Paper AD02

A Scalable Platform For Clinical Trials - Interoperability And Automation In A Web-Based Environment

John R. Gerlach, MaxisIT, Inc.
Maulik Shah, MaxisIT, Inc.

Abstract

Imagine an XML-based platform that is CDISC compliant consisting of a suite of functional modules used to facilitate the plethora of analytical tasks involving clinical trials. XML-based CDISC standards would afford the interoperability across computing platforms, clinical studies and other resources (e.g. MedDRA); whereas, the functional modules, written in SAS and Java, would automate, at least facilitate, a given task. Beginning with the standard items that include a study protocol, an annotated case report form (CRF), the statistical analysis plan (SAP), and the clinical database (CDMS), this platform for clinical trials would expedite such processes as: creating the SAP; mapping raw data to SDTM variables; creating and validating CDISC domain data sets; generating standardized SAS programs for producing reports; creating transport data sets; and producing the Define-XML document. Other functionality would include: monitoring program development, including real-time metrics, and viewing deliverables. This paper presents a conceptual overview of a scalable, flexible computing platform for doing clinical trials.

Introduction

Even with the advent of CDISC standards, which many pharmaceutical companies are just now embracing, clinical studies are often jeopardized by a shoddy infrastructure. Even competent project managers and experienced programmers are caught in overwhelming situations where nobody seems to know either *what needs to be done* or, even, *what has been done*. Thus, the analytical component of clinical studies suffers from eleventh hour scenarios. Consequently, the results are replete with analytical errors and aesthetic oversights that jeopardize the integrity of the deliverables. And, once the client losses faith in the deliverable (after noticing several absurd errors, like having more than one hundred percent of the patients in a treatment arm suffering from a particular misspelled adverse event), the project is doomed.

Certainly, plausible applications for doing data capture, data management, and data analysis for clinical trials have been developed. But, such products usually become just another layer of complexity in the process, especially if the product lacks the flexibility to accommodate a study or client standards, which introduces all kinds of caveats. At best, these products are functionally specific (e.g., data management, report generation) and are not part of a cohesive platform. This paper proposes a computing platform that offers impressive flexibility, real scalability, and true interoperability such that analytical teams need not sit in the same building even; yet every team member knows what to do and what has been done, to do it correctly and on time.

The authors contend that a web-based platform approach surpasses the conventional approach of having a suite of applications software, which afford little with respect to accessibility and interoperability on par with the proposed platform. Also, the authors oppose the *does-everything* macro that generates numerous reports, for example. Why? Because the *does-everything* macro is huge in scope and contains scores of parameters, thereby becoming another language to learn. Even worse, the user of such an environment becomes estranged from the data and the actual analysis, which is ironic. In the proposed computing platform, the Java based interface accesses a library of validated SAS code, including macros, that are used as stand-alone utilities or incorporated into shell programs that dramatically improves the modus operandi and eliminates many menial tasks for the analyst / programmer. In short, SAS macros, for this platform, are intended to facilitate the analytical process, not to mechanize it.

It is important to maintain a holistic understanding of this platform and not to focus on a particular task that is used to illustrate functionality. It is the web-based platform, *The System*, that surpasses current computing environments. Imagine a team of professionals, not geographically bounded, having access to a platform via the Internet complete with secured protocols and security mechanisms, that has a user-friendly interface and offers the functionality needed to accomplish the numerous tasks involving the analysis of a clinical trial, including the monitoring of the work using real-time metrics.

Looking for Consistency → Templates

Even though every clinical trial is unique and sponsors have their own methods, a clinical trial is well defined and employs industry standards. For example, the reporting of adverse events that employs MedDRA classification consists of several well-known reports, namely:

- ➢ Adverse Events by System Organ Class and Treatment Group
- ➢ Adverse Events by System Organ Class by Relationship to Study Drug and by Treatment Group
- ➢ Adverse Events by System Organ Class by Severity and by Treatment Group

These reports on adverse events (AE) have several common traits, that is, rows defined by System Organ Class (Body System and Preferred Term) and columns defined by Treatment Group. The AE tables denoting relationship to study drug and severity merely apply another categorical component, either juxtaposed with the System Organ Class or embedded within each Treatment Group, as follows:

Body System / Preferred Term	Group A (N=xxx)			Group B (N=xxx)			Placebo (N=xxx)		
	Not Related n(%)	Possibly Related n(%)	Probably Related n(%)	Not Related n(%)	Possibly Related n(%)	Probably Related n(%)	Not Related n(%)	Possibly Related n(%)	Probably Related n(%)

Body System / Preferred Term	Group A (N=xxx)			Group B (N=xxx)			Placebo (N=xxx)		
	Mild n(%)	Moderate n(%)	Severe n(%)	Mild n(%)	Moderate n(%)	Severe n(%)	Mild n(%)	Moderate n(%)	Severe n(%)

Besides such common AE tables, consider tables in terms of a handful of layouts, as follows:

➢ Categorical Variable(s)	Body System, Preferred Term, Lab Test, PK/PK Parameters
➢ Time Element	Visit, Study Day
➢ Item of interest	Statistic (e.g., Mean, Standard Deviation)
➢ Across Variable(s)	Treatment Groups (e.g., Group A, Group B, Placebo)
Nested Item	AE Severity, Relationship to Study Drug
	Normal & Abnormal (e.g., Shift tables)
	Gender, Ethnicity
➢ Cells	Counts & Percentages, Other Statistics (e.g. X^2, Confidence Intervals)
	Row Totals and/or Column Totals (*n*,*%*)

Of course listings pose another problem, albeit not as challenging to program as tables since these reports simply show the data from which the tables are based. And, certainly, graphs offer other challenges, as well. In either case, an appropriate template program is attainable, which will be discussed later. Overall, the objective is to facilitate the work of the analyst such that the professional can focus on analyzing the data.

The proposed platform is based, in part, on recognizing common features in any clinical study, such as reports. In fact, there are many tasks that are common to clinical trials. This is nothing new, of course. Unfortunately, this practice is done in the form of cutting and pasting poorly written (undocumented and ugly) code into another program. In contrast, the platform contains well-written, validated code that could be inserted into a template SAS program pertaining to a given task, such as creating a SDTM domain data set or producing a report, specific to a study. In fact, the platform contains a library of

SAS code that the user selects through the web-based interface.   And, of course, the code follows CDISC naming conventions.

To emphasize the notion of template programs consider the following common reports:

- Concomitant Medication by WHO Drug Class and Treatment Group, which closely resembles the adverse events report by System Organ Class and Treatment Group
- Baseline Characteristics for vital signs and physical examinations
- Shift tables for lab tests and vital signs representing a change from Baseline to a particular visit or endpoint.
- Plots (e.g., Analysis Variable by Time) by Treatment Groups.

Of course, the names and number of treatment arms vary, but the layout is consistent. And, in the event that SAS/Graph is not used to produce the graphics, the platform can export the appropriate analysis data files for a preferred graphics application or it can interface directly with that application.

Similar to CDISC standards, naming conventions and file organization are important aspects of the proposed platform.   Rather than have programs called by such names as: AE01, AE_MINE, AE_01_06 and AE_EA, the platform creates a ‘shell’ program (see Appendix A) for each table and names the program according to the table identifier.   This function will be explained in the next section.

The following tasks are explained to convince the reader of the power and flexibility of the proposed platform:

- Generating *n*-shell programs that have a common look and feel
- Maintaining and using of titles and footnotes
- Monitoring program development by using real-time metrics
- Mapping and Validating SDTM domain data sets
- Creating the Define.xml document.
- Other Utilities

Generating N-Programs for N-Reports

The Statistical Analysis Plan (SAP) for a clinical trial identifies all the reports for a study, often consisting of well-over one hundred tables, graphs, and listings.   The usual modus operandi declares poorly written programs, filled with caveats and nuances from other studies, as viable programs, because the protocols are related. Subsequently, the pirating, cutting and pasting, morph into something very ugly and less discernible. Even worse, programmers have their own styles of writing that preclude a common *look and feel* for the collection of programs, despite so-called programming standards.

The proposed platform takes advantage of the SAP and implements several conventions, specifically: there is one program for each report; and, the name of the SAS program identifies the report identifier. A utility produces a program for each report, such as: T_14_3_1.sas, L_16_2_1_1, G_15_2_4.   But, then, what about the contents of such programs? Again, there is consistency. Consider the programmatic process of producing a report, as follows:

- Specify a program Header
- Initialize the environment
- Define formats (if necessary)
- Define the Subject population (e.g., ITT, Safety, Per Protocol, PK Analysis)
- Create the analysis (tabulation) data set
- Perform the analysis (e.g., counts, percentages, P-values)
- Create the reporting data set
- Produce the report

Now consider a template ‘shell’ program (see Appendix A), tailored for a specific client / study, that contains this information, including proper LIBNAME statements and even macro invocations to obtain titles and footnotes. Thus, even at the onset of producing the reports, all the SAS programs would exist, albeit as shells. Added functionality allows the inclusion of validated code into these programs via the interface.

Besides the obvious advantage of getting a good head start on a given study, imagine how this would instruct the junior-level programmers with respect to proper methods and style. Also, the project manager would be able to perform an immediate analysis of the project producing real-time metrics on progress, which will be discussed later.

Eliminating More Tedium

Consider the work involving a large Phase III study having over 3,000 subjects and suppose that the clinical database (CDMS) is not CDISC compliant. Also, there are over 200 reports, including graphs, listed in the Statistical Analysis Plan. Clearly, there's a lot of work to be done. Now take the seemingly simple issue of titles and footnotes that have a propensity of changing per the client, a medical writer or lead statistician. Certainly, it would be advantageous to have such information in a single repository that is easily accessible for modification. Better yet, this information would be obtained upon execution of the Report program, rather than having it hard coded in the program. Thus, the task of updating titles and footnotes could be delegated to someone other than the analyst / programmer. In fact, the programmer need not be concerned about titles and footnotes at all, since the template program dynamically inserts the macro invocation with the specific report identifier that includes the proper TITLE and FOOTNOTE statements into the report.

Initially, this information is usually found in the Statistical Analysis Plan (SAP) from which it can be converted into an appropriate format, as illustrated below. Once this information is imported into the platform, specific to a study, the task becomes little more than a clerical chore. The following illustration shows the hierarchical structure of titles and footnotes, from the main title to the last footnote that includes the date of execution of the program.

```
14.3.1.1       T  5  Adverse Events by System Organ Class by Relationship to Study Drug
               T  6  Safety Population
               F  1  Note: ‘N’ is the number of subjects in each treatment group …
               F  2        AE=Adverse Event.
               F  3  Program 14311.sas
               F  4  Date:
14.3.1.2       T  5  Adverse Events by System Organ Class and Treatment Group
               T  6  Safety Population
               F  1  Note: ‘N’ is the number of subjects in each treatment group …
               F  2        AE=Adverse Event.
               F  3  Program 14311.sas
               F  4  Date:

       :       :       :       :       :       :       :       :       :       :       :

14.3.5.1       T  5  Summary of Clinical Laboratory Parameters: Lipids
               T  6  Safety Population
               F  1  Note: ‘N’ is the number of subjects randomized in each treatment group
               F  2        and ‘n’ is the number of subjects with non-missing lab assessment
               F  3        at each visit.  Discharge visit includes subject records at …
               F  4        SD=Standard Deviation.
               F  5  Program 14311.sas
               F  6  Date:
```

Monitoring Program Development

As stated several times already projects get into trouble because nobody seems to know what's been done or what needs to be done. The well-known Tracking Sheet is often outdated, unreliable and quite useless for producing real-time metrics that give a true reality check about a project. It is one thing to know whether a particular program has been written or validated; however, it is far better to know the big picture in terms of real-time metrics, in contrast to a static unreliable document. The platform capitalizes on the *well-formed* structure of the SAS programs, specifically the header portion, and is able to produce real-time metrics that are rendered on the web-based interface. Of course, it is incumbent for the programmers to include certain information, such as: name and date, which the platform would require upon leaving the programming task.

Consider the following real-time report that shows the status of program development for creating SDTM domains, listed by Observation class. But first, keep in mind that a sponsor of a study submits only the data domains that are actually collected, not the full collection of SDTM domains. From the report it is readily apparent that the domains Findings and Trial

Design need attention, since only several domains have been created. Moreover, the validation component of this process requires immediate attention, since only 13.0% of the programs have been validated. Overall, it is obvious that the creation of SDTM domains requires more resources.

SDTM	Written				Validated			
Domain Class	Yes		No		Yes		No	
Special Purpose	2	100.0%	0	0.0%	0	0.0%	2	100.0%
Interventions	3	100.0%	0	0.0%	0	0.0%	3	100.0%
Events	3	100.0%	0	0.0%	2	66.7%	1	33.3%
Findings	1	14.3%	6	85.7%	0	0.0%	7	100.0%
Trial Design	2	33.3%	4	66.7%	1	16.6%	5	83.3%
Special Purpose Relationship	0	0.0%	2	100.0%	0	0.0%	2	100.0%
	11	47.8%	12	52.2%	3	13.0%	20	87.0%

This is a simple idea, yet it is a powerful way to monitor scores of SAS programs dynamically. And, of course, such metrics are available for Derived data sets and reports (Tables, Listings, and Figures). Similar to the metrics about the programs that create SDTM domains, the programs for reports can be categorized by genre, which bears a strong resemblance to SDTM domains, such as: Demography (DM), Adverse Events (AE), Laboratory Results (LB), Vital Signs (VS), Concomitant Medications (CM), etc. In this case, the monitoring report would show the status of all the tables, listings, and figures by genre (e.g., Adverse Events). In fact, the report could focus on the *type* of report, that is, tables, listings, or figures, as well. And, once again, the platform is flexible to accommodate the SAP that might have its own classification of reports, such as: Anti-psychotics and Cognition, outside the standard CDISC realm, thereby creating non-standard SDTM domains. Again, the platform has the flexibility to monitor whatever is germane to the study.

With these metrics the project manager can act promptly, prior to the notorious *eleventh hour*, avoiding that time crunch, which creates the worst circumstances for doing sound data analysis. For example, imagine learning that 100% of the programs that create the Derived data sets have been written and all, but one (i.e., an oversight) was validated; and, that program happens to be the one that creates the less popular SUPPQUAL domain data set that contains the population flags (e.g., Safety, ITT) – And, there's a bug in the program such that the population flags are defined incorrectly. Whoops!

Mapping and Validating SDTM Domains

With the advent of CDISC standards the mapping of a CDMS to CDISC standards has become a virtual cottage industry. Unfortunately, this very important task often fails miserably in two ways: 1) the lack of CDISC expertise to properly map data; and 2) a shoddy infrastructure to create correct CDISC domain data sets. Once again, the results can be disastrous such that the deliverable is hardly CDISC compliant; and even worse, the clinical data are incorrectly mapped. CDISC expertise becomes vital when SUPPQUAL and RELREC data sets are required, which are often ignored. The proposed platform offers an environment that ensures CDISC compliance, of course assuming expertise in clinical data and CDISC standards.

CDISC domains are well defined, even with respect to the order of variables and the content of their respective labels. Thus, creating domain data sets is not just a matter of mapping raw data to appropriate domain variables. And, let's not forget those Expected, Required, and Permissible variables, along with their particular requirements. Also, the release of SDTM, Version 3.1.1 introduced new domains such as PC (PK Concentrations), as well as new variables for existing domains. In fact, the Findings class has almost doubled in size. To ensure compliance, the platform uses metadata that contains all the pertinent information about the domains in order to properly create domain data sets.

Once the metadata becomes part of the system (See Appendix C), there's an easy interface to map the raw data to prospective domain variables, including a mechanism for specifying rules for imputing values (e.g., USUBJID from SITE and PATID). Along with the mapping process, the platform uses the metadata to generate the SDTM programs that include all of the information needed to produce the domain data sets. Similar to the Report programs, there is one program for each domain data set, which is premised on a template program and the metadata.

Besides the two dozen plus standard SDTM domains, the platform allows the inclusion of non-standard domains for which respective programs can be generated. Once again, the platform uses metadata and a template program in order to generate the respective domain program. The contents of the program should be obvious: a well formed program header, a Data step that contains the beginnings of assignment statements, the proper variable labels, and even the data set label. Thus, the analyst / programmer working from specifications can produce domain data sets without being concerned about oversight. And, if a prospective domain variable (e.g., CMSTRF / CMENRF) is not germane to the study, the programmer can simply delete the assignment statement. Here again, consider the junior professional (in SAS or CDISC) who has the advantage of working with such a program at the onset.

Even though the platform enhances the certainty of creating appropriate domain data sets, it is always a good idea to validate such data sets. The platform is able to perform rigorous validation checks on the domain data sets, such as:

- Complete conformance to the metadata associated with domains, from the data set label, domain names, variable names, respective labels, and data attributes
- Primary key is unique
- Required variables must exist and cannot be null
- VISITNUM and VISIT are pair-wise appropriate
- Date variables are ISO-8601 compliant
- The value of the Sequence variable is unique

Appendix B contains the code for a macro that validates all CDISC date variables in a SAS data library.

Utility Macros

The platform would not be complete without a collection of utility macros that are available to the analyst / programmer. The following is a brief list of the kinds of tasks these macros perform. Two of these utility macros are found in Appendix B. These utility macros are accessed via the interface, which explains their purpose and proper use.

- Concatenation of RTF documents into a single Reports Document, including TOC.
- Pagination of standard SAS output in *“Page x of y”* format, as part of the header information.
- Validation of SDTM domains.
- Imputing ISO 8601 Dates from character variables containing partial date and/or time components.
- Producing a cross-reference of a clinical database with respect to variables and data sets.
- Producing a hardcopy of all or selected SAS programs.
- Using the latest version of other sources files (e.g., MedDRA, WHO).
- Creating transport files from a SAS data library
- Checking for consistent data type and length across data variables for XML.
- Producing Patient Profiles

Creating the CRT-DDS (define.xml) Document

In 1999 the FDA published a guide on electronic submissions (eSub) that specifies the document for describing the content and structure of data provided, as part of a submission, called the Data Definition Document (i.e., define.pdf). This document has since been supplanted by the Case Report Tabulation Data Definition Specification (CRT-DDS) document, known as “define.xml” which renders the information about the study data and, more importantly, provides an environment for referencing the domain data sets (SDTM, ADaM), as well as the annotated CRF.

CDISC has four main standards, thus far: the Study Data Tabulation Model (SDTM), the Analysis Data Model (ADaM), the Operational Data Model (ODM), and the Lab model (LAB). While SDTM, ADaM, and LAB pertain to the actual data, the ODM is a vendor neutral, platform independent format for the interchange and archive of data. For simplicity, this paper addresses SDTM and ODM to produce the Define XML document.

Given an annotated Case Report Form (aCRF) and a collection of SDTM domain data sets, along with the XML Style Sheet, the platform facilitates the inclusion of other pertinent information to build the Define XML document, such as information about the study and so-called Computation Methods. Other information pertaining to the domains and variables is attained directly from the SDTM domain data sets and related metadata about those domains (See Appendix C). Of course, prior to the creation of the Define XML document, the platform validates the domain data sets with respect to XML, that is, to ensure consistency with respect to data attributes (including upper and lower case). Also, the platform produces SAS Version 5 transport files from the domain data sets and produces a diagnostic report of this process.

Summary Points of Interest – Flexibility, Accessibility, Interoperability

The flexibility of this platform allows the use of external proprietary files, such as the Medical Dictionary for Regulatory Activities (MedDRA) and WHO (World Health Organization) Drug files other resources, even ad hoc information for converting units for lab data. And, besides SAS, the platform is able to use other applications software.

The Web-based environment allows access to the platform and client servers from virtually anywhere. Along with a reliable computing environment that has a consistent interface and file organization, a team of professionals scattered across the globe can work well together, even on multiple studies.

The scalability allows the simultaneous implementation of many studies. For example, consider a large Integrated Safety Summary (ISS) involving twenty-five legacy studies, which span over a decade. Thus, except for having a common therapeutic area of interest, the data in the studies might vary tremendously greatly. For example, some date / time variables might be bona fide SAS variables; whereas, other studies might have date parts and other studies might have character variables containing partial dates; yet, the dates must all become ISO 8601 compliant. Besides implementing the ISS, the secondary objective would be to convert all the data into SDTM domain data sets as part of the deliverable. Such a project would be a huge undertaking; yet, the platform would handle it easily, as if it were just another study.

Conclusion

The proposed Web-based platform is a computing environment designed for doing clinical trials. Obviously, however, the concepts presented here can be transferred to similar computing endeavors. In the realm of clinical trials, where the deliverable must be perfect and done expeditiously, this platform offers many advantages, from honestly knowing the status of the project to implementing methods that ensure integrity of the deliverable.

The proposed platform goes beyond the scope of a given project for a particular client at one site. Indeed, this platform allows the means to work on multiple projects by a team of professionals spread across the globe. And, needless to say, it's a small world and it's about time.

References

Adams, John H. and Kirill Tchernakov. *CDISC for Electronic Submission – A Table Translation Program.* PharmaSUG Proceedings, 2005.

Brucken, Nancy and Paul Slagle. *Mapping Clinical Data to a Standard Structure: A Table Driven Approach.* PharamSUG Proceedings, 2006.

CDISC Standards (http://www.cdisc.org/standards/).

Friebel, Anthony, et al. *Defining and Validating CDISC Data Standards to XML in SAS Technology*. PharmaSUG Proceedings, 2004.

Gerlach, John R. *A Relational Understanding of SDTM Tables.* SESUG Conference, 2006.

Gerlach, John R. *Imputing ISO8601 Dates From Partial Dates.* PharmaSUG Proceedings, 2006.

Gerlach, John R. *Validating Data Using SAS Formats as a Programming Tool.* PharmaSUG Proceedings, 2005

Gerlach, John R. *Generating N Shell Programs for N Reports in a Clinical Trial.* PharmaSUG Proceedings, 2004.

Kenny, Susan J. *Strategies for Implementing SDTM and ADaM Standards.* PharmaSUG Proceedings, 2005.

Kihullen, Michael. *Implementing CDISC Data Models in the SAS® Metadata Server.* PharmaSUG Proceedings, 2006

Palmer, Michael and Julie Evans. *Testing CDISC's Operational Data Model in SAS.* PharmaSUG Proceedings, 2003.

Palmer, Michael. *It's Easy If You Know How: Importing, Processing, and Exporting CDISC XML with SAS®.* PharmaSUG Proceedings, 2002.
Peterson, Don and John R. Gerlach. *User-Specified Text Flow Inside the Report Procedure.* SUGI, 1999.
Russell, Chan and Wayne Kubick. *Overview of Techniques for Reading and Writing ODM Data.* CDISC White Paper, June 2001. (www.cdisc.org/publications/index.html).
SAS Institute Inc. 2004. *The CDISC Procedure for SAS® Software, Release 8.2 and Later.* Cary, NC: SAS Institute Inc.
Shostak, Jack. *Implementation of the CDISC SDTM at the Duke Clinical Research Institute.* PharmaSUG Proceedings, 2005.

Trademark Citation

Contact Information

Maulik Shah; VP Global Sales & Delivery
mshah@maxisit.com

John R. Gerlach; Senior Developer
jgerlach@maxisit.com

MaxisIT, Inc.
1199 Amboy Avenue, Suite 2H
Edison, NJ 08837
732-205-1717
877-MAXISIT
www.maxisit.com

Keywords: Interoperability, CRT-DDS, XML, CDISC, SDTM.

Appendix A – The Shell Program

```
/* ====================================================================================
   Program   :  <Repid>.sas

   Author    :
   Written   : ddmonyyyy

   Validator :
   Validated : ddmonyyyy by

   Client    : <Client>
   Study     : <Study>

   Purpose   : Produce table.
   Report    : <Title>

   Input     : DERIVED.DM
   Output    : Report

   Macros    : %hdrftr – Generates titles and footnotes.

   Notes     :

   ==================================================================================== */

/* ------------------------------------------------------------------------------------
   Initialize environment.
   ------------------------------------------------------------------------------------ */

   %let repid = <Repid>;
   %include “&projid.\Programs\Utility\Init.sas”;

/* ------------------------------------------------------------------------------------
   Define formats.
   ------------------------------------------------------------------------------------ */


/* ------------------------------------------------------------------------------------
   Define population (e.g., Safety, ITT).
   ------------------------------------------------------------------------------------ */

 * proc sort data=derived.dm(keep=usubjid safety);
 *    by usubjid;
 *    where safety eq ‘Y’;
 * run;

/* ------------------------------------------------------------------------------------
   Create the analysis data set.
   ------------------------------------------------------------------------------------ */


/* ------------------------------------------------------------------------------------
   Perform the analysis.
   ------------------------------------------------------------------------------------ */


/* ------------------------------------------------------------------------------------
   Create the reporting data set.
   ------------------------------------------------------------------------------------ */


/* ------------------------------------------------------------------------------------
   Produce the report.
   ------------------------------------------------------------------------------------ */

 * proc report data=rdset nowindows headline headskip split=’!’;

 *    %hdrftr(&repid.) ;
 * run;

/* ==================================================================================== */
```

Appendix B – Utilities

Create Transport (XPT) Files

```
%macro xpt(model,loc);
   libname &model. "&loc.\Data\&model.";
   proc sql noprint;
      select count(memname) into :nds
         from dictionary.tables
         where libname eq "%upcase(&model.)" and length(memname) eq 2;
   quit;
   %let nds = &nds.;
   proc sql noprint;
      select memname into :ds1 - :ds&nds.
         from dictionary.tables
         where libname eq "%upcase(&model.)" and length(memname) eq 2;
   quit;
   %do i = 1 %to &nds.;
      %put ==========>>> Creating Transport File #&i.: %upcase(&&ds&i..);
      libname xpt xport "&loc.\Data\&model.\&&ds&i...xpt";
      proc copy in=&model. out=xpt memtype=data;
         select &&ds&i..;
      run;
      %end;
   libname xpt clear;
%mend xpt;
```

Validate ISO 8601 Dates

```
%macro qc_dates;
   %do i = 1 %to &ndoms.;
      proc sql noprint;
         select name into :vars separated by ' '
            from date_vars
            where domain eq "&&dom&i..";
      quit;
      %put ====>>> Domain: &&dom&i.. / &vars.;
      data &&dom&i..;
         length element $4;
         array dates{*}$19 &vars.;
         set sdtm.&&dom&i..(keep=&vars.);
         do i = 1 to dim(dates);
            token = 1;
            dates{i} = translate(dates{i},':','T');
            do while(scan(dates{i},token,'-:') ne '');
               select(token);
                  when(1) if input('01JAN'||substr(dates{i},1,4), ?? date9.) eq .
                     then error = 1;
                  when(2) if input(substr(dates{i},1,7)||'-01',?? yymmdd10.) eq .
                     then error = 1;
                  when(3) if input(substr(dates{i},1,10), ?? yymmdd10.)      eq .
                     then error = 1;
                  when(4) if input(substr(dates{i},1,13),?? datetime13.)     eq .
                     then error = 1;
                  when(5) if input(substr(dates{i},1,16),?? datetime16.)     eq .
                     then error = 1;
                  when(6) if input(substr(dates{i},1,19),?? datetime16.)     eq .
                     then error = 1;
                  otherwise;
                  end;
               token = token + 1;
               end;
            if length(dates{i}) ge 11
               then substr(dates{i},11,1) = 'T';
            if error
               then output;
         end;
       * drop i token error;
      run;
      proc print data=&&dom&i.;
         title2 "Non-Compliant ISO8606 Date Variables";
         title3 "( Domain: %upcase(&&dom&i..) )";
      run;
      %end;
%mend qc_dates;
```

Appendix C – CDISC Meta Data with Partial AE Domain

Domain	Name (e.g., AE, DM, VS)
Class	Domain Class (e.g., Events, Findings)
Name	Variable Name
Sequence	Order of Variable in Domain Data Set
Type	Data Type (Character, Numeric)
Label	Variable Label
Origin	Origin of Variable
Role	Purpose of Variable (e.g., Identifier)
Core	Required, Expected, or Permissible
Terms	Values, Standards (e.g., ISO8601)

Domain	**Name**	Type	Class	Seq	Core	Label
AE	STUDYID	Char	Events	1	Req	Study Identifier
AE	DOMAIN	Char	Events	2	Req	Domain Abbreviation
AE	USUBJID	Char	Events	3	Req	Unique Subject Identifier
AE	AESEQ	Num	Events	4	Req	Sequence Number
:	:	:	:	:	:	:
AE	AETERM	Char	Events	8	Req	Reported Term for the AE
AE	AEMODIFY	Char	Events	9	Perm	Modified Reported Term
AE	AEDECOD	Char	Events	10	Req	Dictionary-Derived Term
AE	AECAT	Char	Events	11	Perm	Category for Adverse Event
AE	AESCAT	Char	Events	12	Perm	Subcategory for Adverse Event
AE	AEOCCUR	Char	Events	13	Perm	Adverse Event Occurrence

Domain	**Name**	Origin	Role	Terms
AE	STUDYID	CRF	Identifier	
AE	DOMAIN	Derived	Identifier	AE
AE	USUBJID	Sponsor Defined	Identifier	
AE	AESEQ	CRF or Derived	Identifier	
:	:	:	:	:
AE	AETERM	CRF	Topic	
AE	AEMODIFY	Sponsor Defined	Synonym Qualifier	
AE	AEDECOD	Derived	Synonym Qualifier	
AE	AECAT	Sponsor Defined	Grouping Qualifier	
AE	AESCAT	Sponsor Defined	Grouping Qualifier	
AE	AEOCCUR	CRF or Sponsor	Result Qualifier	Y, N or Null