PharmaSUG 2018 - Paper AD11

A Simple Method for Integrating Analysis Result Metadata into Existing Define.xml 2.0

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

Shunbing Zhao, Merck & Co., Inc., Rahway, NJ, USA

Anjana Grandhi, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

Analysis Result Metadata (ARM) adds significant value to a regulatory submission. It provides agency reviewers with a means to trace results back to their source documents, such as key pieces of SAS programs used for analysis, datasets, data selection criteria, and even specific pages in a statistical analysis plan. However, the define.xml for many completed studies does not have ARM included due to earlier technical constraints. It is of great interest to add ARM features to these existing define.xml if there is a business need to include them in future submissions. The Pinnacle 21 (P21) Community version becomes increasingly popular for generating define.xml 2.0 for submissions by the pharmaceutical industry. The latest version of P21 (version 2.20) still lacks the capability to generate define.xml with the feature of ARM.

This paper presents a simple method for integrating ARM in the existing define.xml version 2.0. An Excel spreadsheet is designed to collect all required information for ARM, then a macro is utilized to convert the collected information to ASCII text in valid xml syntax and add them to proper locations in existing define.xml. The final define.xml has all ARM features, passes P21 define.xml schema validation, and is ready to be included in a submission.

INTRODUCTION

Traceability is one of the basic principles in CDISC standards as it provides reviewers a route to understand the data flow from data collection to SDTM, ADaM, and analysis results. Analysis Result Metadata (ARM) is part of the traceability that the reviewers are looking for as it provides a necessary linkage from the statistical analysis display back to their corresponding analysis datasets.

The define.xml for many completed studies did not have the ARM included in the past. One primary reason is that the ARM was not a part of requirements at that time. Technical constraints could be another reason for this deficiency as not so many tools in the past were available for generating define.xml with the feature of ARM. Even now, the Pinnacle 21 Community version has not provided this capability yet. It adds great value if we can integrate ARM in the existing define.xml for these studies. Moreover, it could be very beneficial for small pharma/CRO if they have existing utilities and tools to generate define.xml version 2.0 but are not able to include ARM in their final product.

DISCUSSION

The document from CDISC “Analysis Results Metadata v1.0 for Define-XML v2.0” specifies a list of required or optional elements of ARM for define.xml version 2.0. In order to collect information for populating these elements, we designed an Excel spreadsheet that study statisticians or study programmers can easily fill out. See below for the description of each collected item.

Excel Spreadsheet Column	Description
Table No	Unique table number to be included in ARM, it can contain special characters such as “.”, “_”
Table Name in PDF	The unique name of CSR table in PDF format

Excel Spreadsheet Column	Description
Table Page No	If desired, page number can be supplied so that a link will be created to that page in define.xml
Table Description	A brief description of that analysis table
Dataset Name	The ADaM dataset that the analysis is based on
Analysis Parameter	The parameter that the analysis is based on. Usually it is “PARAMCD”
Analysis Variable	The variable name that contains the value for the analysis. Usually it is “AVAL” or “AVALC”
Analysis Purpose	The purpose of the analysis. i.e., Efficacy outcome measure.
Selection Criteria	The ID of identifier for selection criteria. It has to match the one in the Selection Criteria section in define.xml, otherwise the link will not work. Error will be captured in Pinnacle 21 when performing schema validation if there is such a mismatch.
Documentation	Further explanation of the analysis.
Documentation Name	The PDF name of the documentation, i.e., sap.pdf
Documentation Page Number	If desired, page number can be supplied so that a link in define.xml will be created to that page of documentation in PDF format
Comments	Any additional comments if applicable
Programming Statement	Key SAS code snippet that will be included in ARM
Name of SAS Source Code in text format	File name of the text file that contains the Key SAS code.
SAS Version	SAS version number, such as 9.3, or 9.4

Table 1. Design of Excel spreadsheet for information collection of ARM

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Table No	Table Name in PDF	Table Page No	Table Description	Description of Analysis	Dataset Name	Analysis Parameter	Analysis Variable	Analusys Purpose	Selection Criteria	Documentation	Doc Name in PDF	Doc Reference	Doc Page Number	Programming Statement	SAS Source Code in Text	SAS Version
14.2.1.2a	T_14_2_1_2a.pdf	1	Summary of Investigator's Assessment of Clinical Outcome at EOIV, EOT and TOC (mMITT Population)	Clinical outcome based on Investigator's assessment of clinical response (cure, improved, failure or non-evaluable) at End of Intravenous Therapy Visit, End of Intravenous+Oral Therapy Visit and Test-of-Cure / Safety Visit in the microbiological Modified Intent-to-Treat (mMITT) population	ADEF	PARAMCD	AVALC	EFFICACY OUTCOME MEASURE	WC.ADEF.PARAMCD.EQ.OVAL-63176f67	Clinical success rates were evaluated by calculating a 95% confidence interval around the difference in rates among the 2 treatment groups (daptomycin group minus standard of care comparator) using the Newcombe hybrid score (Wilson) confidence limits for the risk difference without continuity correction.	sap.pdf	SAP Section 10.1.1	11	proc freq data=ADEF; tables TRTPN*AVALC / riskdiff (method = NEWCOMBE); run; proc freq data=ADEF; tables TRTPN*Y_var; weight Weight_var; exact riskdiff; run;		9.3
14.2.2.2a	T_14_2_2_2a.pdf	1	Summary of Pathogen-Level Microbiological Outcome at TOC by Blinded Central Review (mMITT Population)	Pathogen-Level Microbiological Response at Test-of-Cure / Safety Visit in the microbiological Modified Intent-to-Treat (mMITT) population for each baseline infecting pathogen	ADZB	PARAMCD	AVALC	EFFICACY OUTCOME MEASURE	WC.ADZB.PARAMCD.EQ.ZBALL-21dfced9	Pathogen-level microbiological success rates are calculated for each pathogen subgroup (MRSA, MSSA, other) as the proportion of subjects in a population with that pathogen and susceptibility who have an outcome of success. 95% confident intervals of percentage of subjects reaching “Success” are constructed, based on binomial distribution, for the differences between the daptomycin and comparator arm following Wilson score method.	sap.pdf	SAP Section 10.1.2	11	proc freq data=ADZB; tables TRTPN*AVALC / riskdiff (method = NEWCOMBE); run; proc freq data=ADZB; tables TRTPN*Y_var; weight Weight_var; exact riskdiff; run;		9.3

Figure 1. Screen print of Excel Spreadsheet used for ARM information collection.

CONVERT THE COLLECTED INFO TO XML SYNTAX

We developed a list of SAS programs to read in the Excel spreadsheet, then convert the collected information to valid XML syntax, please see below for some key parts of the SAS programs:

```
/*read in the collected information in Excel spreadsheet format*/
proc import datafile = 'C:\doc\MK9999_KN999-ARM_For_Define_V1.xlsx'
    dbms = excel
    out = arm replace;
    sheet = 'MK9999-KN999';
    getnames = yes;
run;

/*Basic house-keeping for further information manipulation*/
data arm;
    set arm;
    length table_no1 $30;
    if compress(table_no) = '' then delete;
    if index(upcase(table_no), 'TABLE') = 0 then
        table_no1 = 'Table '||strip(table_no);
    if substr(strip(upcase(Selection_Criteria)), 1, 3) ne 'WC.' then do;
        Selection_Criteria = 'WC.'|| strip(Selection_Criteria);
    end;
    table_no2 = tranwrd(strip(table_no), ' ', '_');
    table_no2 = tranwrd(strip(table_no), '.', '_');
    sequence = _n_;/*row number of the record, useful in further processing*/
    Programming_Statement = compbl(Programming_Statement);
run;

/*Prepare the dataset for output in text format*/
data armtext(keep =  xmlcode) ;
    set arm end = last;
    by table_no;
    length xmlcode $2000;
    suborder = _n_;
    /*************************************************************************
        Step 2: Generate Leafs Definition for ARM
    *************************************************************************/
    xmlcode = '<!-- ****************************** -->';
    output;
    xmlcode = '<!-- ARM Definitions Section          -->';
    output;
    xmlcode = '<!-- ****************************** -->';
    output;

    if _N_ = 1 then do;
        xmlcode = '<arm:AnalysisResultDisplays>';
        output;
    end;

    if first.table_no then do;
        xmlcode = '    <arm:ResultDisplay OID="RD.'
                  || strip(table_no2)
                  || '" Name="'
                  || strip(table_no1)
                  || '">';
        output;
        xmlcode = '        <Description>';
        output;
        xmlcode = '            <TranslatedText xml:lang="en">'
```

```
                || strip(Table_Description)
                || '</TranslatedText>';
        output;
        xmlcode = '        </Description>';
        output;
        xmlcode = '        <def:DocumentRef leafID="LF.'
                || strip(scan(Table_Name_in_PDF, 1, '.'))
                || '">';
        output;
        xmlcode = '            <def:PDFPageRef PageRefs="'
                || strip(vvalue(table_page_no))
                || '" '
                || 'Type="PhysicalRef"/>';
        output;
        xmlcode = '        </def:DocumentRef>';
        output;
    end;

    xmlcode = '       <arm:AnalysisResult';
    output;
    xmlcode = '         OID="AR.'
             || strip(table_no2)
             || '.R.'
             || strip(vvalue(sequence))
             || '"';
    output;
    xmlcode = '         ParameterOID="IT.'
             || strip(Dataset_Name)
             || '.'
             || strip(Analysis_Parameter)
             || '"';
    output;
    xmlcode = '         AnalysisReason="SPECIFIED IN SAP"' ;
    output;
    xmlcode = '         AnalysisPurpose="'
             || strip(Analusys_Purpose)
             || '">' ;
    output;
    xmlcode = '         <Description>';
    output;
    xmlcode = '             <TranslatedText xml:lang="en">'
             || strip(Description_of_Analysis)
             || '</TranslatedText>';
    output;
    xmlcode = '         </Description>';
    output;


    xmlcode = '         <arm:AnalysisDatasets>';
    output;
    xmlcode = '            <arm:AnalysisDataset ItemGroupOID="IG.'
              || strip(Dataset_Name)
              || '">';
    output;
    xmlcode = '               <def:WhereClauseRef WhereClauseOID="'
              || strip(Selection_Criteria)
              || '"/>';
```

```
    output;
    xmlcode = '               <arm:AnalysisVariable ItemOID="IT.'
              || strip(Dataset_Name)
              || '.'
              || strip(Analysis_Variable)
              || '"/>';
    output;
    xmlcode = '         </arm:AnalysisDataset>';
    output;
    xmlcode = '         </arm:AnalysisDatasets>';
    output;

    xmlcode = '         <arm:Documentation>';
    output;
    xmlcode = '           <Description>';
    output;
    xmlcode = '             <TranslatedText xml:lang="en">'
             || strip(Documentation)
             || '</TranslatedText>';
    output;
    xmlcode = '         </Description>';
    output;
    if strip(Doc_Name_in_PDF) > '' then do;
        xmlcode = '         <def:DocumentRef leafID="LF.'
                  || strip(scan(Doc_Name_in_PDF, 1, '.'))
                  || '.'
                  || strip(vvalue(sequence))
                  || '">';
        output;
        xmlcode = '            <def:PDFPageRef PageRefs="'
                  || strip(vvalue(Doc_Page_Number))
                  || '" '
                  || 'Type="PhysicalRef"/>';
        output;
        xmlcode = '        </def:DocumentRef>';
        output;

    end;
    xmlcode = '         </arm:Documentation>';
    output;

    if strip(Programming_Statement) > '' then do;
        xmlcode = '         <arm:ProgrammingCode Context="SAS version '
                  || strip(vvalue(SAS_Version))
                  || '">';
        output;

        xmlcode = '         <arm:Code>';
        output;
        xmlcode = strip(Programming_Statement);
        output;

        xmlcode = '         </arm:Code>';
        output;
        xmlcode = '         </arm:ProgrammingCode>';
        output;
    end;
```

```
    xmlcode = '          </arm:AnalysisResult>';
    output;
    xmlcode = '          </arm:ResultDisplay>';
    output;

    if strip(SAS_Source_Code_in_Text) > '' then do;
        xmlcode = '          <arm:ProgrammingCode Context="SAS version '
                  || strip(vvalue(SAS_Version))
                  || '">';
        output;

        xmlcode = '<def:leaf ID="LF.'
                  || strip(scan(SAS_Source_Code_in_Text, 1, '.'))
                  || '.'
                  || strip(vvalue(sequence))
                  || '" />';
        output;
        xmlcode = '          </arm:ProgrammingCode>';
        output;
    end;

    if last then do;
        xmlcode = '</arm:AnalysisResultDisplays>';
        output;
    end;
run;

/*generate the xml syntax in text format*/
data _null_;
        file "C:\doc\ARMTEXT.txt" lrecl = 8196 nopad encoding = "utf-8"; ;
        set armtext(keep = xmlcode);
        x = length(xmlcode)- length(left(xmlcode));
        put @x xmlcode ;
run;
```

The above SAS code generates the valid xml syntax in text format, which is the main part of the ARM. Similarly, we have to generate the valid xml syntax for PDF files referenced the ARM. See below for an example:

```
data pdftext(keep = sect xmlcode) ;
    set arm end = last;
    by table_no;
    length xmlcode $2000;
    /*************************************************************************
        Step 1: Generate Leafs Definition for ARM
     *************************************************************************/
    xmlcode = '<!-- ****************************** -->';
    output;
    xmlcode = '<!-- Leafs Definitions Section      -->';
    output;
    xmlcode = '<!-- ****************************** -->';
    output;

    if first.table_no then do;
        if strip(Table_Name_in_PDF) >''  then do;
            sect = 2.5;
```

```
            xmlcode = '<def:leaf ID="LF.'
                 || strip(table_no)
                 || '" xlink:href="'
                 || strip(Table_Name_in_PDF)
                 || '">';
            output;
            xmlcode = '<def:title>'
                 || strip(table_no1)
                 || '</def:title>';
            output;
            xmlcode = '</def:leaf>';
            output;
        end;
    end;

    if strip(Doc_Name_in_PDF) >''  then do;
        sect = 2.6;
        xmlcode = '<def:leaf ID="LF.'
             || strip(scan(Doc_Name_in_PDF, 1, '.'))
             || '.'
             || strip(vvalue(sequence))
             || '" xlink:href="'
             || strip(Doc_Name_in_PDF)
             || '">';
        output;
        xmlcode = '<def:title>'
             || strip(Doc_reference)
             || '</def:title>';
        output;
        xmlcode = '</def:leaf>';
        output;
    end;

    if strip(SAS Source_Code_in_Text ) >''  then do;
        sect = 2.7;
        xmlcode = '<def:leaf ID="LF.'
             || strip(scan(SAS_Source_Code_in_Text, 1, '.'))
             || '.'
             || strip(vvalue(sequence))
             || '" xlink:href="'
             || strip(SAS_Source_Code_in_Text)
             || '">';
        output;
        xmlcode = '<def:title>'
             || strip('SAS Source Code')
             || '</def:title>';
        output;
        xmlcode = '</def:leaf>';
        output;
    end;
run;
data _null_;
        file "C:\doc\Leaf.txt" lrecl = 8196 nopad encoding = "utf-8"; ;
        set pdftext(keep = xmlcode);
        x = length(xmlcode)- length(left(xmlcode));
        put @x xmlcode ;
run;
```

Once the text file of valid xml syntax is generated, we can readily copy each section of the text and paste them into existing define in proper locations, then we will get the correct display of ARM in existing define.xml. See below for an example:

Date of Define-XML document generation: 2017-01-23T20:12:03

Stylesheet version: 2015-01-16

Standard	ADaM-IG 1.0
Study Name	MK9999-KN999
Study Description	Analysis Datasets for Study MK9999-KN999
Protocol Name	Study MK9999-KN999
Metadata Name	Study MK9999-KN999 Data Definitions
Metadata Description	Analysis Datasets for Study MK9999-KN999

Analysis Results Metadata (Summary) for Study MK9999-KN999

Table 14.2.1.2a Summary of Investigator's Assessment of Clinical Outcome at EOIV, EOT and TOC (mMITT Population) Clinical outcome based on Investigator's assessment of clinical response (cure, improved, failure or non-evaluable) at End of Intravenous Therapy Visit, End of Intravenous+Oral Therapy Visit and Test-of-Cure / Safety Visit in the microbiological Modified Intent-to-Treat (mMITT) population
Table 14.2.2.2a Summary of Pathogen-Level Microbiological Outcome at TOC by Blinded Central Review (mMITT Population) Pathogen-Level Microbiological Response at Test-of-Cure / Safety Visit in the microbiological Modified Intent-to-Treat (mMITT) population for each baseline infecting pathogen
Table 14.2.3.2a Summary of Subject-Level Microbiological Outcome at TOC (mMITT Population) Subject-Level Microbiological Response at Test-of-Cure / Safety Visit in the microbiological Modified Intent-to-Treat (mMITT) population, derived from the Pathogen-Level Microbiological Response for all of the subject's Baseline Infecting Pathogens and from the presence or absence of a Persisting and/or Super-infecting Pathogen (Gram-positive)

Figure 2. Screen print of ARM (Summary) added in existing define.xml version 2.0

Analysis Results Metadata (Detail) for Study MK9999-KN999

Table 14.2.1.2a

Display	SAS Programs Summary of Investigator's Assessment of Clinical Outcome at EOIV, EOT and TOC (mMITT Population)
Analysis Result	**Clinical outcome based on Investigator's assessment of clinical response (cure, improved, failure or non-evaluable) at End of Intravenous Therapy Visit, End of Intravenous+Oral Therapy Visit and Test-of-Cure / Safety Visit in the microbiological Modified Intent-to-Treat (mMITT) population**
Analysis Parameter(s)	PARAMCD = "OVAL" (Overall Response)
Analysis Variable(s)	AVALC (Analysis Value (C))
Analysis Reason	SPECIFIED IN SAP
Analysis Purpose	EFFICACY OUTCOME MEASURE
Data References (incl. Selection Criteria)	ADEF [PARAMCD = "OVAL"]
Documentation	Clinical success rates were evaluated by calculating a 95% confidence interval around the difference in rates among the 2 treatment groups (daptomycin group minus standard of care comparator) using the Newcombe hybrid score (Wilson) confidence limits for the risk difference without continuity correction. SAS Programs

Programming Statements	[SAS version 9.3]
	`proc freq data=ADEF;`
	`tables TRTPN*AVALC / riskdiff (method = NEWCOMBE);`
	`run;`
	`proc freq data=ADEF;`
	`tables TRTPN*Y_var; weight Weight_var;`
	`exact riskdiff;`
	`run;`

Figure 3. Screen print of ARM (Details) added in existing define.xml version 2.0

CONCLUSION

Analysis Result Meta is a valuable feature in define.xml version 2.0 with functions including providing traceability from analysis display back to analysis datasets. This paper presents a simple method to

convert the collected information in Excel spreadsheet to valid xml syntax, and embeds the ARM into existing define.xml, as well as prepares them for future submissions.

REFERENCES

Frank Dilorio and Jeffery Abolafia, 2015. Results-Level Metadata: What, How, and Why." PHUSE 2015 Proceedings.

Lex Jansen, 2017. Creating Define-XML version 2 including Analysis Results Metadata with the SAS® Clinical Standards Toolkit. PharmaSUG 2017 Proceedings.

ACKNOWLEDGMENTS

The authors would like to thank Mary Varughese and Cynthia He for their great support and valuable input of this paper.

CONTACT INFORMATION

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

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

Name: Shunbing Zhao
Enterprise: Merck
Address: 126 E. Lincoln Avenue
City, State ZIP: Rahway, NJ 07065-4607
Work Phone: 732-594-3976
Fax:
E-mail: shunbing.zhao@merck.com
Web: www.merck.com

Name: Anjana Grandhi
Enterprise: Merck
Address: 126 E. Lincoln Avenue
City, State ZIP: Rahway, NJ 07065-4607
Work Phone: 732-594-4308
Fax:
E-mail: Anjana.grandhi@merck.com
Web: www.merck.com