PharmaSUG 2013 – Paper DS12

The Y2K17 Bug! Using Metadata to Respond to PDUFA V Requirements

Vincent J. Amoruccio, Alexion Pharmaceuticals, Cheshire, CT

ABSTRACT

On July 9, 2012, President Obama signed into law the Food and Drug Administration Safety and Innovation Act (FDASIA). As part of FDASIA, the Prescription Drug User Fee Act (PDUFA) was reauthorized for the fifth time. PDUFA V requires submitting standardized data to the FDA. If the data submitted are not in standardized format, then PDUFA V gives the FDA the authority to refuse the submission. At the 2012 CDISC Interchange in Baltimore, Maryland the FDA's CDER and CBER Divisions reiterated the sentiments of PDUFA V and the eStudy Data Guidance with a very clear message: Standards will be mandated by 2017!

While the standards are still in draft form, we have been enlightened by the FDA and CDISC of what to expect. First, the FDA has already announced its expectation to only receive electronic submissions. Second, the FDA has strongly alluded to mandating SDTM and ADaM Standards. Third, we anticipate the requirement of metadata as a standard component of SDTM and ADaM.

At Alexion Pharmaceuticals, we have been using a platform neutral metadata system as a standard way to produce clinical data output for submissions to regulatory agencies such as the FDA. This cost effective solution uses basic SAS® software and will enable us to easily respond to the PDUFA V requirements. In this paper, I will introduce this simple system and explain how it can be immediately used for regulatory submissions to the FDA and will meet the expected PDUFA V requirements.

INTRODUCTION

The Food and Drug Administration Innovation Act ("FDASIA") is an Act approved by President Obama to amend the Federal Food, Drug, and Cosmetic Act ("FDC Act"). It is intended to revise and extend the user-fee programs for prescription drugs and medical devices, to establish user-fee programs for generic drugs and biosimilars, and for other purposes. Our industry is most affected by Titles One and Eleven.

FDASIA gained most of its recognition from Title One which re-authorized the Prescription Drug User Fee Act for the fifth time ("PDUFA V"). PDUFA was first enacted in 1992 to generate revenue from user fees paid by drug and biologic manufacturers in exchange for the FDA's agreement to expedite the review process for sponsors submitting certain New Drug Applications ("NDAs") under the FDC Act and Biologics License Applications ("BLAs") under the PHS Act[1].

With each authorization of PDUFA, the FDA communicates goals and procedures of the reauthorization for the fiscal years following its release. Shortly after FDASIA went into effect, the FDA released a document titled "PDUFA Re-authorization Performance Goals and Procedures Fiscal Years 2013 Through 2017" document. This document highlights the changes between PDUFA IV and V. Most of PDUFA V is the same as PDUFA IV, however, two sections will impact our industry: Section 1 and Section 7.

Section 1 of PDUFA V, "Review Performance Goals", establishes a new review model that will apply to most NDAs and original BLAs received between October 1, 2012 and September 30, 2017 including applications that are resubmitted as a result of a Refuse-to-File action. This new review model will promote greater transparency and improve communication between the FDA and the application and improve the efficiency and effectiveness of the review process and decrease the number of review cycles required for approval.

Section 7 of PDUFA V, "Improving the Efficiency of Human Drug Review Through Required Electronic Submissions and Standardization of Electronic Drug Application Data" also has a significant impact on our industry. First, Section 7 of PDUFA V will mandate ICH M2 EWG Electronic Common Technical Document Specifications unless the FDA determines another version to be used. This is directly related to Section 745A of Title Eleven (XI) in FDASIA. Second, and most important, it will require Clinical Terminology Standards through open standards development organizations such as the Clinical Data Interchange Standards Consortium ("CDISC"). The goal of completing and implementing controlled terminology is by Fiscal Year 2017. In other words, CDISC will be required by 2017!

Existing Study Data Standards

The FDA is comprised of two distinct drug information centers ("DIC"): The Center for Drug Evaluation and Research ("CDER") and The Center for Biologic Evaluation and Research ("CBER"). Both CDER and CBER promote and

protect the public's health by ensuring their products are safe and effective. CDER regulates prescription and non-prescription drugs while CBER regulates biologic products including blood, vaccines, allergenics, and technologies.

In the current guidance from both CBER and CDER, they "strongly encourage" but do not formally require the use of data standards for the submission of applications. In December 2011, CDER issued an update to its Common Data Standards Issues Document. This document provided specific suggestions towards the use of the most up-to date Study Data Tabulation Model ("SDTM") and Analysis of Data Sets Model ("ADaM") along with metadata (aka: "Define") files in both XML and PDF.

CDER also provides specifications for submitting study data electronically (V2.0 Released July 18, 2012). These specifications directly correspond to the most recent SDTM, ADaM, and ICH eCTD guidelines.

Impact of PDUFA V

The impact of PDUFA V and the FDA's Performance Goals and Procedures provide a transition from weak, inoperable guidelines to strong, enforceable requirements. To those who are already well versed in SDTM, ADaM, Define files, and ICH eCTD Requirements, PUDFA V will have the least impact. PDUFA V will provide nothing further than an imposed requirement to follow standards that had already been used. Those who have not submitted data to the FDA following the "strongly encouraged" standards will be impacted the most. They will need to learn and implement these standards within a certain time frame before being at risk for a Refusal-to-File action from the FDA.

Both FDASIA and PDUFA V send a message of increased speed. FDASIA requires organizations to pay more money for their applications. The FDA has agreed to have a quicker review cycle to approve drug applications quicker and to put medicine in the patient's hands faster. This will greatly benefit public health, but will also put a demand Pharmaceutical Companies do more, faster, and better!

METADATA

Metadata can help Pharmaceutical Companies do more, faster, and better! The concept of metadata is often misunderstood since the term metadata is ambiguous. Bretheron & Singley distinguish between two distinct classes: structural/control metadata and guide metadata[2]. According to them, structural metadata are used to describe the structure of computer systems such as tables, columns and indexes. Guide metadata are used to help people find specific items and are usually expressed as a set of keywords in a natural language. The metadata used in this paper is a combination of both. It is purely data about data. We will use data to define the structure of other data, specifically the attributes of data sets, variables, and parameters. We will also use data to help end users find specific data.

Metadata Platforms

A platform is the underlying hardware or software for a computing system. A metadata platform is the underlying software or file type on which metadata is stored. SAS® is a powerful system of software products and has a wide array of data types it can read and convert into SAS® data. SAS® can easily read files from Microsoft Excel, Microsoft Access, dBASE, Lotus, and other delimited files (tab, comma, etc.).

Depending on how the metadata is used in SAS®, there might be limitations from each platform. If metadata is going to be dynamically accessed by people, SAS®, or both, file-access permissions must be taken into consideration. Microsoft Excel and various delimited files, such as CSV and TXT files, will be problematic if they need to be accessed simultaneously by multiple people or programs. These file types are usually "single-access" files where only one person or program may access it at time.

Platform Neutrality

Metadata, or data about data, is platform neutral as long as the data it stores can be clearly identified by rows and columns. If this is true, then the platform storing the data is irrelevant and the metadata should be easily hosted by any other platform other than the one it is stored in. The metadata system at Alexion was initially created using Microsoft Excel and was transformed into Microsoft Access since Microsoft Access is an easy to use relational database. The concepts of metadata described in this paper can be established using any platform which stores data in a row-by-column structure.

I would highly recommend a relational database such as Microsoft Access since it can be accessed by multiple users, remain open while being used by SAS®, and use structured query language to join and query data. Using other platforms, such as Microsoft Excel and other delimited files (tab, comma, etc.), will more than suffice but will require more SAS® programming code to join and query data.

The Role of Metadata in a SAS® Programming Environment

Metadata in SAS® programming should be used to create efficiency. First and foremost, it can create efficiencies in the generation of SAS® code. Second, it can create efficiencies in the reporting of data about various SAS® objects. The metadata system at Alexion evolved, and is still evolving, from a process improvement initiative to a system which now defines many of our programming processes.

The metadata concepts I have defined in this paper and have put in place at Alexion are meant to reduce the amount of code required to create SAS® output. It also provides a centralized repository to find information about the SAS® output. Metadata is similar to macros in that it can be used to create repeatable processes that are portable from one area to another.

The rest of this paper will focus on using metadata to increase efficiencies for the reporting and analysis of SDTM and ADaM data for an FDA submission. This addresses the requirement in PDUFA V Section 7 pertaining to CDISC. Secondly, it will incorporate compliance checks for ICH eCTD requirements. This will be discussed mostly in the section about Tables, Listings, and Figures since SDTM and ADaM guidelines are already ICH eCTD compliant.

Alexion's metadata platform uses Microsoft Access and macros written using SAS®/Base. The only costs involved are Microsoft and SAS® licenses which most users already have. There are many advanced systems out there like SAS® Data Integration ("DI"), and Clinical Data Integration ("CDI"). These solutions offer similar metadata along with enhanced functionality with a higher cost.

METADATA AS A SYSTEM

Both SDTM and ADaM require the define.xml file ("Define file"). Version 1.0 of CDISC's Metadata Submission Guideline Package explains the requirements of the Define file in greater detail. In this section, I will incorporate all of the CDISC SDTM required data columns ("variables"), however, I will only discuss the usage of variables required to create the SAS® output. For further information or details, refer to the CDISC guidance. This paper will not differentiate between SDTM and ADaM Define files since there is little difference between the two.

The Define file is defined by CDISC as the metadata describing the structure and content of the submitted data sets. It contains three main components: Data Set-level metadata, Variable-level metadata, and Parameter Value-level metadata. Alexion's metadata system uses these three components not only to build the Define.xml files but also to dynamically create the SAS® data set used in a submission. We integrated these components into a single file which provides a full specification of the database being created ("database specification").

Data Set-Level Metadata

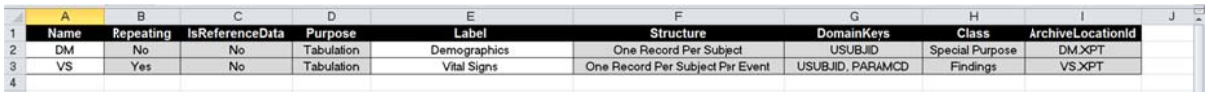
Data Set-level metadata describes the attributes which define the data set. The format of this metadata is characterized by the Define file. Using CDISC's Metadata Submission Guideline Package, Table 1 below provides a description of each metadata variable required for Data Set-level Metadata. It also indicates whether the variable is required for the Define file, SAS®, or both. Data set name and label are minimum requirements for defining a SAS® data set. At Alexion, we have integrated the metadata for SAS® and the Define file into one file. As a result, our metadata has all of the variables in Table 1.

Variable	Required	Description
Name	Define, SAS®	The name of the data set
Repeating	Define	Valid values are "Yes" or "No". Used to identify data sets having more than one record per subject.
IsReferenceData	Define	Valid values are "Yes" or "No". Used to identify data sets with reference data only (e.g. Trial Design Data sets).
Purpose	Define	The purpose of the data set.
Label	Define, SAS®	A short description about the variable.
Structure	Define	Used to describe the individual records in the data set.
DomainKeys	Define	The variables which uniquely identify a record.
Class	Define	The CDISC Class for the domain.
ArchiveLocationId	Define	Provides a reference to the transport file path.

Table 1. Data Set-Level Metadata Variables

Figure 1 below provides an example of Data Set-Level Metadata used in Microsoft Excel. Columns A and E ("Name" and "Label") in white are metadata we want to use in SAS® to define the data set attributes. Columns B,C,D,F,G, H

and I in grey are the additional metadata CDISC requires for the define file. They will be stored in the metadata but not used by SAS® to provide an attribute to the data set.



	A	B	C	D	E	F	G	H	I	J
1	Name	Repeating	IsReferenceData	Purpose	Label	Structure	DomainKeys	Class	ArchiveLocationId	
2	DM	No	No	Tabulation	Demographics	One Record Per Subject	USUBJID	Special Purpose	DM.XPT	
3	VS	Yes	No	Tabulation	Vital Signs	One Record Per Subject Per Event	USUBJID, PARAMCD	Findings	VS.XPT	
4										

Figure 1. Data Set-Level Metadata Variables shown in Microsoft Excel

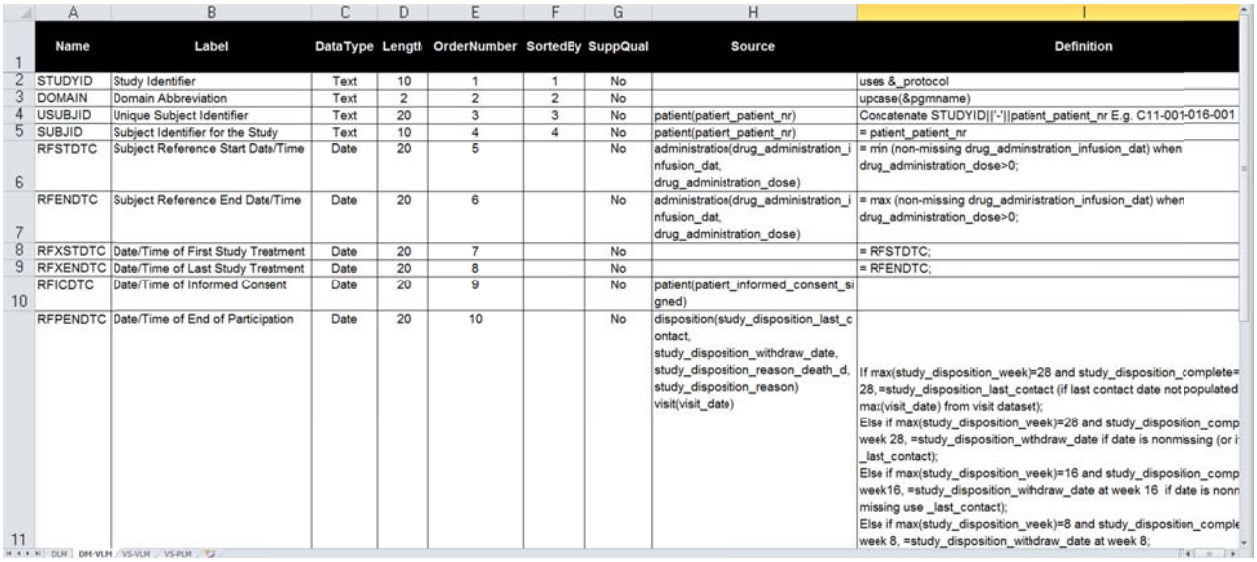
Variable-Level Metadata

Variable-level metadata contains the attributes for each variable within a data set. Using CDISC's Metadata Submission Guideline Package, Table 2 below provides a description of each metadata variable required for Variable-level Metadata. It also indicates whether the variable is an Alexion Utility, Define file, or SAS® variable. The SAS® variables needed in the metadata to define variable attributes are name, label, data type, order number, sort order ("SortedBy"), and significant digits.

Attribute	Required	Note
Name	Define, SAS®	Variable name.
Label	Define, SAS®	Variable label.
DataType	Define, SAS®	The permissible XML file types (text, integer, float, datetime, date, time). Integer and float translate to "Num" in SAS®. The others translate to "Char".
Length	Define, SAS®	Variable length.
OrderNumber	Define, SAS®	Provides the order of the variable within the data set.
SortedBy	SAS®	Identifies the variable as a sort variable and its position in the array of variables.
SuppQual	Alexion Utility	Valid values are "Yes" or "No". "Yes" if the variable should be used in the data set's supplemental qualifier. "No" if the variable will only be used in the parent data set.
ParamVLM	Alexion Utility	Valid values are "Yes" or "No". "Yes" if the value is for normalizing data.
Source	Alexion Utility	Lists the data set name and variables used to create the variable. The convention used is data set(variables).
Definition	Alexion Utility	A technical definition of the variable.
Comments	Define	Remarks or observations about the variable.
Mandatory	Define	Identifies if the variable is required as per CDISC Implementation Guidelines.
Role	Define	Describes how a variable is used according to CDISC Implementation Guidelines.
SignificantDigits	Define, SAS®	Identifies the number of digits after the decimal point for float data types. This is not a requirement for SAS® but can be used dynamically.
Origin	Define	Indicator for the origin of the variable.
CodeList	Define	A reference to the Controlled Terminology used.
ComputationalMethod	Define	Provides guidance towards the algorithm used to derive or impute the variable values.

Table 2. Variable-Level Metadata Variables

Figure 2 below shows a sample of variable-level metadata used in Microsoft Excel for the variables required by SAS®.



	A	B	C	D	E	F	G	H	I
1	Name	Label	DataType	Length	OrderNumber	SortedBy	SuppQual	Source	Definition
2	STUDYID	Study Identifier	Text	10	1	1	No		uses &_protocol
3	DOMAIN	Domain Abbreviation	Text	2	2	2	No		upcase(&pgmname)
4	USUBJID	Unique Subject Identifier	Text	20	3	3	No	patient(patient_patient_nr)	Concatenate STUDYID||'-'||patient_patient_nr E.g. C11-001-016-001
5	SUBJID	Subject Identifier for the Study	Text	10	4	4	No	patient(patient_patient_nr)	= patient_patient_nr
6	RFSTDTC	Subject Reference Start Date/Time	Date	20	5		No	administration(drug_administration_infusion_dat, drug_administration_dose)	= min (non-missing drug_administration_infusion_dat) when drug_administration_dose>0;
7	RFENDTC	Subject Reference End Date/Time	Date	20	6		No	administration(drug_administration_infusion_dat, drug_administration_dose)	= max (non-missing drug_admiristration_infusion_dat) when drug_administration_dose>0;
8	RFXSTDTC	Date/Time of First Study Treatment	Date	20	7		No		= RFSTDTC;
9	RFXENDTC	Date/Time of Last Study Treatment	Date	20	8		No		= RFENDTC;
10	RFICDTC	Date/Time of Informed Consent	Date	20	9		No	patient(patient_informed_consent_signed)	
11	RFPENDTC	Date/Time of End of Participation	Date	20	10		No	disposition(study_disposition_last_contact, study_disposition_withdraw_date, study_disposition_reason_death_d, study_disposition_reason) visit(visit_date)	If max(study_disposition_week)=28 and study_disposition_complete=28, =study_disposition_last_contact (if last contact date not populated ma:(visit_date) from visit dataset); Else if max(study_disposition_week)=28 and study_disposition_comp week 28, =study_disposition_wthdraw_date if date is nonmissing (or i _last_contact); Else if max(study_disposition_week)=16 and study_disposition_comp week16, =study_disposition_withdraw_date at week 16 if date is nonn missing use _last_contact); Else if max(study_disposition_week)=8 and study_disposition_comple week 8, =study_disposition_withdraw_date at week 8;

Figure 2. Variable-Level Metadata Variables required for SAS® shown in Microsoft Excel

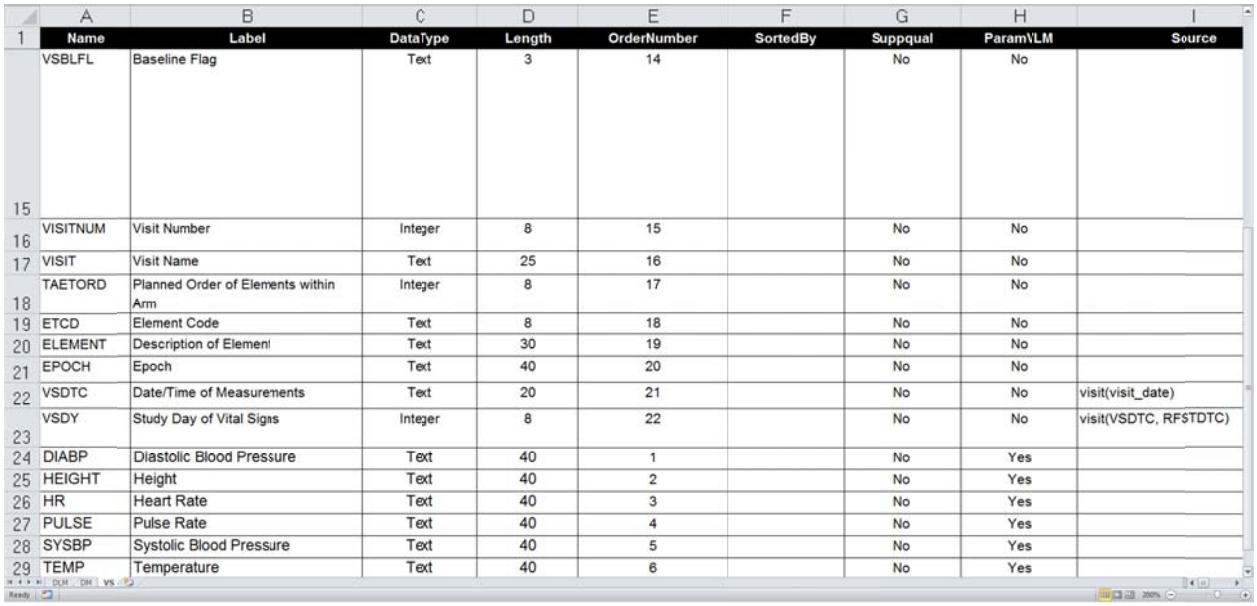
Figure 3 below shows a sample of variable-level metadata used in Microsoft Excel for the variables required by the Define file.

	A	J	K	L	M	N	O	P
1	Name	Comments	Mandatory	Role	SignificantDigits	Origin	CodeList	ComputationalMethod
14	SITEID		Yes	Identifier		Assigned		
15	INVNAM		No	Record Qualifier		Pg. 1		
16	BRTHDTC		No	Record Qualifier		Pg. 1		
17	AGE		No	Record Qualifier		Derived		See Age-Calculations.doc
18	AGEU		No	Variable Qualifier		Derived		
19	SEX		Yes	Record Qualifier		Pg. 1	SEX	
20	RACE		No	Record Qualifier		Pg. 1	RACE	
21	ARMCD		Yes	Record Qualifier		Pg. 1		
22	ARM		Yes	Synonym Qualifier		Assigned		
23	ACTARMCD		Yes	Synonym Qualifier		Assigned		
24	ACTARM		Yes	Synonym Qualifier		Assigned		
25	COUNTRY		Yes	Record Qualifier		Pg. 1		
26	DMDTC		No	Record Qualifier		Pg. 1		
27	DMDY		No	Timing		Pg. 1		
28	AGEYEAR		No	Record Qualifier		Derived		
29	AGEMONTH		No	Record Qualifier		Derived		
30	RACEOTH		No	Record Qualifier		Pg. 1		
31	PTTELFAX		No	Record Qualifier		Pg. 1		
32	ORGBRAIN		No	Record Qualifier		Pg. 5		
33	ORGKIDNY		No	Record Qualifier		Pg. 5		

Figure 3. Variable-Level Metadata Variables required for Define Files shown in Microsoft Excel

Parameter Value-Level Metadata

A horizontal data set is a data set where the columns are characteristics, the rows are objects to describe (usually people), and the values of the row-by-column cell hold the value of the characteristic for the object. A vertical data set, also known as a “normalized” data set, is a data set where each row is the object and the characteristic. The column is a general place holder for each object’s value of the characteristic. Parameter value-level metadata provides data about the records in a normalized or vertical data structure. Since they are described using the same attributes as the variable-level metadata, we have combined them into the Variable-level metadata structure. Figure 4 below shows the metadata variables used for the Vital Signs data set using ParamVLM = “Yes” in Column H.



	A	B	C	D	E	F	G	H	I
1	Name	Label	DataType	Length	OrderNumber	SortedBy	Suppqual	Param\VLM	Source
15	VSBLFL	Baseline Flag	Text	3	14		No	No	
16	VISITNUM	Visit Number	Integer	8	15		No	No	
17	VISIT	Visit Name	Text	25	16		No	No	
18	TAETORD	Planned Order of Elements within Arm	Integer	8	17		No	No	
19	ETCD	Element Code	Text	8	18		No	No	
20	ELEMENT	Description of Element	Text	30	19		No	No	
21	EPOCH	Epoch	Text	40	20		No	No	
22	VSDTC	Date/Time of Measurements	Text	20	21		No	No	visit(visit_date)
23	VSDY	Study Day of Vital Signs	Integer	8	22		No	No	visit(VSDTC, RFSTDTC)
24	DIABP	Diastolic Blood Pressure	Text	40	1		No	Yes	
25	HEIGHT	Height	Text	40	2		No	Yes	
26	HR	Heart Rate	Text	40	3		No	Yes	
27	PULSE	Pulse Rate	Text	40	4		No	Yes	
28	SYSBP	Systolic Blood Pressure	Text	40	5		No	Yes	
29	TEMP	Temperature	Text	40	6		No	Yes	

Figure 4. Parameter Value-Metadata Variables shown in Microsoft Excel

A Metadata Attribute Macro

The purpose of the attribute macro is to combine the Data Set-Level Metadata, Variable-Level Metadata, and Parameter Value-Level Metadata to dynamically create and assign the data set, variable, and parameter value attributes.

The attribute macro “ATTR” will create three temporary data sets with the domain name as the prefix: <domain>ATTR, <domain>SUPP, and <domain>PVLM. For each variable provided in the Variable-level Metadata, the data set <domain>ATTR (e.g. dmATTR) will contain the null variables with their attributes. To assign the attributes, it will be concatenated with the data set containing all of the variables identified in the Variable-Level Metadata (e.g. sdtm_code below).

```
data final;
 set dmATTR
     sdtm_code;
run;
```

This data set may contain other variables, but must minimally contain variables specified in the Variable-Level Metadata. If it does not contain all of the variables in the Variable-level Metadata, they will be created by default and will be null. The ATTR macro will also create the macro variables “Domain”, “ListAll”, “DataLbl”, and “SortedBy” which will be used in a SQL statement to create the final, permanent data set.

```
proc sql feedback;
  create table sdtm.&domain. (label = "&datalbl.") as
  select &listall
  from final
  order by &SortedBy;
quit;
```

An explanation of the ATTR macro is specified below.

```
%macro attr;

/* Set Global Macros */
%global Domain ListAll DataLbl SortedBy;
```

All of our programs use a standard macro variable “PGMNAME”. Our SDTM and ADaM data set programs are named by the domain name (e.g. dm.sas, adae.sas). As a result, we can use the “PGMNAME” macro parameter in the attribute macro to identify the domain name.

```
/* Get the Domain Name */
proc sql feedback noprint;
     select upcase(strip(value)) into : Domain
     from Sashelp.Vmacro
     where name = "PGMNAME"; /* Program Name is the domain name (e.g dm.sas)*/

quit;

/* Get the data from the module */
libname libref pcfiles server="alxnct-sasd01.alxn.net" port=8621
        path="C:\users\amorv1982\PharmaSUG2013\Data\SDTM\Metadata\sdtm-database-
              specifications.accdb";

data &Domain. &Domain.SUPP &Domain.PVLM SortedBy(Keep = VarName SortedBy);
     length SASVarType $4.;
     set libref.&qry;
```

SAS® and the Define file report variable type and length differently. The macro takes the values from the Define file and creates similar variables for SAS®.

```
     /* SAS Variable Length */
     if upcase(DataType) in ("INTEGER", "FLOAT") then SASVarType = "Num";
             else if DataType ^= '' then SASVarType = "Char";

     /* SAS Variable Length*/
     if upcase(SASVarType) = "CHAR" then SASVarLength =
             compress('$'||put(VarLength,4.));
             else SASVarLength = put(VarLength,4.);

     /* Domain Label */
     if DomainLabel > '' then call symput('datalbl',strip(DomainLabel));
```

The data set <domain>SUPP (e.g. dmSUPP) will contain the Variable-level Metadata for all of the variables marked as Supplemental Qualifier and can be used as needed to develop the domain's Supplemental Qualifier (SuppQual="Yes"). Finally, the normalized data will be contained in the <domain>PVLM data set. It can be joined to any data set to bring over the normalized data attributes.

```
       if SuppQual = "No" and ParamVLM = "No" then Output &Domain;
       if SuppQual = "Yes" and ParamVLM= "No" then output &Domain.SUPP;
       if SuppQual = "No" and ParamVLM = "Yes" then output &Domain.PLM;
       if SortedBy ne . then output SortedBy;
run;
libname libref clear ;

proc sql feedback noprint;
    /* List of All Variable Names - for final */
    select VarName into :listall separated by ', '
    from &Domain
    order by OrderNumber;
    %let listall = &listall;

    /* order by pulled from DB Spec */
    select VarName into :SortedBy
    separated by ', '
    from SortedBy
    order by SortedBy;
quit;
```

```
* automate attrib statement iteratively *;
data _null_;
     set &Domain end=last;
     i+1;
       call symput('var'||strip(put(i,best.)), cats(VarName));
       call symput('lbl'||strip(put(i,best.)), cats(VarLabel));
       call symput('lgt'||strip(put(i,best.)), cats(SASVarLength));
       call symput('typ'||strip(put(i,best.)), cats(SASVarType));

     if last then call symput('total', strip(put(i,best.)));
run;

/* Create shell of module data set */
data final;
     /* Automatic Attributed Statement*/
     attrib
     %do i = 1 %to &total.;
          &&var&i.
          length = &&lgt&i.
          label = "&&lbl&i"
     %end;;

     /* Create null variable by type */
     %do j = 1 %to &total.;
          %if %upcase(&&typ&j) = CHAR %then %do;
               &&var&j = '';
          %end;
          %if %upcase(&&typ&j) = NUM %then %do;
               &&var&j = .;
          %end;
     %end;

     /* Delete the OBS = 1 and make null data set */
     if _n_ = 1 then delete;
run;

/* Create Permanent Attributed Shell */
proc sql feedback;
     create table &Domain.ATTR as
     select &listall.
     from final;
quit;

%mend attr;
```

TLF metadata

One particular area of CDISC where there are no standards is for analysis result-level metadata. CDISC has strongly implied that this will be addressed in future guidance. Analysis result-level metadata will be data about each analysis display, specifically each Table, Listing, and Figure (“TLF”). Despite the lack of structure and guidance from CDISC, Alexion has implemented a very useful metadata process for the development of TLFs.

Our TLF metadata contains all of the attributes which define each particular display. This metadata might vary because each company’s preferences are different. For example at Alexion, we produce a single display for each analysis set (e.g. demographics-ITT.rtf, demographics-PP.rtf, etc.). The purpose of the TLF metadata is to store all of the TLF attributes in a single, independent file so that it can be linked to the TLF program. As a result, the TLF program will never have to be updated if a TLF attribute, such as a table number, title or footnote changes.

Our TLF Metadata contains three metadata components: Analysis Set Metadata, Section Metadata, and TLF Metadata.

Analysis Set Metadata

A number is assigned to distinctly identify each analysis set and along with a short name and long name descriptor. The number will be used as a part of the display number. The short name descriptor is used in the file name, while

the long name descriptor is used in display titles. For example 1 = “FAS” (Full Analysis Set), 2=”PPS” (Per Protocol Set), and 3 = “SFS” (Safety Set).

Section Metadata

Sections are used to distinctly organize the content of the TLF output into logical types. When appropriate, the ICH E3 Guidelines for the Structure and Content of Clinical Study Reports are followed for the section numbers. Within a section, numbering may be applied to breakdown the contents of the section by subtype. Short Names are used in the output file name to help define the content of the file. Table 3 below provides an example of ICH E3 Sections used at Alexion.

Title	Content	Number	Short Name
Demographics	Demographics	14.1.1	DEM
Demographics	Disposition	14.1.2	DISP
Demographics	Medical History	14.1.3	MEDHX
Efficacy	Primary Endpoints	14.2.1	EFF-PRIM
Efficacy	Secondary Endpoints	14.2.2	EFF-SEC
Efficacy	Tertiary Endpoints	14.2.3	EFF-TERT
Efficacy	Exploratory Endpoint	14.2.4	EFF-EXPL

Table 3. Section Metadata following ICH E3 Guidance

TLF Metadata

The TLF Metadata capture data about the display output which would normally be generated through SAS® code. These variables are considered “Required”. Other variables are not required, but are used to guide the user’s access and understand the data.

Variable	Required	Description
RecordID	Yes	ID which uniquely identifies each display
AnalysisSetShort	Yes	Links the display to the Analysis Set Metadata to identify the displays analysis.
SectionContent	Yes	Links the display to the Section Metadata to identify the section it belongs to.
Order	Yes	The number which identifies where the display falls in each section.
DisplayShortName	Yes	A short descriptor of the analysis performed in the display (e.g. KAPPM for Kapplan Meier).
Title	Yes	The Title of the Display
Footnote<n>*	No	The Footnote which will get passed into FootNote<n> in SAS®. *Alexion’s standard display reserve the last two footnotes (usually 9 and 10). Footnotes 1 to 8 are available.
ProgramName	No	The program creating the display
ActiveYN	Yes	Identifies if the record is active.
UniqueYN	No	Identifies Unique versus Repeatable output
ModificationDate ModificationHistory	No	Allow for modification history of changes.

Table 4. Alexion’s TLF Metadata

Figure 5 provides an example of the Table Metadata.

	A	B	C	D	E	F	G	H	O	P	Q
1	RecordId	AnalysisSetShortName	SectionContent	order	DisplayShortName	Title	Footnote1	Footnote2	ProgramName	ActiveYN	UniqueYN
2	1	FAS	DEMOGRAPHICS	1	SUM	Summary of Characteristics			dem	Y	Y
3	2	FAS	DEMOGRAPHICS	2	SUM-BASEL	Summary of Baseline Characteristics			dem	Y	N
4	3	PPS	DEMOGRAPHICS	3	SUM	Summary of Characteristics			dem	Y	N
5	4	PPS	DEMOGRAPHICS	4	SUM-BASEL	Summary of Baseline Characteristics			dem	Y	N
6	5	FAS	SECONDARY ENDPOINTS	1	LOGISTIC-CREATNORM	Summary of Creatinine Normalization	Data after 04SEP2011 was not considered.	Quasi-Separation Warnings are due to low frequency of covariae.	logistic	Y	Y

Figure 5. Table Metadata shown in Microsoft Excel

The TLF Metadata Macro

The macro below will create 14 macro variables to be used in a table, listing, or figure program. The macro requires the parameter “RecordId” be specified. “RecordId” is the link to the TLFMetaData for the display being created. For example, the program “Logistic.sas” in Figure 5 above, would have a call to the macro specifying “RecordId=5”. By doing so, the macros Title1, Title2, Footnotes1-Footnote8 will be pulled in from the metadata and provide the full SAS® code to create the title and footnote. Additionally, the macro will create a macro for the filename and the SAS® data set for the table.

The output filename for each display will consist of a padded output number, section short name, display short name, and analysis set short name, each separated by a hyphen. It will be prefixed with a “t” for Tables, “l” for Listings, and “f” for figures. The output number ensures that output will list in the correct order in an explorer window such as Windows Explorer or the eCTD system used for FDA Submissions. See Table 5 below.

Table Number	Padded Number	Short Name	Output File Name
14.1.1.1.1	140101000101	DEM-SUMBASEL-CHARS-FAS	t140101000101-dem-sumbasel-chars-fas.rtf
14.1.1.1.2	140101000102	DEM-SUMBASEL-CHARS-PPS	t140101000102-dem-sumbasel-chars-pps.rtf
14.2.1.2.1	140201000201	EFF-PRIM-KAPPM-CREAT-FAS	t140201000201-eff-prim-kappm-creat-fas.rtf
14.2.1.2.2	140201000202	EFF-PRIM-KAPPM-CREAT-PPS	t140201000202-eff-prim-kappm-creat-pps.rtf

Table 5. Alexion’s TLF Output File Names

An explanation of the TLFMetaData macro is specified below.

```
%macro TLFMetaData(RecordId=);

/* Global Macros */
%global title1 title2 footnote1 footnote2 footnote3 footnote4 footnote5 footnote6
footnote7 footnote8 flnme bdat;

/* Import the Analysis Set MetaData */
proc import table = "AnalysisSetMetaData" out = AnalysisSetMetaData dbms = AccessCS
replace; database = "&database"; server = 'alxnct-sasd01.alxn.net' ; port = 8621 ;
run;

/* Import the Section MetaData */
proc import table = "SectionMetaData" out = SectionMetaData dbms = AccessCS
     replace; database = "&database"; server = 'alxnct-sasd01.alxn.net'; port = 8621 ;
run;

/* Import the TLF MetaData */
proc import table = "TablesMetaData" out = TLFMetaData dbms = AccessCS
     replace; database = "&database";server = 'alxnct-sasd01.alxn.net' ; port = 8621 ;
run;

/* Combine the Data to Get all Numeric Code, Short and Long Names */
proc sql;
     create table tlf1 as
       select a.*, b.AnalysisSetNumber, b.AnalysisSetLongName
       from TLFMetaData as a
            left join
            AnalysisSetMetaData as B
        on a.AnalysisSetShortName = B.AnalysisSetShortName
        where a.RecordId = &RecordId.;/* Keep only the TLF MetaRecord being read in*/

       create table tlf2 as
       select a.*, b.SectionTitle, b.SectionShortName, b.SectionContentNumber
       from tlf1 as a
            left join
            SectionMetaData as b
     on a.SectionContent = b.SectionContent;
quit;
```

```
data _null_;
     set tlf2;

     /* Title 1 */
     TitleNumber = catx('.', SectionContentNumber, Order, AnalysisSetNumber);

     call symput("ttl1", compbl("Table "||TitleNumber));

     /* Title 2 */
     call symput("ttl2", compbl(AnalysisSetLongName));

     /* Footnotes */
     %do i = 1 %to 8;
          call symput("footn&i", cats(footnote&i.));
     %end;

     /* Output Number Padding */
     o1 = compress(SectionContentNumber, ".");

     if order ne '0' then o2 = put(input(Order,best.), z2.);
     else if order = '0' then o2 = '';

     o3 = put(AnalysisSetNumber, z2.);
     outnumber = cats("t", o1, o2, o3);


     /* File Name */
     call symput("flnme", lowcase(cats(outnumber,"-",SectionShortName,"-
     ",DisplayShortName,"-",AnalysisSetShortName)));

     /* SAS Output Data set */
     call symput("bdat", lowcase(cats(outnumber)));

     /* Population Macro Variable */
     call symput("anls",  cats(AnalysisSetShortName));
     call symput("anlsn", cats(AnalysisSetNumber));
run;

/* Create the title and footnote macros */
%let title1 = %bquote(title1 &ttl1); %let title2 = %bquote(title2 &ttl2);

%let footnote1 = %bquote(footnote1 j=l "&footn1");
%let footnote2 = %bquote(footnote2 j=l "&footn2");
%let footnote3 = %bquote(footnote3 j=l "&footn3");
%let footnote4 = %bquote(footnote4 j=l "&footn4");
%let footnote5 = %bquote(footnote5 j=l "&footn5");
%let footnote6 = %bquote(footnote6 j=l "&footn6");
%let footnote7 = %bquote(footnote7 j=l "&footn7");
%let footnote8 = %bquote(footnote8 j=l "&footn8");
```

Section 7 of PDUFA V mandates ICH M2 EWG Electronic Common Technical Document Specifications unless the FDA determines another version is to be used. To comply with ICH eCTD requirements:

1. The maximum length of a file name will not exceed 64 characters including the extension.
2. Only lower case letters will be used in the file name.
3. No spaces in the file name.
4. Only letters (“a” to “z”), digits (“0” to “9”), and hyphen (“-“) will be used in the file name.

This simple check can be employed to check the file name for ICH eCTD requirements.

```
data _null_;
     length specchar $100;
     specchar=compress("&flnme ","abcdefghijklmnopqrstuvwxyz0123456789-");
     if specchar^='' then spec=0; else spec=1;
```

```
     if length("&flnme") >64  then len=0 ; else len=1;
     if index(trim(left("&flnme ")),' ')>0 then spac=0; else spac=1;
     if spec=1 and len=1 and spac=1 then do;
            call symput("meetsreg","TRUE");
     end;
     else do;
          call symput("meetsreg","FALSE");
     end;
run;
%put  Filename &flnme meets regulatory submission is &meetsreg;

%mend TLFMetaData;
```

CONCLUSION

FDASIA and PDUFA V have been implemented to allow for regulatory agencies, such as the FDA, to respond to public health issues quicker and more efficiently. The Pharmaceutical Industry is most impacted by Sections One and Seven of PDUFA V. This law will require Pharmaceutical companies to file submission electronically and to follow CDISC data standards. Metadata can be used to facilitate compliance with CDISC standards and create efficiencies in the generation of SAS® code to build SAS® data sets. This paper has provided several examples of how Alexion uses a platform neutral metadata system as a standard way to produce clinical data output for submissions to regulatory agencies such as the FDA. This simple system can be immediately adopted for regulatory submissions to the FDA and expanded to meet the anticipated PDUFA V requirements by the time they become enforceable in 2017.

REFERENCES

"Hyman, Phelps, &McNamara, PC." . N.p., 11 Jul 2012. Web. 30 Mar 2013. <www.hpm.com>.

Bretherton, F. P.; Singley, P.T. (1994). "Metadata: A User's View, Proceedings of the International Conference on Very Large Data Bases (VLDB)". pp. 1091–1094.

ACKNOWLEDGMENTS

I would like to thank John Kaye, Senior Clinical Programmer at Alexion, for his assistance with code to check for ICH eCTD compliance. I would also like thank Daniel Amoruccio for his unbounded support in making this paper possible.

CONTACT INFORMATION

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

Vincent J. Amoruccio
Alexion Pharmaceuticals
352 Knotter Drive
Cheshire, CT 06410
(203) 439-9623
AmoruccioV@alxn.com
www.alxn.com