PharmaSUG 2024 - Paper AP-252

Externally Yours – Adeptly Managing Data Outside Your EDC System

Frank Canale, SoftwaRx LLC, St. Augustine, FL

ABSTRACT

Programmers in the pharmaceutical industry are used to working with data that is entered into, and extracted from, a system commonly known as an EDC (Electronic Data Capture) system. When using data that is sourced from one of these systems, you can reliably count on the type of data you will receive (normally SAS datasets), and if the EDC is set up well, a standard structure that provides output data containing CDISC/CDASH variable names. But what does one do when receiving data that is sourced outside the EDC system and received from other vendors? How do you manage this data…retrieve it…validate the structure…even export it to a format allowing you to merge it with other more conventional SAS datasets?

INTRODUCTION

Data resulting from a clinical trial takes on many forms…Demography, Medical History, Physical Exam, Adverse Events, etc. As stated above - normally this data is entered into (and subsequently extracted from) an EDC system, to be processed into CDISC data structures (first SDTM - then ADaM) and subsequently into TLF output files. One of the best things about an EDC system and its resulting data files is that users can create modules for each data type and hopefully, utilize those data modules within the system with CDISC/CDASH variable names as the standard for several protocols within a single compound, or even better – across the entire organization in some circumstances.

However, there are other forms of data that emanate from outside sources and are never entered into the EDC system…for example, data could be received from external clinical laboratories, ECG/EKG centers, PK/PD labs, collectors of external efficacy measurements/parameters, and many other sources. These external, non-conventional data sources present many different challenges. Not only is the external data not standard or housed in a standard, reusable structure, but there are also the following issues - how does one validate the structure and some of the content of the external data? How can you be sure your external data structure matches the proposed structure found within the data transfer agreement between your organization and an outside lab/external data provider? How can you be sure that controlled terminology values within the external data file will match with those called for within the data transfer agreement? How do you compare the most recently transferred data with the previously transferred data file to ensure that you did not receive less records in the current transfer than you had in the previous transfer – or to insure consistency in the structure

itself across transfers? Your edit checks within EDC will not perform these functions…your externally written SAS edit checks will not do the trick either.

This paper and presentation will display and depict one method using some tools written in basic SAS code and SAS macro that will help manage this situation more efficiently.

THE CHALLENGES

The challenges presented with externally received data are numerous and varied and consist of the following:

- Numerous data types
  - SAS datasets
  - Excel/XLSX
  - Comma Separated files (CSV)
  - XPT/SAS export files

- Variable naming and data structure issues between the data and the agreement
  - Variable names within Excel files (xlsx) contain spaces
  - Variable names within the transfer don't match those proposed and accepted within the data transfer agreement
  - In some rare cases, variable names contain special characters (%, &, etc.)
  - Variable length issues
  - Variable type issues (character vs numeric)
  - Variables that may exist in one source (agreement) but not within the transfer itself, or vice versa

- Sometimes the transferred files reside within ZIP file folders – we need a way to extract this data within a SAS process

- Sometimes the transferred files have dates in the file names

In my experience, I have had situations where I was lucky enough to have the ability to manage the external data providers and request prior to having the transfer agreement accepted and signed off on, that they try to adhere to certain standards and practices before sending my sponsor company the external data. I would request that they send SAS files or XPT/SAS export files as the preferred form, followed by Excel. I would request that they NOT put the date of the external file creation in the file name (they could put this within the name of the ZIP file containing the data but NOT in the name of the file itself). I would try to guide them with variable naming (not utilize spaces or special characters, use CDISC/CDASH terminology or sponsor-centric names to facilitate merging with the EDC data, etc.). Finally, I would ask them to provide the data in the same structure for every transfer.

For my last experience with this process, I was not so lucky. I had to work with external data where the agreement had long been approved prior, and the transfers were what they were. I couldn't control this process - so I had to come up with a solution that

worked within this limitation. But a limitation isn't always a limitation…it's an opportunity! Therefore, I wrote some macro code that got around just about every situation presented to me while working with the external data sources. I wrote a general macro that searched for, and reported on, data structure and controlled terminology issues for just about every external transfer the company had received. For one specific transfer that was well outside of the parameters allowing a general purpose macro, I created an ad-hoc program that could be called within the main general macro with the passing of two macro parameters…this way, I could call the main macro, pass a value into an optional parameter, and the main macro would branch off into the ad-hoc code, provide the structure issues, and then branch back to the main macro to create an issue report.

A few factors to consider when working with external data transfers:

- The external data transfers are being created within a Lab or other medical processing organization…NOT a data management or programming organization
- The providers consist of lab workers or clinicians…NOT data managers or programmers
- Conversely…WE are the data and programming experts/SMEs - so it is on US to insure the data quality

Knowing these factors helps reduce the frustration when working with external data…if you understand these things, it keeps you focused. The providers did their job providing the data…we are responsible for doing our job of ensuring the quality of the data and guide the providers as needed.

The MS_DTS_CONTENT_CHECK macro

The MS_DTS_CONTENT_CHECK macro is incredibly involved and detailed. This macro will perform the following checks on external data files:

- Compares the contents of the transfer to expected (per a metadata spreadsheet described below...variable names, variable length, variable types)
- Compares the values of variables with expected controlled terminology to the values in the controlled terminology values in the metadata spreadsheet
- Compares the current transfer to the previous transfer...make sure there are an equal or greater number of records in the current transfer vs the previous transfer
- Compares structures between the current transfer and the previous transfer and note any differences between the two (if a previous transfer does not exist, these 2 checks are not performed)
- For each issue found, the macro creates a discrepancy record and outputs this to a report...if no discrepancies are found, messages are written to the log and output file noting this.

The following is a description of the macro parameters:

- CURRDIR : required - Directory/folder where current transfer file exists

- FILENAME : required - Name of the actual transferred file to check (ex: LABS, PK, <STDY>_BWH CMR_CMR_[date])...if the FILENAME parameter contains the string "[date]", then the DSDATE parameter must be passed in...the macro will substitute the "[date]" string for the value in DSDATE

- PREVDIR : optional - Directory/folder where previous transfer file exists (if not passed in, macro will not perform a comparison of the current transfer to the prior one, in case the current transfer is the first one)...if the previous transfer is contained within a ZIP file, the string ".zip" must be added to the value of PREVDIR (example:prevdir=G:\Prod\<PROJ>\<STDY>\analysis\final_lock\ rawdata\other\archive\ <STDY>_CELERION_PK_14SEP2023.zip")

- DSDATE : optional - Date value in case the transfer file itself contains a date in the name (ex: 18SEP2023, 20230918)...must be passed in exactly the way it appears in the name of the file or the macro will produce a file not found error in the log and stop processing

- VENDOR : optional - Name of the vendor (ex: ICON LABS, Synteract)...this is needed in case there are multiple instances of a filename for different vendors...if passed in the macro will select records from the metadata file based on the values of the FILENAME and VENDOR parameters

- STDYMAC : optional - Name of a macro containing study-specific checks created just for the study or situation...if this is passed in, the study-specific macro must exist

- STDYN : optional - Tells the macro whether to run the standard checks or proceed directly to the study-specific macro call. Default=Y (meaning to run the standard checks), valid responses are Y or N

- DEBUG : optional - Tells macro whether to keep the work datasets, for debugging/QC purposes only. Default=N, valid responses are Y or N. “Y’ means to keep the work datasets for further debugging or QC, “N” means to delete the work datasets.

Sample Call:

```
%ms_dts_content_check(currdir=G:\Prod\<PROJ>\<STDY>\analysis\final_lock\rawdata\other,
prevdir=G:\Prod\<PROJ>\<STDY>\analysis\final_lock\rawdata\other\archive\<PROJ>_BWH_CMR
CMR 15JUN2023,filename=<PROJ>_BWH CMR_CMR_[date],dsdate=20230715,debug=Y);
```

This macro will note the following macro use errors and issues:

ERRORS

- The CURRDIR and FILENAME macro parameters must be passed in (both are required)

- If the FILENAME parameter contains the string "[date]" or "[DATE]", the DSDATE parameter must be passed in

- Conversely, if the DSDATE parameter is passed in, the FILENAME parameter must contain the string "[DATE]" or "[date]"

- The value passed into the FILENAME parameter must exist in the metadata file

- The external transfer file in the FILENAME parameter must exist in the folder passed into the CURRDIR parameter

- If any of the above issues occur, the macro will produce error messages and stop processing

ISSUES:

- If no Controlled Terminology exists in the metadata file for the file passed into FILENAME, then no controlled terminology checks are performed, and a Note is written to the log

- If a folder is passed into the optional parameter PREVDIR, the external file in the FILENAME parameter must exist in the PREVDIR folder...if it does not, no checks between the current and previous transfer are done and a Note is written to the log

OTHER ITEMS:

- If no comparison issues between the metadata and the current transfers are found, this is noted in the log and the output report

- If no comparison issues between the previous and the current transfers are found, this is noted in the log and the output report

- If no controlled terminology issues are found, this is noted in the log and the output report

PREREQUISITES

- Before using the macro, the creation of an Excel file containing metadata from your organization's external transfer agreements is necessary. The macro will use this spreadsheet as the "master" for comparison.
- In one study, there were @15 separate external transfers, and each one had an agreement…I was able to create the metadata file for this study in about 3 days' time.

- Creating the metadata file is “busy work” and a bit tedious but once it’s created, it’s mandatory for this process and a very valuable tool
- The Excel workbook file is named <STDY>_DTS_METADATA.XLSX
- There are 2 worksheets needed within the metadata Excel workbook:
    - Datasets: Contains the expected Vendor Name, External File Name, Field Names, Field Lengths, Field Types (character or numeric), File Type, and the name of the Controlled Terminology applicable to each field
    - Controlled Terminology: Contains the Controlled Terminology Name, Field Name, and an entry for each expected value within that field.
- Worksheet Examples:

DATASETS

VENDOR	DATASET	VARIABLE	LENGTH	VARTYPE	FILETYPE	CT
ICON	ICON_LABS	SUBJECT	30	CHAR	SAS	
ICON	ICON_LABS	SEX	6	CHAR	SAS	ICON_SEX
ICON	ICON_LABS	RACE	100	CHAR	SAS	ICON_RACE
ICON	ICON_LABS	AGE	8	NUM	SAS	
ICON	ICON_LABS	LABDT	10	CHAR	SAS	
ICON	ICON_LABS	LBTESTCD	20	CHAR	SAS	ICON_TESTCD
ICON	ICON_LABS	LBTEST	60	CHAR	SAS	ICON_TEST
ICON	ICON_LABS	LBORRES	8	NUM	SAS	
ICON	ICON_LABS	LBORRESU	10	CHAR	SAS	

…etc…

CONTROLLED TERMINOLOGY

CT	VARIABLE	CTVALUE
ICON_SEX	SEX	MALE
ICON_SEX	SEX	FEMALE
ICON_RACE	RACE	WHITE
ICON_RACE	RACE	BLACK OR AFRICAN AMERICAN
ICON_RACE	RACE	ASIAN
ICON_RACE	RACE	PACIFIC ISLANDER
…etc…		
ICON_TESTCD	LBTESTCD	ALKPHOS
ICON_TESTCD	LBTESTCD	BILI
ICON_TESTCD	LBTESTCD	CREAT
ICON_TESTCD	LBTESTCD	CHOL
…etc…		
ICON_TEST	LBTEST	Alkaline Phosphorus
ICON_TEST	LBTEST	Bilirubin
ICON_TEST	LBTEST	Creatinine
ICON_TEST	LBTEST	Cholesterol
…etc…		

MACRO PROCESSING

Initial Process and Metadata Processing

The initial process of this macro and metadata processing as follows:

- Perform initial error checking:
  - Current Directory (CURRDIR) and File Name (FILENAME) parameters are required and therefore must not be missing
  - If the FILENAME parameter contains the string "[DATE]", the DSDATE parameter must not be missing, and vice-versa. These two parameters work in conjunction with each other to process files with dates in the name
  - If STDYN is passed in as N, the STDYMAC parameter must not be missing/null (STDYN=N tells the macro to bypass standard processing and run a study-specific macro)
  - If any of the above issues occur, the macro returns an error message to the log and aborts processing.

- Acquire metadata from the above-described metadata Excel Worksheet for both the external file and any associated controlled terminology. If the macro parameter VENDOR is passed in, add an additional condition to only select records for that specific vendor and the file passed into FILENAME (as described above, the VENDOR parameter allows the user to generate processing on a specific vendor's files in the event that several vendors have provided files named the same).

- Create 2 SAS datasets – one containing the actual file metadata depicted above (field names/lengths/types, file type - SAS, XLSX, XPT, CSV), controlled terminology value list name…and another one containing the actual values for each variable needing controlled terminology. If no controlled terminology exists for the external file, set a macro variable telling the macro not to run those checks.

- Check to ensure that the requested external file exists…if not, present an error message within the log, and abort processing…otherwise, continue.

- Using the parameters FILENAME and DSDATE, allow processing on external files with dates in the name of the file. The macro will check to see if the FILENAME parameter contains the string "[DATE]" and if so, it will replace that string in the parameter with the date found within the parameter DSDATE. Naturally, this date must be in the same format and value as found within the filename itself…if this is not the case, an error will be produced that the file isn't found, and processing will abort as described in the point above.

- Using the FILETYPE field within the Excel metadata spreadsheet, create a SAS dataset for the file called for in the FILENAME parameter. The FILETYPE field value dictates how to do this:
    - CSV - PROC IMPORT with DBMS of CSV
    - XLSX - PROC IMPORT with DBMS of EXCEL2007
    - XPT - PROC COPY with XPORT Libname
    - SAS - PROC SQL using CURRDIR parameter value as Libname
    - See below Macro code for details

- Perform a PROC CONTENTS on the SAS dataset created in the above step to acquire the metadata on the actual data file, and output this to another SAS dataset. This will be compared with the values found in the metadata Excel spreadsheet items described above. One note about the dataset created using the metadata fields in the Excel spreadsheet created from the transfer agreement…when an external file is converted from CSV or Excel into SAS, the conversion will replace spaces in the field names with underscores ("_"). The same will need to be done for the items found in the dataset created from the metadata spreadsheet and the macro handles this.

- Start to perform checks of the actual external data metadata versus the metadata called for within the transfer agreement:
    - Compare variable types (character vs numeric)
    - Compare variable lengths (if available in the metadata spreadsheet…some transfer agreements do not provide these so they will not be in the metadata spreadsheet. Also, this only applies to SAS or XPT files, since an Excel to SAS conversion automatically assigns lengths using the values in the fields)
    - Check to see if variables that are called for in the metadata file exist in the external file, or if variables exist in the external file that are not called for in the metadata file
    - If any of the above checks fails, output issue records to a temporary error dataset…otherwise, if none of the checks fail, create a record saying that no errors/issues were found, and write a message to the log. The output dataset will be reported on later in the macro.

Checks of Current Transfer to Prior Transfer

The macro allows a user to also compare contents of the current externally transferred file to a previous version of the file. This process is as follows:

- If the macro parameter PREVDIR is passed in, this will enact this part of the process…if not, no checks of current to prior transfer are done.

- First, check to see if the previous version of the external file is found within a ".zip" folder. The ".zip" extension must be passed into PREVDIR for this check to happen. If the previous file exists within a zip folder, set an internal macro variable to "Y"…otherwise, set it to "N".

- If the FILENAME parameter contains the string "[DATE]" and DSDATE contains a date (checked for above), check for the dated version of the file by replacing the "[DATE]" string with the value in DSDATE parameter…if this file does not exist, bypass the checks of the current transfer to the prior transfer.

- If the previous transfer file indeed exists within a .zip folder (internal macro variable set to "Y" in the second step above), some special processing is needed to find and utilize this transfer file:
    - Create a ZIP LIBNAME of the folder passed into the PREVDIR parameter
    - Create a contents dataset of the files within the Zip Libname, using the DOPEN, DNUM, and DREAD functions
    - Read this contents dataset and search for the external file called for in the FILENAME parameter…if found, pass it into a macro parameter, and create a copy of it within the WORK folder created in the SAS session. If not found, send a message to the log and bypass the checks of the current transfer to the prior transfer.
    - The above processing was found within our favorite website: **lexjansen.com** (See Reference 1 below for actual publication)
    - Please see the code below for details

- Once the prior transfer file is located, convert it to a SAS dataset as follows:
    - CSV - PROC IMPORT with DBMS of CSV
    - XLSX - PROC IMPORT with DBMS of EXCEL2007
    - XPT - PROC COPY with XPORT Libname
    - SAS - PROC SQL using CURRDIR parameter value as Libname
    - See below Macro code for details

  A few notes on this…if the prior transfer file is in XPT format, the subsequent converted dataset will need to be renamed with "_PRV" as a suffix so the current transfer dataset is not overwritten by the PROC COPY process.

- Once the prior external transfer file is found and converted to SAS, perform metadata checks as follows:

- Check to ensure there are the same or greater number of observations in the current transfer versus the prior transfer. The assumption here is that all external transfers are cumulative (all data contained…this is normally the case). The latest transfer must have the same number or more observations as the prior version.
- Check to ensure that both the current and prior transfer files both have the same fields contained within.
- Check to ensure that the field types are the same across both the current and the prior transfer files
- Check to ensure that the field lengths are the same across both the current and prior transfer files
- If any of the above checks fails, output issue records to a temporary error dataset…otherwise, if none of the checks fail, create a record saying that no errors/issues were found, and write a message to the log. The output dataset will be reported on later in the macro.

Controlled Terminology Processing

The macro will check the data within the external transfer for any controlled terminology issues. For this process, Controlled Terminology entails expected values for certain fields. These are values you would normally expect to exist within a field, and no other fields are permissible within these fields. Examples of this are Gender/Sex, Race, Lab Test Codes, Lab Test Descriptions, and the like. The process of checking controlled terminology within this macro is as follows:

- Acquire all fields within the metadata where controlled terminology values exist…get the variable name, the variable type, and the controlled terminology name (found in the metadata examples above). The type is needed because the process differs very slightly between character variables and numeric ones.

- For each variable that contains controlled terminology, perform a merge of the external file data with the controlled terminology values. For example, if the external transfer file contains controlled terminology for variables SEX and RACE, perform the merge and check for both variables.

- Then, perform the following checks:
  - The controlled terminology variable in the external transfer is blank/has no value
  - A value is found within the external transfer variable that does not exist within the expected controlled terminology
  - If any of the above checks fails, output issue records to a temporary error dataset…otherwise, if none of the checks fail, create a record saying that no errors/issues were found, and write a message to the log. The output dataset will be reported on later in the macro.

Study-Specific Processing

The MS_DTS_CONTENT_CHECK macro was created in a fashion to efficiently perform standard checks on externally transferred data files. This macro was written with certain expectations in mind. However, as we well know, not everything in our industry is standard, or expected! Many times, a data file is sent to us that goes against the grain and requires us to process it differently. In my experience while creating this macro, a certain transfer would not allow me to perform much of the above-described process. Here are some items that caused this additional challenge:

- Field names in the first record of the Excel transfer were not useful (i.e. F1, F2, F3, etc.)
- Actual field names in the Excel transfer were in the second record
- The actual field names contained special characters not transferrable to SAS in a simple PROC IMPORT…examples are as follows:
    - % and & signs
    - '/' or '\' characters
    - '-' (dashes)
    - Other "unusable" characters
- Date fields contained values other than dates
- Study-specific cross-field content checks were asked for by the statistician…these required checking values of one field to those of another (the main macro did not perform actual data checks except for the controlled terminology checks described above)

In addition, the controlled terminology checks described above needed to be performed.

Therefore, I had to create a completely different program. In addition, I had to find a way to have the MS_DTS_CONTENT_CHECK main macro control this process. Therefore, the STDYN and STDYMAC parameters were added to that macro. The processing used those 2 additional parameters to control this. A user then could create a "driver" program that will run all external transfer checks in the single program, including the standard checks, and special checks described above.

If a user wishes to only perform the study-specific or file-specific check using a non-standard macro, they can do so using the MS_DTS_CONTENT_CHECK macro. In my case, I created a special program that referenced the non-standard external file and passed in STDYN=N and the name of the study-specific program within the STDYMAC parameter. In this case, the main macro will see that these parameters were set accordingly and will then immediately branch to a section that will generate the study-specific macro/program (the program should be set up as a macro). The main program will check to ensure that the study-specific macro/program exists…if not, an error message is generated, and the study-specific processing is not done. Otherwise, the study-specific macro/program is generated. Once this is done, the process branches back to the main macro for error data processing and reporting.

Error Dataset Processing, Reporting, and Process End

Once all checks are performed as described above, all temporary error datasets are set into a main "bucket" and each error is categorized and ordered. This main dataset is sorted, and a final error/issue report is created in Excel. The Excel report is created for each externally transferred file and is named using the study, external file passed into the FILENAME parameter, and run date as follows:

<STUDY>_content_check_<DATE>_<TRANSFER FILE NAME>.xlsx

An example of an expected output Excel error report is as follows:

Type of Check Performed	Transfer File Name	Variable Name	Description of Issue
Check of DTS Transfer file contents to metadata	ICON_LABS	LBTEST	Differences found in variable length between DTS document metadata (22) and transfer file (21).
Check of Current DTS Transfer file to Previous DTS Transfer file	ICON_LABS	LBORRES	Differences found in variable type between Current transfer (CHAR) and Previous transfer (NUM).
Check of controlled terminology values to metadata	ICON_LABS	LBTESTCD	Value for variable LBTESTCD (CRAET) for observation 7 not found within Controlled Terminology ICON_LABS_TESTCODE.
…etc…			

Creation of this report ends the main process.

Finally, the macro will save the work datasets for debugging and QC if desired. A simple passing of "Y" to the DEBUG parameter will maintain the work datasets.

CONCLUSION

Working with data that is received from an external lab or outside clinical organization can be quite painful and difficult, but there are ways to utilize the SAS and Excel tools already found within your organization to lessen this pain and difficulty somewhat. We can harness the power that SAS provides us to manage just about any data source…using this power, we can access the external data and quickly provide details on potential structure issues and report on how an external transfer may not be following the agreed upon format. In addition, many sections of the MS_DTS_CONTENT_CHECK macro can be copied and modified for additional usage when accessing/merging/reporting on externally transferred data.

The author attempted to present one such process using basic SAS and SAS macro code to efficiently manage external data files and uncover potential issues with the transfers.

REFERENCES

1) Article in Conference Proceedings: Rick Langston, "Reading and Writing ZIP Files with SAS®" - SAS Institute Inc., Cary, NC
   Available at https://support.sas.com/resources/papers/proceedings14/SAS264-2014.pdf

CONTACT INFORMATION

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

Frank Canale, SoftwaRx, LLC
E-mail: frankocb@yahoo.com

APPENDIX – ENTIRE MACRO

Here is the entire macro. As with any piece of existing SAS code adapted for other purposes, please be sure to make the proper modifications for use in your use within your organization. There are a few items within the code that are specific to the client I was working with and will need to be edited/changed to allow the macro to work properly within your organization:

- References to CK_STUDY, CK_TLF, and CK_AUTO3…these were macro variables that were created within my former client's initial setup program for the Study, TLF folder location, and Program folder location – these will need to be altered for the same purposes per your setup program, or will need to be set up within the calling program
- Folder location specifications – were centric to my former client and will need to be changed accordingly
- References to RAWDATA/OTHER – this is the location of the current external transfer file and will obviously need to be changed per your organization/folder structure
- Etc. as needed

```
/*************************************************************************
PRODUCT      : External data transfer acceptance macro
PROJECT      : <ALL>
PROGRAM      : ms_dts_content_check.sas
PURPOSE      : Performs checks on externally transferred data sources...the following checks are performed:
                      - Compare the contents of the transfer to expected (per the metadata spreadsheet...variable names,
                     variable length, variable types)
                      - Compare the values of variables with expected controlled terminology to the values in
                        the controlled terminology values in the metadata spreadsheet
                      - Compare the current transfer to the previous transfer...make sure there are an equal or greater
                        number of records in the current transfer vs the previous transfer...also, compare structures  between the
                        current transfer and the previous transfer and note any differences between the two  (if a previous transfer
                        does not exist, these checks are not performed) For each issue found, create a discrepancy record and output
                        to a report...if no discrepancies are found, messages                 are written to the log noting this.

                  Also, this macro will note the following macro use errors and issues:
                        ERRORS:
                        - The CURRDIR and FILENAME macro parameters must be passed in (both are required)
                        - If the FILENAME parameter contains the string "[date]" or "[DATE]", the DSDATE parameter must be passed in
                        - Conversely, if the DSDATE parameter is passed in, the FILENAME parameter must contain the string "[DATE]" or
                        "[date]"
                        - The value passed into the FILENAME parameter must exist in the metadata file
                        - The external transfer file in the FILENAME parameter must exist in the folder passed into the CURRDIR
                        parameter
                        - If any of the above issues occur, the macro will produce error messages and stop processing
```

ISSUES:
- If no Controlled Terminology exists in the metadata file for the file passed into FILENAME, then no controlled terminology checks are performed and a Note is written to the log
- If a folder is passed into the optional parameter PREVDIR, the external file in the FILENAME parameter must exist in the PREVDIR folder...if it doesn't, no checks between the current and previous transfer are done and a Note is written to the log

OTHER ITEMS:
- If no comparison issues between the metadata and the current transfers are found, this is noted in the log and the output report
- If no comparison issues between the previous and the current transfers are found, this is noted in the log and the output report
- If no controlled terminology issues are found, this is noted in the log and the output report

Author : Frank Canale
Created : 18-Sep-2023
SAS Version : Windows 9.4
Input : N/A
Output : Excel report containing list of discrepancies found, or record with message if no discrepancies found

Parameters :

- CURRDIR : required - Directory/folder where current transfer file exists (if not passed in, error message sent to log and macro processing ends)

- FILENAME: required - Name of the actual transferred file to check (ex: LABS, PK, 6022_BWH CMR_CMR_[date])...if the FILENAME parameter contains the string "[date]", then the DSDATE parameter must be passed in...the macro will substitute the "[date]" string for the value in DSDATE

- PREVDIR : optional - Directory/folder where previous transfer file exists (if not passed in, macro will not perform a comparison of the current transfer to the prior one, in case the current transfer is the first one)...if the previous transfer is contained within a ZIP file, the string ".zip" must be added to the value of PREVDIR (example:
"prevdir=G:\Prod\CK274\CY6022\analysis\final_lock\rawdata\other\archive\CY6022_CELERION_PK_14SEP2023.zip")

- DSDATE : optional - Date value in case the transfer file itself contains a date in the name (ex: 18SEP2023, 20230918)...must be passed in exactly the way it appears in the name of the file or the macro will produce a file not found error in the log and stop processing

- VENDOR : optional - Name of the vendor (ex: ICON LABS, Synteract)...this is needed in case there are multiple instances of a filename for different vendors...if passed in the macro will select records from the metadata file based on the values of the FILENAME and VENDOR parameters

- STDYMAC : optional - Name of a macro containing study-specific checks created just for the particular study...if this is passed in, the study-specific macro must exist

- STDYN : optional - Tells the macro whether or not to run the standard checks, or proceed directly to the study-specific macro call. Default=Y (meaning to run the standard checks), valid responses are Y or N

```
       - DEBUG  : optional - Tells macro whether or not to delete the work datasets, for debugging purposes only.   Default=N, valid
       responses are Y or N

      Sample call: %ms_dts_content_check(currdir=G:\Prod\<PROJ>\<STDY>\analysis\final_lock\rawdata\other,
                     prevdir=G:\Prod\CK274\<PROJ>\analysis\final_lock\rawdata\other\archive\<PROJ>_BWH_CMR CMR 15JUN2023,
                     filename=<PROJ>_BWH CMR_CMR_[date],dsdate=20230715);

*************************************************************************************/

%macro ms_dts_content_check(currdir =,
                            prevdir =,
                            filename=,
                            vendor  =,
                            dsdate  =,
                            stdymac =,
                            stdyn   =Y,
                            debug   =N);

/*****************************************************************************************************************
 * If the STDYN parameter is passed in as N, bypass all of the standard checking code and proceed directly to
 * the study-specific generation...first check to make sure STDYN and STDYMAC are passed in correctly...
 *****************************************************************************************************************/
%if &STDYN=N and &STDYMAC=  %then %do;
   %put %sysfunc(compress(ER ROR: )) Study specific macro name must not be missing if standard checks are not being run.;
   %RETURN;
%end;
%else %if &STDYN=N and &STDYMAC ne  %then %do;
   %let currds=&filename;
   %goto STDY_SPEC;
%end;

/*************************************************************************
 * Get location of metadata file...also set up macro variable of metadata file
 * name...
 *************************************************************************/
%global metaloc metafil;

%let metaloc = &CK_RAWDATA\other;
%let metafil = &CK_STUDY._DTS_metadata;

%let filename=%upcase(&filename);

/*************************************************************************
 * Perform checks on input parameters and if files exist:
 *      - CURRDIR and FILENAME parameters are REQUIRED, so neither one can be missing
 *      - If DSDATE is not missing, then FILENAME value must have the string "[date]"
```

```
 *             contained within it
 *     - Conversely, if FILENAME value has the string "[date]" contained within it,
 *             then DSDATE must not be missing
 *  - If the STDYN parameter is passed in as N, the STDYMAC parameter must be
 *             passed in/not missing
 ************************************************************************/
    %global ec datc;
    %let ec=0;
    %let datc=0;

    %if &currdir=  and &filename=  %then %do;
       %put %sysfunc(compress(ER ROR: )) Both macro parameters CURRDIR and FILENAME must have values passed in.;
       %let ec=1;
    %end;
    %else %if &currdir=  and &filename ne  %then %do;
       %put %sysfunc(compress(ER ROR: )) Macro parameter CURRDIR must have a value passed in.;
       %let ec=1;
    %end;
    %else %if &filename=  and &currdir ne  %then %do;
       %put %sysfunc(compress(ER ROR: )) Macro parameter FILENAME must have a value passed in.;
       %let ec=1;
    %end;

    data _null_;

       if index("&filename","[DATE]") > 0 and "&DSDATE" = " " then call symput("datc","1");
       if index("&filename","[DATE]") = 0 and "&DSDATE"^= " " then call symput("datc","1");
    run;

    %if &datc=1 %then %put %sysfunc(compress(ER ROR: )) FILENAME and DSDATE parameter values not passed in properly...please check.;

/*************************************************************************************************
 * If any of the error conditions above occur, send an error message to the log and abort the macro...
 *************************************************************************************************/
    %if &ec=1 or &datc=1 %then %do;
       %put %sysfunc(compress(ER ROR: )) Error(s) encountered...macro will abort.;
       %RETURN;
    %end;

/*************************************************************************************************
 * CONTINUE PROCESSING...Open the DTS Metadata file...create 2 datasets...one for the actual field names and one
 * for the controlled terminology values...
 *
 * Then, only keep records where the passed in value of FILENAME equals the value in the VARIABLE field in the
 * spreadsheet output dataset...
 *************************************************************************************************/
proc import datafile="&METALOC\&METAFIL..xlsx"
```

```
                out=metadata_fields
   dbms=excel2007
   replace;
   sheet=Datasets;
run;

proc import datafile="&METALOC\&METAFIL..xlsx"
                out=metadata_contterm
   dbms=excel2007
   replace;
   sheet="Controlled Terminology";
run;

/*****************************************************************************************************
 * Subset the records in the METADATA_FIELDS dataset created above to the following:
 *      - If the VENDOR macro parameter is passed in, only those pertaining to that Vendor name
 *      - The dataset/data file found in the FILENAME macro parameter value
 *****************************************************************************************************/
proc sql noprint;
   create table metadata_fields2 as
      select *
         from metadata_fields
            where
               %IF &VENDOR NE  %THEN %DO;
                  vendor = "&VENDOR" and
               %END;
                  upcase(dataset) = "&FILENAME";

/*****************************************************************************************************
 * Next, acquire all controlled terminology records from the METADATA_CONTTERM dataset created above where records in the
 * METADATA_FIELDS2 dataset have controlled terminology and match the controlled terminology within METADATA_CONTTERM...
 * once we have these, create a macro variable for an "IN" statement...if there are no controlled terminology
 * records associated with the external file, create a macro variable to NOT run the CT checks...
 *
 * Also, check the METADATA_FIELDS2 dataset for existance of the passed in FILENAME value...if there are no records
 * found, produce an error message to the log and abort processing, because most likely the user made a mistake
 * passing in the filename value or they misspelled it...
 *****************************************************************************************************/
   create table control_terms as
      select distinct ct
         from metadata_fields2
         where ct ne " ";

   select count(*) into: ctrecs from control_terms;

   select count(*) into: mdrecs from metadata_fields2;
quit;
```

```
%IF &MDRECS=0 %THEN %DO;
    %put %sysfunc(compress(ER ROR: )) File name &FILENAME does not exist in metadata file...macro will abort.;
    %return;
 %end;

%IF &CTRECS=0 %THEN %DO;
   %LET CTCHK=N;
   %put %sysfunc(compress(NOTE: )) No Controlled Terminology for file &FILENAME so no CT checks will be performed.;
%END;
%ELSE %DO;
   %LET CTCHK=Y;

   data _null_;
      set control_terms end=eof;

      length contterm $200.;
      retain contterm;

      if _n_ = 1 then contterm = "'" || strip(ct) || "'";
      else            contterm = strip(contterm) || " '" || strip(ct) || "'";

      if eof then call symput("allcts",strip(contterm));
   run;

   proc sql;
      create table control_terms2 as
         select * from metadata_contterm
            where ctname in (&allcts);
   quit;
%END;

/*****************************************************************************************************
 * BEGIN TO PROCESS DATA STRUCTURE CHECKS...
 * Go to the METADATA_FIELDS2 file and get the value of the FILETYPE field for the particular external
 * file in question...
 *      - check to see if the external transfer file actually exists...if not, output an error message and exit
 *              the macro process...
 * Otherwise...
 *      - if the filetype is CSV or XLSX, use PROC IMPORT to read the external file into a SAS dataset...
 *      - if the filetype is XPT, run PROC COPY to convert to a SAS dataset...
 *      - if the filetype is SAS, create a SAS Libname and read in the data as a dataset would normally be processed...
 *
 * First create macro variables containing the output file name, file type, and whole file name combination
 * (NAME.TYP)...make this conditional upon whether there is a date in the file name (FILENAME parameter contains the
 * string "[date]" and the DSDATE parameter is not missing)...
 *****************************************************************************************************/
```

```
data _null_;
   set metadata_fields2;

   length dataset_name $100.;

   if _n_ = 1 then do;
      dataset = upcase(dataset);

      if index(upcase(dataset),"[DATE]") and "&DSDATE" ne " " then do;
         dataset_name = tranwrd(strip(dataset),"[DATE]","&DSDATE");
         call symput("DATED","Y");
      end;
      else do;
         dataset_name = strip(dataset);
         call symput("DATED","N");
      end;

      call symput("dsname",strip(dataset_name));
      stop;
   end;
run;

%PUT &=DSNAME &=DATED;

proc sql noprint;
   select distinct strip(filetype) length=4 into: filtyp from metadata_fields2;
quit;

proc sql noprint;
   %IF &FILTYP=SAS %THEN %DO;
      select distinct strip("&DSNAME") || ".sas7bdat" into: extfile from metadata_fields2;
   %END;
   %ELSE %DO;
      select distinct strip("&DSNAME") || "." || strip(filetype) into: extfile from metadata_fields2;
   %END;
quit;

data _null_;

   length ext_file $100.;
   ext_file = strip("&EXTFILE");

   call symput("extfile",strip(ext_file));
run;

%PUT &=EXTFILE;
```

```
/*************************************************************************************************
 * Check if the external transfer file exists...if the external file does not exist, produce an error
 * message and abort processing...
 *************************************************************************************************/
 %if not (%sysfunc(fileexist(&CURRDIR\&EXTFILE))) %then %do;
    %put %sysfunc(compress(ER ROR: )) External transfer file &EXTFILE does not exist...macro will abort.;
    %return;
 %end;

/*************************************************************************************************
 * If the file exists, read it into a temporary SAS dataset using code depending on the file type...
 *************************************************************************************************/
%IF &FILTYP=CSV %THEN %DO;
   proc import datafile="&CURRDIR\&EXTFILE"
      out=&DSNAME
      dbms=csv replace;
      getnames=yes;
      datarow=2;
   run;
%END;
%ELSE %IF &FILTYP=XLSX %THEN %DO;
   proc import datafile="&CURRDIR\&EXTFILE"
                out=&DSNAME
      dbms=excel2007
      replace;
   run;
%END;
%ELSE %IF &FILTYP=XPT %THEN %DO;
   libname xptin xport "&CURRDIR\&EXTFILE";

   proc copy in=xptin out=work;
   run;
%END;
%ELSE %IF &FILTYP=SAS %THEN %DO;
   libname curdir "&CURRDIR";

   proc sql;
      create table &DSNAME as
         select * from curdir.&DSNAME;
   quit;
%END;

/*************************************************************************************************
 * Now that we have the external data available to process as SAS data, begin to perform checks...
 * First, perform a PROC CONTENTS on the output SAS dataset created in the steps above and output the
 * contents data to a dataset...this will be compared to the Metadata_fields2 dataset created above for
 * consistency...
```

```
 ************************************************************************************************/
proc contents data=&DSNAME noprint
              out=transfer_file_contents(keep=varnum memname name type length nobs);
run;

proc sort; by varnum; run;

data transfer_file_contents(drop=type);
   set transfer_file_contents;

   length vartype $6.;

   select (type);
      when (1) vartype = "NUM";
      when (2) vartype = "CHAR";
      otherwise;
   end;
run;

/************************************************************************************************
 * If the filetype is CSV or XLSX, spaces in the field names within the metadata file will need to be
 * converted to underscores so the comparison merge of field names works...SAS converts them to under-
 * scores when converting the external file to a dataset...
 ************************************************************************************************/
%IF &FILTYP=CSV OR &FILTYP=XLSX %THEN %DO;
   data metadata_fields2(drop=varlen);
      set metadata_fields2;

      varlen = length(strip(variable));
      if index(variable," ") then variable = substr(tranwrd(variable," ","_"),1,varlen);
   run;
%END;

/************************************************************************************************
 * Merge the TRANSFER_FILE_CONTENTS and the METADATA_FIELDS2 datasets together by field names and start
 * the actual checks...
 *      - compare variable types (numeric vs character) and note differences
 *      - compare variable lengths if lengths are available in metadata file (for SAS/XPT files ONLY, since
 *              CSV and XLSX create the lengths at conversion time depending on the existing data) and note differences
 *      - if a variable is in the metadata file and not the transfer file, or vice versa, note this
 *      - then check how many observations are found with differences...if none, place a note in the log stating that no
 *              differences were found between metadata and transfer data contents
 * First, sort the 2 datasets and rename certain variables...then, merge together and flesh-out issues...
 * save these to an output dataset...
 *
 * If no issues are found, output a record to the error dataset saying that no compare issues were found
 * between the Current transfer and the metadata...
```

```
 ************************************************************************************************/
data metadata_fields2(drop=variable dataset rename=(variable2=variable dataset2=dataset));

   attrib variable2 dataset2 length=$32. informat=$32. format=$32.;

   set metadata_fields2;
   variable2=strip(variable);
   dataset2 =strip(dataset);
run;

proc sort data=metadata_fields2(rename=(variable=name length=metalen));
   by name;
run;

proc sort data=transfer_file_contents;
   by name;
run;

data meta_merge_to_contents(keep=dsname varname issudesc);
   merge transfer_file_contents (in=cont)
         metadata_fields2       (in=meta);
   by name;

   length dsname $20. varname $50. issudesc $200.;

   label dsname  = "Dataset"
         varname = "Variable"
         issudesc= "Problem with Transfer";

   if (cont and not meta) then do;
      dsname = "&DSNAME";
      varname = strip(name);
      issudesc = "Field found in transfer file but it does not exist in DTS document.";
      output;
   end;
   else if (meta and not cont) then do;
      dsname = "&DSNAME";
      varname = strip(name);
      issudesc = "Field found in DTS document but it does not exist in transfer file.";
      output;
   end;

   else if meta and cont then do;
      if type not in ("DATETIME","DATE","TIME") then do;
         if upcase(type) ne upcase(vartype) then do;
            dsname = "&DSNAME";
            varname = strip(name);
```

```
            issudesc = "Differences found in variable types between DTS document metadata (" || strip(type) || ") and transfer file ("
|| strip(vartype) || ").";
            output;
         end;
      end;
      else do;
         if vartype ne "NUM" then do;
            dsname = "&DSNAME";
            varname = strip(name);
            issudesc = "Differences found in variable types between DTS document metadata (" || strip(type) || ") and transfer file ("
|| strip(vartype) || ").";
            output;
         end;
      end;
      %IF &FILTYP=SAS OR &FILTYP=XPT %THEN %DO;
         if metalen ne . then do;
            if metalen ne length then do;
               dsname = "&DSNAME";
               varname = strip(name);
               issudesc = "Differences found in variable length between DTS document metadata (" || strip(put(metalen,best.)) || ")
and transfer file (" || strip(put(length,best.)) || ").";
               output;
            end;
         end;
      %END;
   end;
run;

proc sql noprint;
   select count(*) into: metadisc from meta_merge_to_contents;
quit;

%IF &METADISC=0 %THEN %DO;
   %put %sysfunc(compress(NOTE: )) *** Comparison between Metadata and Current Transfer file &EXTFILE produced no issues! ***;

    data meta_merge_to_contents;
       length dsname $20. varname $50. issudesc $200.;

       label dsname  = "Dataset"
             varname = "Variable"
             issudesc= "Problem with Transfer";

       dsname = "&DSNAME";
       varname= "N/A";
       issudesc = "No comparison issues found between Metadata and structure of Current transfer data file.";
       output;
    run;
```

```
 %END;


/*************************************************************************************************
 * If the PREVDIR macro parameter has been passed in, perform checks to see if the previous transfer file
 * exists...
 *      - if the folder in the PREVDIR macro parameter contains the string ".zip", perform logic to extract the
 *              zipped transfer file in the ZIP file folder and place into a temporary "work" folder for later access
 *      - otherwise, check if the prior external transfer file exists in the conventional way using the
 *              "%sysfunc(fileexist(...))" functionality
 *      - either way, store the location and name of the previous external data transfer file to a filename called
 *              "xl"
 *
 * NOTE: For transfer files with the date within the filename, the macro will have to search for the prior
 * file in the PREVDIR folder in a different fashion...in these cases, go to the previous folder passed into PREVDIR
 * and search for/process the file contained within that folder...the date in that filename is unknown to the macro
 * but the PREVDIR folder drives this process...
 *************************************************************************************************/
%let currds=&dsname;

%IF &PREVDIR NE  %THEN %DO;

   data _null_;
      length prev_direct $200.;

      prev_direct = symget("PREVDIR");

      if index(upcase(prev_direct),".ZIP") then call symput("ZIPFND","Y");
      else                                      call symput("ZIPFND","N");
   run;

   %put &=ZIPFND;

   %IF &ZIPFND=N %THEN %DO;
/*************************************************************************
 * If the PREVDIR value exists and the previous folder IS NOT a zip file:
 *
 * If the FILENAME macro parameter contains "[date]" and DSDATE is not missing,
 * locate and process the data file located in the previous folder passed within
 * the PREVDIR parameter...this file will be the one processed as the previous
 * tranfer...
 *************************************************************************/
      %if &DATED=Y %then %do;
         data fndprev;

            keep filename filnoext revfile;
```

```
            length fref $8. filename filnoext revfile $80.;

            rc = filename(fref, "&PREVDIR");

            if rc = 0 then do;
               did = dopen(fref);
               rc = filename(fref);
            end;

            dnum = dnum(did);

            do i = 1 to dnum;
               filename = dread(did, i);
               revfile = strip(reverse(filename));
               if revfile ne " " then filnoext= reverse(substr(strip(revfile),index(revfile,".")+1));
               fid = mopen(did, filename);
               if fid > 0 then output;
            end;

            rc = dclose(did);
         run;

         proc sql noprint;
            select distinct strip(filename) into: extfile from fndprev;

            select distinct strip(filnoext) into: dsname  from fndprev;
         quit;

         %let dsname=&dsname;
      %end;
      %if not (%sysfunc(fileexist(&PREVDIR\&EXTFILE))) %then %do;
         %put %sysfunc(compress(NOTE: )) Previous external transfer file does not exist in &PREVDIR...;
         %put %sysfunc(compress(NOTE: )) comparison between current and previous transfers will not be done.;
         %let prevproc=N;
      %end;
      %else %do;
         %let prevproc=Y;

         filename xl "&PREVDIR\&EXTFILE" ;
      %end;
   %END;
/*****************************************************************************
 * If the PREVDIR value exists and the previous folder IS a zip file:
 *
 * Open the Zip file folder passed into the PREVDIR parameter and locate the
 * previous transfer file...copy it from the zip folder into a work location...
 * then process it from the work location...
```

```
*******************************************************************************/
  %ELSE %DO;
     filename inzip ZIP "&PREVDIR";

     data contents(keep=memname isFolder);
        length memname $200 isFolder 8;

        fid=dopen("inzip");
        if fid=0 then do;
           %let prevproc=N;
           stop;
        end;
        else %let prevproc=Y;

        memcount=dnum(fid);
        do i=1 to memcount;
           memname=dread(fid,i);
           isFolder = (first(reverse(trim(memname)))='/');
           output;
        end;

        rc=dclose(fid);
     run;

     data _null_;
        set contents(where=(index(upcase(memname),".TXT")=0)) end=eof;

        length ext_file_name $100.;
        if _n_ = 1 then ext_file_name = " ";

        %if &DATED=Y %then %do;
           if memname ne " " then do;
        %END;
        %ELSE %DO;
           if index(upcase(memname),upcase("&DSNAME")) then do;
        %END;
              ext_file_name = strip(memname);
              goto out;
           end;

        if eof then do;
           if ext_file_name = " " then call symput("zipfil","NOPE");
           stop;
        end;

        out:;
           call symput("zipfil",strip(ext_file_name));
```

```
      run;

      %if &ZIPFIL=NOPE %then %do;
         %put %sysfunc(compress(NOTE: )) Previous external transfer file does not exist in &PREVDIR...;
         %put %sysfunc(compress(NOTE: )) comparison between current and previous transfers will not be done.;
         %let prevproc=N;
      %end;
      %else %do;
         %let prevproc=Y;

         filename xl "%sysfunc(getoption(work))/&ZIPFIL" ;

         data _null_;
            infile inzip(&ZIPFIL) lrecl=256 recfm=F length=length eof=eof unbuf;
            file xl lrecl=256 recfm=N;

            input;
            put _infile_ $varying256. length;
            return;

            eof:
               stop;
         run;
      %end;
   %END;

   %IF &DATED=Y AND &PREVPROC=Y %THEN %DO;
      proc sql noprint;
         select distinct substr(strip(memname),1,index(memname,".")-1) into: dsname from contents;
      quit;

      %let dsname=&dsname;
   %END;

/*********************************************************************************************
 * If the previous file exists, read it into a temporary SAS dataset using code depending on the file type...
 * A few things to consider with this process...
 *      - The system needs to point in different folder areas depending upon if the previous file is located
 *              in a ZIP folder or not...if the previous file is located in a ZIP folder, the above process copies it to a
 *              "work" folder so the system retrieves it from there...otherwise it will use the value of PREVDIR
 *              to locate and process it...
 *      - For XPT files, the current transfer output dataset will need to be temporarily renamed so it's not
 *              overwritten when the previous transfer output dataset is created...once the output dataset is
 *              created using PROC COPY, the current and the previous transfer datasets are renamed...
 *      - The suffix "_PRV" is appended to the value passed into the DSNAME parameter when the previous
 *              transfer dataset is created...
 *********************************************************************************************/
```

```
%IF &PREVPROC=Y %THEN %DO;

  %IF &FILTYP=CSV %THEN %DO;
    proc import datafile=xl
       out=&DSNAME._PRV
       dbms=csv replace;
       getnames=yes;
       datarow=2;
    run;
  %END;
  %ELSE %IF &FILTYP=XLSX %THEN %DO;
    proc import datafile=xl
                  out=&DSNAME._PRV
       dbms=excel2007
       replace;
    run;
  %END;
  %ELSE %IF &FILTYP=XPT %THEN %DO;
    proc datasets nolist library=work;
       change &DSNAME=&DSNAME._CUR;
    run; quit;

    %IF &ZIPFND=N %THEN %DO;
       libname xptin xport "&PREVDIR\&EXTFILE";
    %END;
    %ELSE %DO;
       libname xptin xport "%sysfunc(getoption(work))\&EXTFILE";
    %END;

    proc copy in=xptin out=work;
    run;

    proc datasets nolist library=work;
       change &DSNAME=&DSNAME._PRV
              &DSNAME._CUR=&DSNAME;
    run; quit;
  %END;
  %ELSE %IF &FILTYP=SAS %THEN %DO;
    %IF &ZIPFND=N %THEN %DO;
       libname predir "&PREVDIR";
    %END;
    %ELSE %DO;
       libname predir "%sysfunc(getoption(work))";
    %END;

    proc sql;
       create table &DSNAME._PRV as
```

```
            select * from predir.&DSNAME;
      quit;
    %END;

/*************************************************************************************************
 * If a previous version of the file passed within the DSNAME parameter exists, and a SAS dataset is produced
 * using the previous transfer file, perform a check between the Current Transfer file and the Previous Transfer File...
 *      - compare data structure between the 2 and note differences
 *      - compare the number of observations within each file...there should be an equal or greater number of
 *             observations in the current transfer...if not, note the difference
 *
 * If no issues are found, output a record to the error dataset saying that no compare issues were found
 * between the Current and Previous transfers...
 *************************************************************************************************/
    proc contents data=&DSNAME._PRV noprint
                  out=prev_transfer_file_contents(keep=varnum memname name type length nobs);
    run;

    proc sort; by name; run;

    proc sort data=transfer_file_contents; by name; run;

    data prev_transfer_file_contents(drop=type rename=(memname=memname_p varnum=varnum_p length=length_p nobs=nobs_p
vartype=vartype_p));
       set prev_transfer_file_contents;

       length vartype $6.;

       select (type);
          when (1)     vartype = "NUM";
          when (2)     vartype = "CHAR";
          otherwise;
       end;
    run;

    data compare_curr_to_prev(keep=dsname varname issudesc);
       merge transfer_file_contents      (in=curr)
             prev_transfer_file_contents (in=prev);
       by name;

       length dsname $20. varname $50. issudesc $200.;

       label dsname  = "Dataset"
             varname = "Variable"
             issudesc= "Problem with Transfer";

      if (curr and prev) and _n_ = 1 then do;
```

```
         if nobs lt nobs_p then do;
            dsname = "&CURRDS";
            varname = "<ALL>";
            issudesc = "Previous transfer of &EXTFILE (&DSNAME) has more observations than current transfer.";
            output;
         end;
      end;

      if curr and not prev then do;
         dsname = "&CURRDS";
         varname = strip(name);
         issudesc = "Field found in Current transfer file but it does not exist in the Previous transfer file &DSNAME.";
         output;
      end;
      else if prev and not curr then do;
         dsname = "&CURRDS";
         varname = strip(name);
         issudesc = "Field found in Previous transfer file but it does not exist in the Current transfer file &DSNAME.";
         output;
      end;
      else if curr and prev then do;
         if length ne length_p then do;
            dsname = "&CURRDS";
            varname = strip(name);
            issudesc = "Differences found in variable length between Current transfer (" || strip(put(length,best.)) || ") and
Previous transfer &DSNAME (" || strip(put(length_p,best.)) || ").";
            output;
         end;
         if vartype ne vartype_p then do;
            dsname = "&CURRDS";
            varname = strip(name);
            issudesc = "Differences found in variable type between Current transfer (" || strip(vartype) || ") and Previous transfer
&DSNAME (" || strip(vartype_p) || ").";
            output;
         end;
      end;
   run;

   proc sql noprint;
      select count(*) into: currprev from compare_curr_to_prev;
   quit;

   %IF &CURRPREV=0 %THEN %DO;
      %put %sysfunc(compress(NOTE: )) *** Comparison between Previous Transfer and Current Transfer file &EXTFILE produced no issues!
***;

      data compare_curr_to_prev;
```

```
         length dsname $20. varname $50. issudesc $200.;

         label dsname  = "Dataset"
               varname = "Variable"
               issudesc= "Problem with Transfer";

         dsname = "&CURRDS";
         varname= "N/A";
         issudesc = "No comparison issues found between Previous and Current transfer data files.";
         output;
      run;
   %END;

  %END;

%END;

/*****************************************************************************************************
 * Process Controlled Terminology checks...
 *      - If a value exists for a variable having controlled terminology that is not in the CT list in the
 *             metadata, output an error record
 * First, acquire all of the fields in the dataset that will contain controlled terminology, from the
 * Metadata_fields2 dataset created above...then read these into macro variables and keep a count...
 * controlled terminology checks will be processed for each variable containing control terminology...
 *****************************************************************************************************/
%IF &CTCHK=Y %THEN %DO;
   proc sql;
      create table vars_with_ct as
         select distinct upcase(name) as name, upcase(type) as type, ct, ct_cond
            from metadata_fields2
            where ct ne " "
            order by name;
   quit;

   data _null_;
      set vars_with_ct end=eof;

      cnt = _n_;

      call symput("CTVAR" || strip(put(cnt,best.)),strip(name));
      call symput("CTNAM" || strip(put(cnt,best.)),strip(ct));
      call symput("CTTYP" || strip(put(cnt,best.)),strip(upcase(type)));

      if ct_cond ne " " then call symput("CTCND" || strip(put(cnt,best.)),strip(upcase(ct_cond)));
      else                   call symput("CTCND" || strip(put(cnt,best.)),"X");

      if eof then call symput("CTNUM",strip(put(cnt,best.)));
```

```
  run;

/*****************************************************************************************************
 * Now, for each of the fields found and set into macro variables above, get the actual controlled term-
 * inology from the dataset Control_terms2 and sort by those values...we will merge this with the actual
 * values in the data and flush out any in the data not existing in the metadata...any values found in the
 * data that don't exist in the metadata CT will be output to an error dataset...also, if the value is mis-
 * sing in the data, this will be output to the error dataset as well...
 *
 * For each variable having controlled terminology...
 *      - loop through each variable having a controlled terminology value
 *      - merge the actual data for that variable with the CT values in the metadata by the values
 *      - output any mismatches in the data (values not existing in the CT) to a "bucket" dataset
 *      - set the "bucket" to a main error dataset and delete the "bucket" dataset
 *      - if there are no issues, create a record in the main error dataset saying so
 *      - if the transfer file does not contain any controlled terminology, create a record saying so
 *****************************************************************************************************/
  data control_term_issues;
     stop;
  run;

  %DO I = 1 %TO &CTNUM;
     proc sql;
        create table ct_vals_data as
           select distinct &&CTVAR&I
                           %IF &&CTTYP&I=CHAR %THEN %DO;
                              length=200
                           %END;
              from &CURRDS
              %IF &&CTCND&I NE X %THEN %DO;
                 where &&CTCND&I
              %END;
              order by &&CTVAR&I;

        create table ct_vals_meta as
           select distinct %IF &&CTTYP&I=NUM %THEN %DO;
                              input(ctvalue,best.) as &&CTVAR&I
                           %END;
                           %ELSE %DO;
                              ctvalue as &&CTVAR&I length=200
                           %END;
              from control_terms2
              where ctname = "&&CTNAM&i"
              order by &&CTVAR&i;
     quit;

/*****************************************************************************************************
```

```
* If a controlled terminology metadata value contains the string "[*]", this will signify that the data
* has dates in it...these will need to be removed from the data and the [*] will need to be removed from
* the controlled terminology as well...
*************************************************************************************************/
     %IF &&CTTYP&I=CHAR %THEN %DO;
        data ct_vals_meta;
           set ct_vals_meta end=eof;

           &&CTVAR&I = left(strip(&&CTVAR&I));

           retain braccnt;
           if _n_ = 1 then braccnt = 0;

           if index(&&CTVAR&I,"[*]") then do;
              &&CTVAR&I = substr(strip(&&CTVAR&I),1,index(&&CTVAR&I,"[*]")-1);
              braccnt + 1;
              call symput("varbrac"||strip(put(braccnt,best.)),strip(&&CTVAR&I));
           end;

           if eof then call symput("totbracs",strip(put(braccnt,best.)));
        run;

        proc sort data=ct_vals_meta; by &&CTVAR&I; run;

        data ct_vals_data;
           set ct_vals_data;

           %IF &TOTBRACS NE 0 %THEN %DO J = 1 %TO &TOTBRACS;
              if index(&&CTVAR&I,"&&VARBRAC&J") then &&CTVAR&I = strip("&&VARBRAC&J");
           %END;
        run;

        proc sort data=ct_vals_data; by &&CTVAR&I; run;
     %END;

     data ct_merge_&I(keep=dsname varname issudesc);
        merge ct_vals_data (in=ctd)
              ct_vals_meta (in=ctm);
        by &&CTVAR&I;

        length dsname $20. varname $50. issudesc $200.;

        label dsname  = "Dataset"
              varname = "Variable"
              issudesc= "Problem with Transfer";

        %IF &&CTTYP&I=NUM %THEN %DO;
```

```
            if &&CTVAR&I = .
         %END;
         %ELSE %DO;
            if &&CTVAR&I = " "
         %END;  then do;
            dsname = "&CURRDS";
            varname = "&&CTVAR&I";
            issudesc = "&&CTVAR&I is missing for observation " || strip(put(_n_,best.)) || ".";
            output;
         end;
         else if ctd and not ctm then do;
            dsname = "&CURRDS";
            varname = "&&CTVAR&I";
            %IF &&CTTYP&I=NUM %THEN %DO;
               issudesc = "Value for variable &&CTVAR&I (" || strip(put(&&CTVAR&I,best.)) || ") for observation " ||
strip(put(_n_,best.)) || " not found within Controlled Terminology &&CTNAM&I" || ".";
            %END;
            %ELSE %DO;
               issudesc = "Value for variable &&CTVAR&I (" || strip(&&CTVAR&I) || ") for observation " || strip(put(_n_,best.)) || "
not found within Controlled Terminology &&CTNAM&I" || ".";
            %END;
            output;
         end;
      run;

      data control_term_issues;
         set control_term_issues ct_merge_&I;
      run;

      proc datasets nolist library=work;
         delete ct_merge_&I;
      run; quit;
   %END;

    proc sql noprint;
       select count(*) into: ctisscnt from control_term_issues;
    quit;

    %IF &CTISSCNT=0 %THEN %DO;
       %put %sysfunc(compress(NOTE: )) *** Checks of Controlled Terminology for Current Transfer file &EXTFILE produced no issues!
***;

       data control_term_issues;
          length dsname $20. varname $50. issudesc $200.;

          label dsname  = "Dataset"
                varname = "Variable"
```

```
               issudesc= "Problem with Transfer";

          dsname = "&CURRDS";
          varname= "N/A";
          issudesc = "No controlled terminology issues found in Current transfer data file.";
          output;
       run;
    %END;
%END;
%ELSE %DO;
   data control_term_issues;
      length dsname $20. varname $50. issudesc $200.;

      label dsname  = "Dataset"
            varname = "Variable"
            issudesc= "Problem with Transfer";

       dsname = "&CURRDS";
       varname= "N/A";
       issudesc = "Controlled Terminology does not exist within Current transfer data file...checks not performed.";
       output;
    run;
%END;

/*****************************************************************************************************
 * If there's a study-specific macro (found in the parameter STDYMAC), generate it here...
 *****************************************************************************************************/
%STDY_SPEC: ;

%IF &STDYMAC NE  %THEN %DO;
   %IF NOT (%sysfunc(fileexist(&CK_AUTO3\&STDYMAC..SAS))) %THEN %DO;
      %put %sysfunc(compress(NOTE: )) *** A run of study-specific macro %upcase(&STDYMAC) requested, but this macro program does not
exist... ***;
      %LET STDYMAC=  ;
   %END;
   %ELSE %DO;
      %&STDYMAC;;
   %END;
%END;

/*****************************************************************************************************
 * Set all of the above created DTS issue datasets together...
 *      - Comparison of current transfer structure to DTS/metadata
 *      - Comparison of current transfer to previous tranfer
 *      - Controlled terminology issues
 *      - Study Specific items (if STDYMAC macro parameter is passed in)
 * Create an ordering variable and a variable containing the type of check...
```

```
 ***************************************************************************************************/
data all_dts_issues;

   %IF &STDYN=N AND &STDYMAC NE  %THEN %DO;
      set study_specific_checks (in=stud);
   %END;
   %ELSE %DO;
      set meta_merge_to_contents (in=meta)
          %IF &PREVDIR NE  %THEN %DO;
             Compare_curr_to_prev   (in=comp)
          %END;
          Control_term_issues    (in=cont)
          %IF &STDYMAC NE  %THEN %DO;
             study_specific_checks (in=stud)
          %END; ;
   %END;

   length checktyp $100.;

   %IF &STDYN=N AND &STDYMAC NE  %THEN %DO;
      ordervar = 1; checktyp = strip(study_checks);
   %END;
   %ELSE %DO;
      if meta then do;
         ordervar = 1; checktyp = "Check of DTS Transfer file contents to metadata";
      end;
      %IF &PREVDIR NE  %THEN %DO;
         if comp then do;
            ordervar = 2; checktyp = "Check of Current DTS Transfer file to Previous DTS Transfer file";
         end;
      %END;
      if cont then do;
         ordervar = 3; checktyp = "Check of controlled terminology values to metadata";
      end;
      %IF &STDYMAC NE  %THEN %DO;
         if stud then do;
            ordervar = 4; checktyp = strip(study_checks);
         end;
      %END;
   %END;
run;

proc sort; by ordervar checktyp; run;

/***************************************************************************************************
 * Create an output spreadsheet all of the data for each type of check performed...
 ***************************************************************************************************/
```

```
ods listing close;

data _null_;

   call symput("sysdat",strip(put("&sysdate"d,yymmdd10.)));
run;

ods excel file="&CK_TLF.\&CK_STUDY._DTS_content_check_&SYSDAT._&CURRDS..xlsx" options(sheet_name="DTS Checks");

proc report data=all_dts_issues nowd split="|";
    columns ordervar checktyp dsname varname issudesc;

    define ordervar       / order noprint;

    define checktyp       / style(column)={cellwidth=4in}
                            style(header)={cellwidth=4in} "Type of Check Performed";

    define dsname         / style(column)={cellwidth=1.5in}
                            style(header)={cellwidth=1.5in} "Transfer File Name";

    define varname        / style(column)={cellwidth=1in}
                            style(header)={cellwidth=1in} "Variable Name";

    define issudesc       / style(column)={cellwidth=4in}
                            style(header)={cellwidth=4in} "Description of Issue";
run;


ods excel close;

ods listing;


/*****************************************************************************************************
 * Depending on the value of the DEBUG parameter...if it's not Y, delete all of the work datasets
 * produced above...
 *****************************************************************************************************/
%IF %UPCASE(&DEBUG) NE Y %THEN %DO;
   proc datasets nolist library=work kill;
   run; quit;
%END;

%mend ms_dts_content_check;
```