PharmaSUG 2016 - Paper AD09

The Devil is in the Details – Reporting from Pinnacle 21 (OpenCDISC) Validation Report

Amy Garrett, Novella Clinical, Columbus, OH
Chris Whalen, Clovis Oncology, Boulder, CO

ABSTRACT

Pinnacle 21 Community Validator (P21C, previously known as OpenCDISC Community) is a valuable tool for SDTM implementers, but the resulting report has some limitations. The P21C report does not display full information about erroneous records, making it difficult for the reviewer to discern which issues are caused by dirty data versus incorrect programming or data mapping. This limitation means programmers must manually look up records based on observation number (provided on the details tab of the P21C report), which is extremely time-consuming. This cumbersome process creates the need for a detailed listing report to expedite the review of P21C validation findings. The resulting report includes a series of customizable listings organized by domain and issuer ID (FDA Publisher ID). The final output saves time, helps programmers understand issues more thoroughly, and provides a tangible product that can be delivered to other team members for further investigation, including data management for querying. This paper focuses on how to create a detailed report of data issues requiring further inquiry and is intended for use by the data management team.

INTRODUCTION

Pinnacle 21 Community Validator is the leading industry tool for validating SDTM data sets against CDISC standards (for more information about CDISC standards, please see cdisc.org). After the validator has finished checking the SDTM data sets, findings are made available to the user, typically in Excel format. The findings report consists of four tabs: Datasets Summary, Issue Summary, Details, and Rules.

The Datasets Summary tab provides an overview of the contents for each input file and contains summary information about the total number of records, errors, warnings, and notices for each domain. The Issue Summary tab breaks down issues by severity (error, warning, and notice) and by type for each domain. Each issue type is categorized by FDA Publisher ID, which represents the FDA's published business rules. A description of each rule can be found on the Rules tab. The Details tab includes all issues in an expanded format and is presented on the record level. This tab includes the domain, record number, count, variables, values, rule ID, message, category, and severity for each issue[1]. A sample details tab is provided below in figure 1, along with the flow of a record back to its source data set in figure 2.

Figure 1 is a modified sample output of a the Details tab of a P21C report

Domai	Recor	Variables	Values	Publisher I	Message	Category	Severit
CM	5880	CMENRF, CMSTDTC, CMENDTC	ONGOING, null, null	FDAC122	Missing values for CMSTDTC, CMSTRF and CMSTRTPT, when CMENDTC, CMENRF or CMENRTPT is provided	Consistency	Warning
CM	5881	CMENRF, CMSTDTC, CMENDTC	ONGOING, null, null	FDAC122	Missing values for CMSTDTC, CMSTRF and CMSTRTPT, when CMENDTC, CMENRF or CMENRTPT is provided	Consistency	Warning
CM	5882	CMENRF, CMSTDTC, CMENDTC	ONGOING, null, null	FDAC122	Missing values for CMSTDTC, CMSTRF and CMSTRTPT, when CMENDTC, CMENRF or CMENRTPT is provided	Consistency	Warning

Figure 1. Record 5,882 in the CM domain is non-compliant with FDAC122.

Figure 2 is a screenshot of the source CM data set

	Domain Abbreviation	Study Identifier	Unique Subject Identifier	Sequence Number	End Relative to Reference Period	Start Date/Time of Medication	End Date/Time of Medication	Reported Name of Drug, Med, or Therapy	Standardized Medication Name	Category for Medication
5880	CM	ABC-123	ABC-123-123456	1	ONGOING			COLACE	Docusate sodium	CONCOMITANT MEDICATIONS
5881	CM	ABC-123	ABC-123-123456	2	ONGOING			LIPITOR	Atorvastatin calcium	CONCOMITANT MEDICATIONS
5882	CM	ABC-123	ABC-123-123456	3	ONGOING			METFORMIN	Metformin	CONCOMITANT MEDICATIONS

Figure 2. In the CM domain, record 5,882 has a missing value for CMSTDTC, when CMENRF is provided and is not in compliance with FDAC122.

In a 2014 paper by Peterson and Adams, the authors stated "the validator tool in many cases does not provide specific enough detail for a data reviewer to quickly identify the relevant record, locating this within the EDC system and associated documentation"[2]. This limitation has mandated the practice of tracing records from the Details tab back to source data sets so that the programmers can communicate their findings to data management (D.M.) with complete subject information including USUBJID, which the Details tab does not routinely report by default. While tracing back just a few records can be simple, if the details tab contains thousands of issues, this task quickly becomes overwhelming and communicating findings becomes a complete nightmare with e-mails going every which way only to leave all parties thoroughly confused. This has led some to advocate against using P21C for data cleaning alltogether[2]. We recognize that while the process is not perfect, there is value in using P21C for data review. Programmers need an efficient tool for sorting through P21C findings as well as an effective process for communicating issues to team members. This paper shows the reader how to create a custom multi-tabbed listing which clearly displays suspected issues and provides sufficient detail to trace problems back to source data. D.M. can use the report to quickly address data integrity problems within the EDC.

STEP ONE: REVIEW P21C FINDINGS

Understanding P21C findings can be difficult and it is the responsibility of the programmer to discern which issues can or cannot be resolved. It is common to have issues come from multiple sources such as dirty data, incorrect mapping, and programming errors. Additionally, some issues are present simply because the study is ongoing (this is especially true with issues related to the Disposition domain). Before the programmer can communicate issues to D.M., the programmer must discern which issues are suspected to be data related. Annotating the P21C Issue Summary tab is a good place to note issues that are believed to be data related as well as any caveats that may apply. Furthermore, annotating the P21C report can help the programmer track new issues as they appear as well as monitor issues over the course of the study to ensure their resolution.

Figure 3 is a sample Issue Summary with user annotations

Issue Summary					
Source	Publisher ID	Message	Severity	Found	Comment
CM					
	FDAC107	CMSTDTC is after CMENDTC	Error	1	DATA ISSUE
	FDAC183	Missing value for CMDOSU, when CMDOSE, CMDOSTXT or CMDOSTOT is provided	Error	4	DATA ISSUE where CMDOSTXT NOT in('UNK','UNKNOWN')
LB					
	FDAC212	Duplicate records	Warning	11	DATA ISSUE
PE					
	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	Warning	14	DATA ISSUE
	FDAC212	Duplicate records	Warning	13	DATA ISSUE
	FDAC176	Missing value for PEREASND, when PESTAT is 'NOT DONE'	Warning	1166	DATA ISSUE if PETESTCD = PEALL, else not collected
TR					
	FDAC178	Missing value for TRORRES, when TRSTAT or TRDRVFL is not populated	Warning	2	DATA ISSUE
VS					
	FDAC178	Missing value for VSORRES, when VSSTAT or VSDRVFL is not populated	Warning	2	DATA ISSUE

Figure 3. An Issue Summary tab annotated for data issues

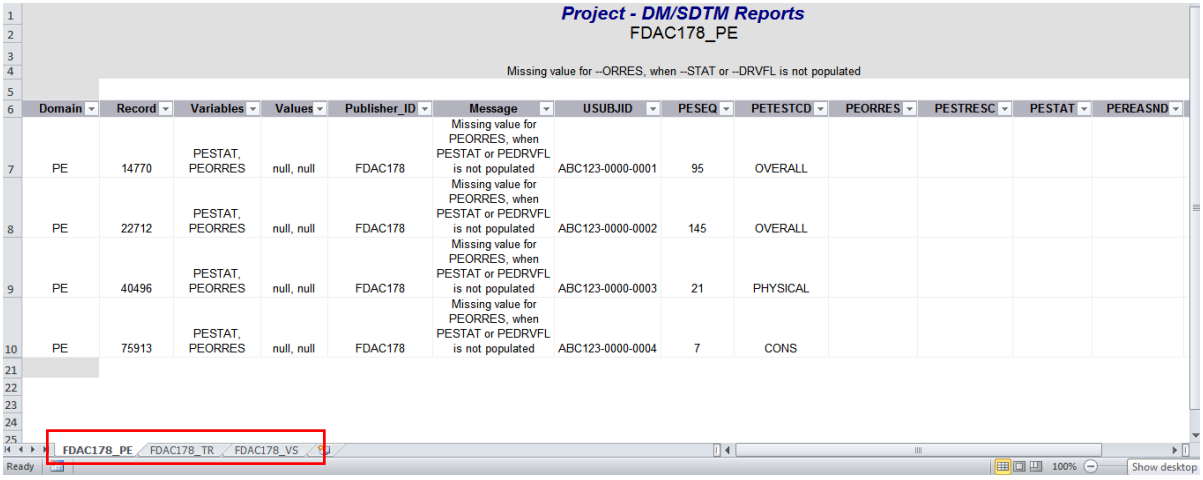
DISPLAYING FINDINGS: A PREVIEW

Once the Issue Summary tab has been annotated, we'll take these steps to extract user friendly information from the report:

1. Look up each record listed on the details tab in the source domain
2. Filter findings by applying any pre-defined conditions
3. Output the record, along with a description of the issue, into specific tabs based on domain/Publisher ID combination.

In the example below, FDAC178 (missing –ORRES when –STAT or –DRVFL is not populated) affects PE, TR, and VS domains; as such, a separate tab for each domain is created containing their respective FDAC178 findings. Each tab is clearly labeled with the applicable Publisher ID and domain. When findings are presented like the sample below, D.M. can quickly identify the subject and visit (not pictured) within the EDC and take the appropriate action. The remainder of the paper will focus on the technical aspects of how to create a report similar to figure 4.

Figure 4 is a sample output report based on P21C validator details tab



Project - DM/SDTM Reports
FDAC178_PE

Missing value for --ORRES, when --STAT or --DRVFL is not populated

Domain	Record	Variables	Values	Publisher_ID	Message	USUBJID	PESEQ	PETESTCD	PEORRES	PESTRESC	PESTAT	PEREASND
PE	14770	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	ABC123-0000-0001	95	OVERALL				
PE	22712	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	ABC123-0000-0002	145	OVERALL				
PE	40496	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	ABC123-0000-0003	21	PHYSICAL				
PE	75913	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	ABC123-0000-0004	7	CONS				

FDAC178_PE FDAC178_TR FDAC178_VS

Figure 4. In the sample report, only relevant sections have been kept due to spacing. STUDYID, PETEST, PECAT, VISIT, VISINUM, PEDTC, PEDY, have been hidden due to limited space.

STEP TWO: PROGRAM SETUP

Prior to any real programming, we must first set the global macro variable **lib.** This should represent the LIBNAME where the domains are stored in sas7bdat format. These files must be the same data sets that were used to create the transport files for P21C validation.

```
* Set the path where SAS dataasets are stored in sas7bdat format ;
libname sdtm "C:\SDTM\Data"
* Set the global macro variable: lib  ;
* This global macro is used throughout - make sure to upcase! ;
%global lib;
%let lib = SDTM ;
```

Next, the details tab of the P21C report is imported (import code not provided) and variables not needed for the final report are dropped from the newly created details data set. For the purpose of this paper, count, category, severity, and Pinnacle 21 ID are considered to be extraneous information not helpful for data review. Depending on the audience, different variables may need to be kept or dropped. At a minimum, domain and record number are required as they will serve as keys in the hash programming portion of the base macro (more on that later). This is the first place that custom programming can be entered. For example, if the user does not wish to print LB records for a specific Publisher ID, those records can be removed from the details data set directly after import. When a Publisher ID also contains records belonging domains that we want to report, removing unneeded records from the details data saves processing time but is not required; there will a second opportunity to add custom programming.

```
* Delete LB records where Publisher ID = FDAC178;
Data details;
    Set details;
    If domain = 'LB' and Publisher_ID = 'FDAC178' then delete;
Run;
```

STEP THREE: CREATE PROGRAM FOR ALL UNIQUE PUBLISHER IDS

We need to refer to the notes on the Issue Summary tab (step one) and determine the number of unique publisher IDs to be reported. In the sample below, there are four unique Publisher ID codes. Each unique Publisher ID requiring output in the final report is included in the base macro parameter. After the macro has been executed, the second opportunity to add custom code is presented (see sample program format below based on notes in figure 5). Adding custom code ensures D.M. is only looking into records that are thought to be data issues.

Figure 5 is a sample of an annotated Issue Summary tab with unique Publisher IDs numbered.

Issue Summary					
Source	Publisher ID	Message	Severity	Found	Comment
CM					
	FDAC183 1	Missing value for CMDOSU, when CMDOSE, CMDOSTXT or CMDOSTOT is provided	Error	4	DATA ISSUE where CMDOSTXT NOT in('UNK','UNKNOWN')
LB					
	FDAC212 2	Duplicate records	Warning	11	DATA ISSUE
PE					
	FDAC178 3	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	Warning	14	DATA ISSUE
	FDAC212	Duplicate records	Warning	13	DATA ISSUE
	FDAC176 4	Missing value for PEREASND, when PESTAT is 'NOT DONE'	Warning	1166	DATA ISSUE if PETESTCD = PEALL, else not collected
TR					
	FDAC178	Missing value for TRORRES, when TRSTAT or TRDRVFL is not populated	Warning	2	DATA ISSUE
VS					
	FDAC178	Missing value for VSORRES, when VSSTAT or VSDRVFL is not populated	Warning	2	DATA ISSUE

Figure 5. There are four unique Publisher IDs that need to be included in the report

```
/*** Call the base macro to find observation numbers ***/
%base(FDAC183 FDAC212 FDAC178 FDAC176);

/*** Insert custom code if necessary ***/
data FDAC183;
    set FDAC183;
    if domain = 'CM' and CMDOSTXT in('UNK','UNKNOWN') then delete;
run;

data FDAC176;
    set FDAC176;
    if domain = 'PE' and PETESTCD ne 'PEALL' then delete;
run;

/*** Set the tiles and report path ***/
%let FDAC183rpttitle = %str(this is title for FDAC183);
%let FDAC212rpttitle = %str(this is title for FDAC018);
%let FDAC178rpttitle = %str(this is title for FDAC178);
%let FDAC176rpttitle = %str(this is title for FDAC176);

%let ReportPath = C:\Users\agarrett\Desktop;

/*** Make report(s) ; **/
%report(FDAC183 FDAC212 FDAC178 FDAC176);
```

STEP THREE CONTINUED: BASE MACRO

The first macro, *base,* can be found in entirety in the appendix. Base macro has one parameter, ID_string, which is a space delimited list representative of the Publisher IDs to be processed. The macro loops through the string and subsets the details data set based on the Publisher ID and creates a new data set with the same name as the Publisher ID being processed.

```
%let j=1;         /* Start loop through the string of Publisher ID's */
%let PubID = %scan(&ID_string, &j, %str( ));
%put Processing &PubID ;
%do %while (&PubID ne);

data &PubID;      /* Subset Details based on Publisher ID */
    set details;
    where Publisher_ID =  "&PubID.";
run;
```

Next, the macro queries the subset data set and selects distinct domains into a macro variable called **WDOMAIN**.

```
proc sql noprint;
    select distinct quote(strip(domain)) into :wdomains separated by ','
    from &PubID;
    quit;
```

The macro then creates a list of all affected domains by querying dictionary.tables (this ensures that the domains listed as being affected on the details tab actually exist in the SDTM library) and selecting domains that match the existing list of affected domains generated in the above statements. This list is re-named and assigned to be a global macro variable with the naming convention: **DSS&PUBID**. After this, the macro loops through the affected domains and creates a new data set in work with a new variable, *record,* which represents the observation number for each row in the domain.

```
proc sql noprint;
    select memname into: dss separated by ' '
    from dictionary.tables
    where libname="&lib." and memname in(&wdomains);
    quit;

%global &dss&PubID ;
%let dss&PubID = &dss ;
%put DSS PUBID =  &&dss&PubID;

%let i=1;
%let ds = %scan(&&dss&PubID, &i, %str( ));
%put &ds;
    %do %while (&ds ne);
           data &ds;
           set &lib..&ds;
                  record = _N_;
           run;
    %let i = %eval(&i+1);
    %let ds = %trim(%scan(&&dss&PubID, &i, %str( )));
    %end;
```

Lastly, the macro loads the original subset data set, PUBID, into a hash table and then creates a new output data set containing only records with a match in the hash table. Matches are defined as records having identical domain name and identical record number. Additionally, the helpful messages in the details data set are appended on to each record. A sample output data set is presented in Figure 6.

```
data outds;
if 0 then set &PubID ;

dcl hash hh (dataset: "work.&PubID", hashexp: 10) ;

    hh.definekey ('record','domain');
    hh.DefineData ('Message','Publisher_ID','Variables','Values');
    hh.DefineDone ();

    do until (eof) ;
    set &&dss&PubID end = eof;
    if hh.find () = 0 then output outds;
    end;
    stop;
run;
```

Figure 6 shows a sample output data set after running base macro on FDAC178.

Domain	Variables	Values	Publisher_ID	Message	record	STUDYID	USUBJID	PESEQ	PESPID	PETESTCD
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75913	ABC123	ABC123-0000-0000	7	1	CONS
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75918	ABC123	ABC123-0000-0000	12	6	CRANIAL
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75923	ABC123	ABC123-0000-0000	17	5	FUNDUS
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75928	ABC123	ABC123-0000-0000	22	9	GAIT
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75936	ABC123	ABC123-0000-0000	30	2	MENTAL
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75941	ABC123	ABC123-0000-0000	35	7	MOTOR
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75952	ABC123	ABC123-0000-0000	46	11	OVERALL
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75969	ABC123	ABC123-0000-0000	63	10	REFLEX
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75977	ABC123	ABC123-0000-0000	71	8	SENSORY
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75982	ABC123	ABC123-0000-0000	76	3	SPEECH
PE	PESTAT, PEORRES	null, null	FDAC178	Missing value for PEORRES, when PESTAT or PEDRVFL is not populated	75987	ABC123	ABC123-0000-0000	81	4	VISION
TR	TRSTAT, TRORRES	null, null	FDAC178	Missing value for TRORRES, when TRSTAT or TRDRVFL is not populated	5332	ABC123	ABC123-1111-0000	.		
TR	TRSTAT, TRORRES	null, null	FDAC178	Missing value for TRORRES, when TRSTAT or TRDRVFL is not populated	5336	ABC123	ABC123-1111-0000	.		
VS	VSORRES, VSSTAT	null, null	FDAC178	Missing value for VSORRES, when VSSTAT or VSDRVFL is not populated	4773	ABC123	ABC123-2222-0000	.		
VS	VSORRES, VSSTAT	null, null	FDAC178	Missing value for VSORRES, when VSSTAT or VSDRVFL is not populated	34852	ABC123	ABC123-5555-0000	.		

Figure 6. PESEQ is missing for TR and VS records, indicating that if one were to scroll over, TRSEQ and VSSEQ would be populated for those records, respectively

STEP THREE CONTINUED: REPORT MACRO

After the output Publisher ID data sets have been created and any custom programming has been implemented, it is time to call the report macro (full code in the appendix). The user can specify titles prior to running report macro and should specify the report path destination. In the first line of the macro, the macro variable, COMMON_COLS, is assigned and should be modified by the user based on the information desired to be reported; the variables in COMMON_COLS will be present on each tab of the report. Next, the report macro starts looping through the parameter, ID_string similar to base macro. Within each Publisher ID loop, the macros loops through the string of affected domains (recall the global macro DSS&PUBID), identifies variables that apply to the domain, subsets the Publisher ID data sets by domain, and reports variables listed in the COMMON_COLS as well domain-specific variables. Domain variables are identified during each iteration and stored in the macro variable, DOMAIN_COLS. A sample output report can be found in figure 4.

With three easy steps, a multi-tabbed report is now ready to be given to the D.M. team!

CONCLUSION

The Details tab of a P21 report provides only a glimpse into each issue; this limited insight is not always sufficient for the programmer to truly discern the nature of the problem. To comprehend an issue, complete information is needed for the record to be properly identified in source data. This paper provides a solution to assist the programmer in understanding the issue fully, creates a listing from which other teammates can benefit. The listing can also serve as documentation of communication surrounding the issue between team members, resulting in fewer e-mails and more productivity.

We understand D.M. may not always be familiar with SDTM terminology/ mapping and find the SDTM annotated CRF (aCRF) to be a great resource for them to use during review. Referring to the aCRF helps identify form(s) from which an erred record is sourced and acts as an informal QC because if D.M. is unable to determine where the data originated, the FDA is unlikely to meet this challenge as well. Furthermore, by providing D.M. the opportunity to review SDTM data with this report, they will develop an awareness of how certain data patterns disrupt the flow on the back-end.

Beyond aiding D.M. in data review, we believe this reporting process could benefit the generation of a Define.XML file by identifying values that may need to be added to an extensible code list (FDAC314). It may also be useful in documenting certain findings in the Study Data Reviewer's Guide. We invite others to develop their own application and hope you will share your knowledge with us.

REFERENCES

1. Becker, Matt 2012. SDTM, ADaM and define.xml with OpenCDISC. *Proceedings of the PharmSUG 2012 Conference*

2. Peterson, Terek; Adams, Gareth 2014. OpenCDISC Validator Implementation*:* A Complex Multiple Stakeholder Process. *Proceedings of the PharmSUG 2014 Conference*

3. HTTP://WWW.OPENCDISC.ORG

CONTACT INFORMATION

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

Name: Amy Garrett
Enterprise: Novella Clinical
Address: 580 N. 4th St.
City, State ZIP: Columbus, OH 43215
Work Phone: 614-572-0243
E-mail: agarrett@novellaclinical.com

Name: Christopher Whalen
Enterprise: Clovis Oncology
Address: 5500 Flatiron Parkway, Suite 100
City, State ZIP: Boulder, CO 80301
Work Phone: 303-625-5060
E-mail: cwhalen@clovisoncology.com

Appendix

```
%macro base(ID_string);
%let j=1;                        /* Start to loop through the string of Publisher ID's */
%let PubID = %scan(&ID_string, &j, %str( ));
%put Processing &PubID;

%do %while (&PubID ne);

        data &PubID;                    /* Subset Details based on Publister ID */
                set details;
                where Publisher_ID =  "&PubID.";
        run;

/*** BEGIN PROGRAMMING FOR DUPLICATES, FDAC212 ***/
        %if %upcase(&PubID) = FDAC212 %then %do;
                proc sort nodupkey data = &PubID;
                        by Domain Message Publisher_ID Variables Values;
                run;

                data &PubID;                    /* Add keyset for duplicates */
                        set &PubID;
                        by domain Values;
                        if first.domain then
                        key_num = 1;
                        else key_num + 1;
                        Message=cats("Duplicate Records  (keyset ",strip(put(key_num,z5.)),")");
                        drop key_num;
                run;

                proc sql noprint;       /* Get Duplicate keys from P21 report for FDAC212 only */
                        select distinct compress('"'||strip(tranwrd(variables, ',','","'))||'"')
                        into :DupeKeys separated by '#'
                        from &PubID;
                quit;

        %end;

/*** END PROGRAMMING FOR DUPLICATES, FDAC212 ***/
        %let wdomains=;

        proc sql noprint;                       /* Get domains for affected by Publisher ID */
                select distinct quote(strip(domain)) into :wdomains separated by ','
                from &PubID;
        quit;

        %if (&wdomains ne ) %then %do;

                proc sql noprint;                       /* Verify domains exist in the library */
                        select memname into: dss separated by ' '
                        from dictionary.tables
                        where libname="&lib." and memname in(&wdomains);
                quit;

                %global dss&PubID;
                %let dss&PubID = &dss;
                %put DSS PUBID =  &&dss&PubID;
                %let i=1;
                %let ds = %scan(&&dss&PubID, &i, %str( ));
                %put &ds;

                %do %while (&ds ne);

                        data &ds;       /* Move domains to work, add variable for record number */
                                set &lib..&ds;
                                record = _N_;
                        run;

                        %let i = %eval(&i+1);
                        %let ds = %trim(%scan(&&dss&PubID, &i, %str( )));
                %end;
```

```
                data outds;            /* Find record in domain by matching on record number */
                        if 0 then set &PubID ;
                        dcl hash hh (dataset: "work.&PubID", hashexp: 10);

                        hh.definekey ('record','domain');
                        hh.DefineData ('Message','Publisher_ID','Variables','Values');
                        hh.DefineDone ();

                        do until (eof);
                        set &&dss&PubID end = eof;
                        if hh.find () = 0 then output outds;
                        end;
                        stop;
                run;

                data &PubID ;                   /* Re-name data set to FDA Publsiher ID */
                        set outds;
                run;

/*** BEGIN HASH PROGRAMMING FOR DUPLICATES, FDAC212 ***/
                %if %upcase(&PubID) = FDAC212 %then %do;

/* Prep for loop by finding the domains/keys affected by duplicates */
                        %let i=1;
                        %let ds = %scan(&&dss&PubID, &i, %str( ));
                        %put &ds;
                        %let DSKeys = %scan(%bquote(&DupeKeys), &i, '#');
                        %put &DSKeys;

                        %do %while (&DSKeys ne and &ds ne);

                                data outds&i;  /* Find duplicate of affected records by matching on
                                               key variables from in P21 report */
                                        if 0 then set &PubID ;
                                        dcl hash hh (dataset: "work.&PubID", hashexp: 10);
                                        hh.definekey (&DSKeys,'domain');
                                        hh.DefineData
                                        ('Message','Publisher_ID','Variables','Values');
                                        hh.DefineDone ();

                                        do until (eof);
                                        set &ds end = eof;
                                        if hh.find () = 0 then output outds&i;
                                        end;
                                        stop;
                                run;

                                %let i = %eval(&i+1);
                                %let ds = %trim(%scan(&&dss&PubID, &i, %str( )));
                                %let DSKeys = %scan(%bquote(&DupeKeys), &i, '#');
                                %let k = %eval(&i-1);
                                %put k = &k;
                        %end;

                        data &PubID ;          /* Re-name data set to FDA Publsiher ID */
                                set outds1-outds&k;
                        run;

                %end;

/*** END HASH PROGRAMMING FOR DUPLICATES, FDAC212 ***/
                %let j = %eval(&j+1);
                %let PubID = %scan(&ID_string, &j, %str( ));
        %end;
%end;
%mend;
```

```
%macro report(ID_string);
	%let common_cols=Domain Variables Values Publisher_ID Message Record;

	* Assign common variables;
	%let today=%sysfunc(date(),yymmdd10.);
	%put &today;
	ODS _ALL_ CLOSE;
	title1 "Project - DM/SDTM Reports";
	ods tagsets.excelXP
		file="&ReportPath\SDTM_Checks_&today..xls"
		options (
		AUTOFILTER ="all"
		);
	%let j=1;    /* Start to loop through the string of Publisher ID's */
	%let PubID = %scan(&ID_string, &j, %str( ));
	%put Processing &PubID;

	%do %while (&PubID ne);
		%let i=1;    /* Start to loop through the domains affected by Publisher ID */
		%let ds = %scan(&&dss&PubID, &i);
		%put &ds;

		%do %while (&ds ne);

			proc sql noprint;
				/* Any SDTM variable in common_cols should be listed here */
				select distinct varnum, name into :dummy,
							:domain_cols separated by ' '
					from dictionary.columns
						where name not in ('DOMAIN' )
							and libname="&lib" and memname = "&ds"
				order by varnum;
			quit;

			* Set the title;
			%if (%SYMEXIST(&PubID.rpttitle) eq 1) %then
				%do;
					title4 height=2 "&&&PubID.rpttitle.";
					%let row_repeat=6;
					%let frozen_headers=6;
				%end;
			%else
				%do;
					title4;
					%let row_repeat=3;
					%let frozen_headers=3;
				%end;

			* Open the ODS sandwich and set the options;
			ods tagsets.excelxp
				options (SHEET_NAME="&PubID._&ds."
				SHEET_INTERVAL    ="NONE"
				EMBEDDED_TITLES   ="yes"
				ORIENTATION       ="landscape"
				FITTOPAGE         ="yes"
				PAGES_FITHEIGHT   ="100"
				ROW_REPEAT        ="1-&row_repeat"
				FROZEN_HEADERS =  "&frozen_headers"
				AUTOFIT_HEIGHT    ="yes"
				CENTER_HORIZONTAL ="yes"
				ABSOLUTE_COLUMN_WIDTH = "10"
				);

			data print;
				set &PubID;
				where domain = "&ds";
			run;

			options nolabel;

			proc report data=print nowindows missing
```

```
                                style(column)=[foreground=black background=WHITE font_face=arial
                                               font_size=10pt just=center]
                                style(header)=[foreground=black background=LIBGR font face=arial
                                               font_size=10pt just=center fontweight=bold];
                                column &&common_cols  &domain_cols;
                        run;

                        %let i = %eval(&i+1);
                        %let ds = %trim(%scan(&&dss&PubID, &i));
                %end;

                %let j = %eval(&j+1);
                %let PubID = %scan(&ID_string, &j, %str( ));
        %end;

        ods tagsets.excelxp close;
        ods listing;
%mend;
```