Paper SM01

# SAS ODS Package for Efficient Creation of Zip File for P21 Validation

Venkatesh Burla, Merck & Co., Inc., Rahway, NJ, USA


## ABSTRACT

Pinnacle 21 validation is a critical step to prepare FDA eSubmission package. It ensures compliance with the regulatory guidelines and validates the accuracy and consistency of clinical trial data. ZIP file is a mandatory input to perform P21 validation. Manually creating a ZIP file leads to inefficiencies and potential errors. Although ZIP compression tools (7-zip and WinZip) are widely used, they are not convenient with in the SAS program. Leveraging the power of ODS, comprehensive solution is developed to automate the process and reduce the time/effort while maintaining data integrity. This paper presents a novel approach utilizing the SAS Output Delivery System (ODS) to streamline the process of creating ZIP files for P21 validation. Our solution offers a seamless macro with ODS capabilities, enable users to create transport files from the libraries, verify the existence of submission files, such as ADRG, define and ARM in the directory, and finally create ZIP file to perform P21 validation.


## INTRODUCTION

Pinnacle21 is a software suite designed for clinical data management and regulatory submission process in the pharmaceutical and life sciences industries. It plays a crucial role in ensuring the quality, compliance, and submission readiness of clinical data. Pinnacle 21 is widely utilized for the validation of SDTM and ADaM datasets, with a specific focus on datasets formatted according to CDSIC (Clinical Data Interchange Standards Consortium) standards. Some of the key features and reasons for using Pinnacle 21 include: CDISC Compliance, Data Quality Assurance, Submission Readiness, Efficiency and Productivity. To execute P21, it is essential to have a mandatory zip file containing the necessary xpt files of the datasets and submission files – adrg.pdf, define.xml.

Although Zip compression tools are widely used, they are not convenient to use with in the SAS program. Beginning with SAS 9.2. it is possible to generate the custom zip file using the ODS package statements. This paper delves into the details of leveraging the SAS ODS Package to facilitate the efficient creation of Zip file with macro (**XPT0DEF0ZIP.SAS).**

## HOW IT WORKS?

This macro creates XPT files from input parameters by invoking macro (**setxpt.sas)** and subsequently produces a zip file using SAS ODS functionality. Detailed descriptions of the macro variables can be found in the table below, while the comprehensive macro and a sample call to the macro are provided in Appendix 1, 2, and 3.

## SYNTAX: ODS PACKAGE STATEMENT

ODS PACKAGE statement Opens, adds to, publishes, or closes one SAS Output Delivery System (ODS) package object. Syntax:

```
ODS PACKAGE (<name>) OPEN <options>;
ODS PACKAGE (<name>) PUBLISH
   Transport PROPERTIES(transport-property-1”value-1” …transport-property-n =”value-n”);
ODS PACKAGE (<name>) ADD FILE=file-specification | DATA=member-specification
MIMETYPE=”string” <PATH=path-specification> <options>;
ODS PACKAGE (<name>) CLOSE <CLEAR>;
```

## DESCRIPTION AND FUNCTIONS OF THE MACRO

**MACRO VARIABLES AND THEIR DESCRIPTION USED IN THE MACRO: XPT0DEF0ZIP.SAS**

| MACRO VARIABLE | DESCRIPTION | VALID VALUES |
|---|---|---|
| Sdtm_in | SDTM datasets to be included in the ZIP file | %str(), datasets names separated by “\|”. EX: AE\|DM |
| sdtm_libname | Library reference for the location of SDTM datasets | Lptss |

| | | |
|---|---|---|
| adam_in | ADaM datasets to be included in the ZIP file | %str(), datasets names separated by "\|". EX: adsl\|adae |
| adam_libname | Library reference for the location of ADaM datasets | lptda |
| create_xpt | Option to create XPT files passed from the sdtm_in and adam_in | %str(), Y |
| xpt_libref | Location for the XPT files (Req) to be created | lptda |
| include_sfiles | Option for the submission files (Define, ADRG, ARM) to be included in the ZIP file | %str(), Y |
| Exclude_file | Option to exclude any file to be excluded from the zip file | %str(), ae\|dm |
| zip_file | Name of the ZIP file to be created | |
| zip_file_loc | Location of the ZIP file to be created | |
| delete_xpt | Option to delete XPT files after creating the ZIP file | %str(), Y |
| delete_old_zip | Option to delete old zip files in the xpt_libref location | %str(), Y |

The macro has the following functions:

#### 1. DELETING OF PREVIOUS XPT FILES

Throughout the P21 run, xpt files are generated multiple times to address issues. To avoid the unnecessary retention of outdated files, all xpt files referenced in the '**xpt_libref**' are deleted.

#### 2. CREATION OF XPT FILES

The macro xpt0def0zip.sas invokes setxpt.sas macro to generate XPT files when parameter '**create_xpt**' set to 'Y' for SDTM and ADaM datasets based on input parameters, with '**sas_input'** specifying the dataset location. By default, the macro selects SDTM datasets ('dm', 'ae', and 'ds'), and all ADaM datasets are selected for xpt files creation. Additionally, the parameter '**filter'** allows the selection of specific SDTM or ADaM datasets. The macro implementation setxpt.sas can be seen in Appendix 1.

#### 3. VERIFY THE EXISTENCE OF SUBMISSION FILES

Submission files, including adrg, define, and optionally arm, are essential for the analysis package. While not obligatory for running P21 and often not created during early stages or interim analysis, the parameter **include_sfiles** assesses their existence of the submission files. When the parameter is set to 'Y', puts a WARNING message in the log if these submission files are not found.

#### 4. SELECTION OF FILES TO BE INCLUDED IN THE ZIP FILE

The final zip file encompasses all selected XPT files from '**sdtm_in**' and '**adam_in**', along with 'adrg.pdf', 'define.xml', 'define.xls', and 'analysis-results-metadata.pdf'. If these is a need to exclude any non-ADaM (intermediate) dataset or any file from the '**xpt_libref**', specify the file(s) to be excluded without the filename extension in the '**exclude_file**' parameter. Multiple filenames can be passed, separated by the '|' character. By default, the zipfile is generated with 'round0systemdatetime' (ex: round012JAN2024T11_41_32) in the xpt_libref location. However, you have the flexibility to control the name through the '**zip_file**' parameter, and the location of the zip file can be managed using the '**zip_file_loc**' parameter.

#### 5. CLEANUP

Following the creation of the zip file, you have the option to delete the generated XPT files by setting the parameter '**delete_xpt**' to 'Y'. Similarly, you can retain the old zip file by setting the parameter '**delete_old_zip**' to 'N'. The default values ('Y') preserve both the XPT files and zip file.

## CONCLUSION

In conclusion, The SAS ODS package offers a streamlined method for generating ZIP files. This approach not only streamlines the conversion of SAS datasets to XPT files but also seamlessly integrates the packaging of these files along with submission files into a zip format. The macro not only enhances productivity by reducing manual intervention but also saves considerable amount of time comparing to manual creation of zip file. During a test study, manually creating a zip file consumed 16 minutes, while employing the macro reduced the process to less than 1-2 minutes.

## REFERENCES

https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/odsug/p19kppnlyy01n7n18jcfsy0eutvq.htm

## ACKNOWLEDGMENTS

The author would like to thank Merk's PI and ADaM CoE team for the collaborative efforts and valuable suggestions in developing the macro, and Amy Gillespie, Sai Govind Chenna, Dileep Penmetsa, Liu Li and Himanshu Patel for reviewing the paper and providing great suggestions.

## CONTACT INFORMATION

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

Venkatesh Burla
Merck & Co., Inc., Rahway, NJ, USA
215-631-5295
Email: venkatesh.burla@merck.com

## APPENDIX 1: SETXPT MACRO.SAS

## EXAMPLE FOR LIBNAME:

**ADaM:** libname lptda "/opt/bardsar/test/mkxxx-xxx/protxx-csr01/dataanalysis/";
**SDTM:** libname lptss "/opt/bardsar/test/mkxxx-xxx/protxx-csr01/datasdtm/";

| SAS SCRIPT | FUNCTION |
|---|---|
| Macro variables:<br>sas_input<br>xpt_out<br>filter | <br>libname reference for the SAS datasets<br>output location for the XPT files<br>option to select specific datasets (missing will create XPTs for all datasets) |
| %let _subset_dsn = %str();<br>%if %length(&filter) > 0 %then %do;<br>%local _subset_dsn _n_ndsets _dsin;<br><br>%let _ndsets=%eval(%sysfunc(countw(&filter.,"\|")));<br>%do _d=1 %to &_ndsets.;<br>%let _dsin = %upcase(%sysfunc(quote(%scan(&filter, &_d,'\|'))));<br>%let _subset_dsn= &_subset_dsn &_dsin;<br>%end;<br>%end; | Processing when only specific datasets are needed to create XPTs with &filter macro variable. |
| proc sql noprint;<br>create table sastoxpt as<br>(select lowcase(memname) as memname, nobs from dictionary.tables<br>where libname=%upcase("&sas_input.") and memtype="DATA" and nobs ne 0<br>%if %length(&_subset_dsn) %then %do;<br>and memname in (&_subset_dsn)<br>%end; )<br>order by memname;<br>quit; | Getting the list of SAS files to be converted to XPT files. When missing value is assigned for &filter, all the datasets from the sas_input are selected. |
| %let path=%sysfunc(cats(%sysfunc(pathname(&xpt_out)),/));<br>%put &path.; | Step to add '/' at the end of location of the libname reference to output XPT file |
| data _null_;<br>retain _outxpt "&path";<br>set sastoxpt;<br>call execute("libname _xptout xport %nrstr('"\|\|trim(_outxpt)\|\|trim(memname)\|\|".xpt');");" | Using XPORT engine to create the XPT files for the SAS datasets selected from the above step. |

| | |
|---|---|
| call execute('data _xptout.'\|\|trim(memname)\|\|'; set &sas_input..'\|\|trim(memname)\|\|'; run;');<br>call execute('libname _xptout clear;');<br>run;<br><br>proc datasets nolist memtype=data;<br>delete sastoxpt;<br>run;<br>quit; | Deleting the work dataset selected in the above step. |

**APPENDIX 2: XPT0DEF0ZIP.SAS MACRO**

| SAS SCRIPT | FUNCTION |
|---|---|
| %let err1 = ERR; %let err2 = OR; %let war1 = WAR; %let war2 = NING;<br><br>%*** set up default parameter values if the value is not entered;<br>%if %nrbquote(&sdtm_in)= %nrbquote( ) %then %let sdtm_in = %str(ae\|dm\|ex);<br>%if %nrbquote(&sdtm_libname)= %nrbquote( ) %then %let sdtm_libname = %str(lptss);<br>%if %nrbquote(&adam_libname)= %nrbquote( ) %then %let adam_libname = %str(lptda);<br>%if %nrbquote(&create_xpt)= %nrbquote( ) %then %let create_xpt = %str(Y);<br>%if %nrbquote(&include_sfiles)= %nrbquote( ) %then %let include_sfiles = %str(Y);<br>%if %nrbquote(&zip_file)= %nrbquote( ) %then %let zip_file = %str(round);<br>%if %nrbquote(&delete_xpt)= %nrbquote( ) %then %let delete_xpt = %quote(N);<br>%if %nrbquote(&delete_old_zip)= %nrbquote( ) %then %let delete_old_zip = %quote(Y); | Assign default values when missing values are passed. Ex: dm\|ex\|ae for SDTM datasets, all datasets for ADaM datasets (lptda) |
| %*** Get the path of the xpt_libref to create/delete the xpt files **;<br>%local xptpath pfiles ;<br><br>proc sql noprint;<br>select path into :xptpath<br>from dictionary.libnames<br>where libname= %upcase("&xpt_libref");<br>quit;<br><br>%if %length(&xptpath)=0 %then %do;<br>%put ==Usage for Macro &sysmacroname==;<br>%put ;<br>%put &err1&err2: (&sysmacroname) xpt_libref is a required parameter. Please assign libname ;<br>%put &err1&err2: (&sysmacroname) reference in xpt_libref variable and resubmit. ;<br>%put ;<br>%put ===Usage for Macro &sysmacroname====;<br>%goto exit;<br>%end;<br><br>%if %nrbquote(&zip_file_loc) ne %nrbquote() %then %do;<br>%let _char= %sysfunc(first(%sysfunc(reverse(&zip_file_loc.))));<br>%if %nrbquote(&_char) ne %nrbquote(/) %then %let zip_file_loc = %sysfunc(cats(&zip_file_loc, /));<br>%put &zip_file_loc.;<br>%end; | Fetching the location for the XPT files to be created.<br><br>Abort the execution of the macro when null value is passed to the xpt_libref. Xpt_libref is a required parameter.<br><br>Check and add the “/” character at the end of the path. |

| | |
|---|---|
| %if %nrbquote(&xptpath) ne %nrbquote() %then %do;<br>%let _char= %sysfunc(first(%sysfunc(reverse(&xptpath.))));<br>%if %nrbquote(&_char) ne %nrbquote(/) %then %let xptpath =<br>%sysfunc(cats(&xptpath, /));<br>%put &xptpath.;<br>%end;<br><br>%*** check Zip File location ;<br>%if %nrbquote(&zip_file_loc)= %nrbquote( ) %then %let<br>zip_file_loc = %str(&xptpath); | |
| %*** Delete the XPT/ZIP files ***;<br>**%macro** delete_files (filetyp);;<br>filename filelist "&xptpath.";<br>data _null_;<br>dir_id = dopen('filelist');<br>total_members = dnum(dir_id);<br>do i = **1** to total_members;<br>member_name = dread(dir_id,i);<br>if scan(lowcase(member_name),**2**,'.')="&filetyp" then do;<br>file_id = mopen(dir_id,member_name,'i',**0**);<br>if file_id > **0** then do;<br>freadrc = fread(file_id);<br>rc = fclose(file_id);<br>rc = filename('delete',member_name,,'filelist');<br>rc = fdelete('delete');<br>end;<br>rc = fclose(file_id);<br>end;<br>end;<br>rc = dclose(dir_id);<br>run;<br>**%mend** delete_files;<br><br>%if %qupcase(&create_xpt) = %quote(Y) %then %do;<br>%***delete_files***(xpt);<br>%*** Invoking xpt macro to craete ADaMs ***;<br>%***setxpt***(sas_input = &adam_libname.<br>,xpt_out =&xpt_libref<br>,filter = %str(&adam_in)<br>);<br><br>%*** Invoking xpt macro to craete SDTMs ***;<br>%***setxpt***(sas_input = &sdtm_libname.<br>,xpt_out =&xpt_libref<br>,filter = %str(&sdtm_in)<br>);<br>%end; | Deleting the old XPT files if any before creating the XPT files in a directory.<br><br>Invoking the setxpt macro to create the XPT files based on the input parameters. |
| **data** files(keep=files);<br>length fref $**8** files fileloc $**1000**;<br>rc = filename(fref, "&xptpath");<br>if rc = **0** then do;<br>did = dopen(fref);<br>rc = filename(fref);<br>end;<br>else do;<br>length msg $200.;<br>msg = sysmsg();<br>put msg=;<br>did = .;<br>end;<br><br>if did <= **0** then putlog 'ER' 'ROR: Unable to open directory.';<br>dnum = dnum(did);<br>do i = **1** to dnum; | Fetch the files to be zipped from the xptpath directory. |

| | |
|---|---|
| files = dread(did, i);<br>fid = mopen(did, files);<br>if fid > 0 then do;<br>fileloc="&xptpath/"\|\|files;<br>output;<br>end;<br>end;<br>rc = dclose(did);<br>**run**;<br><br>**data** ds;<br>set files;<br>if upcase(scan(files,2,'.')) in ("XPT") %if %qupcase(&include_sfiles) = %quote(Y) %then or upcase(scan(files,2,'.')) in ("PDF") or upcase(scan(files,2,'.')) in ("XML") or upcase(scan(files,2,'.')) in ("XSL");;<br>**run**; | Select the files with extension: XPT, PDF, XML, and XSL. |
| %if %qupcase(&include_sfiles) = %quote(Y) %then %do;<br>**proc sql**;<br>create table _docs (docs char(50));<br>insert into _docs<br>values('adrg.pdf')<br>values('analysis-results-metadata.pdf')<br>values('define.pdf')<br>values('define.xml')<br>values('define2-0-0.xsl');<br>**quit**;<br><br>**proc sql**;<br>create table missfiles as select * from _docs<br>where upcase(docs) not in (select upcase(files) from ds<br>where upcase(scan(files,2,'.')) in ("PDF", "XML", "XSL")) ;<br>**quit**;<br><br>**proc sql** noprint;<br>select distinct docs, count(*) into :pfiles separated by ",", :nmissf from missfiles;<br>**quit**;<br><br>%if &nmissf > 0 %then %do;<br>%do a = 1 %to &nmissf.;<br>%let efile = %scan(%bquote(&pfiles.), &a, ',',);<br>%if not(%sysfunc(fileexist(&efile))) %then %do;<br>%put &war1&war2: (&sysmacroname) The &efile file is missing in the zip file.;<br>%end;<br>%end;<br>%end;<br><br>%end; | Check for the submission files when include_sfiles is set to Y. |
| **data** _null_;<br>timenow=put(time(),time.);<br>datenow=put(date(),date9.);<br>time=translate(timenow,'_',':');<br>call symputx ('dtnow',strip(datenow) \|\| "T" \|\| strip(time));<br>**run**;<br><br>%*** Exclude files to be zipped if exclude_files is populated ***;<br>%if &exclude_file ne %str() %then %do;<br>**data** exl;<br>length exclude $10;<br>%let dex = 1; | Get the System run date and time to use as a suffix in the zip file name.<br><br>Exclude the files from the zip file if any wanted or non-standard datasets are present in the xptpath directory. |

| | |
|---|---|
| %do %while (%scan(%bquote(&exclude_file.), &dex, '|',) ne %str( ));<br>exclude = "%scan(%bquote(&exclude_file.), &dex, '|',)";<br>output;<br>%let dex = %eval(&dex + 1);<br>%end;<br>**run**;<br><br>**proc sql**;<br>  create table dsf as select *<br>  from ds<br>  where upcase(scan(files,**1**,'.')) not in (select upcase(exclude) from exl);<br>**quit**;<br><br>%*** Read the file names to create ZIP file ***;<br>**proc sql** noprint;<br>  select files into :adams separated by "," from dsf;<br>  select count(*) into :nfiles from dsf;<br>**quit**;<br>%end;<br><br>%else %do;<br>**proc sql** noprint;<br>  select files into :adams separated by "," from ds;<br>  select count(*) into :nfiles from ds;<br>**quit**;<br>%end;<br><br>%*** Delete old zip files ***;<br>%if %qupcase(&delete_old_zip) = %quote(Y) %then<br>%***delete_files***(zip);; | Selecting all the files from xptpath when no files are selected to be excluded.<br><br>Invoking the delete_files macro to delete old zip files if needed. |
| %*** ODS PACKAGE for creating the ZIP file ***;<br>ods package (SapphireStyle) open nopf;<br>%do i = **1** %to &nfiles.;<br>%let dname = %scan(%bquote(&adams.), &i, ',',);<br>%put &dname.;<br><br>ods package (SapphireStyle)  add file="&xptpath.&dname.";<br>%end;<br>ods package(SapphireStyle) publish archive properties(archive_name="&zip_file.0&dtnow..zip" archive_path="&zip_file_loc");<br>ods package (SapphireStyle)  close;<br><br>%*** Delete old XPT files ***;<br>%if %qupcase(&delete_xpt) = %quote(Y) %then<br>%***delete_files***(zip);;<br><br>%exit: | Invoking the ODS package to ZIP the files selected in the above step |

## APPENDIX 3: SAMPLE MACRO CALL

**EXAMPLE 1:**

```
%xpt0def0zip(
 sdtm_in = %str()
,adam_in= %str()
,sdtm_libname = %str()
,adam_libname = %str()
,create_xpt  = %str(Y)
,xpt_libref = %str(lptda)
,include_sfiles = %str()
,exclude_file = %str(ae)
,zip_file = %str()
,zip_file_loc = %str()
,delete_xpt = %str()
,delete_old_zip = %str()
);
```

**EXAMPLE 2:**

```
%xpt0def0zip(
 sdtm_in = %str(ae|ds)
,adam_in= %str(adsl|adae|adex|adexsum)
,sdtm_libname = %str(lptss)
,adam_libname = %str(lptda)
,create_xpt  = %str(Y)
,xpt_libref = %str(lptda)
,include_sfiles = %str(Y)
,exclude_file = %str(axpb)
,zip_file = %str(adamrun)
,zip_file_loc = %str(lpteasd)
,delete_xpt = %str(Y)
,delete_old_zip = %str(N)
);
```