Paper 129-26

Preparing the SAS® Software Programming Environment for Regulatory Submission

Sunil K. Gupta, Gupta Programming, Simi Valley, CA

ABSTRACT

For pharmaceutical companies, there is a push to prepare SAS® programs for clinical study reporting and analysis. By utilizing a clinical data warehouse design strategy, SAS® programs can be set up in advance of the study completion to facilitate the generation of the final study analysis, tables, listings, and graphs.

Issues in data warehouse structure and technology and industry standards will be discussed. This paper reviews the process of establishing a standard method to prepare the SAS® software programming environment (SAS/BASE®, SAS/ACCESS®, SAS/AF®, SAS/FSP®, SAS/STAT®, SAS/GRAPH®) for regulatory submissions. By incorporating best practices in the data collection, data entry, data cleaning and data reporting of clinical studies, the generation of the final end result will be more efficient. In addition, because quality assurance is a key component throughout the system, pharmaceutical companies have greater control of the quality and completeness of the regulatory submission. The intended audience for this presentation is the intermediate to advanced SAS® user.

The paper is divided into the following sections for review: project objectives and milestones, data warehouse design issues, clinical data management, system documentation and standards, statistical analysis plan, and regulatory submission.

PROJECT OBJECTIVES & MILESTONES

The purpose of a clinical study needs to be understood by everyone involved. The research, analysis and reporting tasks will require total involvement and understanding of everyone on the team. The team members will need to communicate their requests to each other in terms of understanding each other's strengths and responsibilities. Typical team members include data experts such as physicians and medical writers, and programming experts such as biostatisticians and SAS programmers. The data experts know what they want and rely on the programmers to create the tables and analysis for review.

SAS programmers often have the responsibility of supporting the reporting requirements of Clinical Affairs and other departments of pharmaceutical companies. This involves interacting with department members to define the report. The SAS programmer's task is to understand the request and to design the program to achieve the desired outcome. Without some method to communicate what is available and what can be accomplished, this task can be difficult to complete.

A SAS Service Request Form should be developed to meet the needs of the department and reflect the type of information available for reporting. The form should be organized to guide the customer through a series of questions. Sections on the form should include whom the request is from and the required date, description of the SAS request with the purpose listed, selection criteria to identify the population and time period of the report, and the format and organization of the report along with the method of output desired. Finally, the SAS programmer should log the completion date and time.

Proper tools such as a SAS Service Request Form should be in place to facilitate the communication and documentation of the customer's requirement and the customer's expectations of the programmer. Establishing standards in service requests will increase both efficiency and customer satisfaction.

Typically, the SAS programmer will need to generate a variety of reports to fulfill the regulatory requirements of the clinical investigation. By establishing clinical reporting templates for each of the functional aspects of Clinical Data Management, the SAS programmer can dramatically improve the efficiency of developing and generating reports. Examples of reporting templates include patient listings, summary tables and graphs. Each study would use the same reporting template to generate patient listings, summary tables and graphs.

With appropriate meetings and project status updates, significant milestones can be achieved and monitored. Working closely with the FDA reviewer from the beginning to identify and plan the course of action will facilitate the project schedule and completion.

DATA WAREHOUSE DESIGN ISSUES

The objective of the data warehouse is to develop code to map data from the raw clinical data through the SAS Views, SAS Raw data sets, SAS Integrated data sets and SAS CRT (Case Report Tabulation) data sets to generate a quality, reproducible, error-free FDA submission.

<u>Clinical Data Process Flow</u>

↓ Raw Clinical Data (Informix/Oracle)
  ↓ SAS Views
    ↓ SAS Raw Data Sets
      ↓ SAS Integrated Data Sets
        ↓ SAS CRT (Case Report Tabulation) Data Sets

The following table quantifies the number of data sets and program files that are typically required in an FDA submission:

Typical FDA Submission Statistics

Number of hardcopy pages = up to 1 million pages

Number of clinical studies = 10
Number of directories per clinical study = 10
Number of people involved = 16 (5 Programmers, 4 Statisticians, 3 Medical Writers, 3 Clinical Data Managers, 1 Regulatory Manager)
Time from start of study to FDA submission = 1 ½ to 2 years (Depends on length of longest study)
Time from database lock to FDA submission =3 weeks to 1 year (3 weeks assumes all work completed in advance)
Number of Raw data sets = 100
Number of Raw data set creation programs = 100
Different types of Raw data structure = 16

Number of Integrated data sets = 66
Number of Integrated data set creation programs = 66

Number of CRT data sets = 40
Number of CRT data set creation programs = 40

Number of macro programs = 100
Number of reporting and analysis programs = 55
Number of tables, listings and figures = 200

The task is to consider all the expectations of the clinical data system in order to plan for it. The SAS Data Warehouse system enables features for the management, organization and exploitation of clinical data. Through this process, clinical data can be turned into a quality controlled FDA submission. The benefits of establishing a data warehouse structure include: access to data in timely manner, quality assured data, integrated and consistent data, easily accessible data, and data exploration and discovery.

Database design strategy should be optimal and robust. It should be able to process a variety of clinical datasets, be a flexible and integrated system, enforce system standards and facilitate required CRT data structure.

Different types of data are collected and processed as a snapshot to provide a single view of the data. All the various sources of the data must be correctly related to each other for accurate reporting and analysis. From knowing that all the data from each study must be pooled together for the Integrated Summary of Safety and the Integrated Summary of Efficacy sections, steps should be taken to ensure a smooth convergence. Establishing system standards in all aspects will facilitate structured output data sets. Efficient database maintenance is achieved through standards, quality control and proper documentation. The macro programming language helps to centralize the code for standard reports and analysis of similar studies. Changes to the original specification should be expected and planned for by facilitating the addition of new variables and data entry codes as well as performing complicated queries. Where possible, differences between studies should be accounted for to prevent loss of information.

Functional specifications of the CRT datasets may include the following: creation of the demog crt dataset, merge of all crt datasets with the demog crt dataset, use of vertical file structure if applicable for multiple occurrence of data items at different time points. Having the key demog variables available in all crt datasets eliminates the need to merge with the demog crt dataset for data process and analysis by key group variables. The advantages of the vertical file structure include: a single variable stores the name of the various types of data stored, can use a non-specific variable name for process, and more efficient programming due to multiple records instead of multiple variables. It may be necessary to keep a corresponding horizontal file structure for other types of analysis.

Other analytical datasets may be required to contain summary level of information by patient and visit date for by visit analysis. The crt datasets are expected to be both detail and summary level datasets by key variables.

Data extraction involves the direct mapping of raw source variables and data into SAS raw data sets. It is important to realize that different types of data need to be correctly related to each other to accurately reflect the patient's clinical data. Ideally, a SAS standard dictionary is used to enforce consistency in SAS variable names, type and length across studies.

Raw Clinical Data

Vitals AE
Adminlog Antibody
Diagnose Hospital
Demo
Random Terminat
Followup History
Conmeds Lab

Once data is extracted from the clinical database, it must be transformed before loading into the data warehouse. Transforming the data involves data validation, data scrubbing, data integration, data structuring, time handling, denormalization and data summarization.

Key to Data Warehouse Tasks

Data Validation	Assure programs function as specified
Data Structure	Create new variables/modify existing variables
Integration	Achieve consistency by standardization
Scrubbing	Recode or removal of invalid data

The table below outlines the details of the steps involved in the data transformation and data checks:

Data Validation

- Integrity of data dictionary datasets
- Same number of observations
- Control of variable selection
- Correct mapping of data values

Data Structure

- Impute partial dates
- Create new variables
- Drop unwanted variables

Integration of studies & data sets

- Standardize dataset name, variable name, variable attributes
- Standardize to numeric codes with formats
- Standardize to binary response variables

Scrubbing

- Process multiple records per patient to single record per patient as needed
- Transpose data as needed

Building intelligence into the system requires the utilization of metadata. By accessing information about the content and structure of the clinical data, more robust and automated systems can be developed to perform data processing tasks.

In the area of data exploitation, advance tools can be incorporated to view, analyze and report clinical data for better decision-making. With SAS's new ODS, it is much easier to create rtf, html and SAS data sets from any procedure.

CLINICAL DATA MANAGEMENT

For large companies, another department may be responsible for the data collection and entry, data editing and data cleaning of clinical studies. Where possible, additional methods and programs should be developed to confirm the quality of the data received and analyzed. It is important to discover as soon as possible if any invalid values have been entered into the data set. In addition, statistical and clinical study assumptions may not be correct and need to be verified. At least, the format of coded values must be confirmed to assure correct reading of coded values. Ideally, a data validation manual should be prepared to define all data checks to be performed.

For accessing Informix or Oracle views, the SAS/ACCESS module can be used to create standard SAS views. You may also want to consider SAS/IntrNet capabilities. For using a SAS based data entry system, the following SAS modules can be utilized to create a quality control entry system: SAS/AF, SAS/FSP. Screen Control Language (SCL) allows you to add any logic and field validation for data entry.

Data Validation Plan

- Logical checks - variable level, date consistency
- Check for duplicate records
- Check for required variables
- Check for unique key variables

SYSTEM DOCUMENTATION AND STANDARDS

For each clinical study, system documentation and programming standards should be a requirement. A naming convention should be utilized for all data sets, variables, formats, macro variables and macro programs. By defining a naming system from a global perspective at the start, all documentation and program development will need minimum update at a later stage. Each programmer should have available a code book containing data set contents, sample proc prints and key to all formats.

Good directory structure to store and access data sets and programs is essential for good communication and understanding in a multi-user environment. Considerations should be made for separating raw data from SAS data sets and SAS programs from format libraries. This is a good time to get input from all programmers and statisticians in regards to data set structure and access. Depending on what the FDA reviewer requests, the preferred data file structure for many data sets may be a horizontal file structure to facilitate analysis and review.

Often multi-users will be required to complete all the necessary programming in the time allocated. A central location of programs and method of access allows for a shared environment. Utility macro programs can be used to create data sets from views, provide Proc CONTENTS and sample listing, and Proc FREQ of key categorical variables. A macro library should be established. Having a good and sensible naming convention throughout the process will go a long way to improve the development and maintenance of programs. By designing a modular system, code can be reused by other clinical studies with minimum effort. Time efficiency can be realized. The concept of best practices should be exercised where possible.

A single statistical analysis file can be created from all significant data sets for the study. This data set will contain all the primary and secondary measurements along with demographics and safety information. When doing integrated summary analysis, a similar setup can be utilized. By standardizing at the study level, the integration process becomes much easier. The alternative is to compare each variable for each data set across all studies to assure consistent variable name and range of values before combining all studies. By taking a systematic approach, macros can be written to standardize individual studies into common data sets to be combined into a single set of data sets. The function of these standardization macros would be to recode, rename, keep, drop and assign variables as required for each study.

Several good programming methodologies and strategies include having the relative path in the libname statement to facilitate upward scalability and portability, to archive data sets as backups, and to execute SAS in batch mode to save listing and log files by the same name. Proc DATASETS with the AGE statement can be used to archive data sets as backups. Defining macro variables to be used in footnotes facilitates the program identification, execution date and the path of SAS program. Many of these things can be established in the initialization program.

The tables below outline the advantages and disadvantages in using the two strategies: mass production of programs and set of central macros. The optimal method is a hybrid of both strategies because the advantages of both methods can be utilized.

Software Development	**Option 1:**

Life Cycle	Mass Production of Programs
a. Documentation	Need to assure consistent documentation
b. Development	Simple, straight forward programming
c. Testing	Simple, straight forward testing
d. Maintenance	Low, need to assure all updates are completed

Software Development Option 2:

Life Cycle	Set of Central Macros
a. Documentation	Need to document all features and options.
b. Development	More difficult, time consuming, complex
c. Testing	More thorough, all inclusive
d. Maintenance	High, difficult to understand and to update

Software Development Hybrid Option:

Life Cycle	Programs with Central Macros
a. Documentation	Need to document all features and options
b. Development	Less difficult, time consuming, advanced
c. Testing	More thorough, all inclusive
d. Maintenance	Medium, need to understand and to update

The ideal method is to have a single controller file to call one or more macros, which may call more macros as determined by the task-oriented macros. If possible, do not have more than 3 levels of macro calls. This recommendation will facilitate documentation, development, and testing of the macros.

Level of Macro Calls

```
        Controller File
(1)     → Macros: A, B, C
(2)             → Macros: A1, A2
(3)                     → Macros: A1a, A1b
```

The object is to establish a main source code library with the following design items listed below. This will allow for more robust programming and greater utility.

Macro Source Code Library

- ♦ Be responsive to user's needs
- ♦ Design modular macros
- ♦ Design self contained macros
- ♦ Design robust macros
- ♦ Keep macros easy to read
- ♦ Use keyword parameter
- ♦ Do not overparameterize
- ♦ Make parameters easy
- ♦ Check passed parameters
- ♦ Supply default parameter values
- ♦ Supply a test parameter
- ♦ Avoid hard coding with macros
- ♦ Write messages to the SAS log as needed

Directory Structure

View	SAS View to Informix
Raw	SAS Raw Data sets
Integrated Data sets	Integrated SAS Data sets
CRT Data sets	CRT SAS Data sets
Programs	Analysis Programs
Catalogs	Format Library
Dev	Program Development Directory
Output	Tables, Lists & Graphs
ISS	Integrated Summary Safety
ISE	Integrated Summary Efficacy

Task-Oriented Macros

I. Setup Macros
II. Standardization Macros and Programs
III. Utility Macros
IV. Data Management Analysis Macros
V. Summary Level Analysis Macros
VI. Report Output Layout Macros
VII. Report Output Macros

Macro Variables

%let prot = A01	Protocol number (A01, A02, A03)
%let pgm = demog	Program name (demog, ae)
%let in_dsn = demog	Input data set (demog, ae)
%let dsn = demog	Data set name (demog, ae)
%let dscode = dm	Data set code (dm, ae, cm)
&sysdate	System date

I. Setup Macros

%init	Initialize program
%c_view	Create sas view
%c_rdsn	Create raw data set
%c_frm	Create format library
%df_ae	Define AE evaluable population
%df_eff	Define Efficacy population

Initialize Program - init.sas

```
%macro init( prot   /*   Protocol Number   */ );

options pagesize = 59 linesize=130
sasautos=('c:\drugA\catalogs\macros');

%global protn prgloc;

%let protn = %trim(%left(&prot));
%let prgloc = "c:\drugA\&prot.\programs";

libname v&prot    "c\drugA\&prot.\views";
libname r&prot    "c\drugA\&prot.\raw";
libname i&prot    "c\drugA\&prot.\integrate";
libname c&prot    "c\drugA\&prot.\crt";

libname library    "c\drugA\catalogs";

Proc FORMAT library = library;
quit;

%mend init;
```

Define AE evaluable population macro - df_ae.sas

```
%macro df_ae;
```

```
Proc SORT data=i&protn..eval
 out=ae (keep = inv pat trtgroup);
 by pat;
 where ae = 1;
run;

%mend df_ae;
```

II. Standardization Macros and Program

i_demog.sas	Create integrated demog data set
%dm_atr	Define variables with attribute statements for demog data set
%dm_keep	List of variables to keep in data set
%dm_renm	List of variables to rename into standard variables in demog
%dm_rcod	Recode variable in demog

Create demog data set - i_demog.sas

```
%include '..\catalogs\init.sas';

%init(prot=A01);

%let pgm=demog;
%let in_dsn = demog;
%let dsn=demog;
%let dscode=dm;

data i&protn..&dsn;

 %&dscode._atr;
 set r&protn..&in_dsn.(rename=(%&dscode._renm));
  pat = input(patno, 4.);
  keep %&dscode._keep;

run;
```

Define variables with attribute - dm_atr.sas

```
%macro dm_atr;

 attrib pat length = 8              label='Participant Number';
 attrib site length = 8                     label='Site Identifier';
 attrib aeany length=8 format=yn.  label='Any Aes Occur';

%mend dm_atr;
```

List of variables to keep in data set - dm_keep.sas

```
%macro dm_keep;

 ptid prot site ptnum sex race

%mend dm_keep;
```

III. Utility Macros

%dsn_wk	Create working data set from sort
%dsn_cn	Proc CONTENTS
%dsn_pt	Proc PRINT - sample listing
%dsn_uni	Proc UNIVARIATE
%dsn_tab	Proc TABULATE
%dsn_frq	Proc FREQ
%dsn_dup	Check for duplicate records
%ddl	CRT by Study

Working data set - dsn_wk.sas

```
%macro dsn_wk( dtyp = r,       /*  Directory type  */
                    ds = demog   /*  Data set name  */
);

 Proc SORT data=&dtyp.&protn..&ds out=&ds;
  by pat;
 run;

%mend dsn_wk;
```

Proc Contents - dsn_cn.sas

```
%macro dsn_cn( dtyp = r,       /*  Directory type  */
                    ds = demog   /*  Data set name  */
);

 Proc CONTENTS data=&dtyp.&protn..&ds;
 run;

%mend dsn_cn;
```

Proc Print - dsn_pt.sas

```
%macro dsn_pt( dtyp = r,       /*  Directory type  */
                   ds = demog   /*  Data set name  */
);

 Proc PRINT data=&dtyp.&protn..&ds (obs=10) label;
 run;

%mend dsn_pt;
```

CRT By Study - ddl.sas

```
Ddl.sas;

%include '..\catalogs\init.sas';

%init(prot=A01);
%init(prot=A02);
%init(prot=A03);

Data ccolumn;
 set sashelp.vtable;
 c_ds= memname;
 study = substr(libname, 2, 6);
 keep study c_ds nobs;

 if substr(libname, 1, 1) = 'C';
run;

Proc SORT data=ccolumn;
 by c_ds;
run;

Proc TRANSPOSE data=ccolumn
                    out=tcstatus (drop = _name_ _label_);
 by c_ds;
 id study;
 var nobs;
run;

title 'Existing CRT Datasets';
Proc PRINT data=tcstatus;
 var c_ds _A01 _A02 _A03;
run;
```

STATISTICAL ANALYSIS PLAN

A good idea for clinical reporting and analysis is to create a single statistical analysis data set. This single data set will contain all key demographic information along with all significant efficacy and safety parameters. Variables from demographic and follow-up data sets can be merged by patient and visitdate to create a single statistical analysis file.

Understanding and using latest technology and industry standards are important for effective reporting intelligence tools. Reporting intelligence tools include data-driven data processing and reporting, on-line references to documentation, e-mail communication, and web publishing tools.

Incorporate data-driven reports where layout and structure of the report is determined by the attributes of the input data set (variable labels, formats and lengths). Programs should access the SQL data dictionary tables for automated decision processing. On-line reference to documentation includes dataset contents, format catalog, and program and macro listing. Tools to automate the documentation of program testing and verification and data set definition table eliminate the need to retype this information in another format. Using e-mail technology to better communicate timely information to team members empowers everyone to stay informed and be proactive. Monitoring the e-mails of program status and error reports controls the quality of the output generated. Web publishing tools from SAS's new ODS features enable the creation of rtf and html files with minimum effort.

Utilizing a consistent method for generating reports methods makes good sense. Typical methods to consider include Data *null*, Proc REPORT, Proc TABULATE, and Proc SQL. The table below lists some benefits and features for each approach. A complete list and review of these reporting methods can be found in the Observations - The Technical Journal for SAS Software Users - First Quarter 1994 - Writing Reports with SAS Software. What are your options? page 10-42.

Reporting Approach	
Data _Null_	- For complete control and customized reports
Proc REPORT	- Produces organized output for review
Proc TABULATE	- Produces multi-dimensional tables with descriptive statistics in a finished tabular format
Proc SQL	- Can combine data from several data sets to build the report but does not have much control over the output format

Two important components of any drug approval application are table listings that contain all patient information and table summaries that describe the safety and efficacy of the drug. A SAS procedure such as PROC SQL provides many complex operations and options for generating these results.

In the final step of program completion, program validation and testing should be performed for quality assurance. Test cases should be identified and tested to assure complete accountability. It is very important to first define what the expected results are before the tests are performed. A key objective is to check for consistency of numbers throughout all tables. For optimal performance, it is often best for another programmer to verify the program of the original programmer. A system should be defined to migrate tested and quality assured programs to the production library.

The migration process from development to production ensures that only documented and tested code is utilized by all team members for quality and reliable reporting. Quality Assurance and Edit Check Listings serve to confirm the logic of the program and the quality of the data before it is processed.

Migration Process

- Development and Testing
- Quality Assurance
- Production Library

Edit Check Listings

- Review baseline variables
- Identify any missing key variable
- Confirm dates are logical
- Descriptive statistics on all continuous variables
- Frequency Counts on key categorical variables

IV. Data Management Analysis Macros	
%trans	Transpose data set
%comp	Compare two data sets
%dates	Compare two dates

Test Plan and Log

Test Item	Test Performed	Expected Result	Observed Results

Error Code	**Specification**
Demo_i005	Sex not M or F
Demo_i006	Race not W, B, O, H or X

V. Summary Level Analysis macros

Descriptive Statistics - n, mean, sd, min, max, sum

%prop	General Proportions macro
%cont	Continuous Variables

Statistical Analysis	
%ttest	T-test
%chist	Chi-Square test

General Proportions Macro - prop.sas

```
%macro prop( ds ,      /*  Data set name  */
                  labl,    /*   Line label */
                  xvar,    /*  Variable of analysis  */
                  outfl    /*  Output file name  */
);

 Proc MEANS data=c&protn..&ds noprint;
  by tx;
  var &xvar;
  output out=tmp1 n=n mean=mean sum=sum;
 run;

 Data _null_;
  file &outfl notitles mod ls=180;

  obsnum=1; set tmp1 point=obsnum;
  n1 = n; pct1 = mean*100; sum1=sum;

  obsnum=2; set tmp1 point=obsnum;
  n2 = n; pct2 = mean*100; sum2=sum;

  put @1 &label @17 sum1 5.  ' / ' n1 2. pct1 6.1 '%'
                              sum2 12. ' / ' n2 2. pct2. 6.1 '%' @;
  stop;
 run;

%mend prop;
```

VI. Report Output Layout macros

%header	Header information
%footer	Footer information
%line	Line output
%colcnt	Column counter for put statements

Footer information - footer.sas

```
%macro footer;

 tday = date();
 put @&c1
'_____________________________________________';
 put / @&c1 "Directory: &prgloc. File: &pgm.  " tday
worddate.;

%mend footer;
```

VII. Report Output macros

%t_demog	Demog Table in rtf format
%t_vitals	Vitals Table in rtf format
%l_demog	Demog Listing in html format

Demog Table – t_demog.sas

```
%macro t_demog;

%prop(ds=demog, labl=Age, xvar= age, outfl=demog.txt);

ODS rtf file = 't_demog.rtf';

Title 'Patient Demographics – Age By Tx Table';
Proc PRINT data=tmp1;
 Var tx n mean;
Run;

ODS rtf close;

X "/usr/bin/mailx –s 'DEMOG Table' sgupta < t_demog.rtf";

%mend t_demog;
```

Demog Listing – l_demog.sas

```
%macro l_demog;

ODS rtf file = 'l_demog.rtf';

Title 'Patient Demographics – Listing';
Proc PRINT data= demog label;
 Var patient tx age sex wt hgt;
Run;

ODS rtf close;

X "/usr/bin/mailx –s 'DEMOG List' sgupta";

%mend l_demog;
```

With configuration management, system and program updates can be performed in the production environment. A good reference for documentation and software development is "Taming the Chaos: A Primer on the Software Life Cycle and Programming Standards". This paper does a good job in outlining the benefits and method to document the Software Development Life Cycle. A good reference for software validation is "Software validation for the rest of us". The paper does a good job in explaining the methods and reasons for doing a correct validation.

REGULATORY SUBMISSION

Ultimately, SAS data sets and files will be prepared for the FDA submission. FDA has provided updated guidelines outlining the directory structure, general rules, SAS programs, SAS data sets, list and log files. In addition, tables and listings may need to be saved as Rich Text Formatting (RTF) files and HTML files for ease of review by Microsoft Word and an Internet Browser. For example, the general considerations for data sets include the following: all data sets to have a unique identifier for each patient in the study, all variable names and codes to be consistent across all studies, and all variable formats to be of similar type within and across all studies. In addition, several key grouping variables such as treatment group and sex may be required in each data set to facilitate analysis. Where possible, macros that generate RTF files should be utilized to automate the process.

The significance of empowering the FDA reviewer with PDF files that allow for drill-down by hyper-text features and navigation tools with user instructions will greatly improve the review process. In fact, in the guidelines, it states to provide a hypertext link from the listing of the file to the SAS transport file. Study definitions, data set definitions, variable definitions, program index, and format definitions are additional required items. If possible, write SAS programs that automatically create and index PDF files for faster processing.

As part of the FDA move toward a paperless regulatory submission, the FDA has proposed using Version 5 SAS Transport file format as a standard for electronic data submission and archival. The SAS Version 5 Transport file format provides a mechanism for movement of data between different computer types and operating systems. This ensures the long-term availability of submission data. The data set files should not exceed 25 MB per file. Each transport file should be saved as an individual file representing the SAS data set.

The following is a list of all items required for on-line documentation:

- Study (protocol) Definitions
- User Instructions
- File Definitions
- Variable Definitions
- Data set List
- Program List
- Listings List
- Tables List
- Outputs - Listings and Tables

Study Definition

Protocol	**Description of Study**
DRUG001	Safety
DRUG002	Phase 2 double blind, randomized, placebo-controlled, dose escalation study

File Definition

Dataset Name	**#of obs**	**Filename Description**
AE2	5833	The AE2 data set contains all non-missing adverse event records with selected demographic and treatment group variables. It has one record per participant per adverse event.

Data set AE2

Description:
The AE2 data set contains all non-missing adverse event records with selected demographic and treatment group variables. It has one record per participant per adverse event.

Variable Definition

Variable	**Sort**	**Length**	**Format**	**Label Description**
Ptid	1	$ 13	.	Participant ID
Aeact	2	8	Aeact.	Action Taken
Aeany		8	Yn.	Any AE Occur

Formats

AEACT	**YN**
1='None'	1='Yes'
2='Discontinued'	2='No'

Notes
1. Sort Keys: Ptid, Aeact.
2. The COSTART dictionary was used for the AE preferred terms and the AE body system terms.

SUMMARY

Before starting to write individual programs to produce output for regulatory submission, it is wise to take a systematic global perspective of all the resources available and the best strategy to achieve the outcome. Advance preparation of SAS® programs in Clinical Data Management, statistical analysis and quality control will facilitate the generation of all required output files. By including everyone from the beginning, a greater understanding of the submission objectives can be achieved.

SAS offers the features of a relational database model needed to create an efficient Clinical Data Warehouse System. In addition, there are numerous tools including Proc SQL to facilitate the generation of reports and analysis. Programs can be utilized to build flexibility around a structured system.

TRADEMARK INFORMATION

SAS® is a registered trademark of the SAS Institute Inc., Cary, NC, USA.

http:/www.sas/service/techsup/sas_xport.html

Gupta, Sunil K., Gupta Programming (1995), "Designing Clinical SAS Service Request Forms", WUSS.

Gupta, Sunil K., Gupta Programming (1995), "Utilizing Clinical SAS Report Templates", WUSS.

Gupta, Sunil K., Gupta Programming (1996), "Database Design Strategies in CANDAs", PharmaSUG.

Wilson, Steve A., Kaiser Permante Division of Research, "Developing a SAS System Autocall macro library as an effective toolkit"

Carol Linden and John E. Green III, (1994), "Writing Reports with SAS Software. What are your options?" Observations: The Technical Journal for SAS Software Users - First Quarter, 10-42.

C. Michael Whitney (1996) "Taming the Chaos: A Primer on the Software Life Cycle and Programming Standards", Observations: The Technical Journal for SAS Software Users - Fourth Quarter, 15-21.

Harris, Michael, Amgen Inc. (1998), "Software validation for the rest of Us", WUSS.

DiIorio, Frank, Advanced Integrated Manufacturing Solutions, Co. (1998), "The Elements of SAS

Programming Style”, WUSS.

Iza Peszek, Cindy Song, Olga Kuznetsova, Merck & Co (1999), “Producing Tabular Reports in SAS® Systems in the Form of MS Word® Tables”, PharmaSUG.

Guidance for Industry - Providing Regulatory Submissions in Electronic Format - NDA, U.S. Department of Health and Human Services, Food and Drug Administration, Center for Drug Evaluation and Research (CDER), IT 3 January 1999
http://www.sas.com/fda-esub
http://www.fda.gov/cber/guidelines.htm

ABOUT THE AUTHOR

The author welcomes your comments & suggestions.

Sunil K. Gupta
Gupta Programming
SAS Institute Quality Partner™
213 Goldenwood Circle, Simi Valley, CA 93065
Phone: (805)-577-8877
E-mail: Sunil@GuptaProgramming.com
http://www.GuptaProgramming.com

Sunil is a senior consultant at Gupta Programming. He specializes in SAS/BASE®, SAS/AF®, SAS/FSP®, SAS/STAT® and SAS/GRAPH®. His consulting projects with pharmaceutical companies include the development of a Clinical Study Data Entry System, a Macro-Based Application for Report Generation, and customized plots and charts with SAS/GRAPH®. He has been using SAS® software for over 10 years and is a SAS Institute Quality Partner™.

SAS Alliance
Quality Partner

ACKNOWLEDGEMENTS

The author would like to thank Patricia Gerend and Anita Rocha for their invaluable assistance in the technical review of this paper.