PharmaSUG 2021 - Paper AP-053

A SAS Macro to Generate Slim Version of Define for RTOR

Jeff Xia, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

FDA Real-Time Oncology Review (RTOR) pilot program provides a channel for agency reviewers to access study data earlier on to explore more efficient review of oncological submissions, which could lead to safe and effective treatments for patients as early as possible. One of the required components in RTOR is corresponding ADaM datasets for key efficacy and safety tables/figures for pivotal study, which might be a subset of the entire ADaM datasets (i.e., excluding datasets and variables that are not involved in key efficacy and safety analysis). A corresponding smaller (slim) version of define documents (define.xml and define.pdf) is also required.

This paper introduces a SAS macro that can be used to generate a smaller (slim) version of define document using Pinnacle 21(P21) Enterprise Version.

1. This macro reads the full version of define specification in Excel spreadsheet format into SAS datasets.
2. It programmatically excludes datasets/variables that are not part of the RTOR package, as well as corresponding components associated with these removed datasets/variables, such as value level meta data, where clauses, code lists, comments and methods.
3. Finally, it generates a new version of define spec that can be imported back to P21 to create a slim version of define documents for RTOR deliverables. This macro improves efficiency and accuracy in preparing RTOR package by removing a few trivial manual steps when generating the slim version of define document.

This macro was utilized to generate ADaM define documents in RTOR package for a Keytruda study, a Phase III first-line immunotherapy for patients with MSI-H/dMMR metastatic colorectal cancer, which has been approved by the FDA 5 months ahead of FDA PDUFA date.

INTRODUCTION

Real-Time Oncology Review (RTOR) is a pilot program in Oncology Center of Excellence in FDA, which aims to explore a more efficient review process so that safe and effective treatments can be available to patients as early as possible. Acceptance of RTOR pilot does not guarantee approvability of the application. However, through data and analysis standardization, and early iterative engagement with the applicant, agency reviewers can balance workload in different milestones, improve their efficiency and quality in review process, and make regulatory decision sooner in benefit-risk evaluation.

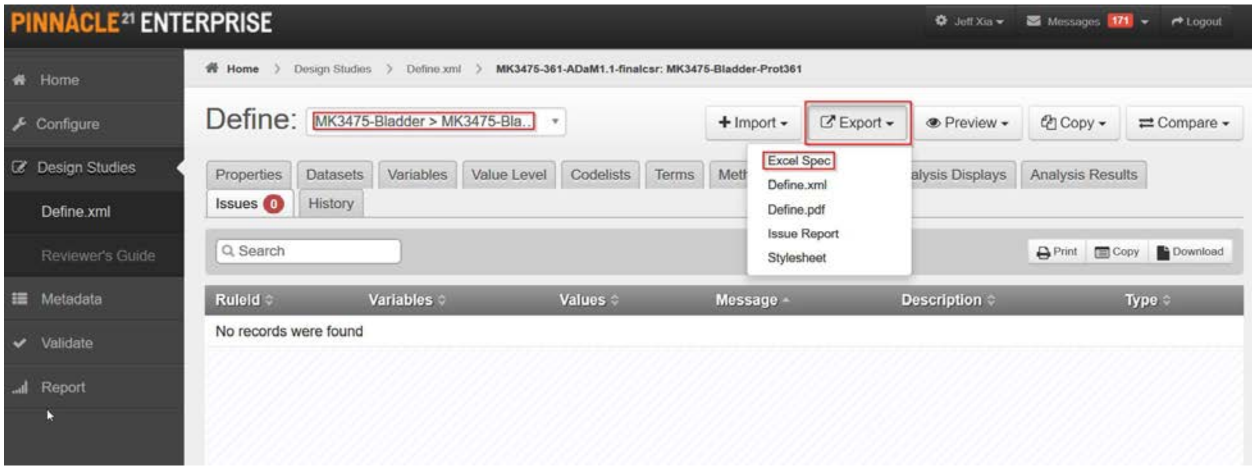
RTOR requires many different components, such as complete SDTM dataset packages; top line efficacy and safety tables/figures; complete ADaM datasets for key efficacy and safety tables/figures for pivotal study, as well as applicable SAS programs, etc. The sponsor will have to apply for the RTOR pilot by notifying the appropriate division regulatory project manager in the FDA. If the application is accepted, the timeline for sponsor to submit the RTOR package is about 6-9 weeks after database lock. Normally, it is quite difficult for the study team to finalize the complete ADaM datasets for the entire CSR. Therefore, it is quite common for sponsors to only include the datasets that are related to the top line results rather than the complete ADaM datasets for the whole CSR.

P21 is a great tool to generate define documents for regulatory submissions. The programming data specification can be read into P21 and converted to define document in valid XML syntax. Also, the define document in P21 can be saved (exported) in the format of PDF or Excel Spreadsheet. Since RTOR only involves a subset of datasets for the complete CSR, it is very time-consuming and attention-demanding to trim the full version of define to a slim version of define which only contains applicable datasets related to

RTOR. This paper introduces a SAS macro to eliminate these trivial manual steps and automate the entire process.

STEP 1: EXPORT THE DEFINE FROM P21 AND READ INTO SAS DATASETS

The first step is to export the define document of the complete ADaM datasets from P21 to an Excel Spreadsheet, and then convert the Excel spreadsheet into SAS datasets for further processing later using SAS. Since some tab names or column headers in the Excel spreadsheet may contain space or other special characters, the SAS option “validvarname” is set to “any” to avoid possible reading errors in SAS. Also, the engine of xlsx is used along with the statement of “libname” to enable SAS to access the file of Excel Spreadsheet directly.



No records were found

Display 1. Screen Print of Exporting Define in the Format of Excel Spreadsheet

```
/* this option is used because Excel field names often have spaces */
options validvarname=any;

libname p21spec XLSX "&Original_P21_spec";

/* discover the list of tab names in the Excel Spreadsheet */
proc contents data = p21spec._all_
              out = myds
              noprint;
run;

proc sort data = myds /* remove duplicate tab names */
          out = myds1(keep = memname)
          nodupkey;
    by memname;
run;
```

Then execute the following data NULL step to read all tabs in the Excel spreadsheet into SAS datasets. The program will dynamically determine how many tabs the Excel Spreadsheet has, and read them into SAS dataset one by one in an alphabetical order.

```
data _null_;
    set myds1;
    call execute("data "
                || compress(memname)
                || "; set p21spec.'"
```

```
               || strip(memname)
               || "'n; run;"); /* ‘n is very important for tabs with space */
run;
```

STEP 2: REMOVE NON-APPLICABLE CONTENTS IN EACH DATASET

A sub macro is designed to remove all non-applicable contents from each dataset:

```
%macro process_data (dsn =);
    data &dsn._1(drop = tmp);
       set &dsn;
       length tmp $200;
       tmp = strip("&keep_dsn");
       if findw(tmp, strip(dataset))  ;
    run;
%mend;

%process_data(dsn = Datasets);
%process_data(dsn = Variables);
%process_data(dsn = ValueLevel);
%process_data(dsn = WhereClauses);
```

It is straightforward to remove the non-applicable contents in the tab of Datasets, Variables, ValueLevel and WhereClauses. On the other hand, it is a little tricky to remove non-applicable contents in the CodeList, Method and Comments tabs. For example, if the dataset ADLB will not be included in the RTOR package, the codelist associated with any variable in ADLB will have to be removed except it is also used in any other datasets, or it appears in the ValueLevel tab that are included in the RTOR package. Specific logic has been developed to handle this exception, see SAS code below.

```
/*process codelist, method and comments for each dataset*/
proc sql noprint;
    create table codelists_1 as
        select * from codeLists
        where ID in
           (select CodeList
            from variables_1
            where compress(CodeList) >""
            union
            select CodeList
            from ValueLevel_1
            where compress(CodeList) >"");

    /*Same logic for handing the CodeList in Method,
      Comment and Dictionaries tab,
      Detailed SAS codes are skipped for the length of the paper */

quit;
```

Another challenge is to ensure the variable information in the define document matches the one in the actual datasets that will be submitted. The below SAS code has been developed to flag the difference if any, either a new variable is added in the dataset but not in define, or a variable appears in the define but not in the dataset. The SAS program also produce warning message to draw user’s attention for these discrepancies.

```
%if %length(&ADaM_path) ne 0 %then %do;
```

```
    libname adam "&ADaM_path" access = readonly;
    proc contents data = adam._all_
                  out = adamds(keep = memname name label type length varnum)
                  noprint;
    run;

    libname adam clear;

    data adamds1(drop = tmp);
        set adamds( rename  = (memname = dataset
                                name    = variable
                                type    = var_type
                                label   = var_label
                                length  = var_length));
        length tmp $200;
        tmp = strip("&keep_dsn");
        if findw(tmp, strip(dataset))  ;
    run;

    proc sort data = Variables_1;
        by dataset variable;
    run;

    proc sort data = adamds1;
        by dataset variable;
    run;

    data Variables_1(drop = var_label var_type var_length varnum);
        merge Variables_1(in = a)
                 adamds1 (in = b);
        by dataset variable;
        length Status $20;
        order = varnum;
        if a and not b then do;
              put 'User Defined Warning: Variable removed from spec:'
                   dataset = variable =;
              delete;
        end;
        else if b and not a then do;
              label = var_label;
              'Data Type'n = var_type;
              length = strip(put(var_length, best.));
              Status = 'NEW';
        end;
        else do;
              Status = 'NO CHANGE';
        end;
    run;

    proc sort data = Variables_1;
        by dataset order;
    run;
%end;
```

Sometimes the Excel spreadsheet might contain some non-printable characters that are very difficult for users to notice or detect by eye browsing. These non-print characters in the ID column might prevent P21 from correctly linking different XML components. Therefore, these characters must be removed before importing back to P21. The following macro can help users to achieve this purpose.

```
%macro remove0blanks(ds = , var =);
    %put ****************************************************************;
    %put Processing dataset &ds by removing non-printable characters from;
    %put variables &var;
    %put ***************************************************************;
    %let ds = &ds;
    %let var = &var;
    %local i cnt tmp;
    %let cnt = %sysfunc(countw(&var));
      data &ds;
          set &ds;
          %do i = 1 %to &cnt;
                %let tmp = %scan(&var, &i);
                &tmp = compress(&tmp);
                &tmp = compress(&tmp, byte(10));
                &tmp = compress(&tmp, byte(13));
            %end;
      run;
%mend;

%remove0blanks(ds = variables_1 , var = codelist method comment predecessor);
%remove0blanks(ds = valueLevel_1, var = codelist method comment predecessor);
%remove0blanks(ds = comments_1 ,  var = ID);
%remove0blanks(ds = methods_1 ,   var = ID);
%remove0blanks(ds = codelists_1 , var = ID);
%remove0blanks(ds = whereClauses_1, var = ID);
```

STEP 3: EXPORT SAS DATASETS TO EXCEL SPREADSHEET FOR P21

The last step is to output all relevant datasets to Excel Spreadsheet in a format that can be imported back to P21 and generate the define document in valid xml syntax.

```
ods html close;
ods listing close;
ods excel file = "&Out_Path.\&Out_P21_Spec_Name";

%macro output_ds(sheet =);
    %local dsn numvar i;
    %let dsn = %sysfunc(compress(&sheet));
    ods excel options (sheet_name = "&sheet"
                       zoom = '100'
                       orientation='landscape'
                       row_repeat = 'header'
                       pages_fitheight = '100'
                       center_horizontal = 'yes'
                       center_vertical = 'no'
                       gridlines='on'
                       frozen_headers = 'yes');
    proc report data = &dsn._1 nofs
        style(header)= {font_weight=bold
                       font_size=10pt
                       just=center
                       protectspecialchars=off
                       borderstyle=solid
```

```
                   bordercolor=black}
                   style(column)={borderstyle=solid
                   bordercolor=black};
%if %upcase(&sheet) = STUDY %then %do;
   define "Attribute"n/style(column)=[whitespace=pre]
           style(column)={width=1.5in};
   define "Value"n/style(column)=[whitespace=pre]
           style(column)={width=10in};
%end;
%else %if %upcase(&sheet) = DATASETS %then %do;
   define "Description"n/style(column)=[whitespace=pre]
           style(column)={width=2.5in};
   define "Class"n/style(column)=[whitespace=pre]
           style(column)={width=2.5in};
   define "Structure"n/style(column)=[whitespace=pre]
           style(column)={width=4.5in};
   define "Key Variables"n/style(column)=[whitespace=pre]
           style(column)={width=4.5in};
%end;
%else %if %upcase(&sheet) = VARIABLES and %length(ADaM_Path)>0 %then %do;
   define "Label"n/style(column)=[whitespace=pre]
           style(column)={width=3.5in};
   define "Data Type"n/style(column)=[whitespace=pre]
           style(column)={width=2.5in};
   define "Significant Digits"n/style(column)=[whitespace=pre]
           style(column)={width=2.5in};
   *define status/noprint;
   compute Status;
     if upcase(strip(Status)) in ('NEW') then do;
       call define(_row_,'style','style={background=yellow}');/*Yellow*/
     end;
   endcomp;
%end;
%else %if %upcase(&sheet)=VARIABLES and %length(ADaM_Path) =0 %then %do;
   define "Data Type"n/style(column)=[whitespace=pre]
          style(column)={width=2.5in};
   define "Significant Digits"n/style(column)=[whitespace=pre]
          style(column)={width=2.5in};
   define "Label"n/style(column)=[whitespace=pre]
          style(column)={width=3.5in};
%end;
%else %if %upcase(&sheet) = VALUELEVEL %then %do;
   define "Where Clause"n/style(column)=[whitespace=pre]
          style(column)={width=5in};
%end;
%else %if %upcase(&sheet) = WHERECLAUSES %then %do;
   define "ID"n/style(column)=[whitespace=pre] style(column)={width=5in};
%end;
%else %if %upcase(&sheet) = CODELISTS %then %do;
   define "Term"n/style(column)=[whitespace=pre]
          style(column)={width=5in};
   define "Decoded Value"n/style(column)=[whitespace=pre]
          style(column)={width=5in};
%end;
%else %if %upcase(&sheet) = METHODS %then %do;
   define "Description"n/style(column)=[whitespace=pre]
          style(column)={width=10in};
```

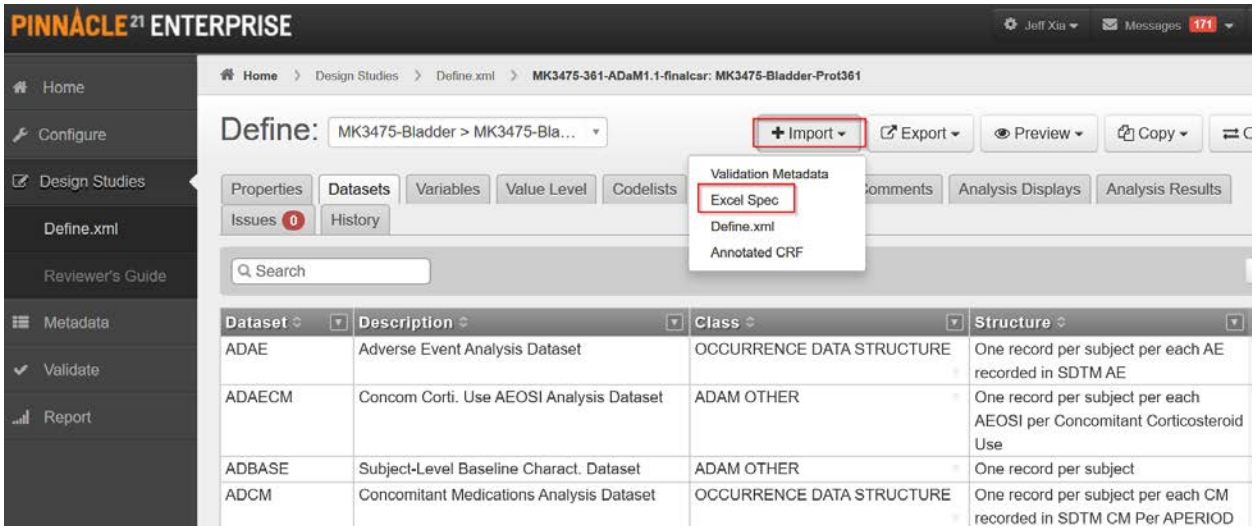
```
    %end;
    %else %if %upcase(&sheet) = COMMENTS %then %do;
       define "Description"n/style(column)=[whitespace=pre]
              style(column)={width=10in};
    %end;
    run;
%mend;
```

The following macro call outputs the contents in the dataset “Study” to the Excel spreadsheet, the contents in the rest datasets such as “Datasets”, “Variables”, “ValueLevel”, “WhereClauses”, “CodeLists”, “Dictionaries”, “Methods”, “Comments”, “Analysis Display”, “Analysis Results” and “Analysis Criteria” can be written to the same Excel spreadsheet by following a similar macro call. Please note, specific logic has been embedded in the macro “output_ds” so that if a variable exists in the dataset but not in the spec, the entire row in the Excel spreadsheet will be highlighted in yellow to draw user’s attention.

```
%output_ds(sheet = Study);
```

STEP 4 IMPORT THE NEW EXCEL SPREADSHEET TO P21

The last step is to import the newly generated spreadsheet back to p21 and generate corresponding define.xml, which is a slim version of the complete define for ADaM datasets.



Dataset	Description	Class	Structure
ADAE	Adverse Event Analysis Dataset	OCCURRENCE DATA STRUCTURE	One record per subject per each AE recorded in SDTM AE
ADAECM	Concom Corti. Use AEOSI Analysis Dataset	ADAM OTHER	One record per subject per each AEOSI per Concomitant Corticosteroid Use
ADBASE	Subject-Level Baseline Charact. Dataset	ADAM OTHER	One record per subject
ADCM	Concomitant Medications Analysis Dataset	OCCURRENCE DATA STRUCTURE	One record per subject per each CM recorded in SDTM CM Per APERIOD

Display 2. Screen Print of Importing Define in the Format of Excel Spreadsheet

CONCLUSION

This paper presents a method that can be used to programmatically remove non-applicable contents from a full version of define in the format of Excel spreadsheet that was exported from P21, and produce a clean version of spec in the format of Excel spreadsheet that is only relevant to datasets included in RTOR. This spreadsheet can be easily imported back to P21 to generate a slim version of define document for RTOR submission. When preparing the slim version of RTOR define, the proposed method significantly improves the efficiency and accuracy, which has been demonstrated in a recent RTOR submission to the FDA.

ACKNOWLEDGMENTS

The author would like to thank Mary Varughese for her great support and valuable input into this paper.

REFERENCE

FDA Real-Time Oncology Review Pilot Program: https://www.fda.gov/about-fda/oncology-center-excellence/real-time-oncology-review-pilot-program

FDA Approves First-Line Immunotherapy for Patients with MSI-H/dMMR Metastatic Colorectal Cancer: https://www.fda.gov/news-events/press-announcements/fda-approves-first-line-immunotherapy-patients-msi-hdmmr-metastatic-colorectal-cancer

CONTACT INFORMATION

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

Name: Jeff Xia
Enterprise: Merck & Co., Inc.
Address: 126 E. Lincoln Avenue
City, State ZIP: Rahway, NJ 07065-4607
Work Phone: 732-594-6439
E-mail: jeff.xia@merck.com
Web: www.merck.com