Check Please: An Automated Approach to Log Checking

Richann Watson

ABSTRACT

In the pharmaceutical industry, we find ourselves having to re-run our programs repeatedly for each deliverable. These programs can be run individually in an interactive SAS® session, which enables us to review the logs as we execute the programs. We could run the individual programs in batch and open each individual log to review for unwanted log messages, such as ERROR, WARNING, uninitialized, have been converted to, and so on. Both of these approaches are fine if there are only a handful of programs to execute. But what do you do if you have hundreds of programs that need to be re-run? Do you want to open every single one of the programs and search for unwanted messages? This manual approach could take hours and is prone to accidental oversight. This paper discusses a macro that searches a specified directory and checks either all the logs in the directory, only logs with a specific naming convention, or only the files listed. The macro then produces a report that lists all the files checked and indicates whether issues were found.

INTRODUCTION

Checking the logs for unwanted messages is one of the first steps in ensuring that a program was executed successfully. When running the program interactively this can be easily done by manually reviewing the log in the interactive session. However, if you execute the program in batch mode or from a 'driver' program and the logs are saved as individual files, then you need to open the log using SAS Universal Viewer or some other type of text editor in order to review the log. Regardless of the method you use, the log still needs to be perused in order to find these pesky messages.

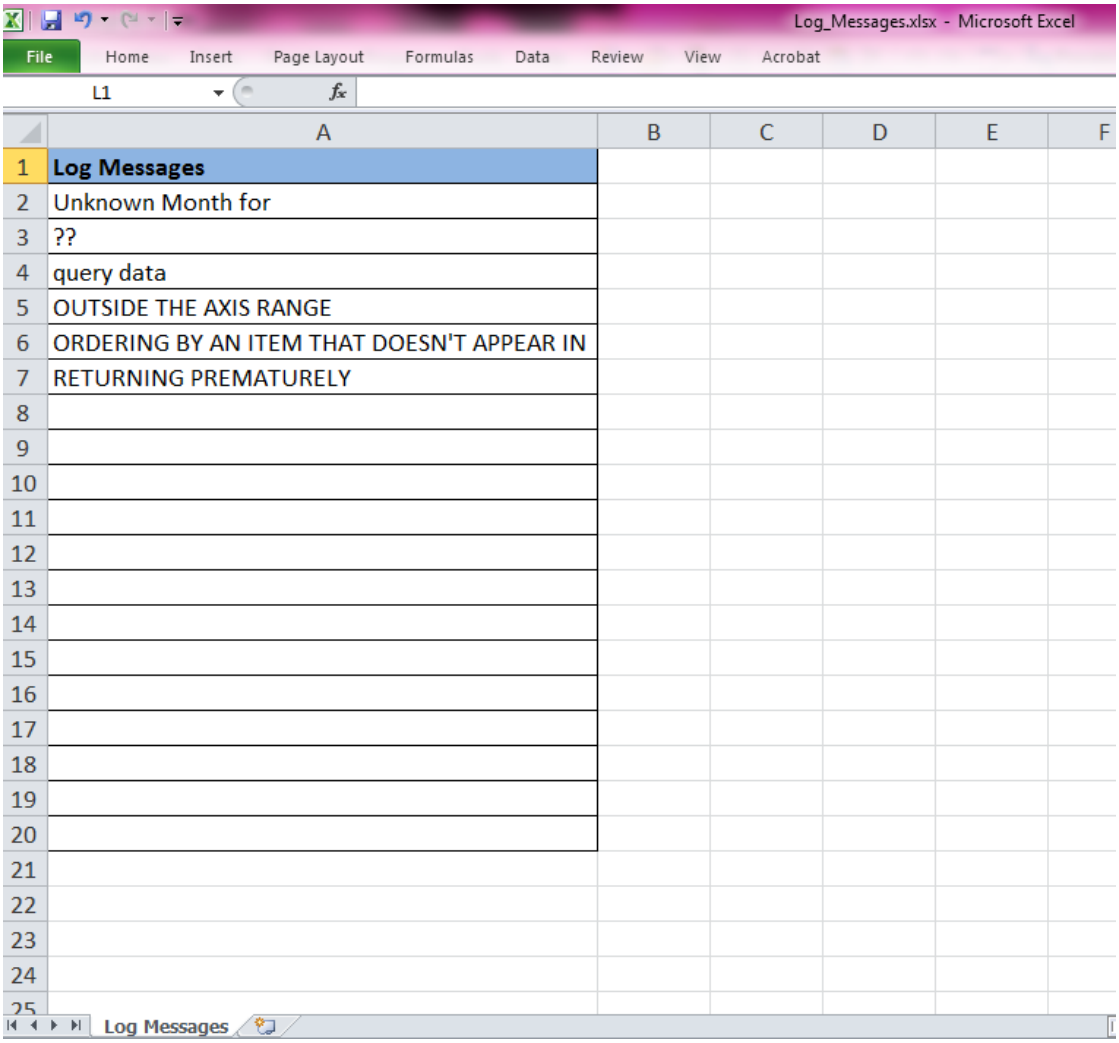
TYPES OF LOG MESSAGES

There are a variety of log messages that may cause problems in the program and these messages need to be investigated. Some of the more common messages include:

- ERROR
- WARNING
- UNINITIALIZED
- NOTE: MERGE
- MORE THAN ONE DATA SET WITH REPEATS OF BY
- VALUES HAVE BEEN CONVERTED
- MISSING VALUES WERE GENERATED AS A RESULT
- INVALID DATA
- INVALID NUMERIC DATA
- AT LEAST ONE W.D FORMAT TOO SMALL

How these log messages are to be fixed is dependent on the data and project needs. 'ERROR' messages should always be corrected since this can affect the output. Additionally, the 'MORE THAN ONE DATA SET WITH REPEATS OF BY' should be fixed so that a many-to-many merge is not done in a data step. Some other messages may not have an effect on the data and may not be as critical to fix; however, that is dependent on what the client deems as critical. There are some clients that do not wish to see any unwanted log messages and request that they all be fixed.

In addition to the common messages that you would usually look for in logs, the client may have a more extensive list of messages that they may not want to appear. These unwanted log messages can be stored in an Excel workbook, such as the one shown in Display 1.



Display 1. Sample Excel Workbook with Client Specific Unwanted Log Messages

OTHER LOG MESSAGES TO CONSIDER

When determining the list of possible unwanted log messages to build into the macro (provided in Appendix A), you may need to consider other types of messages that may cause the macro to flag the program as containing unwanted messages. Additionally, if you are aware of specific types of log messages that will trigger an issue to be reported, then it would be wise to build into the macro the types of messages that are allowed.

For example, when a SETINIT may be close to the expiration date, the log will have several types of WARNING messages. Since these WARNING messages do not have an impact on the actual execution of the program, then logic can be incorporated to look for these types of messages.

- UNABLE TO COPY SASUSER
- BASE PRODUCT PRODUCT
- EXPIRE WITHIN
- BASE SAS SOFTWARE … EXPIRING SOON
- UPCOMING EXPIRATION
- SCHEDULED TO EXPIRE

- SETINIT TO OBTAIN MORE INFO

LOG CHECKING PROBLEM

With all the various types of messages that need to be checked during the log review, one can plainly see how quickly this process of checks and fixes can become tedious and burdensome. This manual approach to reviewing these logs can lead to several other concerns.

1. It is easy to overlook a particular message in a log while scrolling through the log.
2. When searching for a particular message, it is easy to mistype the message or get the casing incorrect and thus overlooking the unwanted message.
3. If there are a lot of logs to check, it is easy to overlook a log altogether.
4. The programmer may be short on time and may just make the assumption that the program ran clean the last time and there is an output produced so it should still be fine.

THE SOLUTION - CHECKLOGS

Rather than leave the log checking to 'chance' and human error an automated alternative can be implemented through use of the CHECKLOGS macro (Appendix A). The macro will parse through each of the indicated logs in a specified directory looking for the unwanted messages, and then produce a report summarizing the findings.

MACRO PARAMETERS

The macro has 8 input parameters, with only one being required:

- 'loc' – location(s) where the log files reside; If more than one location is specified then it should be separated by a delimiter indicated in the 'delm' macro parameter.
- 'fnm' – optional parameter that indicates which types of files to look at; if more than one type of file is specified then it should be separated by a delimiter indicated in the 'delm' macro parameter.
- 'out' – optional parameter that indicates the name of the output report file that will be produced.
- 'loc2' – optional parameter that indicates the location of where the report should be saved.
- 'delm' – optional parameter that indicates what delimiter is used in the 'loc' and 'fnm' parameters.
- 'msgf' – optional parameter that indicates the full file name of the spreadsheet that contains the client specific unwanted log messages.
- 'msgs' – optional parameter that indicates the name of the worksheet within the Excel workbook indicated in the 'msgf' macro parameter.
- 'msgv' – conditionally required parameter that indicates the name of the column within the Excel workbook indicated in the 'msgf' macro parameter.

The 'loc' parameter allows for multiple locations to be specified. The full path of each location must be indicated and they must be separated by the indicated delimiter. The macro will scan through each location specified looking for the necessary log files.

For the optional parameter 'fnm', the macro will build a where clause that will be used to exclusively extract the logs of interest. For example, if all your logs are stored in the same folder and you are only concerned with viewing the logs for the tables and figures, and all the tables and figures logs have a standard naming convention (e.g., 't_#_#_#.log' for tables and 'f_#_#_#.log' for figures where '#_#_#' represents the output number) then you can use the standard portion for each type separated by a delimiter such as '~'.

```
fnm = t_~f_
```

Note that the building of the where clause does not care if the standard portion is a prefix or a suffix as long as there is a standard and that the standard is not used for other log types. So if the table logs had a naming convention of 'xxxxx_t.log' and the figure logs had a naming convention of 'xxxxx_f.log' where

'xxxxx' represents the output name, then you can still use the standard portion for each type separated by a delimiter such as '~'.

```
fnm = _t.~_f.
```

If the 'out' parameter is not specified, then the file name for the report will default to 'all_checklogs.rtf'. Ideally, if you are executing the macro to look at only specific types of files, then the name of the report should be specified so that the report is not overwritten when executed later for another file type.

If the 'loc2' parameter is not specified, then the output location for the report will default to the first location that exists in the 'loc' parameter. For example, you can specify four locations for logs but the first one (C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Adhoc) but the next one in the list of locations does (C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Graphs). The second location is the first location that exists and therefore is the location that will be used as the storage location.

```
loc = C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Adhoc~
      C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Graphs~
      C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Tables
```

If the 'delm' parameter is not specified, then the delimiter defaults to '@'. Note that this needs to match the delimiter that is used if 'loc' has multiple locations and if 'fnm' is specified and more than one file name type is indicated in the 'fnm' macro parameter. For example, above 'loc' and 'fnm' used a '~' as a delimiter therefore 'delm' would need to be specified. Note that it is not ideal to use a space as a delimiter since the directory path and/or the file names could have spaces.

```
delm =~
```

If a client has specific log messages that they want to scan for, they can specify those messages in a spreadsheet. The spreadsheet can be read into a SAS data set and the messages can be incorporated into the macro so that the macro can scan for those particular messages during execution. In order for the macro to read in the spreadsheet, the following two macro parameters must be specified: 'msgs' and 'msgv'.

If 'msgs' is specified:

1. The FULL file name must be specified; this includes the location of the file. For example,

   ```
   msgs = C:\Users\user\Desktop\SAS Tips\Check Logs\Log_Messages.xlsx
   ```

2. The macro parameter 'msgv' must also be specified. This is the column name with spaces and special characters converted to an underscore. For example, refer to Display 1,

   ```
   msgv = LOG_MESSAGES
   ```

   Note that the worksheet must have a column header in order for the program to execute accurately.

3. If the spreadsheet has a worksheet with a name other than 'Sheet1', then the name of the worksheet must also be specified. If the worksheet is 'Sheet1', then the worksheet name does not have to be specified, since the default value is 'Sheet1'. For example, refer to Display 1,

   ```
   msgs = Log Messages
   ```

Table 1. illustrates some sample calls of the macro and what the expected outcome is. Note that the samples in this paper are based off of the Windows environment. However, the program will work with a Unix environment as well. The user just needs to make sure the slashes in the 'loc' and 'loc2' if specified parameter(s) are pointing in the correct direction.

Sample Call	Expected Outcome
`%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All)`	• Check **ALL** the logs in the specified directory. • Use default name "all_checklogs" for the report name. • Store report in location where logs are stored.

Sample Call	Expected Outcome
	• There are no client specific log messages to look for.
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, out=all_output_logs)	• Check **ALL** the logs in the specified directory. • Use specified name "all_output_logs" for the report name. • Store report in location where logs are stored. • There are no client specific log messages to look for.
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, fnm=table, out=Tables_Logs)	• Check **ONLY** the logs that contain "table" in the file name in the specified directory. • Use specified name "Table_Logs" for the report name. • Store report in location where logs are stored. • There are no client specific log messages to look for.
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, out=Analysis Dataset and Tables Logs, fnm=ad@table)	• Check **ONLY** the logs that contain "ad" **OR** "table" in the file name in the specified directory. • Use the default delimiter "@" • Use specified name "Analysis Dataset and Tables Logs" for the report name. • Store report in location where logs are stored. • There are no client specific log messages to look for.
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, out=Analysis Dataset and Tables Logs delm, fnm=ad~table, delm=~)	• Check **ONLY** the logs that contain "ad" **OR** "table" in the file name in the specified directory. • Use the specified delimiter '~' • Use specified name "Analysis Dataset and Tables Logs delm" for the report name. • Store report in location where logs are stored. • There are no client specific log messages to look for.
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, loc2=C:\Users\user\Desktop\SAS Tips\Check Logs\Log Report, out=Tables 1 and 2 Logs delm, fnm=table1!table2, delm=!)	• Check **ONLY** the logs that contain "table1" **OR** "table2" in the file name in the specified directory. • Use the specified delimiter '!' • Write the report to another location • Use the specified name "Tables 1 and 2 Logs delm" for the report name. • There are no client specific log messages to look for.
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, loc2=C:\Users\user\Desktop\SAS Tips\Check Logs\Log Report, out=Tables 1 and 2 Logs delm-Specific Log, fnm=table1!table2, delm=!, msgf=C:\Users\user\Desktop\SAS Tips\Check Logs\Log_Messages_Sheet.xlsx, msgs=Log Messages, msgv=Messages)	• Check **ONLY** the logs that contain "table1" **OR** "table2" in the file name in the specified directory. • Use the specified delimiter '!' • Write the report to another location • Use the specified name "Tables 1 and 2 Logs delm-Specific Log" for the report name. • The client specific workbook specified with the column name of 'Messages' and specified worksheet name of 'Log Messages' is incorporated into the list of log messages to scan for.
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, loc2=C:\Users\user\Desktop\SAS Tips\Check Logs\Log Report, out=Tables 1 and 2 Logs delm-Specific Log Sheet, fnm=table1!table2, delm=!, msgf=C:\Users\user\Desktop\SAS Tips\Check Logs\Log_Messages_Sheet.xlsx, msgs=Log Messages, msgv=Messages)	• Check **ONLY** the logs that contain "table1" **OR** "table2" in the file name in the specified directory. • Use the specified delimiter '!' • Write the report to another location • Use the specified name "Tables 1 and 2 Logs delm-Specific Log Sheet" for the report name. • The client specific workbook specified with the column name of 'Messages' and specified worksheet name of 'Log Messages' is incorporated into the list of log messages to scan for.

Table 1. Sample CHECKLOG Calls and Expected Outcome

Note that the all the sample calls can also be done with multiple locations specified. In the following example, all the macro parameters are specified and multiple log locations are also indicated. Thus in this example, all four locations will be scanned looking for the standard error messages as well as the client specific messages provided in the spreadsheet 'Log_Messages_Sheet.xlsx'. However, only the log

files with 'table1' **OR** 'graph' in the file name will be parsed. The user specified delimiter '#' will be used to parse out the different locations and the different file name types. Lastly, the report generated will be named 'multiple_table_graph_logs_errxls' and stored in the Log Report folder not the ADaM folder.

```
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\ADaM#
               C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Graphs#
               C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\SDTM#
               C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Tables,
           loc2=C:\Users\user\Desktop\SAS Tips\Check Logs\Log Report,
           fnm=table1#graph,
           delm=#,
           out=multiple_table_graph_logs_errxls,
           msgf=C:\Users\user\Desktop\SAS Tips\Check Logs\Log_Messages_Sheet.xlsx,
           msgs=Messages
           msgv=Log_Messages)
```

NUTS AND BOLTS OF THE MACRO

So how does this macro actually work? There are a variety of steps in the macro and each step will be explained.

Determining Working Environment

The first thing the macro does is determine in what working environment the command is being executed. The macro is able to run in either Windows or Unix environment. Depending on the environment, the macro will determine what form the pipe command should take and in which direction the slash in the path name should be pointing.

During this initial part of the macro, the following two macro variables are set and will be used in the macro:

- ppmcd – represents the pipe command
- slash

If the execution environment is Windows (i.e., &SYSSCP = WIN), then following macro variables are set as

```
%let ppcmd = %str(dir);
%let slash = \;
```

If the execution environment is Unix (i.e., &SYSSCP = LIN X64), then following macro variables are set as

```
%let ppcmd = %str(ls -l);
%let slash = /;
```

If the environment is something other than WIN or LIN X64, then the macro will abort. The macro will need to be updated to allow for different operating environments.

What Log Files Are Being Checked?

If the 'fnm' input parameter is specified, the macro will then parse through each token in the parameter to build a where clause. If there is more than one token in 'fnm', then either the default delimiter or the one specified in 'delm' parameter, will be used to parse 'fnm'. The where clause will use the SAS function INDEX to look for a specific value in the filename. If no input parameter is specified, then the macro will check every log file in the specified directory.

Using the sample calls in Table 1. the associated where clauses for each call are illustrated (see Table 2.).

Sample Call	Where Clause Built
`%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All)`	No ***where*** clause. Macro will check all logs in the directory.
`%checklogs(loc=C:\Users\user\Desktop\SAS`	No ***where*** clause. Macro will check all logs in the directory.

Sample Call	Where Clause Built
Tips\Check Logs\Logs\All, out=all_output_logs)	
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, fnm=table, out=Tables_Logs)	where index(flog, “table”)
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, out=Analysis Dataset and Tables Logs, fnm=ad@table)	where index(flog, “ad”) or index(flog, “table”)
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, out=Analysis Dataset and Tables Logs delm, fnm=ad~table, delm=~)	where index(flog, “ad”) or index(flog, “table”)
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, loc2=C:\Users\user\Desktop\SAS Tips\Check Logs\Log Report, out=Tables 1 and 2 Logs delm, fnm=table1!table2, delm=!)	where index(flog, “table1”) or index(flog, “table2”)
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, loc2=C:\Users\user\Desktop\SAS Tips\Check Logs\Log Report, out=Tables 1 and 2 Logs delm-Specific Log, fnm=table1!table2, delm=!, msgf=C:\Users\user\Desktop\SAS Tips\Check Logs\Log_Messages_Sheet.xlsx, msgs=Log Messages, msgv=Messages)	where index(flog, “table1”) or index(flog, “table2”)
%checklogs(loc=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All, loc2=C:\Users\user\Desktop\SAS Tips\Check Logs\Log Report, out=Tables 1 and 2 Logs delm-Specific Log Sheet, fnm=table1!table2, delm=!, msgf=C:\Users\user\Desktop\SAS Tips\Check Logs\Log_Messages_Sheet.xlsx, msgs=Log Messages, msgv=Messages)	where index(flog, “table1”) or index(flog, “table2”)

Table 2. Sample CHECKLOG Calls and Where Clause Built

Reading in Log Message Spreadsheet

If a client specific list of unwanted messages is provided in an Excel workbook, this list of messages is read into SAS to create three macro variables. These macro variables are used to build the search criteria that is used for scanning each log.

Depending on how many client specific log messages are provided, only one or two of the macro variables may be populated. The macro will create an index statement for each log message similar to what was done when creating the where clause for subsetting for specific files. The index statements for each log message are appended into one variable that is no longer than 2000 characters to create a macro variable. If the appended index statements for each log message exceed more than 2000 characters, then the second macro parameter is populated with the next set of appended index statements. If there are copious amounts of client specific messages, then the code can be modified to allow for more macro variables to be created in a similar fashion to the blocked code in SAS Code 1. If additional macro variables are created they need to be included in the portion of the macro where the log scan occurs.

```
libname logmsg xlsx "&msgf";

data _null_;
  length fullmsg1 fullmsg2 fullmsg3 $2000;
  retain fullmsg1 fullmsg2 fullmsg3;
  set %if "&msgs" ne "" %then logmsg."&msgs"n;
      %else logmsg."Sheet1"n;
      end = eof; /* need this semicolon to end the set statement */

  partmsg = catt('index(upcase(line), "', (upcase(&msgv)), '")');

  if length(fullmsg1) + length(partmsg) <= 2000 then
       fullmsg1 = catx(" or ", fullmsg1, partmsg);
  else if length(fullmsg2) + length(partmsg) <= 2000 then
       fullmsg2 = catx(" or ", fullmsg2, partmsg);
  else fullmsg3 = catx(" or ", fullmsg3, partmsg);

  if eof then do;
          call symputx('fullmsg1', fullmsg1);
          if fullmsg2 ne '' then call symputx('fullmsg2', fullmsg2);
          if fullmsg3 ne '' then call symputx('fullmsg3', fullmsg3);
  end;
run;
```

SAS Code 1. Creating the Macro Variables for Client Specific Log Messages

Note that the macro uses the libname statement to specify the spreadsheet and read the file.

Extracting the Logs from the Different Log Locations

Determine if the Log Location Exists

Since it is possible for log files to reside in multiple locations, it may be necessary to search all possible locations for the necessary files. The macro will allow for multiple locations to be specified as long as they are separated by the indicated delimiter. Each location will be extracted one at a time and checked to make sure the location exists. For each log location a libname will be created and a system macro variable will be used to determine if the libname was successfully assigned.

```
/* need to make sure the location exists so create a temp library */
libname templib&k "&&lcn&k";

/* if &SYSLIBRC returns a 0 then path exists                      */
%if &syslibrc = 0 %then %do;  /* begin if &syslibrc = 0 */
```

If the location exists, then the location will be scanned for the log files that are to be parsed for unwanted error messages. If the location does not exist, a warning message is written to the log indicating the directory does not exist.

```
WARNING: "directory C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\All Logs does not exist"
```

If the log location does not exist, the macro moves onto next log location. It will loop through all of the log locations determining if each exists and whether or not it should start the log extraction phase. If none of the log locations specified exists, an additional warning message is written to the log.

```
WARNING: "None of the log locations specified exist"
```

Log Extraction

After the pipe command is built based off of the macro variables set during the determination of the working environment and the where clause is determined (if applicable) the macro will bring in every file within the specified directory that is being processed and create a data set. This data set will only contain the logs files that are indicated by the where clause, in the event that a where clause is built. Additionally, the name of the log, the date and time can be extracted from the various tokens in the filename. When

the pipe command reads in each file all the information for that file is captured on one line (one variable) and the data can be parsed to extract each component.

Table 3. below illustrates how the filename would look if executed in a Windows environment and what the log, date and time are once they are extracted. Then numtok indicates how many tokens are in the filename. This is used to determine if there are any spaces within the filename, so that the macro will correctly assign flog. Note that the numtok is based on the time stamp having AM/PM. If 24-hour clock is used, then the code will need to be adjusted accordingly (i.e., 03:00 PM versus 15:00 – has one less token).

filename	flog	ftim	fdat	numtok
09/07/2016 03:00 AM 48,511 adpr.log	adpr	7-Sep-16	3:00 AM	5
09/19/2016 08:03 AM 76,849 adpsga.log	adpsga	19-Sep-16	8:03 AM	5
08/30/2016 05:43 AM 58,634 adsl.log	adsl	30-Aug-16	5:43 AM	5
08/28/2015 05:14 AM 20,028 AE.log	AE	28-Aug-15	5:14 AM	5
09/01/2015 04:41 AM 22,392 DA.log	DA	1-Sep-15	4:41 AM	5
08/26/2015 10:59 AM 19,734 dm.log	dm	26-Aug-15	10:59 AM	5
10/13/2016 08:49 AM 23,491 graph 1_1.log	graph 1_1	13-Oct-16	8:49 AM	6
10/13/2016 08:49 AM 11,934 graph 1_3.log	graph 1_3	13-Oct-16	8:49 AM	6
09/11/2015 08:06 AM 46,331 LB.log	LB	11-Sep-15	8:06 AM	5
10/17/2016 07:27 AM 24,652 table 3_1_1.log	table 3_1_1	17-Oct-16	7:27 AM	6
10/17/2016 07:28 AM 33,002 table 3_1_2.log	table 3_1_2	17-Oct-16	7:28 AM	6
09/28/2016 10:02 AM 45,680 table1-1.log	table1-1	28-Sep-16	10:02 AM	5
10/08/2015 08:21 AM 24,857 table1_2.log	table1_2	8-Oct-15	8:21 AM	5
10/05/2016 11:08 AM 91,155 table2.1.2.log	table2.1.2	5-Oct-16	11:08 AM	5
10/08/2015 08:21 AM 36,413 table3_2_1.log	table3_2_1	8-Oct-15	8:21 AM	5

Table 3. Files Retrieved and Log Name, Date and Time Extracted

Parsing Each Log File

Once it is determined which logs are to be checked, the macro will then go through each file and check for 'common' undesired log messages as well as other 'not-so-common' undesired log messages. At this point, the user may wish to add 'user-defined' log messages that should be investigated. If a client has a list of specific messages that should be included in the parsing of each log file, then an Excel spreadsheet which contains all the unwanted log messages should be provided. The spreadsheet needs to be specified during the macro call and it should have a column header that is specified during the macro call as well. If the spreadsheet has a worksheet named something other than 'Sheet1', then the worksheet name needs to be specified during the macro call. It is not required that a client specific list of log messages be provided, so if a client does not provide such a list, the macro is still able to run The macro will automatically scan for any of the pre-defined messages that were specified in Types of Log Messages.

If an undesired log message is found, it is saved to a data set so it can be included in the check log report. If there are no undesired log messages, then a 'dummy' record is created for that file with a generic message so that the user can be confident that the log was actually checked and not inadvertently left out of the checking process.

Creating the Log Report

After each log is checked a report will be generated. If the 'loc2' parameter is specified, then a report will be written to the indicated location otherwise the report will be written to the first log location that exists in the list of locations provided. If the 'out' parameter is specified during the macro call, the value of the parameter will be used as the name of the report otherwise the name of the report will default to

'all_checklogs'. The report will contain the name of the log, the date the log was created, the time it was created (if applicable) and the undesired log message or generic message. In addition to this, the report will contain a blank column that will allow the user to input any additional information such as a comment as to why that specific message was allowed while they are reviewing the report.

For example, in Display 2 the logs found in the C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\ADaM directory show that ADAAPASI and ADPSGA had merge issues. In addition, for ADAAPASI, there was a note regarding missing values. During the review of the report it was noted that for the particular deliverable this was fine but that it must be fixed for the next deliverable. For ADSL, it had a message that contained the word 'warning', which was part of a comment so was not an actual program warning, thus the message is allowed and a reason is manually added to the report during the review process. For the logs found in C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Graphs directory, GRAPH1_1 ran clean but GRAPH1_3 had two warning messages that were actually non-issues and was documented in the log report. But GRAPH1_3 had some other log messages that contained the word 'ERROR' which is cause for concern and would definitely need to be investigated.

The report will be generated by log location which will allow the person reviewing the report to know exactly where the log was found when it found the unwanted log message. In addition, the log line number associated with the message is also displayed. Note that the line number is associated with the line of code that was closest to the log message and in some cases, such as macro processing, it may not actually refer to the line in the program.

Summary of Log Issues

Log Location=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\ADaM

Log Name	Log Date	Log Time	Line Number	Log Message	Reason Message is Allowed
ADAAPASI	19SEP2016	11:08 AM	15501	NOTE: Missing values were generated as a result of performing an operation on missing values.	OKAY for this deliverable. To be fixed before next deliverable.
	19SEP2016	11:08 AM	15645	NOTE: MERGE statement has more than one data set with repeats of BY values.	
ADPR	07SEP2016	04:00 AM		No undesired messages. Log is clean.	
ADPSGA	19SEP2016	09:03 AM	12708	NOTE: MERGE statement has more than one data set with repeats of BY values.	
ADSL	30AUG2016	06:43 AM	12735	12735 /* otherwise put "warning exendtc= " exendtc;	This is a comment. There is a non-issue.
ADVS	02SEP2016	06:11 AM		No undesired messages. Log is clean.	

Summary of Log Issues

Log Location=C:\Users\user\Desktop\SAS Tips\Check Logs\Logs\Graphs

Log Name	Log Date	Log Time	Line Number	Log Message	Reason Message is Allowed
GRAPH 1_1	13OCT2016	09:49 AM		No undesired messages. Log is clean.	
GRAPH 1_3	13OCT2016	09:49 AM	3797	NOTE: 1 TOTAL ERRORS.	
	13OCT2016	09:49 AM	3797	NOTE: 1 TOTAL ERRORS.	
	13OCT2016	09:49 AM	3797	NOTE: ERROR DETECTED IN ANNOTATE= DATASET WORK.ANNO1.	
	13OCT2016	09:49 AM	3797	NOTE: ERROR DETECTED IN ANNOTATE= DATASET WORK.ANNO1.	
	13OCT2016	09:49 AM	3797	NOTE: ERROR LIMIT REACHED IN ANNOTATE PROCESS. PROCESSING IS TERMINATED.	
	13OCT2016	09:49 AM	3797	NOTE: ERROR LIMIT REACHED IN ANNOTATE PROCESS. PROCESSING IS TERMINATED.	
	13OCT2016	09:49 AM	3797	NOTE: PROCESSING TERMINATED BY INDIVIDUAL ERROR COUNT.	
	13OCT2016	09:49 AM	3797	NOTE: PROCESSING TERMINATED BY INDIVIDUAL ERROR COUNT.	
	13OCT2016	09:49 AM	3797	WARNING: The intervals on the axis labeled "Analysis Visit (N)" are not evenly spaced.	This is expected since the visits where not evenly spaced in regards to time. However, the graph was not a time series. The AVISIT variable was used for grouping the data. This is a non-issue.
	13OCT2016	09:49 AM	3797	WARNING: The intervals on the axis labeled "Analysis Visit (N)" are not evenly spaced.	This is expected since the visits where not evenly spaced in regards to time. However, the graph was not a time series. The AVISIT variable was used for grouping the data. This is a non-issue.

Display 2. Sample CHECKLOGS Report

CONCLUSION

Checking the logs for a deliverable is a task we all dread doing, but it must be done. It is a necessary 'evil'. However, with the CHECKLOGS macro, this process can be simplified while simultaneously producing a report that can be saved with the deliverable to help explain any log messages that were not addressed.

REFERENCES

UNIX commands http://www.shareittips.com/data/unix.png

CMD commands http://ss64.com/nt/

ACKNOWLEDGMENTS

Thanks to Karl Miller and Deanna Schreiber-Gregory for reviewing the paper and providing additional insight.

CONTACT INFORMATION

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

Richann Watson
DataRich Consulting
richann.watson@datarichconsulting.com

APPENDIX A: CHECKLOGS MACRO

```
/* retrieve all the logs in the specified directory */
%macro checklogs(loc=,  /* location of where the log files are stored       */
                        /* can add multiple locations but they need to be   */
                        /* separated by the default delimiter '@' or the    */
                        /* user specified delimiter                         */
                 loc2=, /* location of where report is stored (optional)    */
                 fnm=,  /* which types of files to look at (optional)       */
                        /* e.g., Tables - t_, Figures - f_, Listings - l_   */
                        /* separate types of files by delimiter indicated in*/
                        /* the delm macro parameter (e.g., t_@f_)           */
                 delm=@,/* delimiter used to separate types of files (opt'l)*/
                 msgf=, /* FULL file name (includes location) of spreadsheet*/
                        /* where the user specified log messages are stored */
                 msgs=, /* sheet/tab name in spreadsheet that contains the  */
                        /* unwanted log messages default to 'Sheet1' (opt'l)*/
                 msgv=, /* name of column in spreadsheet (convert spaces to */
                        /* underscores - must be specified if file specified*/
                        /* (conditionally required)                         */
                 out=   /* log report name (optional)                       */);

  /* need to determine the environment in which this is executed   */
  /* syntax for some commands vary from environment to environment */
  %if &sysscp = WIN %then %do;
    %let ppcmd = %str(dir);
    %let slash = \;
  %end;
  %else %if &sysscp = LIN X64 %then %do;
    %let ppcmd = %str(ls -l);
    %let slash = /;
  %end;
  /* end macro call if environment not Windows or Linux/Unix */
  %else %do;
     %put ENVIRONMENT NOT SPECIFIED;
     %abort;
  %end;
```

```
/* if a filename is specified then build the where clause */
%if "&fnm" ne "" %then %do;  /* begin conditional if "&fnm" ne "" */
  data _null_;
    length fullwhr $2000.;
    retain fullwhr;

    /* read in each log file and check for undesired messages */
    %let f = 1;
    %let typ = %scan(&fnm, &f, "&delm");

    /* loop through each type of filename to build the where clause */
    /* embed &typ in double quotes in case filename has special characters/spaces */
    %do %while ("&typ" ne "");  /* begin do while ("&typ" ne "") */

      partwhr = catt("index(flog, '", "&typ", "')");
      fullwhr = catx(" or ", fullwhr, partwhr);

      call symputx('fullwhr', fullwhr);

      %let f = %eval(&f + 1);
      %let typ = %scan(&fnm, &f, "&delm");
    %end;  /* end do while ("&typ" ne "") */

  run;
%end;  /* end conditional if "&fnm" ne "" */

/* if a spreadsheet is provided with unwanted log messages */
/* then need to use that to build search criteria to be    */
/* used later in the program                               */
%let fullmsg = "";
%if "&msgf" ne "" %then %do;  /* begin conditional if "&msgf" ne "" */
  libname logmsg xlsx "&msgf";

  /* need to make sure spreadsheet exists if it is specified */
  %if %sysfunc(fileexist(&msgf)) = 1 %then %do;  /* begin conditional if
                                                    %sysfunc(filexist(&msgf) = 1 */

    data _null_;
      length fullmsg1 fullmsg2 fullmsg3 $2000;
      retain fullmsg1 fullmsg2 fullmsg3;
      set %if "&msgs" ne "" %then logmsg."&msgs"n;
          %else logmsg."Sheet1"n;
          end = eof; /* need this semicolon to end the set statement */

      partmsg = catt('index(upcase(line), "', (upcase(&msgv)), '")');

      if length(fullmsg1) + length(partmsg) <= 2000 then
           fullmsg1 = catx(" or ", fullmsg1, partmsg);
      else if length(fullmsg2) + length(partmsg) <= 2000 then
           fullmsg2 = catx(" or ", fullmsg2, partmsg);
      else fullmsg3 = catx(" or ", fullmsg3, partmsg);

      if eof then do;
               call symputx('fullmsg1', fullmsg1);
                    if fullmsg2 ne '' then call symputx('fullmsg2', fullmsg2);
                    if fullmsg3 ne '' then call symputx('fullmsg3', fullmsg3);
      end;
    run;
  %end;  /* end conditional if %sysfunc(fileexist(&msgf)) = 1 */

  libname logmsg clear;
%end;  /* end conditional if "&msgf" ne "" */
```

```
  /* need to make sure alllogs does not exist before start processing */
  proc datasets;
    delete alllogs;
  quit;

  /* need to process each location for logs separately */
  %let g = 1;
  %let lcn1 = %scan(&loc, &g, "&delm");

  /* attempt to create a libname for each location specified */
  %do %while ("&&lcn&g" ne "");
    %let g = %eval(&g + 1);
    %let lcn&g = %scan(&loc, &g, "&delm");
  %end;

  /* initialize the location that will act as a place holder till the end of the
program */
  %let dloc=;

  /* loop through each directory specified */
  %do k = 1 %to %eval(&g - 1);  /* begin do i = 1 to %eval(&g - 1) */

      /* need to make sure the location exists so create a temp library */
      libname templib&k "&&lcn&k";

      /* begin looking through each log location for specified log types */
      /* if &SYSLIBRC returns a 0 then path exists                       */
      %if &syslibrc = 0 %then %do;  /* begin if &syslibrc = 0 */

           /* create default location for report if report location not specified */
           /* only create default location for first location that exists         */
           %if "&dloc" = "" %then %do;
              %let dloc = &&lcn&k;
             %end;

          /* need to build pipe directory statement as a macro var  */
          /* because the statement requires a series of single and  */
          /* double quotes - by building the directory statement    */
          /* this allows the user to determine the directory rather */
          /* than it being hardcoded into the program               */
          /* macro var will be of the form:'dir "directory path" '  */
          data _null_;
            libnm = "&&lcn&k";
            dirnm = catx(" ", "'", "&ppcmd", quote(libnm), "'");
            call symputx('dirnm', dirnm);
          run;

          /* read in the contents of the directory containing the logs */
          filename pdir pipe &dirnm lrecl=32727;

          data logs&k (keep = flog fdat ftim filename numtok);
            infile pdir truncover scanover;
            input filename $char1000.;

            length flog $50 fdat ftim $10;

            /* keep only the logs */
            if index(filename, ".log");

            /* count the number of tokens (i.e., different parts of filename) */
            /* if there are no spaces then there should be 5 tokens           */
            numtok = countw(filename,' ','q');
```

```
  /* parse out the string to get the log name */
  /* note on the scan function a negative #   */
  /* scans from the right and a positive #    */
  /* scans from the left                      */
  /* for log keep only first part (e.g. 'ae') */
  /* need to build the flog value based on number of tokens */
  /* if there are spaces in the log name then need to grab  */
  /* each piece of the log name                             */
  /* the first token that is retrieved will have '.log' and */
  /* it needs to be removed by substituting a blank         */
  /* need to do within conditional if statements since num  */
  /* of tokens for Windows is different than Unix           */
  *flog = scan(scan(filename, -1, " "), 1);

  /* entire section below allows for either Windows or Unix */
  /*********** WINDOWS ENVIRONMENT ************/
  /* the pipe will read in the information in */
  /* the format of: date time am/pm size file */
  /* e.g. 08/24/2015 09:08 PM 18,498 ae.log   */
  /*    '08/24/2015' is first token from left */
  /*    'ae.log' is first token from right    */
  %if &sysscp = WIN %then %do;  /* begin conditional if &sysscp = WIN */
    do j = 5 to numtok;
      tlog = tranwrd(scan(filename, 4 - j, " "),  ".log", "");
      flog = catx(" ", tlog, flog);
    end;
    ftim = catx(" ", scan(filename, 2, " "), scan(filename, 3, " "));
    fdat = put(input(scan(filename, 1, " "), mmddyy10.), date9.);
  %end;  /* end conditional if &sysscp = WIN */

  /************************ UNIX ENVIRONMENT ************************/
  /* the pipe will read in information in the format of: permissions, user,*/
  /* system environment, file size, month, day, year or time, filenam      */
  /* e.g. -rw-rw-r-- 1 userid sysenviron 42,341 Oct 22 2015 ad_adaapasi.log*/
  /*    '-rw-rw-r--' is first token from left                              */
  /*    'ad_adaapasi.log' is first token from right                        */
  %else %if &sysscp = LIN X64 %then %do;  /* begin conditional if
                                             &sysscp = LIN X64 */

    do j = 9 to numtok;
      tlog = tranwrd(scan(filename, 8 - j, " "),  ".log", "");
      flog = catx(" ", tlog, flog);
    end;

    _ftim = scan(filename, 8, " ");

    /* in Unix if year is current year then time stamp is displayed */
    /* otherwise the year last modified is displayed                */
    /* so if no year is provided then default to today's year and if*/
    /* no time is provided indicated 'N/A'                          */
    if anypunct(_ftim) then do;
      ftim = put(input(_ftim, time5.), timeampm8.);
      yr = put(year(today()), Z4.);
    end;
    else do;
      ftim = 'N/A';
      yr = _ftim;
    end;

    fdat = cats(scan(filename, 7, " "), upcase(scan(filename, 6, " ")), yr);
  %end;  /* end conditional if &sysscp = LIN X64 */

run;
```

```
/* create a list of logs, dates, times and store in macro variables */
/* count number of logs in the specified folder and retain in macro variable */
proc sql noprint;
  select flog,
         fdat,
         ftim,
         count (distinct flog)
         into : currlogs separated by "&delm",
              : currdats separated by " ",
              : currtims separated by "@",
              : cntlogs
  from logs&k
  %if "&fnm" ne "" %then where &fullwhr;
    ; /* need to keep extra semicolon */
quit;

/* only loop thru the directory if the number of logs found is greater than 0 */
%if &cntlogs ne 0 %then %do;  /* begin conditional if &cntlogs ne 0 */

  /* read in each log file and check for undesired messages */
  %let x = 1;
  %let lg = %scan(&currlogs, &x, "&delm");
  %let dt = %scan(&currdats, &x);
  %let tm = %scan(&currtims, &x, '@');

  /* loop thru each log in the directory and look for undesirable messages */
  /* embed &lg in double quotes in case filename has special characters/spaces */
  %do %while ("&lg" ne "");  /* begin do while ("&lg" ne "") */
    /* read the log file into a SAS data set to parse the text */
    data logck&k&x;
      infile "&&lcn&k.&slash.&lg..log" truncover pad;

      /* use $char1000 in order to maintain spacing to get correct line number */
      input line $char1000.;

      /* need to retain the line number so that when a message is encountered */
      /* then will know hwere in log to find it                               */
      retain lineno;

      if _n_ = 1 then lineno = .;
      if anydigit(line) = 1 then
         lineno = input(substr(line, 1, notdigit(line)-1),best.);

      /* keep only the records that had an undesirable message */
      if index(upcase(line), "WARNING") or
         index(upcase(line), "ERROR") or
         index(upcase(line), "UNINITIALIZED") or
         index(upcase(line), "NOTE: MERGE") or
         index(upcase(line), "MORE THAN ONE DATA SET WITH REPEATS OF BY") or
         index(upcase(line), "VALUES HAVE BEEN CONVERTED") or
         index(upcase(line), "MISSING VALUES WERE GENERATED AS A RESULT") or
         index(upcase(line), "INVALID DATA") or
         index(upcase(line), "INVALID NUMERIC DATA") or
         index(upcase(line), "AT LEAST ONE W.D FORMAT TOO SMALL")
         /* allow for user specific messages to be stored in a spreadsheet */
         %if %symexist(fullmsg1) %then or &fullmsg1;
         %if %symexist(fullmsg2) %then or &fullmsg2;
         %if %symexist(fullmsg3) %then or &fullmsg3;
         ; /* need extra semicolon to end if statement */

      /* create variables that will contain the log that is being scanned */
      /* as well as the and date and time that the log file was created   */
```

```
  length lognm $25. logdt logtm $10. loglc $200.;
  lognm = upcase("&lg");
  logdt = "&dt";
  logtm = "&tm";

  /* create a dummy variable to create a column on report that will allow */
  /* users to enter a reason if the message is allowed                    */
  logrs = ' ';

  /* need to create a variable that captures the location */
  /* in case there are multiple log locations - need to be*/
  /* print the report by log location                     */
  /* nolog will be used to flag the directories with no   */
  /* logs found                                           */
  loglc = "&&lcn&k";
  label loglc = 'Log Location';
  nolog = .;
run;

/* because there are sometimes issues with SAS certificate */
/* there will be warnings in the logs that are expected    */
/* these need to be removed                                */
data logck&k&x._2;
  set logck&k&x.;
  if index(upcase(line), 'UNABLE TO COPY SASUSER') or
     index(upcase(line), 'BASE PRODUCT PRODUCT') or
     index(upcase(line), 'EXPIRE WITHIN') or
     (index(upcase(line), 'BASE SAS SOFTWARE') and
      index(upcase(line), 'EXPIRING SOON')) or
     index(upcase(line), 'UPCOMING EXPIRATION') or
     index(upcase(line), 'SCHEDULED TO EXPIRE') or
     index(upcase(line), 'SETINIT TO OBTAIN MORE INFO') then delete;
run;

/* determine the number of undesired messages were in the log */
data _null_;
     if 0 then set logck&k&x._2 nobs=final;
  call symputx('numobs',left(put(final, 8.)));
run;

/* if no undesired messages in log create a dummy record for report */
%if &numobs = 0 %then %do;  /* begin conditional if &numobs = 0 */
  data logck&k&x._2;
    length lognm $25. line $1000. logdt logtm $10. loglc $200.;
    line = "No undesired messages.  Log is clean.";
    lognm = upcase("&lg");
    logdt = "&dt";
    logtm = "&tm";

    /* create a dummy variable to create a column on the report that */
    /* allow users to enter a reason if the message is allowed       */
    logrs = ' ';

    /* adding variables for line number and log location */
    /* nolog will be used to flag the directories with   */
    /* no logs found                                     */
    lineno = .;
    loglc = "&&lcn&k";
    label loglc = 'Log Location';
    nolog = .;
    output;
  run;
%end;  /* end conditional if &numobs = 0 */
```

```
            /* append all the results into one data set */
            /* need to check to see if data set exists-during 1st iteration it should */
            /* not exist because it was deleted at top of program                      */
            %let exist = %sysfunc(exist(alllogs));

            %if /*&x = 1*/ &exist = 0 %then %do;
              data alllogs;
                set logck&k&x._2;
              run;
            %end;
            %else %do;
              proc append base=alllogs
                          new=logck&k&x._2;
              run;
            %end;

            %let x = %eval(&x + 1);
            %let lg = %scan(&currlogs, &x, "&delm");
            %let dt = %scan(&currdats, &x);
            %let tm = %scan(&currtims, &x, '@');
          %end;  /* end do while ("&lg" ne "") */

        %end;  /* end conditional if &cntlogs ne 0 */

        %else %do;
          data nolog&k;
            length lognm $25. line $1000. logdt logtm $10. loglc $200.;
            line = '';
            lognm = '';
            logdt = '';
            logtm = '';
            logrs = '';
            lineno = .;
            loglc = "&&lcn&k";
            label loglc = 'Log Location';
            nolog = 1;
            output;
          run;

          %let exist = %sysfunc(exist(alllogs));

          %if &exist = 0 %then %do;
            data alllogs;
              set nolog&k;
            run;
          %end;
          %else %do;
            proc append base=alllogs
                        new=nolog&k;
            run;
          %end;
        %end;
      %end;  /* end if &syslibrc = 0 */

      %else %do;
        %put %sysfunc(compress(W ARNING:)) "directory &&lcn&k does not exist";
      %end;

%end;  /* begin do i = 1 to %eval(&g - 1) */

/* if the name of the output file is not specified then default to the name */
%if "&out" =  "" %then %do;
```

```
    %let out=all_checklogs;
  %end;

  /* if the name of the output file is not specified then default to the name */
  %if "&loc2" = "" %then %do;
    data _null_;
      call symputx("loc2", "&dloc");
    run;
  %end;

  %let exist = %sysfunc(exist(alllogs));
  %if &exist ne 0 %then %do;
        /* since a list of files can be provided then the files may not be in order */
        proc sort data=alllogs presorted;
          by lognm line;
        run;

        /* sort the final report by location */
        proc sort data = alllogs;
          by loglc nolog lognm lineno;
        run;

        /* create the report */
        ods listing close;
        options orientation=landscape missing='';

        ods rtf file="&loc2.&slash.&out..rtf";
        title "Summary of Log Issues";

        /* nolog will determine if message about no */
        /* logs found in directory will be printed  */
        proc report data=alllogs ls=140 ps=43 spacing=1 missing nowindows headline;
          by loglc;
          column nolog lognm logdt logtm lineno line logrs;
          define nolog / analysis sum noprint;
          define lognm / order   style(column)=[width=12%]      "Log Name";
          define logdt / display style(column)=[width=12%]      "Log Date";
          define logtm / display style(column)=[width=12%]      "Log Time";
          define lineno/ display style(column)=[width=12%]      "Line Number";
          define line  / display style(column)=[width=30%] flow "Log Message";
          define logrs / display style(column)=[width=20%] flow "Reason Message is Allowed";

          /* force a blank line after each file */
          compute after lognm;
            line " ";
          endcomp;

          /* if there are no logs in the directory then display message indicating that */
          /* do this to verify that the directory was indeed checked and not overlooked */
          compute after _page_;
            if nolog.sum ^= . then addnote='There are no log files found in the directory';
            line @20 addnote $50.;
          endcomp;
        run;

        ods rtf close;
        ods listing;
  %end;
  %else %do;
        %put %sysfunc(compress(W ARNING:)) "None of the log locations specified exist";
  %end;
%mend checklogs;
```