Paper AD08

Those pesky SAS v5 transport files, what's inside?

John H. Adams, Boehringer Ingelheim Pharmaceutical , Inc., Ridgefield, CT

ABSTRACT

The use of SAS V5 transport files, unfortunately, is still currently required for FDA submissions. Many users in the pharmaceutical industry, however, are not well versed on the structure or determining the contents of transport files.

It is quite common for many of us to receive a SAS transport file from a vendor or a CRO. The first question is always – is it a .XPT file format or a .CPT file format. Next, we don't know what the contents are. Is there one dataset or are there multiple datasets.

Of course, a proficient programmer could quickly write some code to look into and /or extract the desired datasets. The average user that doesn't do this task very often, however , needs a simpler solution. This paper describes a utility macro that automatically determines the transport file type (.XPT or .CPT), the list of datasets it contains, the list of variables (plus their attributes) in each dataset and allows the user to extract any or all datasets.

INTRODUCTION

SAS transport files are still very much in use in the pharmaceutical industry because of the FDA submission requirement. This requirement will change in the future to one of using XML and HL7 transport protocols. In the meanwhile, however, we still have to deal with this left-over relic from SAS V5.

Since most users don't have to deal with transport files very often, they typically tend to forget the necessary SAS code to work with these files. There are basically two types of transport files that we use. The first one is created with the XPORT engine and typically has a .XPT filename suffix. The second type of transport file is created with the CPORT procedure and typically has a .CPT filename suffix. Unfortunately we cant always be sure what type of files it is as some vendors / CROs don't follow the naming convention We've seen many instances where the transport file had a .CPT suffix and was really a .XPT file. We also see the inverse.

The solution was to give the occasional user a simple to use utility macro. This macro would be able to quickly tell the user what type of file it was (regardless of what file extension was used) and what it's contents was. The macro would not only detail the names of the contained datasets, but also their variable names and their attributes. Sometimes we don't need everything that is contained in a transport file. So by using the information the macro provides, the user could selectively extract what was needed.

The UinTransport macro was designed to allow occasional users of transport files an easy way to look at the content of a transport file and retrieve it, if necessary. This paper describes the UinTransport macro and how it performs these transport file tasks. The code and the techniques are discussed in some detail in case someone wanted to adapt the macro for their application. Although this paper is about the Uintransport macro, the code for a macro that makes transport files (UtransportOut) is also furnished for the reader.

1 SAS TRANSPORT FILES

Transport files may contain one or multiple datasets. As stated in the Introduction section, there are two types of transport files. The first type is created by using Proc Copy with the XPORT engine. The second one is a file created with Proc Cport . By default, this file is compressed. How can we tell what type of file and what it's contents are? First, let's first look at the file structures.

1.1 XPORT transfer files.

All records are 80 bytes long. The data is stored as character in ASCII format. Integers are stored using IBM-style integer format, and all floating-point numbers are stored using the IBM-style double (truncated if the variable's length is less than 8). See reference[2].for more detailed information.

This type of file should use a .xpt suffix, but not everybody follows this convention. As you'll see below, it's relatively easy to find out if the file is a XPORT file by looking inside and recognize the way the it's header records are structured. Looking at the rest of the details of the file, however, can be much more difficult. Let's look at the file record structure:

1. The first record consists of the following character string, in ASCII:
   HEADER RECORD*******LIBRARY HEADER RECORD!!!!!!!000000000000000000000000000000

2. The first real header record uses the following layout:
   aaaaaaaabbbbbbbbccccccccddddddddeeeeeeee ffffffffffffffff

   In this record:
   -- aaaaaaaa and bbbbbbbb specify 'SAS '
   -- cccccccc specifies 'SASLIB '.
   -- dddddddd specifies the version of the SAS(r) System under which the file was created.
   -- eeeeeeee specifies the operating system that creates the record.
   -- ffffffffffffffff specifies the date and time created, formatted as ddMMMyy:hh:mm:ss. Note that only a 2-digit year appears.

3. Second real header record
   ddMMMyy:hh:mm:ss

   In this record, the string is the datetime modified. Most often, the datetime created and datetime modified will always be the same. Pad with ASCII blanks to 80 bytes. Note that only a 2-digit year appears. If any program needs to read in this 2-digit year, be prepared to deal with dates in the 1900s or the 2000s.

4. Member header records

   Both of these records occur for every member in the transport file.
   HEADER RECORD*******MEMBER HEADER RECORD!!!!!!!000000000000000001600000000140
   HEADER RECORD*******DSCRPTR HEADER RECORD!!!!!!!000000000000000000000000000000

Note the 0140 that appears in the member header record above. This value specifies the size of the variable descriptor (NAMESTR) record that is described later in this document. On the VAX/VMS operating system, the value will be 0136 instead of 0140. This means that the descriptor will be only 136 bytes instead of 140.

5. Member header data
   aaaaaaaabbbbbbbbccccccccddddddddeeeeeeee ffffffffffffffff

In this member header:
-- aaaaaaaa specifies 'SAS '.
-- bbbbbbbb specifies the data set name
-- cccccccc is SASDATA (if a SAS data set is being created)
-- dddddddd specifies the version of the SAS System under which the file was created.
-- eeeeeeee specifies the operating system.
-- ffffffffffffffff is the datetime created, formatted as in previous headers.

The second header record is as follows:
ddMMMyy:hh:mm:ss aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbb

In this record the datetime modified appears using the DATETIME16. format, followed by blanks up to column 33, where aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa corresponds to a blank-padded data set label and bbbbbbbb is the blank-padded data set type. Note that data set labels can be up to 256 characters as of SAS 8, but only the first 40 characters are stored in the second header record.

6. Namestr header record

   One for each member.

   HEADER RECORD*******NAMESTR HEADER RECORD!!!!!!!000000xxxx00000000000000000000

   In this header record, xxxx is the number of variables in the data set, displayed with blank-padded numeric characters. For example, for 2 variables, xxxx=0002. xxxx occurs at offset 54 (base 0 as in C language use).

7. Namestr records

   Each namestr field is 140 bytes long, but the fields are streamed together and broken in 80-byte pieces. If the last byte of the last namestr field does not fall in the last byte of the 80-byte record, the record is padded with ASCII blanks to 80 bytes.

1.2 CPORT transfer files.

The file structure of a transport file created by Proc Cport has no comparable easy documentation. Record lengths vary and can be extremely long. You'll need a file manager program that can open most types of files, like Quickview, in order to look at the contents of the file (Notepad cant handle it)

This type of file should use a .cpt suffix, but not everybody follows this convention. As you'll see below, it's not that easy to find out if the file is a CPORT file unless you have a file manager program capable of poking into the file. Looking at the rest of the details of the file is even:

The first 72 characters in the file contain the text (unless the option NOCOMPRESS was used):
COMPRESSED **COMPRESSED** **COMPRESSED** **COMPRESSED** **COMPRESSED

Following this, the is a string of characters outlining the OS and SAS versions used to create the file:
********LIB CONTROL WIN_PROÿ_WINÿ8 SAS8.2ÿ

After that you'll see some readable names like the variable names/labels, some text like 'SPAN CONTROL' and a lot of unreadable characters, e.g.:
ACTEVENTVisit numberÿ_ÿ_ÿ_ÿªSPAN CONTROL -1ÿé115ÿä2ÿå35ÿèLÿç115

2 THE MACRO INTERFACE

Following is the simple **UinTransport** macro's interface with it's default values for some arguments :

```
%macro Uintransport(file=, detail=N, import=N, outlib=WORK, dsname=_ALL_);
```

Macro arguments:

- File : full filename (including path) of transport file
- Detail : Do you wish to see details about the contents? (Y,N)
- Import : Do you wish to extract contents into SAS? (Y,N)
- Outlib : Assigned Libname for location of output dataset(s)
- Ds_name: Name or name list of datasets to extract

3 THE MACRO OUTPUT

This section shows several sample calls of the macro and the associated log and output.

Note: Sample 5 deals with an improperly named transport file (wrong file extension).

3.1 Sample 1 – show detail only for a transport file with single dataset

The Call:

```
%Uintransport(file=F:\temp2\lablab.xpt);
```

The Log:

```
************************ Reading File (F:\temp2\lablab.xpt) ************************

>>> This is a XPORT type file.

>>>  1 dataset(s) were found : LABLAB1

******************************** Finished Reading File ********************************
```

The Output:

No Detail requested

3.2 Sample 2 - show detail only for a transport file with mutiple datasets

The Call:

```
%Uintransport(file=F:\temp2\lablabM.xpt, import=N, detail=Y);
```

The Log:

```
************************ Reading File (F:\temp2\lablabM.xpt) ************************

>>> This is a XPORT type file.

>>>  2 dataset(s) were found : LABLAB1 LABLAB2

******************************** Finished Reading File ********************************
```

The Output:

List of variables by dataset in file F:\temp2\lablabM.xpt

Dataset name=LABLAB1

Obs	Dataset name	Variable name	Variable type	Variable label	Variable format	Variable length
1	LABLAB1	ACTEVENT	Numeric	Visit number	8.2	8
2	LABLAB1	ADDLNM	Character	Test name	$35.	35
3	LABLAB1	EXAMU	Character			40
4	LABLAB1	LABSTD	Numeric			8
5	LABLAB1	PTNO	Numeric	Patient number	10.0	8
6	LABLAB1	STUDY	Character	Trial number	$15.	9

Dataset name=LABLAB2

Obs	Dataset name	Variable name	Variable type	Variable label	Variable format	Variable length
7	LABLAB2	ACTEVENT	Numeric	Actual event	6.2	8
8	LABLAB2	ADDLNM	Character			35
9	LABLAB2	EXAMU	Character	Lab units - std	$10.	40
10	LABLAB2	LABSTD	Numeric	Lab value - std	15.5	8
11	LABLAB2	PTNO	Numeric	Patient number	10.0	8
12	LABLAB2	STUDY	Character	Study name	$9.	9

3.3 Sample 3 - show detail only for a transport file with mutiple datasets

The Call:

```
%Uintransport(file=F:\temp2\lablabM.cpt, import=N, detail=Y);
```

The Log:

```
************************** Reading File (F:\temp2\lablabM.cpt) **************************

>>> This is a CPORT type file.

>>>  2 dataset(s) were found : LABLAB1 LABLAB2

********************************** Finished Reading File **********************************
```

The Output:

List of variables by dataset in file F:\temp2\lablabM.cpt

Dataset name=LABLAB1

Obs	Dataset name	Variable name	Variable type	Variable label	Variable format	Variable length
1	LABLAB1	ACTEVENT	Numeric	Visit number	8.2	8
2	LABLAB1	ADDLNM	Character	Test name	$35.	35
3	LABLAB1	PTNO	Numeric	Patient number	10.0	8
4	LABLAB1	STUDY	Character	Trial number	$15.	9
5	LABLAB1	examu	Character			40
6	LABLAB1	labstd	Numeric			8

Dataset name=LABLAB2

Obs	Dataset name	Variable name	Variable type	Variable label	Variable format	Variable length
7	LABLAB2	ACTEVENT	Numeric	Actual event	6.2	8
8	LABLAB2	LABSTD	Numeric	Lab value - std	15.5	8
9	LABLAB2	PTNO	Numeric	Patient number	10.0	8
10	LABLAB2	STUDY	Character	Study name	$9.	9
11	LABLAB2	addlnm	Character			35
12	LABLAB2	examu	Character	Lab units - std	$10.	40

3.4 Sample 4 – import a selected dataset from a transport file

The Call:

```
%Uintransport(file=F:\temp2\lablabM.xpt, import=Y, dsname=lablab2);
```

The Log:

```
************************** Reading File (F:\temp2\lablabM.xpt) **************************

>>> This is a XPORT type file.

>>>  2 dataset(s) were found : LABLAB1 LABLAB2

*********************************** Finished Reading File ***********************************


*******************************Importing datasets:lablab2*********************************

NOTE: Input library UINTRANS is sequential.
NOTE: Copying UINTRANS.LABLAB2 to WORK.LABLAB2 (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines. System Option for BUFSIZE was used.
NOTE: There were 848 observations read from the data set UINTRANS.LABLAB2.
NOTE: The data set WORK.LABLAB2 has 848 observations and 6 variables.
NOTE: PROCEDURE COPY used:
      real time          0.10 seconds
      cpu time           0.01 seconds

*****************************All selected datasets Imported *****************************
```

The Output:

No Detail requested

3.5 Sample 5 – Show details and import all datasets from a transport file

The Call:

```
%Uintransport(file=F:\temp2\lablabMD.cpt,import=y, detail=Y);
```

The Log:

```
************************* Reading File (F:\temp2\lablabMD.cpt) *************************

>>> This is a XPORT type file.

>>>  2 dataset(s) were found : LABLAB1 LABLAB2

************************************ Finished Reading File ********************************

*********************************Importing datasets:_ALL_*********************************

NOTE: Input library UINTRANS is sequential.
NOTE: Copying UINTRANS.LABLAB1 to WORK.LABLAB1 (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines. System Option for BUFSIZE was
used.
NOTE: There were 76097 observations read from the data set UINTRANS.LABLAB1.
NOTE: The data set WORK.LABLAB1 has 76097 observations and 6 variables.
NOTE: Copying UINTRANS.LABLAB2 to WORK.LABLAB2 (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines. System Option for BUFSIZE was
used.
NOTE: There were 848 observations read from the data set UINTRANS.LABLAB2.
NOTE: The data set WORK.LABLAB2 has 848 observations and 6 variables.
NOTE: PROCEDURE COPY used:
      real time          0.70 seconds
      cpu time           0.13 seconds

*****************************All selected datasets Imported *****************************
```

NOTE: This filename had a .CPT extension, but the log showed that it was really a .XPT type file

The Output:

List of variables by dataset in file F:\temp2\lablabMD.cpt

Dataset name=LABLAB1

Obs	Dataset name	Variable name	Variable type	Variable label	Variable format	Variable length
1	LABLAB1	ACTEVENT	Numeric	Visit number	8.2	8
2	LABLAB1	ADDLNM	Character	Test name	$35.	35
3	LABLAB1	EXAMU	Character			40
4	LABLAB1	LABSTD	Numeric			8
5	LABLAB1	PTNO	Numeric	Patient number	10.0	8
6	LABLAB1	STUDY	Character	Trial number	$15.	9

Dataset name=LABLAB2

Obs	Dataset name	Variable name	Variable type	Variable label	Variable format	Variable length
7	LABLAB2	ACTEVENT	Numeric	Actual event	6.2	8
8	LABLAB2	ADDLNM	Character			35
9	LABLAB2	EXAMU	Character	Lab units - std	$10.	40
10	LABLAB2	LABSTD	Numeric	Lab value - std	15.5	8
11	LABLAB2	PTNO	Numeric	Patient number	10.0	8
12	LABLAB2	STUDY	Character	Study name	$9.	9

4 THE UINTRANSPORT MACRO CODE

The Macro first reads the file and scans each records for the distinctive type of records that exist for eather XPORT or CPORT type of files. This helps it decide what type of transport file it is dealing with (setting the internal macro variable ***tranflag*** correspondingly to CP or XP). Dependant on this flag , the appropriate macro code section (see the line comments) will be used. Three internal sub-macros `%Cont_mac` (get contents), `%Prt_mac` (print details) and `%Copy_mac` (import dataset) are used to facilitate the sub-tasks. The code is well documented with line comments in order for the reader to understand the purpose of the different code blocks.

```
/*===================================================================*
*                        Source code for Uintransport macro         *
* 12/09/2008             Program  by John Adams                 version 1.1  *
*                                                                   *
*===================================================================*
* Uintransport: Read an existing transport file to                  *
*             1.determine what type of file it is (Xport or Cport)  *
*             2.list datasets the file contains                     *
*             3.determine the datsets variables and attributes      *
*             4.import the selected (or all) datasets               *
*   usage    : %Uintransport(file=F:\temp2\lablab.xpt,              *
*                      detail=Y, import=Y);                         *
*-------------------------------------------------------------------*
* arguments: file  = transport file name (full path)        (req)   *
*          detail= list D/S variables?  (Y,N) [default=N]   (opt)   *
*          import= import dataset(s)?   (Y,N) [default=N]   (opt)   *
*          dsname= select dataset names     [default=ALL]   (opt)   *
*          outlib= target Lib for datasets  [default=WORK]  (opt)   *
* examples : %Uintransport(file=F:\temp2\lablabM.xpt, detail=Y);    *
*          %Uintransport(file=F:\temp2\lablabM.xpt);                *
*          %Uintransport(file=F:\temp2\lablabM.xpt, import=Y);      *
*          %Uintransport(file=F:\temp2\lablabM.cpt, import=Y,       *
*                      dsname=lablab2);                             *
*===================================================================*/;

%macro Uintransport(file=, detail=N, import=N, outlib=WORK, dsname=_ALL_);

  %if (%upcase(&import)=YES or %upcase(&import)=Y ) and &dsname eq %then %let dsname=_ALL_;
  %if &outlib eq %then %let outlib=WORK;

  %global  Ndsname dsnlst ;

   %let notes =%sysfunc(getoption(NOTES));                          /* retain Notes option setting */
   options  nonotes validvarname=v7;

 %macro Cont_mac;                                                   /* get contents of input file */
       proc contents data=Uintrans._all_ noprint    out=ds_inf1 ;      run;
   proc format;      value Vtypef   1 = 'Numeric'   2 = 'Character' ;   run;

   proc sql noprint;                                                /* get the dataset info*/
     create table ds_inf2 as
      select memname as dsname  label='Dataset name' ,   name  as vname  label='Variable name',
            type    as vtype   label='Variable type'    format=Vtypef.,
            length  as Vlength label='Variable length', label as Vlabel label='Variable label',
            case
              when   vtype = 2 and format  eq ' ' then ' '
               when  vtype = 2 and format  ne ' ' then left(trim(format)||trim(left(put(formatl,best8.)))||'.')
                when vtype = 1 and formatl eq 0   then ' '
                 else                             left(trim(format)||trim(left(put(formatl,best8.)))||'.'||left(put(formatd,best8.)))
                 end as Vformat label='Variable format'
      from ds_inf1 ;
     select      distinct dsname  into: dsnlst separated by ' ' from ds_inf2  ;
     select count(distinct dsname) into: Ndsname                from ds_inf2  ;
    quit;
     %put; %put %str(>>> ) %trim(&Ndsname) dataset(s) were found : &dsnlst;
     %put; %put ******************************** Finished Reading File *******************************;
%mend Cont_mac;
```

```
%macro Prt_mac;
  %if %upcase(&detail)=YES or %upcase(&detail)=Y %then %do;              /*print details for the dataset */
     Title1 "List of variables by dataset in file &file";
     proc print data=ds_inf2 label;
      by dsname ;
      var dsname vname vtype vlabel vformat vlength;
     run;
  %end;
%mend Prt_mac;

%macro copy_mac;                                                          /* copy the selected datsets */
  %if (%upcase(&import)=YES or %upcase(&import)=Y ) %then %do;
    %put;  %put; %put **********Importing datasets:&dsname**********;
     options  &notes;
   %if %upcase(&dsname)=%str(_ALL_) %then %do;
     proc copy in=Uintrans out=&outlib;                    run;
   %end;
   %else %do;
     proc copy in=Uintrans out=&outlib;   select &dsname ;   run;
   %end;
    %put **********All selected datasets Imported **********;
  %end;
    options nonotes;
%mend copy_mac;

%if %length(&file) lt 2  %then %put ERROR: You must specify a file name - Macro ending;

%else %do;

 data _null_;
  rc = filename('in_file',"%trim(&file)");
 run;

 data _null_;                                  /****** read thetransport file to determine the file type **********/
  if _n_=1 then do;
    retain tranflag Ndsname dsnLst;
    %put; put "********** Reading File (&file) **********" / ' ' ;
    xpflag=0;  cpflag=0;
    Ndsname=0; dsnLst='  ';
  end;
    INFILE in_file LRECL=100000 n=1 truncover  end=eof ;                        /* set file name */
    INPUT   text1  $ascii32000.  text2  $ascii32000. ;                        /* read only first record */

    ix1 = index(text1,'MEMBER  HEADER RECORD') ;  ix2 = index(text1,'DSCRPTR HEADER RECORD') ;
    ic1 = index(text1,'COMPRESSED') ;             ic2 = index(text1,'LIB CONTROL') ;

  if       ic2>ic1 then tranflag = 'CP';
  else if ix2>ix1 then tranflag = 'XP';

  if       tranflag = 'CP' then put  '>>> This is a CPORT type file.' / ' ';
  else if tranflag = 'XP' then put  '>>> This is a XPORT type file.' / ' ';
  call symput('tranflag',tranflag);
  rc = filename('in_file',' ');
  stop;
 run;


/************************************ Start of XPORT data file block ************************************************************/
  %if &tranflag =XP %then %do;
   libname Uintrans xport "&file";
     %Cont_mac;                                          /* prepare contents of file    */
     %Prt_mac;                                           /* print details for dataset   */
     %Copy_mac;                                          /* copy selected datasets      */
   libname Uintrans  ' ';
  %end;
/************************************** End of XPORT file block ************************************************************/
```

```
/************************************ Start of  CPORT data file block ***************************************************/
  %else %if &tranflag =CP %then %do;

                        %let Uintrans = %sysfunc(pathname(WORK))/Uintrans;
    %if %sysfunc(fileexist(&Uintrans)) = 0 %then %do;
       filename xdummy pipe "mkdir ""&Uintrans"" ";                                  /*create temp work folder*/
      data _null_;     infile xdummy;      run;
       filename xdummy;
       libname  Uintrans "&Uintrans";
    %end;

    %else %do;
       libname  Uintrans "&Uintrans";
      proc datasets lib = Uintrans kill nolist; run;                               /*clean out temp work folder*/
    %end;

        filename transfer "%trim(&file)";
      proc cimport lib = Uintrans  infile = transfer;    run;                          /*get all datasets  */
        filename transfer ;

    %Cont_mac;                                                                      * prepare contents of file  */
    %Prt_mac;                                                                       /* print details for dataset */
    %Copy_mac;                                                                    /* copy selected datasets    */

      proc datasets lib = Uintrans kill nolist;    run;                           /* clean  out temp folder    */

    %if %sysfunc(fileexist(&Uintrans)) = 1 %then %do;                                /*remove temp Folder */
       filename xdummy pipe "rmdir ""&Uintrans"" ";
      data _null_;  infile xdummy;              run;
       filename xdummy;
    %end;
      libname  Uintrans ;

  %end;
/************************************* End of CPORT file block ***********************************************************/

 %end;

 proc datasets library=work nolist;  delete ds_inf1 ds_inf2;  quit;                  /*house cleaning */
  options  &notes;

%mend Uintransport;
```

5 THE UTRANSPORTOUT MACRO CODE

So here is a bonus. The code for another small macro, one that can quickly create .CP and .XP transport files, is a shown here. It takes four argument (see description in the header) and is extremely simple to use. No detailed discussion of this macro's code is given in this paper.

```
/*====================================================================*
*                    Source code for Utransportout macro              *
* 05/12/2008              Program by John Adams            version 1.0  *
*                                                                      *
*====================================================================*
* Uouttransport: Create a transport file (Xport or Cport) from one or more datasets *
*                                                                      *
*   usage    : %Utransportout(file=F:\temp2\lablab.xpt,                *
*                       filetype=CPORT, dsname=AEAEA);                 *
*--------------------------------------------------------------------------------------------------------------------*
* arguments: file     = transport file name (full path)        (req)   *
*            filetype = Cport or Xport [default=Xport]          (opt)  *
*            inlib    = Lib for input datasets(default=WORK)  (opt)   *
*            dsname   = list D/S names                         (req)  *
*--------------------------------------------------------------------------------------------------------------------*
* examples : %Utransportout(file=F:\temp2\lablab.xpt, dsname=lab);     *
```

```
*          %Utransportout(file=F:\temp2\lablabM.xpt,                          *
*                    dsname lab1 lab2), filetype=Cport ;                      *
*===================================================================*/;

%macro Utransportout(file=, filetype=XPORT, inlib=WORK, dsname=, memtype=DATA);

  %if &filetype eq %then %let filetype=XPORT;
  %if &inlib eq %then    %let inlib   =WORK;
  %if &memtype  eq %then %let memtype =DATA;

  %if &dsname eq %then %put ERROR: You must specify one or more dataset names - macro ending.;
  %else %if &memtype ne DATA %then  %put ERROR - Memtype=&memtype is not supported at this time;
  %else %if    (%upcase(%scan(&file,2,%str(.)))=CP and %upcase(&filetype)=XPORT)
          or (%upcase(%scan(&file,2,%str(.)))=XP and %upcase(&filetype)=CPORT) %then
       %put ERROR: You specified the wrong file extension. Use .XP for XPORT, .cp for CPORT - macro ending.;
  %else %do;
      %let i=1; %let wd1= %scan(&dsname ,&i,%str( ));  %let dsnlst=;
    %do %while(&wd1 ne );
      %if memtype=DATA and %sysfunc(exist(&inlib..&wd1))=0 %then  %put WARNING: &inlib..&wd1 dataset does not exist.;
      %else %let dsnlst = &dsnlst &wd1;
      %let i=%eval(&i+1);  %let wd1= %scan(&dsname ,&i,%str( ));
    %end;


    %let ndsn = %eval(&i-1);
    %if &ndsn = 0  %then %put ERROR: You must specify one or more valid dataset names - macro ending.;

    %else %do;
      %if %upcase(&filetype)=CPORT %then %do;                                  /*put dataset(s) to cport file */
          filename transfer "&file";
        proc cport  lib=&inlib memtype=&memtype file=transfer;
          select &dsnlst. ;
        run;
          filename transfer ;
      %end;
      %else %if %upcase(&filetype)=XPORT %then %do;                            /*put dataset(s) to xport file */
          libname outdat xport "&file";
        proc copy  in=&inlib force out=outdat memtype=&memtype ;
          select &dsnlst. ;
        run;
          libname outdat ;
      %end;
    %end;
  %end;
%mend Utransportout;
```

CONCLUSIONS

The UinTransport and UtransportOut macros are extremely useful tools to help automate and simplify the handling of transport files by ocasional users. This eliminates the necessity of having detailed programming knowledge about how to read, write or import from the different types of transport files.

REFERENCES

[1] SAS Institute - Product documentation > SAS 9.2 Documentation > Moving and accessing Files :File Headers

[2] SAS Institute - Technote TS-140 The record layout of a dataset in SAS Transport (created with XPORT engine)

[3] SAS Institute - Support Note 22656 difference between transport file created by Proc Copy with XPORT engine or Proc Xcopy

CONTACT INFORMATION

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

John H Adams
900 Ridgebury Road
Ridgefield, CT, 06877-0368
Work Phone: 203-778-7820
Fax: 203-837-4413
Email: john.adams@boehringer-ingelheim.com
adamsjh@mindspring.com