A Comprehensive Automated Data Management System For Clinical Trials

Heather F. Eng, University of Pittsburgh, Pittsburgh, PA
Jason A. Lyons, University of Pittsburgh, Pittsburgh, PA
Theresa M. Sax, University of Pittsburgh, Pittsburgh, PA

ABSTRACT

A successful data coordinating center for multicenter clinical trials and registries must provide timely, individualized, and frequent feedback to investigators and study coordinators over the course of data collection. Investigators require up-to-date reports to help them monitor subject accrual and retention, randomization balance, and patient safety markers. Study coordinators need to know what data are expected or are delinquent, and they need to know about errors to be corrected, such as missing data or data that don't pass validity and logical consistency tests. Frequent monitoring can reveal systemic issues in procedures that require remedial adjustments to keep the project from being at risk.

Data managers at the University of Pittsburgh's Epidemiology Data Center in the Graduate School of Public Health have developed an integrated system to collect and import data from multiple and disparate sources into a central relational database, subject it to comprehensive quality control procedures, create reports accessible on the web, and email individualized reports to investigators and study coordinators, all on an automated and scheduled basis. Post-hoc routines monitor execution logs, so that unexpected errors and warnings are automatically emailed to data managers for same-day review and resolution.

The system is developed almost exclusively using SAS® software. While SAS is best known as statistical software, its strength as a data management tool should not be overlooked. With its strong and flexible programming capabilities for data manipulation, reporting and graphics, web interfacing, and emailing, it provides the infrastructure to serve as a single platform for the management of data collected in clinical trials and registries.

This paper describes the modules of the system and their component programs as they were developed for the Computer-Based Cognitive-Behavioral Therapy Trial currently underway at the University of Louisville and the University of Pennsylvania, with data coordination at the University of Pittsburgh.

INTRODUCTION

CCBT DATA OVERVIEW

The Computer-Based Cognitive-Behavioral Therapy (CCBT) Trial at the University of Louisville and the University of Pennsylvania, with data coordination at the University of Pittsburgh's Epidemiology Data Center (EDC) in the Graduate School of Public Health, is a two center, randomized controlled trial evaluating the efficacy and cost-effectiveness of computer-assisted cognitive-behavior therapy for outpatients with major depressive disorder. Patients receive therapy for 16 weeks, and may be randomized to use a computer-based therapy system in addition. There are two follow-up assessments at months 3 and 6 to establish long term efficacy.

Collected data include structured interviews administered by trained evaluators, participant self-assessments collected by study coordinators, audio recordings of therapy sessions, and information provided by a computer-based cognitive-therapy system accessed by study participants from their home computers.

The interview and self-assessment data are entered into a Microsoft SQL Server database via a front end of menus and screens developed in Microsoft Access. The SQL Server database is housed on the coordinating center's secure server at the EDC, and study personnel at all clinical sites are given password-protected access to this server.

The therapy audio files are collected so that a random selection of them may be reviewed to ensure therapists are adhering to study protocol. The files are placed on computers at the clinical sites where an automated file transfer system developed by the EDC can connect and copy the audio files to the EDC server.

The computer-based cognitive-therapy system, Good Days Ahead (© 2014 Empower Interactive, Inc.), records information about the modules and lessons completed by a participant, and these data are extracted monthly to Excel spreadsheets and sent to the coordinating center at the EDC.

At the EDC, data from these disparate sources are loaded into a SAS data library, from which all quality control and reporting activities are executed, and which serves as a repository for extracting analysis-ready data sets.

REPORTING REQUIREMENTS

To successfully implement a trial with such varied data, it is essential to provide individualized feedback to study personnel both in the field and in the coordinating center.

In CCBT, a weekly report gives study investigators and personnel a comprehensive overview of the study. The report presents tables and charts illustrating participant accrual and randomization status; database status reports including form completion rates and summarized edit (quality control) reports; audio file accounting; and computerized therapy session adherence. It is provided as a PDF and is emailed to all study personnel at the beginning of each week.

Other reports are readily available to study coordinators through a menu selection in the data collection system. Individualized edit reports address questionable and missing data within the study database at a detailed level, as do reports that track completed or delinquent data forms and delivery of audio files. As these reports are meant to encourage prompt action by study coordinators, they are updated three times a day so that work completed can be confirmed within hours, and coordinators can have timely information with which to work.

DATA MANAGEMENT

The programs necessary to meet these requirements number in the hundreds, but they have been integrated into an automated system that runs on a scheduler three times daily. Full database loading, editing, and reporting routines are executed multiple times daily to allow timely alerts and feedback on changes in the data. The system also has features to write its own code for many of these programs, greatly reducing the time required for new project startup.

The EDC has a long history of coordinating multicenter clinical trials and registries, and based on experience and learned best practices, certain conventions are followed. In this paper, reference to some of these EDC conventions will be made amidst the general SAS features being described. To simplify that reference, a brief description is included here.

The shared study directory resides on an EDC server accessible to all data managers and statisticians. It consists of a study root and system of sub-directories.

`CCBT`	root directory
`>prglib`	program library
`>DBLOAD`	database loading programs
`>Edits`	data editing programs
`>Reports`	reporting programs
`>saslib`	SAS data library
`>fmtlib`	SAS format library and PROC Format program

In the program library, a SAS program named INIT.SAS is invoked when opening a SAS session. INIT.SAS establishes titles, libnames, and filenames pointing to locations in this directory structure that will be needed by anyone working on the project. This program is shared by all EDC project team members.

As in most trials, EDC data collection forms are designed to collect data according to intended content and collection criteria. Data collected from each form is stored in its own SQL Server table, and ultimately its own SAS data set, with all three (form/table/data set) sharing the same name. Thus in the descriptions that follow, the terms form, table, and data set may be understood to refer interchangeably to a specific body of data.

METHODS

LOADING THE DATABASE: DBLOAD

The SAS Import procedure works fairly well for importing data from CSVs, Excel, or MS Access, but in the CCBT project most of the data are collected in a SQL Server database. In order for SAS to recognize this external database, a Universal Data Link (UDL) file must be created. The UDL file is a text file containing login information including username, password and IP address that permits access to the database server.

```
[oledb]
PROVIDER=SQLOLEDB.1;PASSWORD=<password>;PERSIST SECURITY INFO=TRUE;
USER ID=<username>;INITIAL CATALOG=<study>;DATA SOURCE=<IP address>;
```

A SAS libname pointing to this UDL file allows SAS programs to refer to external database tables as though they were data sets in a SAS data library. This libname is defined in the program INIT.SAS for the CCBT study.

```
LIBNAME SQL oledb UDL_FILE="L:\<PATH>\CCBT.udl";
```

A separate SAS program is written to import each of 37 tables from the SQL Server database, apply the appropriate labels and formats, and write it to the SAS data library. Additional code is added to some to calculate scores or other analytical variables. This collection of programs, each named for the table it is importing, is stored in the DBLOAD subdirectory in the program library (prglib) folder.

Also in the DBLOAD subdirectory is one program named DBLOAD.SAS that is comprised of many %INCLUDE statements that call the individual importing programs, thus allowing the entire database to be loaded by executing one program:

```
%INCLUDE "L:\CCBT\prglib\DBLOAD\EL.SAS";
%INCLUDE "L:\CCBT\prglib\DBLOAD\DEM.SAS";
%INCLUDE "L:\CCBT\prglib\DBLOAD\RAND.SAS";
... etc.
```

DBLOAD also calls programs that create data management data sets based on the imported data sets (a data dictionary, a catalog of forms received, a master record on each patient), but as contribute nothing to the content of this paper, they will not be discussed beyond acknowledging their part in DBLOAD.

There is one other program run in DBLOAD that warrants discussion, however, because it takes advantage of the SAS software's ability to push out into file structures for data management processes. All CCBT therapy sessions are audio-recorded, and those recording files (MP3 or WMA formats) are to be transferred to the EDC's central server.

Program AUDIO.SAS pushes a command that creates two data files, each containing the filenames for any audio files that have been uploaded for review.

```
OPTIONS noxwait;
x dir U:\DataFiles\MP3\*.mp3 /b > mp3.dat;
x dir U:\DataFiles\MP3\*.wma /b > wma.dat;
```

The resulting .DAT files are read through INFILE and INPUT statements. The audio files are named according to the format ID_SESSNUM.mp3, so once in SAS, the ID and session number are parsed from the audio file name.

These records are stored as a data set in the CCBT Data Library, and in the reporting module of this system, they are compared to a second data set of Therapy Notes (entered through the main data collection system). This comparison informs when audio notes and files should be expected, to allow reporting of missing Notes or Audio files.

Separately from the DBLOAD process, data from the computerized therapy system Good Days Ahead are extracted to a .CSV file once a month and sent to the EDC. This file is imported and data that is relevant for analysis are parsed into a data set in the SAS data library. Because this happens only monthly, it is one of the few manual program submissions taking place in this process.

QUALITY CONTROL: EDITS

Although the CCBT data collection screens are heavily programmed with range checks, trigger/dependency relationships, and completion alerts, there is an additional quality assurance process that is executed when the data reach the SAS data library. This EDITS process is designed to evaluate the variables in each data set for missing values and range violations, but it also includes checks on logical and chronological consistency that cannot be checked at the data entry level.

Structured much like DBLOAD, the EDITS process is comprised of separate SAS programs that check each data set, and create temporary data sets of discovered checks (or "edits") for later reporting. This collection of programs, each named for the table it is checking, is stored in the EDITS subdirectory in the program library (prglib) folder.

Also in the EDITS subdirectory is one program named EDITS.SAS that is comprised of many %INCLUDE statements that call the individual edits programs, thus allowing the entire database to be edited by executing one program:

```
%INCLUDE "L:\CCBT\prglib\DBLOAD\EL_EDIT.SAS";
%INCLUDE "L:\CCBT\prglib\DBLOAD\DEM_EDIT.SAS";
%INCLUDE "L:\CCBT\prglib\DBLOAD\RAND_EDIT.SAS";
... etc.
```

The individual edit programs contain one long DATA step which is a series of statements that evaluate consecutively each variable in the data set for range, missingness, or logical inconsistency. Any discovered problems are classified as to type of problem, and a record is output to a work data set containing variables *ID, DATE, TIMEPT, Site, Form, Varnum, Varname, Value, Edit_chk, Status, OriginDt,* and *UpdateDt.*

After the individual edit programs are run, EDITS.SAS combines the resulting work data sets into a single edits data set. This is merged with a permanent data set from the CCBT SAS data library that contains edits from all prior runs. When merged, if an edit is found in the prior runs and not in the current, its *Status* is changed to "resolved." If an edit in the current set is also found in the prior set, i.e. it remains unresolved, the *OriginDt* from the prior set is carried forward, and the *UpdateDt* from the current is maintained. If an edit from the current set is not in the prior set, both its *OriginDt* and *UpdateDt* are set to the current date.

This resulting merged and updated data set is written back to the CCBT data library for use in reporting details about edits to be resolved, and also creating summary tables that show the rate at which newly discovered edits are being resolved at each site, and possible backlogs of unresolved edits that might require remedial action.

CODE GENERATION

The form-specific SAS programs that are called by DBLOAD are structurally the same, differing only in the table names and variables being processed, and the same is true for the individual EDIT programs. Because of these similarities, their statements can be written by a code-generating program using FILE and PUT statements.

At the time of study start-up, an Excel workbook is created with two spreadsheets. One is a data dictionary containing variable definitions (*Form, Varname, Type, Label, Format, High, Low, Trigger,* and *Condition*) for each table in the SQL Server database, and the other is a codebook, defining the formats (*Name, Value,* and *Label)* to be used across all resulting SAS data sets.

A SAS program called SASGEN reads variable descriptions from the dictionary, and writes a separate SAS importing program for each table, to be called by DBLOAD.SAS. Another code-generating SAS program called FMTGEN reads from the codebook to write one program that creates a central SAS format library. And a third program called EDITGEN reads range and dependency characteristics (i.e. for branching logic) from the dictionary and writes the programs called by EDITS.SAS.

Code review: SASGEN
A macro is used to write the individual programs, being called separately for each data set. Parameters are simply the form (or data set) name, and the keys to the form.

```
%SASGEN(ATQ,ID TIMEPT DATE);
```

Using the form parameter, the first DATA step selects the dictionary records for that data set. The next DATA step initiates the output program file using a FILE statement, and using a PUT statement, writes syntax to start the program. The opening set of code includes statements that apply to all SQL tables being imported, regardless of variables in the table.

```
DATA header;
 * Initiate program file ;
FILE "L:\CCBT\PRGLIB\DBLOAD\NEW\&form..SAS" lrecl=80;
today=date(); FORMAT today mmddyy8.; RETAIN bck -1;
PUT "* &form..SAS: Create CCBT data set of &form data ;"
  / "* Generated by SASGEN.SAS on " today ";"
 // "DATA &form;  SET SQL.&form;"
  / " * Create SITE ;"
  / "SITE=substr(ID,1,2)+0;"
  / "FORMAT SITE site.;"
  / " " ;
RUN;
```

Generated code:
```
* ATQ.SAS: Create CCBT data set of ATQ data ;
* Generated by SASGEN.SAS on 07/24/14 ;

DATA ATQ;  SET SQL.ATQ;
* Create SITE ;
SITE=substr(ID,1,2)+0;
FORMAT SITE site.;
```

Subsequent DATA steps invoke the same file, using the MOD option of the FILE statement to allow more statements to be written to the same program file.

SASGEN next writes code to convert missing values to SAS missing values. In the CCBT data collection system, various kinds of null data (e.g. skipped, not reported, unknown, or refused) are assigned specific negative numbers or dates outside of reasonable ranges. When these values are imported to SAS, they are converted to specific SAS missing values (e.g. .A, .B, .C, etc.)

SASGEN writes this conversion code by sorting the records by *Type*, then writing ARRAY statements to the SAS program, assigning each variable to its appropriate array. After defining all arrays, a next step writes the code necessary to process each array, converting missing values into the appropriate SAS missing values. Note that depending on the variables found in the specific form, not all arrays are required, and SASGEN only writes those statements applicable to the form in question.

The next two DATA steps create the LABEL and FORMAT statements. Pre-sorting by *Varnum* allows the LABEL statement to be written for all variables in data set order, while re-sorting by *Format* then *Varnum* produces one FORMAT statement with variables grouped by *Format*.

The SASGEN macro finishes by writing the Sort procedure statements that will sort the data set by its keys parameter, and a final DATA step to save the new data set to the CCBT data library.

Code review: FMTGEN
This similar but much simpler program reads from the same Excel workbook, this time from the codebook sheet. All records are imported, and missing values are converted to their equivalent SAS missings as described above.

FMTGEN then sorts the records by *Name* and *Value* so that it can use FIRST.Name and LAST.Name to identify the record-groups that comprise each format. Each format is written as a VALUE statement to the external file.

The resulting program FORMATS.SAS is executed once, writing a permanent format library that is referenced by all data sets in the project data library. As needed, additional formats are manually added to FORMATS.SAS in alphabetical sequence, and the program is executed again to overwrite the format library with all formats.

Code review: EDITGEN
EDITGEN.SAS is similar to SASGEN in that it uses a macro to write the individual programs, but its logic is quite different. SASGEN groups variables so that it can write statements using arrays or one LABEL or FORMAT statement, but EDITGEN writes one variable at a time, producing a series of statements for each variable.

EDITGEN is best described by reviewing the resulting code, seen below. The first few statements EDITGEN writes set variable *Edit_Chk* to zero and define the variables needed for reporting. Then EDITGEN writes statements to check for any rule violations, and if any are found, the value of *Edit_Chk* is reset. In this example, variable *Lgrade* is dependent on variable *Educatn* having a value of 1.

```
Edit_Chk=0;
Label ="Last grade completed"; Varnum=18;
Varname='Lgrade'; Value=left(Lgrade);
IF Lgrade=.A OR Lgrade=.                          THEN Edit_Chk=1;
ELSE IF not (1<=Lgrade<=11)and (abs(Lgrade)>0) THEN Edit_Chk=3;
ELSE IF ((Educatn eq 1) AND (Lgrade=.A)) OR
     (not(Educatn eq 1) AND not (Educatn=.A OR Educatn=.)
     AND not(Lgrade=.B OR Lgrade=.E))             THEN Edit_Chk=4;
IF Edit_Chk>0 THEN OUTPUT;
```

REPORTING

As described earlier, there are two kinds of reports produced for the CCBT trial. The first is a set of weekly reports that are created as one PDF document that is emailed to study personnel. These reports are mostly high-level summaries of participant accrual and randomization and database status: forms accrued or missing; edits resolved or outstanding; audio files received or missing; Good Days Ahead lessons completed or missed.

The other kinds of reports are designed as HTML reports, accessible to coordinators through the data collection front end. Some of these feature drill-down capabilities that allow coordinators to move from overview to detailed view of certain reports. These HTML reports are refreshed three times daily, so that coordinators can view reports that promptly reflect work they have done.

Weekly Report Packet
The packet is created through the Report procedure and the SAS output delivery system (ODS). Each report is written to two locations: on the shared server to be viewed from the data collection front end (overwritten each time it is generated), and on the EDC local server, as an archived copy for data management purposes. This copy must have the date included in its filename.

To include the date, a null DATA step is used to create macro variables *fDate, RunDate,* and *PostDate.*

```
DATA _null_;
CALL SYMPUT("fDate",LEFT(PUT("&sysdate"d,yymmddn.)));
     RunDate=TODAY() -1;
CALL SYMPUT("RunDate",LEFT(PUT(rundate,mmddyy10.)));
     PostDate=TODAY() -7;
CALL SYMPUT("PostDate",LEFT(PUT(PostDate,worddate.)));
RUN;
```

These variables can then be used throughout the PDF in titles and FOOTNOTEs. In particular the *fDate* variable is appended to the end of the PDF file name to create the archive copy of that report.

This ODS code will direct the report to multiple paths, one for live access, and one for archiving:

```
ODS PDF (ID=drive2) FILE="<shared server>\rprt.PDF"       (live access)
         NOTOC STYLE=style.Print STARTPAGE=no;
ODS PDF (ID=drive1) FILE="<EDC server>\RPRT_&fdate..PDF" (archive)
         NOTOC STYLE=style.Print STARTPAGE=no;

TITLE2  H=4 " Forms In Summary as of &rundate";
FOOTNOTE h=8pt j=l "%sysfunc(today(),mmddyy10.)" j=r "page ^{thispage}";
```

```
PROC REPORT data=FormsIn nowd;
   column Y1  Y2 Y3 TOTAL;
   define Y1/ "^{style [font_size=3] X1 title}" width=10 style=[just=left];
RUN;
```

The report is then printed to both locations using PROC REPORT.

In creating the Weekly Packet, it may be necessary to force a new page at the beginning of the next report. This is done using the STARTPAGE option in ODS PDF:

```
ODS PDF (ID=drive1) startpage=yes;
ODS PDF (ID=drive2) startpage=yes;
TITLE2  H=4 " Forms In Summary as of &rundate";
FOOTNOTE h=8pt j=l "%sysfunc(today(),mmddyy10.)" j=r "page ^{thispage}";
PROC REPORT data=FormsIn nowd;
   column Y1  Y2 Y3 TOTAL;
   define Y1/ "^{style [font_size=3] X1 title}" width=10 style=[just=left];
RUN;
```

If two reports are small enough to share a page, it is possible to force some white space between the two reports. The NEWLINE option may be used instead of the STARTPAGE option, placing the next report title halfway down the page (again for both the live report and the archived version):

```
ODS PDF (ID=drive1) text = '^{newline 6}';
ODS PDF (ID=drive1) text = '^S={just=center font=("Arial, Helvetica, Helv",4)}
                           DataStream Files Received by Site';
ODS PDF (ID=drive2) text = '^{newline 6}';
ODS PDF (ID=drive2) text = '^S={just=center font=("Arial, Helvetica, Helv",4)}
                           DataStream Files Received by Site';
PROC REPORT data=ArmStrata nowd;
   column (STE) blank ('CBT' _11 _12 _13 _14) blank1 ('CCBT' _21 _22 _23 _24 );
   define STE   / "" order order =internal;
      define blank / "" center format=NoDot. style=[cellwidth=10mm];
   define _11   / "^{style [font_size=3]Neither}" display center;
    ... etc.
   RUN;
```

Both PDFs are closed after all reports are compiled into the single PDF frame; any other options are reset to blank.

```
ODS PDF (ID=drive1) close;
ODS PDF (ID=drive2) close;
GOPTIONS reset=all;
TITLE;
FOOTNOTE;
```

Emailing the Weekly Report

SAS software includes an emailer utility that can send files from within a SAS program. There are three components to an emailer program. First, email parameters are set to define the local email protocol and account information through the OPTIONS statement.

```
OPTIONS
   emailsys= SMTP                specify the email protocol (MAPI, VMI or SMTP)
   emailhost=<HOST>              specify the server used for sending the email
   emailauthprotocol=LOGIN       sender's USERNAME
   emailpw="PASSWORD";           sender's PASSWORD
```

Next, the FILENAME statement sets the email options for sending to one or more recipients, with attached file.

```
FILENAME outmail EMAIL
   Subject="STUDY Weekly Report"
   From   ="user1@domain.edu"
   To     =("user2@domain.edu" "user3@domain.edu")
   CC     =("user4@domain.edu" "user5@domain.edu")
   ATTACH ="L:\CCBT\prglib\Reports\Weekly_report.sas";
```

And finally a DATA _NULL_ step is used to write the email template that will be sent to the recipients.

```
DATA _NULL_;
FILE outmail;
   PUT
       /  'Good morning all,'
       /  ' '
       /  'Attached are the STUDY Reports for data entered through last'
       /  'Friday. Please review at your convenience. If you have any'
       /  'questions, please let me know.'
       /  ' '
       /  'Thanks.'
       /  ' ';
RUN;
```

Creating HTML Drilldown Reports

Some reports were developed using the ODS HTML statement with the intention of having a drilldown ability to go from the overall and site level reports down to individual subject reports.

To create a reports menu in the data collection front end, an HTML parent directory was constructed by storing the report names in a data set. Each of these values is a string of HTML code that when displayed becomes a hotlink to a report. This parent directory is a static table that is not dependent on study data, but simply identifies the reports to be available through this menu.

```
DATA HTMLhardcode; SET tempHTML;
LENGTH HTMLname $30  htmlMENU $100;
IF MENU =1 THEN HTMLname ='Accrual Reports';
IF MENU =2 THEN HTMLname ='Unresolved Forms Reports';
IF HTMLname ='Accrual Reports'   THEN htmlMENU="<a href=X:\Reports\HTML\
                          AccrualReports\.html>"||trim(HTMLname)||"</a>";
IF HTMLname ='Unresolved Forms Reports' THEN HTMLmenu="<a href=X:\Reports\HTML\
           UnresolvedForms\UnresolvedformsRprt.html>"||trim(HTMLname)||"</a>";
RUN;
```

Once the data set has been created, a report is run to create the Reports Main Menu, an HTML file that is output to the CCBT report directory.

```
ODS HTML PATH="U:\Reports\HTML\" (URL=none)
         BODY="index.html"
       (TITLE='Computer-Assisted Cognitive-Behavior Therapy for Depression')
        STYLE=styles.HTMLparent;
TITLE ;
ODS escapechar = '^';
PROC REPORT DATA=HTMLhardcard NOWD;
     COLUMN HTMLmenu;
     DEFINE HTMLmenu / "Main Report Index" STYLE=[just=center];
RUN;
ODS HTML CLOSE;
```

This setup program is run once at study startup to establish the first level of the drilldown menu.

The same technique is used for each level of reporting to display data in a drilldown fashion where applicable. For example, an Unresolved Forms Report opens an ODS HTML report at the site level. The names of the sites involved in this particular report are presented as hot link code along with accompanying report data for the respective site.

```
DATA SiteLvl; SET Main;  LENGTH xSITE $10;
IF SITE =1 THEN xSITE ='SITE 1';
IF SITE =2 THEN xSITE ='SITE 2';
LENGTH htmlSITE $120;
htmlSITE="<a href=X:\Reports\HTML\UnresolvedForms\
           SITE\"||trim(xSITE)||".html>"||trim(xSITE)||"</a>";
RUN;
ODS HTML PATH="U:\Reports\HTML\UnresolvedForms\"(URL=none)  ... etc.
PROC REPORT DATA=drill6 NOWD;
       COLUMN htmlSITE MSNGFRMS;
       DEFINE htmlSITE    / "SITE" display style=[just=left];
       DEFINE msngfrms    / "Total Unresolved Forms" DISPLAY CENTER;
RUN;
ODS _all_ CLOSE;
```

From the site level, the same process is repeated only with the individual participant IDs containing the HTML link code. Since not every ID will have unresolved forms, we first use the NULL DATA step to count the number of IDs with unresolved forms. We then run through a macro for each ID found in the data set and output an ODS HTML file with the ID contained in the filename.

```
DATA IDcnt; SET ccbt.Forms_In;
BY id; IF FIRST.id; KEEP id arm exitdt randomdt;
RUN;

* Count the number of IDs to be reported ;
DATA _null_; SET IDcnt END=eof;
IF eof THEN CALL SYMPUT("ID_n",_N_);
RUN;

%macro ID_loop;
* Loop once for each ID found in IDcnt data set ;
%do ID_recno=1 %to &ID_n ;

* Store ID and some dates in macro variables ;
DATA _null_; SET IDcnt;
IF _N_=&ID_recno THEN DO;
   CALL SYMPUT("xID",TRIM(LEFT(id)));
   CALL SYMPUT("xRand",LEFT(PUT(Randomdt,mmddyy8.)));
   CALL SYMPUT("xExit",LEFT(PUT(ExitDt,mmddyy8.)));
   RunDate=today()-1; * From last nights data ;
   CALL SYMPUT("rundate",LEFT(PUT(rundate,worddate.)));
   CALL SYMPUT("fdate",LEFT(PUT("&sysdate"d,yymmddn.)));
END;
RUN;

* Write ID-specific HTML output string to create a link for drilldown report ;
ODS HTML PATH="U:\Reports\HTML\UnresolvedForms\Participants\" (URL=none)
         BODY="&xID..html" (TITLE='CCBT')
        STYLE=styles.HTML3deep;

* Create SAS titles using macro variables ;
DATA titles; SET IDcnt; WHERE id="&xID";
LENGTH TitleStr $200;
TitleStr="Subject &xID, Randomized &xRand";
IF ExitDt>0 THEN
   TitleStr="Subject &xID, Randomized &xRand, Exited Early on &xExit";
CALL SYMPUT("titlestr",titlestr);
RUN;

* Set titles ;
TITLE2 BOLD "CCBT Patient Form Status Report - Data as of
%sysfunc(date(),worddate.) at %sysfunc(time(), TIMEAMPM9.)";
TITLE3 "&titlestr";

PROC REPORT data=ccbt.Forms_In nowd; where id="&xID";
COLUMN Timept ('FORM LIST' EL--CSSRSB SCID--WAIT CSSRS ET) ;
DEFINE Timept / " " ORDER ORDER=internal STYLE=[cellwidth=35mm JUST=left];
DEFINE EL     / DISPLAY WIDTH=1 'E/L' STYLE={background=FORMCOLR. VJUST=B}
                                      FORMAT=FRMSIN.;
...etc.
RUN;
ODS HTML close;
%end;
%mend;
```

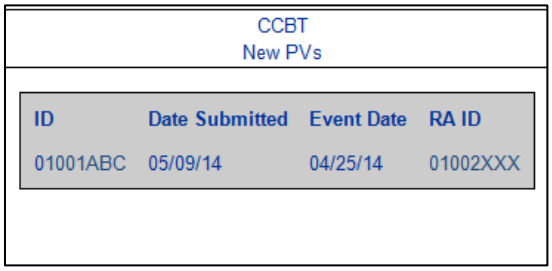
Using the ODS HTML output destination for reporting allows the creation of web-based output that can be accessed through the internet as well as by other programs capable of displaying webpages. Care must be taken to use the same string value for the ODS HTML output as created in the main menu variable, but if done correctly, the resulting report can create easy hotlink navigation throughout the entire report.

Like the Weekly reports, these HTML reports are also archived with date-time stamps for future reference if needed.

Emailed Alerts

Over the course of a study, events occur that must be reported on a timely basis. Serious adverse events, for example, must be handled promptly so that required IRB and DSMB reporting requirements are met. Protocol violations and participant early terminations also are unexpected events that should be reported to data managers for appropriate action.

A special program is included in the automated process that will scan the database for new Serious Adverse Events, Protocol Violations, or participant Early Terminations. If any are found, a notification is emailed automatically to the EDC data managers for follow-up (Display 1).



CCBT
New PVs

ID	Date Submitted	Event Date	RA ID
01001ABC	05/09/14	04/25/14	01002XXX

Display 1. Protocol Violation Alert

AUTOMATION

All the programs discussed thus far are set to run automatically. The obvious caveat in such a system is the danger of unnoticed program errors or small bugs occurring as data change through the system. In order for this system to work creditably, it is crucial to monitor the execution logs of these programs to be sure they run as expected.

Log Scanning

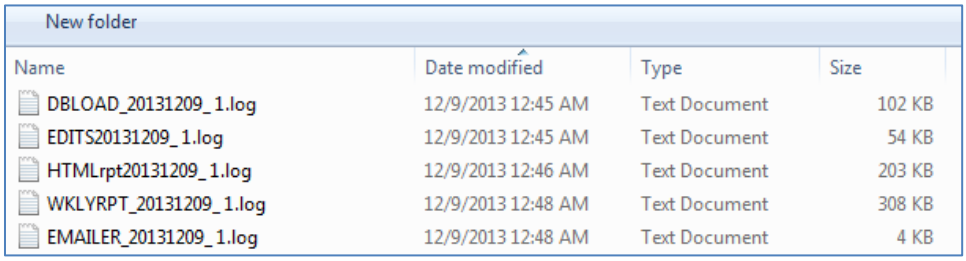
SAS offers an option to write execution logs to external files, using the Printto procedure.

```
PROC PRINTTO LOG= "L:\CCBT\prglib\AutoReporting\DBLOAD_&fdate._&ftime..log";
RUN;
```

This will write the SAS log to the designated file until it is turned off with

```
PROC PRINTTO LOG='null';
```

Each step of the process writes its own log, so that there are several written to the same folder (Display 2).



New folder

Name	Date modified	Type	Size
DBLOAD_20131209_1.log	12/9/2013 12:45 AM	Text Document	102 KB
EDITS20131209_1.log	12/9/2013 12:45 AM	Text Document	54 KB
HTMLrpt20131209_1.log	12/9/2013 12:46 AM	Text Document	203 KB
WKLYRPT_20131209_1.log	12/9/2013 12:48 AM	Text Document	308 KB
EMAILER_20131209_1.log	12/9/2013 12:48 AM	Text Document	4 KB

Display 2. Execution Logs

A program named LOGSCANNER.SAS has been created to read through the saved log files after all of the programs in the automated process have completed.

```
*** Find error and warning messages in the logs;
DATA f_error f_warning;
ATTRIB fName f_name FORMAT=$200.;
INFILE "L:\CCBT\prglib\AutoReporting\*.log " FILENAME=fName;
f_name = fName;
```

Using "`*.log`" in the INFILE statement will result in a successive reading of all .log files found in that location. This is useful because weekly log files do not appear every day, so the files available for reading will vary depending on the run day and time.

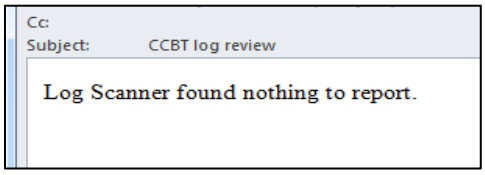
The "`filename=fName`" option will place the name of the file being read into the variable *fName* for use in reporting the location of any discovered problems.

LOGSCANNER then checks for key words in the log such as “ERROR”, “WARNING” and “UNINITIALIZED”. If any of these key words are found in the log, the line from the log is written out to one of two temporary data sets.

```
INPUT txt $200.;
IF INDEX(UPCASE(txt),'ERROR')           >0 THEN OUTPUT f_error;
IF INDEX(UPCASE(txt),'WARNING')         >0 THEN OUTPUT f_warning;
IF INDEX(UPCASE(txt),'UNINITIALIZED')   >0 THEN OUTPUT f_warning;
IF INDEX(UPCASE(txt),'MERGE STATEMENT')>0 THEN OUTPUT f_warning;
RUN;
```

Using the same SAS Emailer logic described earlier, a report written from the temporary data sets is sent to CCBT data managers each time the program is run.

If there is no issue to report, an email is sent confirming there were no errors noted (Display 3).



Display 3. Clean LogScanner Report

If there are issues, data managers receive a report like the one in Display 4 so they can retrieve the log in question to explore and correct the problem.

Log Probe Results

Log File	Problem Result	Description
STUDY.log	Error detected	Neither the PRINT option nor a valid output statement has been given.
STUDY.log	Warning detected	WARNING: number of missing values of impacts is larger than 6
STUDY.log	Uninitialized variable detected	NOTE: Variable X1 is uninitialized.
STUDY.log		NOTE: MERGE statement has more than one data set with repeats of BY values.

Display 4. LogScanner Error Report

After the email has been sent, LOGSCANNER uses the `'x move'` command to move the log files into an archive directory, where they are available for reference, leaving the AutoReporting folder emptied for the next set of log files.

```
OPTIONS NOXWAIT;
X CD L:\CCBT\Prglib\AutoReporting\Archive\;
X MOVE "L:\CCBT\prglib\AutoReporting\*.log";
X CD L:\CCBT\Prglib\;
```

These archived files are zipped regularly to keep the archive folder manageable.

Automation Driver

All of the components presented above are called by a master automation program that is executed via Windows Task Scheduler three times daily: at 12:45 AM, at noon, and at 4:00 PM. These times are selected specifically to provide feedback to study coordinators at useful intervals. When executed, this automation program will:

- initialize the CCBT project environment by running INIT.SAS
- update the SAS data library by running DBLOAD
- update the edit reports by running EDITS
- update the HTML reports three times daily
- run and email the Weekly Report to all study personnel (Mondays at 12:45 only)
- scan for SAEs, protocol violations, and early terminations, and email any alerts to data managers
- save the SAS logs from all executed programs
- scan the SAS logs for errors and warnings, and email scan results to data managers

Code for this driver program is presented in whole, with explanatory comments:

```
* Set up all dates, weekday, and time variables;
DATA _null_;
   CALL SYMPUT("fdate",left(put("&sysdate"d,yymmddn.)));     (filename date stamps)
   CALL SYMPUT("weekday",put(weekday(date()),2.0));          (for weekly report)
   CALL SYMPUT("time",time());                               (for weekly report)
   CALL SYMPUT("ftime",(put(time(),hour.)));                 (for 3 X day reports)
RUN;

* Open log file for DBLOAD, run INIT.sas and DBLOAD, close the log file ;
PROC PRINTTO LOG="L:\CCBT\prglib\AutoReporting\DBLOAD_&fdate._&ftime..log";
RUN;
%INCLUDE "L:\CCBT\prglib\Init.sas";
%INCLUDE "L:\CCBT\prglib\DBLOAD\DBLOAD.sas";
PROC PRINTTO LOG='null'; RUN;

* Open log file for EDITS, run program, close the log file ;
PROC PRINTTO LOG="L:\CCBT\prglib\AutoReporting\EDITS&fdate._&ftime..log"; RUN;
%INCLUDE "L:\CCBT\prglib\Reports\EditsReport.sas";
PROC PRINTTO LOG='null'; RUN;

* Open log file for HTML reports and Alerts, run programs, close the log file ;
PROC PRINTTO LOG="L:\CCBT\prglib\AutoReporting\HTMLrpt&fdate._&ftime..log";
RUN;
%INCLUDE "L:\CCBT\prglib\Reports\HTMLAcc.sas";
... etc. other include statements
* Also run the alert program for SAEs, PVs, and ETs ;
%INCLUDE "L:\CCBT\prglib\Reports\AlertsEmail.sas";
PROC PRINTTO LOG='null'; RUN;


* Only Monday at 12:45 AM: run the Weekly Report and send out the PDF;
%macro WeeklyReport;
%if &weekday =2 %then %do;    *Select Monday - weekday=2 ;
   %if &time <3600 %then %do; *Select the run that occurs before 1am ;
   PROC PRINTTO LOG=L:\CCBT\prglib\AutoReporting\WKLYRPT_&fdate._&ftime..log";
   %INCLUDE "L:\CCBT\prglib\Reports\Weekly_report.sas";
   %INCLUDE "L:\CCBT\prglib\Reports\Emailer.sas";
   PROC PRINTTO LOG='null'; RUN;
   %end;
%end;
RUN;
%mend WeeklyReport;
%WeeklyReport;

* Run the log scanner program to report and email any log issues ;
%INCLUDE "L:\CCBT\prglib\Reports\LogScanner.sas";
```

CONCLUSION

Coordinating data management for a clinical trial, whether single- or multi-center, presents many challenges and requires many concurrent processes that are closely integrated with each other. The advantages of this automated system are many: it effectively documents each of the tasks that need to be accomplished in day-to-day management of trial data; it executes the tasks consistently, completely, and in the proper order; it monitors itself to send alerts if something goes awry, so its execution is dependable; and its 2-3 minutes of execution time translates to a huge savings in resources that would be spent on data managers running programs, checking logs, and emailing and posting reports.

Its flexible modular design, with driver programs calling sub-programs which in turn call other programs, allows changes and additions to be made very easily without disrupting the entire flow of the process. Its ability to write its own programs at study startup provides added value in consistency and dependability of programs that are familiar in style and syntax to all members of the study team.

Its thrice-daily schedule provides refreshed data to study personnel on a fairly useful basis, although it only approximates live data interaction as provided by some other clinical data systems. However the benefit of this model is that data from any source and in any format can be added to its central database, which allows more flexibility in study design and collection.

Many commercially available systems exist for processing clinical trial data, but all have limitations, and most have added cost. This system, developed through years of refinement and expertise, is affordable, completely flexible to implement innovations in clinical trial design, and most importantly in this world of clinical research where funding is scarce and timelines are compressed, it can be implemented quickly and will provide dependable results with fairly minimal personnel cost.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support and helpful suggestions provided by the investigators and study coordinators of the CCBT Trial, namely Jesse H. Wright, MD, PhD, Tracy Eells, PhD, and Kitty B. deVoogd at the University of Louisville; Michael E. Thase, MD, Marna S. Barrett, PhD, Gregory K. Brown, PhD, and Elisabeth Ertel at the University of Pennsylvania; and Stephen R. Wisniewski, PhD, at the University of Pittsburgh.

CONTACT INFORMATION

Your comments and questions are welcomed. You may contact the author at:

Heather F. Eng
Associate Director of Data Management
University of Pittsburgh Epidemiology Data Center
127 Parran Hall
Graduate School of Public Health
Pittsburgh, PA 15261
412-624-5177
: eng@edc.pitt.edu