe "ls -lt &BasePath./input/" ;

DATA dir_list;
INFILE lsinfo TRUNCOVER;
 INPUT permissions $ 1-10 MemType 11-15 Owner $ 16-24
 Group $ 25-33 FileSize 34-44 Month $ 45-48 Day $ 49-51 TimeOrYear $ 52-58
File_Name $ 59-256;
RUN;
```

Permissions	MemType	Owner	Group	FileSize	Month	Day	Time	File_Name
total	7688							.
-rwxrwx---	1	owner	group	0	Apr	4	13:41	dir.csv
-rwxrwx---	1	owner	group	1025089	Apr	1	16:26	DataFeed_Us_052014.txt
-rwxrwx---	1	owner	group	1945180	Apr	1	16:26	event_Datafeed_01APR14.csv
-rwxrwx---	1	owner	group	959683	Apr	1	16:26	Other_Company_01APR14.txt

Figure 6. Sample Output Dataset from Directory.

This list will be much easier to scan for a filename now that is it is in a dataset. This is sent out in an email for reference using the code below:

```
/*****************************************************************/
/*** this report is attached to the e-mail notification***/
/*** %LET SHEET_1 is variable list for the PROC PRINT ***/

%let Sheet_1=Permissions MemType Owner Group FileSize Month Day TimeOrYear
File_Name;

ODS LISTING CLOSE;
ODS tagsets.ExcelXP
FILE="&Basepath./output/Input_Directory_files_&sysdate..xls" STYLE=MINIMAL;
%MACRO export_reports_dir (filen,var_list);
ODS tagsets.ExcelXP OPTIONS (SHEET_INTERVAL='none' SHEET_NAME="&filen.");
PROC PRINT DATA=&filen. NOOBS;  VAR  &&var_list.;  RUN;
%MEND export_reports_dir;
%export_reports_dir (dir_list,&sheet_1.);
```

```
ODS tagsets.ExcelXP CLOSE;
ODS LISTING;
```

STEP 3 - ASSIGNING MACROS FOR PROCESSING FILES

This step assigns macros to be used in processing the files found in the directory. This is the first part of the assign_marcos, macro.

First all the important information for the file is read starting with row 1 (ID = 1) assigning:

1.) The name of the import program to run for that particular Keyword,

2.) the file keyword that is to be identified in the filename from the directory list,

3.) and the file delimiter that should be processed in the import program.

```
/*****************************************************************/
/** Setup Macros for the feeds and programs related to the feeds ***/
/*** Identify the file name for each feed we are expecting based on the
config.xls (File_Keyword Field) *****/

%MACRO assign_macros;
%DO n=1 %TO &OBSCOUNT.;
%GLOBAL PRun_&n. Filename_&n. Flag_Process_&n. Feed_&n. Del_&n.;
/***the keyword is used  to determine if the import file exists. The filename
and the name of the import program for each file feed is based on the config
file***/
/** &PRun_&n. this is the import program to run for the file **/
/** &Feed_&n. this is the feed name (File_Keyword) for use in the macros **/
/** Del_&n. this is the file_delimiter we are expecting to see for each file**/
PROC SQL NOPRINT; SELECT DISTINCT STRIP(File_Keyword) ,
STRIP(Import_Program_Name) , STRIP(File_Delimiter)
INTO :Feed_&n. SEPARATED by ',' ,:PRun_&n. SEPARATED by ',' ,:Del_&n. SEPARATED
by ','
FROM LIB.config WHERE ID IN (&n.); QUIT;

%PUT &&PRun_&n. ;    %PUT &&Feed_&n. ;    %PUT &&Del_&n. ;
```

The following portion of the assign_macro looks for the keyword "&&Feed_&n.“ and if that keyword is found in a File_Name (&&Filename_&n) that file name is associated with the keyword for further processing. The Flag_Process is set to 0 when there is no file_name found – indicating no further processing will occur for that file (as it doesn't exist) or if it is greater than 0, the file will process (the file exists).

```
/***Filename_&n. - assign the filename to a macro to execute the import program
for each file feed :***/
/***Flag_Process_&n. - set a macro to determine if the process for each feed
will run :***/
PROC SQL NOPRINT;
SELECT  File_Name  ,COUNT(*)
INTO :Filename_&n. SEPARATED by ',' ,:Flag_Process_&n. SEPARATED by ','
FROM dir_list  WHERE UPCASE(File_Name) CONTAINS "&&Feed_&n."; QUIT;
%PUT &&Filename_&n.;     %PUT &&Flag_Process_&n.;
```

Continue to last portion of macro which provides documentation for the log:

```
/*** this provides documentation within the log for what files were found and
what macros were assigned in a readable and condensed format ****/
/*** for log documentation ******/
DATA _NULL_; put 77*'='
   / "=== Macros for program execution by file iteration n=&n."
   / "=== The filename that will be processed:"
   / "=== Filename_&n. ========>  &&Filename_&n.“
   / "=== If the flag_process_n is greater than 0 the file will be processed:"
   / "=== Flag_Process_&n. =====> &&Flag_Process_&n.“
   / "=== The program which will processes this file is:"
   / "=== PRun_&n. =============> &&PRun_&n.“
   / "=== The delimiter for this file is:"
   / "=== Del_&n. =============> &&Del_&n.“
   / "=== The macro assigned the Filename is:"
```

```
   / "=== Feed_&n. =============> &&Feed_&n."
      / 77*'=';
  RUN;
%END;
%MEND assign_macros;
%assign_macros;
```

Example Log Output:

```
================================================================================
=== Macros for program execution by file iteration n=1
=== The filename that will be processed:
=== Filename_1 ========>  DataFeed_Us_052014.txt
=== If the flag_process_n is greater than 0 the file will be processed:
=== Flag_Process_1 =====> 1
=== The program which will processes this file is:
=== PRun_1 =============> s100_Import_Files_US.sas
=== The delimiter for this file is:
=== Del_1 =============> '09'x
=== The macro assigned the Filename is:
=== Feed_1 =============> US
================================================================================
NOTE: DATA statement used (Total process time):
    real time           0.03 seconds
      cpu time            0.00 seconds
```

Note: These are the resolved macros above that will actually pass to the execution of the import program for this file. All of this information comes from either the config file or from the dir_list data set – Step 2, assigned to macros in Step 3.

STEP 4 - RUNNING IMPORT PROGRAMS FOR FILES WHICH EXIST

To this point the following items have been done:

- The Config file has been created and imported
- Files in the Input directory have been identified
- Macros have been assigned for all variables needed to run the import processing, if there are files in existence for the processing to proceed.
- Remember this will only process files that exist and that have a keyword within them that match the keyword on the config file.

This macro actually executes the import program for each file found in the directory which matched a keyword. You will notice one portion of the process deals with files that were found and another portion of the process deals with files which were not found.

```
%MACRO executeimp;
%DO n=1 %TO &OBSCOUNT.;
/*** run for each file *****/
%IF &&Flag_Process_&n. ne 0 %THEN %DO;
DATA _null_; put 77*'='
/ "=== Feed_&n. =============> &&Feed_&n."
/ "=== =====> Successful Execution Processing File: &&Filename_&n. “
/ "=== =====> RUNNING PROGRAM : &&PRun_&n. “  / 77*'='; RUN;
    %include "&Basepath./code/&&PRun_&n.";
%END;
%ELSE %DO;
DATA _null_; put 77*'='
/ "=== =====> FILE DOES NOT EXIST  = &BasePath./input/&&Filename_&n. "
/ "=== =====> No Processing was executed for file related to feed."
/ "=== Feed_&n. =============> &&Feed_&n."
/ 77*'=';  RUN;
%END;
```

```
/*** end: run for each file *****/
%END;
%MEND executeimp;
%executeimp;
```

Here is a sample partial log for 1 file:

```
================================================================================
=== Feed_1 =============> US
=== =====> Successful Execution Processing File: DataFeed_Us_052014.txt
=== =====> RUNNING PROGRAM : s100_Import_Files_US.sas
================================================================================
```

Note: Just before this program runs (in this example s100_Import_Files_US.sas is running) you will see this documentation appear in the log prior to the actual program execution.

STEP 5 - EMAIL REPORT OF FILE STATUS

This is optional, if you want to email the status of each file and provide a list of all files found in the directory. Macros for from and to email are assigned using the following code:

```
/*** from e-mail selected based on sysuserid running the program ***/
PROC SQL NOPRINT;
SELECT DISTINCT
    COMPRESS("'"||e.e_mail_address||"'")
INTO :from_email SEPARATED by ","
FROM LIB.email as e
WHERE e.user IN ("&sysuserid.");
QUIT;

%PUT &from_email.;

/*** to e-mail selected based on e_mail_type column from email xls ***/
PROC SQL NOPRINT;
SELECT DISTINCT
COMPRESS("'"||e.e_mail_address||"'")
INTO :to_email SEPARATED by ","
FROM LIB.email as e
WHERE e.e_mail_type IN ("to");
QUIT;

%PUT &to_email.;
```

This sends an e-mail which provides the status of each file in the body and attached is the optional report which lists all the files in the directory when the process ran. This is the start of the body of the e-mail:

```
%MACRO email_int;
FILENAME outmail EMAIL
SUBJECT="Files Processing to Data Warehouse &SYSDATE."
ATTACH=("&Basepath./output/Input_Directory_files_&sysdate..xls")
FROM= &from_email.
/*** seperated by """" ***********/
TO = (&to_email.);
DATA _NULL_;
FILE outmail;
PUT /"Attached you will find the files that were contained at the input location.";
PUT /"Ensure that these files are what you expected.";
PUT /"The following Feeds were checked and their processing status is listed
below:";
PUT /77*"=";
```

This is a conditional portion of the e-mail which provides the status of each file.

```
%MACRO callemailtxt;    %DO n=1 %TO &OBSCOUNT.;
/*** run for each file to notify what file was/wasn't received *****/
%IF &&Flag_Process_&n. ne 0 %THEN %DO;
PUT /77*"=";
PUT /"=== =====> Successful Execution Processing File: &&Filename_&n.";
PUT /"=== =====> RUNNING PROGRAM : &&PRun_&n.";
PUT /77*"="; %END;
%ELSE %DO;
PUT /77*"=";
PUT /"=== =====> FILE DOES NOT EXIST  = &BasePath./input/&&Filename_&n.";
PUT /"=== =====> No Processing was executed for file related to feed.";
PUT /"=== Feed_&n. ==============> &&Feed_&n.";
PUT /77*"=";
%END; %END;
%MEND callemailtxt;
%callemailtxt;
```

Signature and execution of the e-mail macro to send the actual e-mail.

```
PUT /77*"=";
PUT /"Advanced Data and Analytics - Custom Reporting | OptumInsight";
PUT /"ID: &sysuserid.";
PUT /"This is a system generated e-mail.";
;
RUN;

%MEND email_init;
%email_int;
```

Sample E-mail generated:

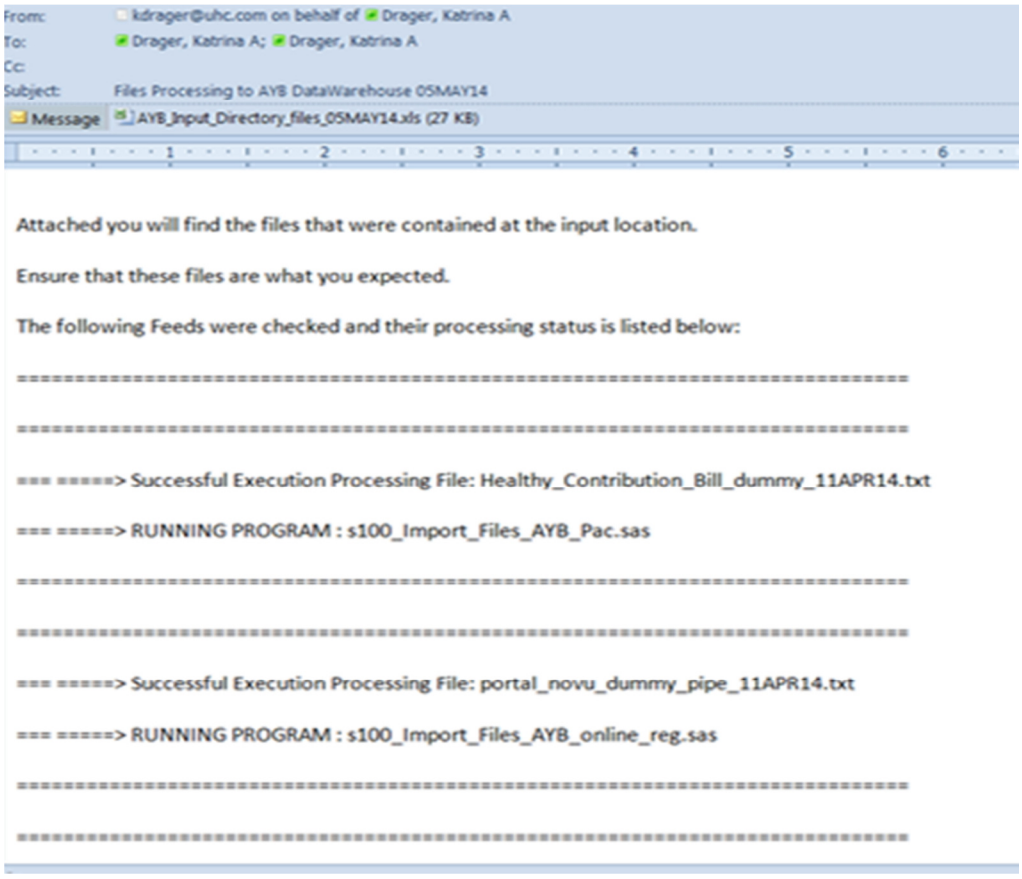

Figure 7. Sample E-Mail Generated using the above code.

CONCLUSION

This program is meant to aid in the identification and processing of files it is a driver based on the use of iterative macros which run based on a configuration file and data existing in the directories with some creativity this program could be used as the basis of other programs which search multiple directories and call multiple import programs, by just expanding some of the information contained in the configuration file and adding some looping to the directory searches.

This program can be used as shown or edited based on your own needs. It is generic and can be changed simply by changing the way the configuration file is setup (config.xls).

REFERENCES

LeBouton, K. & Rice, T. (2000), *Smokin' With Unix Pipes*. http://www2.sas.com/proceedings/sugi25/25/cc/25p103.pdf

CONTACT INFORMATION

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

Katrina Drager
OptumInsight
PO Box 9472
Minneapolis, MN 55440
Work Phone: 952-931-5455
E-mail: Katrina.Drager@Optum.com