

## Developing Clinical Software Systems in SAS/AF®

Carl R. Haske, Ph.D., STATPROBE, Inc., Ann Arbor, MI

### ABSTRACT

Using SAS/AF as a software development platform permits rapid applications development. SAS supplies several classes of objects that the developer can use to quickly prototype and implement systems. However, when developing applications for use in the pharmaceutical industry, it is useful to augment these classes with additional classes that provide robust data management and reporting capabilities. Classes that provide generic tools to subset data, browse data, move data, and report data are useful in the clinical research environment.

Managing clinical trials data requires the efforts of a large team of people including clinical data managers, data entry personnel, programmers, and statisticians. A system that enables concurrent access to data at several stages of the database life cycle enhances the overall work process and accelerates database development. STATPROBE, Inc. developed an integrated data management system, using SAS/AF as the platform. The system incorporates security and tools that assist the valid production and analysis of the clinical data. Various modules in the system support Case Report Form (CRF) tracking, data entry, data cleaning, data querying, and remote data access. The system allows data to progress from data entry slots to a cleaning platform during the critical initiation of a database. The data then emerges to a quality assurance stage for query resolution and preliminary analysis. Data at the quality assurance stage is accessible remotely via the STATPROBE Data Access System. A final database may accumulate in phases or be generated at once when quality assurance is complete. The STATPROBE Data Management System (DMS) and STATPROBE Data Access System provide a systematic, standard, menu-driven method of implementing the necessary tasks for the initiation, development, modification, maintenance, and tracking of client project data at STATPROBE, Inc. This paper describes these systems and the associated class libraries and templates used during the development of these systems.

### INTRODUCTION

This paper describes a clinical data management system developed in SAS/AF. The sections of the paper correspond roughly to the primary system menu (see figure 1).

Pads on the primary menu are:

- User Registration
- Change Password
- Select Project
- Project Administration
- CRF Tracking
- Data Entry
- Data Cleaning
- Query Tracking
- Database Administration
- Quit to SAS
- Exit

The system architecture is logically divided into three sections, a core application section, a section to administrate multiple database management projects, and a section that addresses specific clinical data management tasks for a given project. The core application section handles user login, user registration, and changing passwords. These are functions that are common to many software applications. The project administration and

project selection segment of the system is designed to allow for management of several databases for multiple clinical trials. Multiple project management is also a task common to many software applications. The five main activities specific to the management of clinical data are CRF tracking, data entry, data cleaning, query tracking, and database administration.

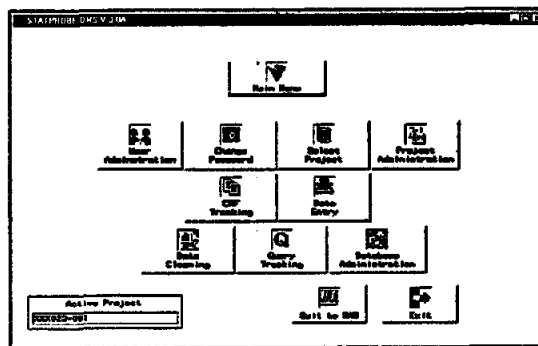


Figure 1. Main System Menu

The first section of the paper, Managing Clinical Data, details the specific tasks of the five main activities associated with managing data in clinical trials. The second section of the paper, Project Administration, discusses the project management aspects of the system. The third section of the paper, System Administration, discusses registering users and the log on system. The next section of the paper describes the STATPROBE Data Access application. The Application Framework section presents a brief background into the development platform used at STATPROBE. The final section discusses reusable templates and classes.

### MANAGING CLINICAL DATA

#### CRF Tracking

The CRF tracking process involves printing CRF page registration, logging CRFs, and reporting the status of CRFs relative to the database (see figure 2). The actual registration of CRFs occurs via the Database Administration menu that is discussed later. The CRF tracking tasks access the CRF registration database.

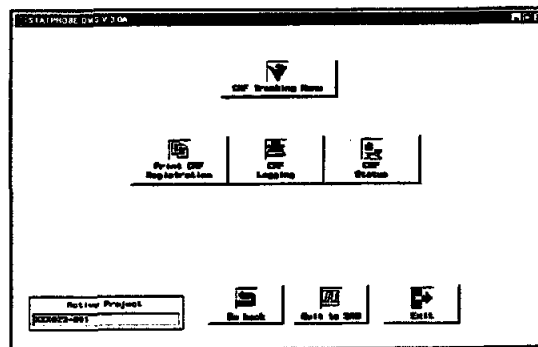


Figure 2. CRF Tracking Menu

CRF pages are registered in the CRF dictionary (see table 1). CRF pages are registered in conjunction with the construction of the clinical database.

| ATTRIBUTE | DESCRIPTION                   |
|-----------|-------------------------------|
| Crf_page  | Unique CRF page number or id  |
| Crf_desc  | CRF description               |
| Crf_keys  | Key identifying fields        |
| Num_flds  | Number of entry fields on CRF |
| Crf_req   | Logical - Is CRF required?    |

Table 1. CRF registration data

The CRF registration data is linked to the clinical database via a relation table (see table 2). This relation table acts as an electronic annotation of the CRFs. The registration table for the clinical database is described later in the Database Administration portion of this section.

| ATTRIBUTE | DESCRIPTION                  |
|-----------|------------------------------|
| Crf_page  | Unique CRF page number or id |
| Crf_fld   | Entry field name             |
| Tablname  | SAS data set name            |
| Varname   | SAS variable name            |

Table 2. CRF entry field to database relation

The date when a CRF arrives at STATPROBE is recorded in the CRF tracking database. The CRF dictionary and relation to the clinical database allows benchmark information to be extracted from the system. This information includes volume and percentage of CRF data that is:

- Double entered into the database
- Cleaned
- In QA status
- Final

To record receipt of a CRF, the user enters the CRF page number or id, the values of the key entry fields, and the date of receipt. For example, the information might include a CRF page number or name, patient number, investigator number, and the current date. All of the standard features of FSEDIT are available to enter the information.

The logging of other CRF tracking processes is automated based on activities performed throughout the DMS. When data for a CRF is double entered, data is cleaned, or edit checks are performed on the data, the system stamps data records with the date and user. When data is modified, the date, user, reason, variable, old value, and new value is entered in an audit trail data set.

### Data Entry

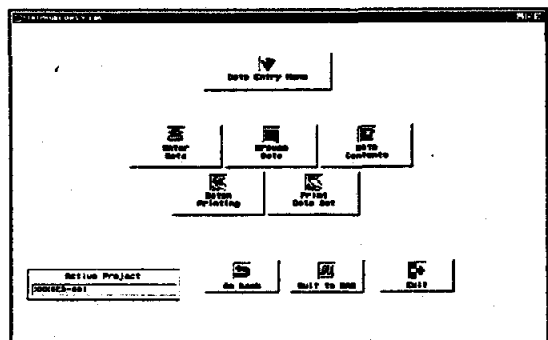


Figure 3. Data Entry Menu

An essential component of the DMS is the ability to enter and modify clinical data. Clinical data at STATPROBE is entered into several physical slots. Data entry personnel are assigned a logical entry level of PRIMARY or DOUBLE to support duplicate entry of data. Users are allocated a slot and are owners of that slot until their data is routed to a cleaning platform. After data has

been entered twice, it passes through a cleaning phase described later.

To sustain the data entry process, users have the option of entering data, browsing data, producing a PROC CONTENTS of the data, or producing a printed copy of existing data either in batch or by specifying a single data set (see figure 3).

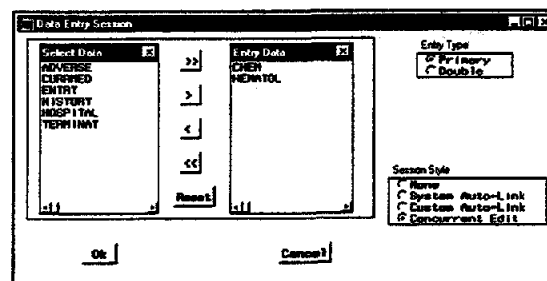


Figure 4. Specifying a data entry session

The system allows the user to select from four types of data entry session styles (see figure 4). By using a mover list object, the user can list the data sets in order and enter data sequentially in the order specified by the list with a system prompt for the option to continue prior to entry in each data set. Alternatively, a sequential edit session can automatically open the next data set in the list when a data set is closed. The user can also select system auto-link. This option links data entry of data sets via an order specified by the database administrator. Finally, the user can select concurrent edit. This allows the user to have multiple data sets open at the same time. To edit data, the user selects the data sets and entry session type, then performs all the standard commands to add, delete, and modify data using an FSEDIT window.

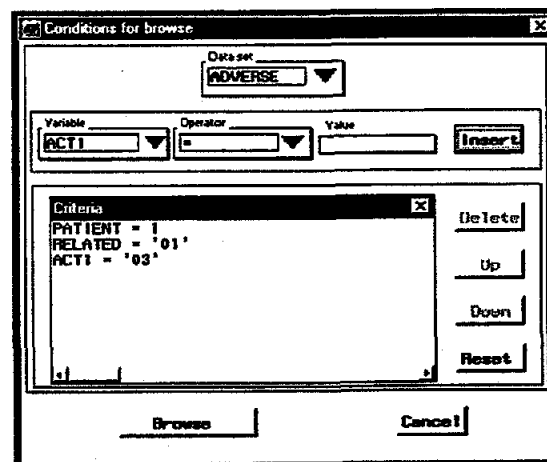


Figure 5. Filter object

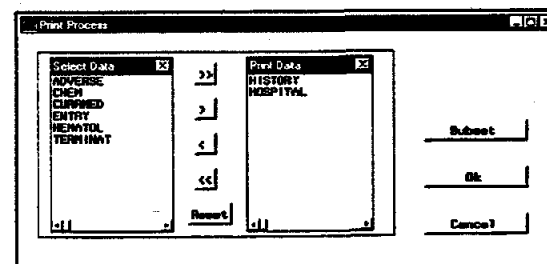


Figure 6. Selecting data for batch printing

To browse or print a single data set, the user selects the data set and specifies a filter condition using an object instantiated from

the filter class (see figure 5). The data that satisfies the list of criteria is printed to the output window or displayed in a viewer.

For batch printing, the user selects the data they wish to print using mover lists (see figure 6) and the user is able to specify a set of criteria to subset all the data sets. The criteria is based on fields that occur in all data sets.

### Data Cleaning

After data is double entered, it is routed from the entry slots to the PRIMARY and DOUBLE slots. Data is cleaned by following systematic steps to clean internal inconsistencies and to resolve inconsistencies in data between the two slots. Inconsistencies are resolved by referencing the original CRFs. See figure 7 for the data cleaning menu. Data cleaning options include:

- Data cleaning tests
- Enter data
- Browse data
- Proc contents of data
- Batch printing of data
- Print a single data set

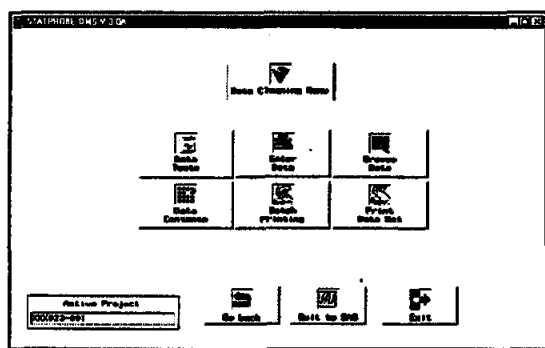


Figure 7. Data Cleaning Menu

Testing the data consists of the following three tasks executed in the order listed (see figure 8):

- resolve duplicate PRIMARY and DOUBLE records
- resolve non-matching PRIMARY and DOUBLE records
- resolve PRIMARY and DOUBLE matching record inconsistencies at the field level

Duplicate records are internal inconsistencies. These are located based on key variables for each data set as defined in the database dictionary section of the DMS when a database is registered.

The non-matching observation process checks for one-to-one correspondence between observations in the PRIMARY and DOUBLE databases. Each record in a given database must have a matching record for the key variables in the other database. Non-matching records are printed in a report for resolution.

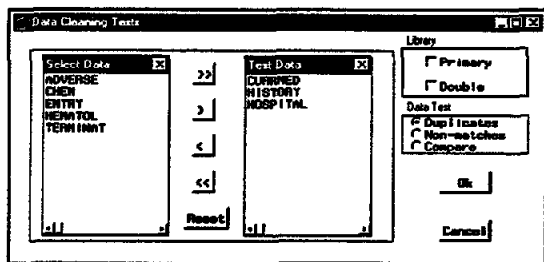


Figure 8. Data Cleaning Tests

In the final cleaning step, data sets are compared for an exact match. A PROC COMPARE report is generated that lists records in PRIMARY and DOUBLE slots that disagree in one or more fields. When all fields in a record agree in PRIMARY and DOUBLE, the record is considered clean and stamped with a clean date. Data is now ready to be locked and moved to the QA slot for quality assurance and query resolution. At this point, data is available for viewing to remote clients via the STATPROBE Data Access application.

When discrepancies are found during the cleaning process, the data cleaner selects the Enter Data option to modify the appropriate data. Additional tools available for the cleaning process are browsing, data contents, batch printing, and printing a single data set. The functionality for these additional data cleaning tools is similar to the analogous tools for data entry.

### Query Tracking

After data has moved to QA, data queries are resolved. Queries are generated when data is entered inconsistently on a CRF. For example, if a male subject had height recorded as 6 feet and weight recorded as 95 pounds, a data query would be generated. Queries are usually resolved at the site where the data was collected and entered on the form. Query resolution is the final step in database quality assurance. The data set that tracks queries is described in table 3.

| ATTRIBUTE | DESCRIPTION                  |
|-----------|------------------------------|
| Crf_page  | Unique CRF page number or id |
| Crf_keys  | Key identifying fields       |
| Crf_fid   | Entry field name             |
| Q_val     | Query value                  |
| R_date    | Resolution date              |
| R_val     | Resolved value               |

Table 3. CRF query tracking

The two primary tasks involved in tracking queries are logging queries to the query data set and modifying data according to a query resolution. When a query is resolved, the data in the QA slot is modified. In order to modify data in the QA slot, the user must have Database Administration clearance and access the data via the Database Administration Utilities menu explained in the next section. Reporting and printing operations on data are available to help with query resolutions.

An audit trail of all modifications to the QA data is generated. The audit trail tracks the person making the change, date of the change, the name of the variable changed, the old variable value, the new variable value, and the reason for the change. After all queries are resolved, data is ready to be frozen and moved to the FINAL slot. Data in the FINAL slot is validated and ready for analysis.

### Database Administration

Database administration incorporates all the high level operations involved in managing a study database. The database administrator works closely with the other data management specialists working to build the clinical database. The database administrator is responsible for making time critical decisions and must be cognizant of the status of data in various slots. The database administrator determines the database structure, designs entry screen, manages the physical movement of data, and manages the data entry sessions.

The tasks captured on the Database Administration menu are as follows:

- System Setup
- Register database
- Register CRFs
- Auto-link specification
- Batch routing of data

- Route a single data set
- Administration utilities

The Administration Utilities menu branches to another menu offering the following tasks:

- Entering data
- Browsing data
- Contents of database
- Batch printing of data
- Printing a single data set
- Allocation of entry slots
- Format designer

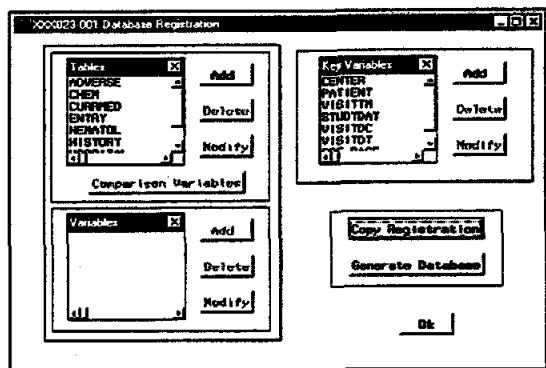


Figure 9. Database Registration

The *register data sets* task is used to design a database dictionary and maintain the database structure (see figure 9). The database dictionary consists of two tables (see tables 4 and 5).

| ATTRIBUTE | DESCRIPTION          |
|-----------|----------------------|
| Tabname   | SAS data set name    |
| Tabdesc   | SAS data set label   |
| Key       | Key table attributes |
| Numattrb  | Number of attributes |

Table 4. Database table dictionary

| ATTRIBUTE | DESCRIPTION       |
|-----------|-------------------|
| Tabname   | SAS data set name |
| Vaname    | Variable name     |
| Varlabel  | Variable label    |
| Varlen    | Variable length   |
| Vartype   | Variable type     |
| Varfmt    | Variable format   |
| Varinfmt  | Variable informat |

Table 5. Database column dictionary

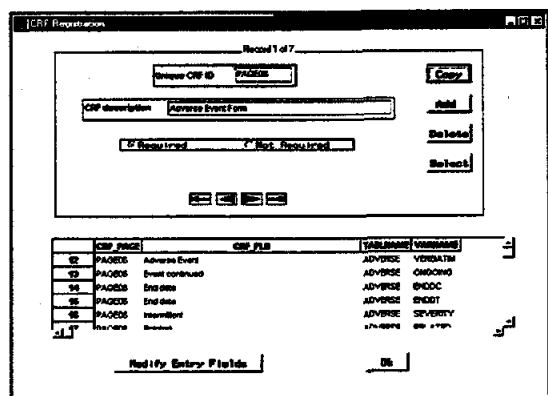


Figure 10. CRF Registration

When the database administrator designs or updates the database, the database dictionary is modified and the file structures are updated.

The CRF registration provides an easy to use interface (see figure 10) for the database administrator to link fields on each CRF page with variables in the data sets. This relation is tracked in the table described in table 2 located in the CRF tracking discussion section.

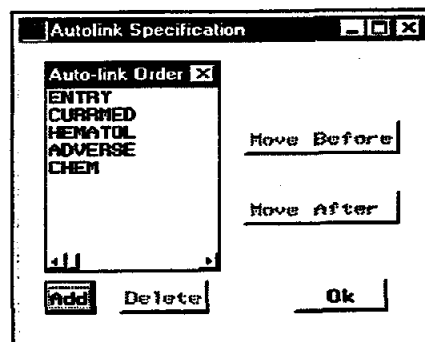


Figure 11. Auto-link Specification

The auto-link specification allows the database administrator to order the data sets in a manner corresponding to the natural flow of the CRFs (see figure 11). This saves time for the data entry person by not requiring them to select data sets and automatically opening data sets for entry in the order specified by the auto-link sequence.

Each time data is routed between physical slots, the database administrator has the option to route an entire data set or subset the data using a filter tool similar to the filter object in figure 5 to define conditions. There is an option to route a single data set and subset on variables that are specific to that data set or perform batch routing and subset on variables that are in all data sets. Batch routing is handled by the frame displayed in figure 12.

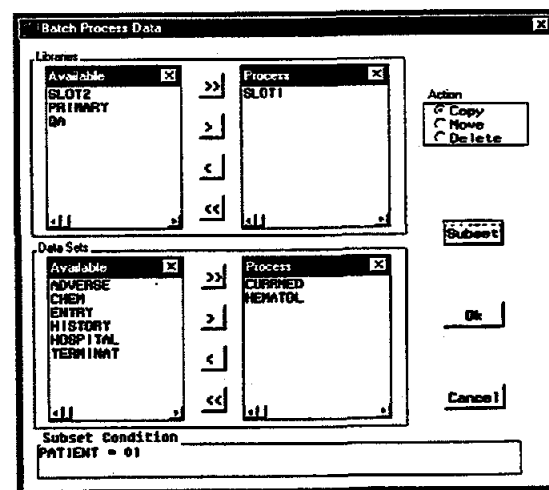


Figure 12. Batch routing

Even when data is frozen, it is likely that it will need to be modified. The database administrator is the only individual that has access to "thaw" data and make modifications. An audit trail of all modifications is maintained.

Status reports are also available to the database administrator. The system tracks key parameters such as entry personnel, cleaning personnel, entry dates, clean dates, lock dates, and freeze dates. The system can scan the data in each slot and

generate reports on the status of the clinical database, entry logs, cleaning logs, and audit trails.

## PROJECT ADMINISTRATION

Project administration is accessible only by a user that has system administration security clearance. The user can manage projects via a composite object (see figure 13). The table at the top of this frame is used for managing the project data. The information in this table is stored in a SAS data set. The four fields for client, product, project, and prefix are concatenated to form the path specification for the standard project file structure. Details of the standard physical layout are omitted.

A table that shows project assignments is displayed at the bottom of the frame. The table displays the data set that defines the relation between the project data set and the user data set. The user data set is described later.

| User ID | Project ID      | Security |
|---------|-----------------|----------|
| 1       | MECKER ABC-1001 | 3        |
| 4       | CHASKE ABC-1001 | 1        |
| 28      | FWALSH ABC-1001 | 5        |
| 28      | FWALSH ABC-1001 | 4        |
| 54      | FWALSH ABC-1001 | 4        |
| 54      | FWALSH ABC-1001 | 4        |

Figure 13. Project Management

Figure 14. Assignment modification

Figure 15. Assignment selection

To modify assignments on a project, the administrator selects the modify assignments button and a form allows additions, deletions, or modifications of personnel that are assigned to the project (see figure 14). When the add button is selected, a list of available users is displayed (see figure 15). This list contains only users that have not been assigned to the project.

Project protocol numbers are displayed in a list box for selection when the user chooses a project from the main menu (see figure 16). The only available projects to a user are those that have a security level assigned for that user. The system accesses the project path to assign standard library references to the physical locations of data files.

Figure 16. Project selection

After selection of a project, the user has a choice on the main menu of the various data management processes; CRF Tracking, Data Entry, Data Cleaning, Query Tracking, and Database Administration. If a user does not have security clearance for a given process, the process button is turned invisible. The SAS data set that stores project information is displayed in table 6.

| ATTRIBUTE | DESCRIPTION                          |
|-----------|--------------------------------------|
| Proj_id   | Unique project system identification |
| Protocol  | Protocol number                      |
| Desc      | Project description                  |
| Path      | Physical path to project files       |

Table 6. Project Profile

Security for the high level processes of the data management system include CRF Tracking (security level 1), Data Entry (security level 2), Data Cleaning (security level 3), Query Tracking (security level 4) and System Setup and Database Administration (security level 5). The security level system is hierarchical, meaning that a user with a security level of 3 will be allowed to access processes of security levels 1, 2, and 3. Security task levels are assigned to personnel for each project (see figure 14). Table 7 displays the data that is stored by the form in figure 14.

| ATTRIBUTE | DESCRIPTION                          |
|-----------|--------------------------------------|
| User_id   | Unique user system identification    |
| Proj_id   | Unique project system identification |
| Security  | Security level                       |

Table 7. Project assignment table

## SYSTEM ADMINISTRATION

All users are required to specify a login id and password. The purpose of logging into the DMS is to provide security for accessing the data management system. A user can only login to the system if the system administrator has added their user profile to the user data set. Table 8 shows the structure of the user data set.

| ATTRIBUTE | DESCRIPTION                       |
|-----------|-----------------------------------|
| User_id   | Unique user system identification |
| Password  | User logon password               |
| Username  | Full user name                    |
| Syslevel  | System level access               |
| Prefs     | SCL list of user preferences      |

Table 8. User Table

The system level security determines if a user has system administration rights. The registering of users employs a composite object similar to that used for registering projects displayed in figure 13. For adding or editing a user the frame in figure 17 is utilized.

Figure 17. User Registration

Another system administration task allows a supervisor to set the path to the system database. This option is typically set at application installation and is rarely used thereafter. This option can be used to copy or backup the entire application to a different physical location. It also allows for multiple data management systems to be installed, each with a different user base and project base. The physical location of the system database is typically the same as that of the application.

## STATPROBE Data Access

The STATPROBE Data Access system is a client system that is integrated into the STATPROBE DMS. This system consists of hardware and software. The system was developed on OS/2®. Since initial development of the system in 1994, it has been ported to Windows 95®. Currently the system is at version 1.6.

The Data Access system is placed at a remote site and allows the client access to a clinical database as it is under development. The client has access to view and query data in the QA stage of development.

Figure 18 displays the main selection frame of the system. The system supports multiple protocols displayed in a radio box at the upper left of the frame. The list box for the data sets at the left refreshes with data for a protocol whenever the protocol is selected in the radio box. As the user selects a specific data set, the variable list to the left refreshes with the variables in that data set.

The transfer data button at the top initializes a modem that connects with the data management site and transfers the database of the selected protocol. This allows the user to periodically update their system with recent data.

Figure 18. Data Access Main Frame

There are two ways to view data, either in tabular format or in record format. The view data button displays the data in tabular format (figure 19). The browse data launches a FSEDIT session in browse mode.

Figure 19. Viewing data in tabular format

When viewing data, the user has access to a very user friendly query procedure referred to in the system as a "Search". Figure 19 shows a search button at the bottom of the screen. This system allows the novice SAS user to construct a query that is displayed to the user in plain English.

Figure 20. Constructing a query

Figure 20 shows one of the dialogs used in constructing a query. The English text of the query is displayed in a text box at the bottom of the screen. An actual query can contain several conditions like that displayed in figure 20 and these lines can be connect with logical operators AND and OR.

## APPLICATION FRAMEWORK

The foundation of these applications is STATPROBE's generic SAS/AF classes and templates. The class library and template library development was independent from the development of the clinical aspects of the system.

These classes and templates are used so that all the database applications developed at STATPROBE will have similar characteristics. Using the classes and templates also allows for a rapid application development environment. Figure 21 shows some entries in the class library.

```

CLASS      STANDARD VEE open button      14/06/88
CHOOSEIN   CLASS      CHOOSEIN CLASS      06/02/88
CHOOSEIN2  CLASS      Tiny shesher      06/02/88
CHOOSEIN3  CLASS      Choose 1 from between two lists 06/02/88
CHOOSEIN4  CLASS      Tiny shesher      06/02/88
CHOOSEIN5  CLASS      Choose 1 to display a list of libraries 06/02/88
COMBOIN    CLASS      Combobox with library driver      06/02/88
COMBOL     CLASS      Combobox with alias driver      06/02/88
COMBOL2    CLASS      Combobox 1 to display a list of datasets 06/02/88
COMBOL3    CLASS      Combobox with dataset driver      06/02/88
CONTRLIST  CLASS      List of data set labels      03/06/88
COPYLIST  CLASS      Filter driver      06/02/88
DATA_LAST CLASS      Button - Last Record      01/08/88
DEMOIN2    CLASS      Button - Next Record      01/08/88
DEMOIN3    CLASS      Button - Print Record      01/08/88
DEMOIN4    CLASS      Data form control button set same copy 01/08/88
DEMOIN5    CLASS      Data form control button set same copy 01/08/88
DEMOIN6    CLASS      Data form addt mode control button set 06/02/88
DLIN1     CLASS      List of data set labels v. S.I.I 03/06/88
DLIN2     CLASS      List of data set labels v. S.I.I 03/06/88
ENTRDATE  CLASS      ENTRDATE CLASS      12/06/88
ENTRTEXT  CLASS      Headed Text Entry      12/06/88
FILTER1    CLASS      Filter Class      06/02/88
FILTER2    CLASS      Filter Class for specific batch vers 06/02/88
FILTER3    CLASS      Filter Class      06/02/88
FILTER4    CLASS      Filter Class with library selection 06/02/88
IIMAGEIN  CLASS      Image Class      06/02/88
INCLNT    CLASS      Combobox label-entry box      01/08/88

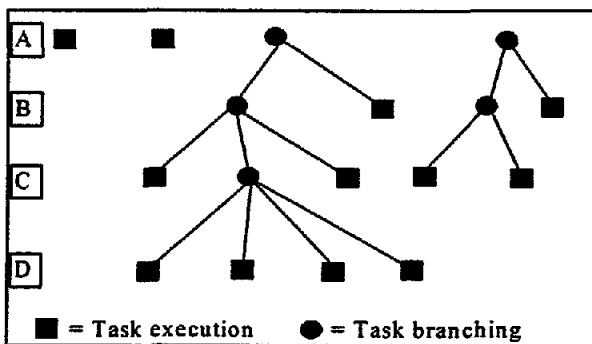
```

**Figure 21. Foundation class library**

One example of a useful class for database applications is the combination form and table object to manage the relationship between a parent and child table. Instantiations of this object have been shown in figures 10, 13, and 17. An object of this class is parametrized using the parent and child data sets, and the name of the administration frame that is called to modify the child data. The classes and templates provide essential processes such as a login procedure and basic utilities.

An application framework consists of a set of tasks. Tasks are organized in a hierarchical manner. Higher level tasks can branch to additional sub-tasks. For example, an application task might be "Database Administration", however this task may consist of several tasks including "Setup Database", "Modify Database", "Copy Database", "Browse Database", and "Print Database". Therefore, when the user tells the application to perform a specific task, the application will either branch to a set of sub-tasks or complete the execution of the task.

The natural structure for this task management is a tree (see diagram 1). Each node represents a task. Each node or task when selected, either branches to additional tasks or performs a specific task. In the example depicted in the diagram, a single task branches to as many as four levels in the task hierarchy.



### Diagram 1. Application Schematic

Note that the navigation through such a system, i.e. branching from node to node and executing actions, is independent of the

actual content at each node. STATPROBE has developed a general system that models this process with a user interface commonly referred to as a "switchboard". Successive switchboards are represented by parameters in the system.

To realize this application structure in SAS/AF, STATPROBE developed a sub-class of the Image Icon class. The sub-class is referred as the Menu Image Icon class. A switchboard in an application consists of a set of Menu Image Icon objects. Every application is prototyped by defining a set of switchboards. To complete an application, it is only necessary to program the frames for Menu Image Icon objects that execute a specific function as opposed to branching to another switchboard.

The Menu Image Icon class has four new instance variables listed in table 9.

| Instance Variable | Description                                        |
|-------------------|----------------------------------------------------|
| DISP_APP          | Catalog entry to display                           |
| SLIST             | Slist entry corresponding to a Menu Image icon set |
| SYSLEV            | Display attribute for system administration        |
| TASKLEV           | Display attribute for task levels                  |

**Table 9. Menu Image Icon Class instance variables**

Designing an application front-end has been reduced to a matter of minutes. The developer can produce several prototypes and meet with the users to discuss the direction of the application before investing substantial development time.

Another advantage of this framework arises during the application development process. There are many cases where the specific frame behind a Menu Image Icon object is general enough that the frame can be saved as a template and reused in other applications.

Other generic functions are independent of application specifics. In our framework, all independent functions include:

- Login system
- Change password
- Application node navigation
- Error log
- Quit to SAS
- Exit

All applications use an entry called MAIN.SCL. This module is a generic application driver and the main entry to launch the application. As an application prototype is developed, the code in main.scl may be modified to accomplish design changes at the high level function of the application, however it is typical for this program to remain static throughout the application development. Main.scl has four primary sections consisting of

- Initialize DBFS
- Initialize system environment
- Login user
- Execute application

The standard library DBFS is included in all applications. The acronym DBFS stands for "Database File System". The DBFS consists of the user table, project table, assignments table, and any other specific files necessary to track data for an application. The DBFS files typically reside in a sub-directory DBFS of the application directory. If an application is very basic and does not need to track system data in a database file system, the statements to initialize the DBFS library are removed from main.scl.

The section of `main.scl` that initializes the system environment invokes a method called `environ`. Each application must always have the `environ` method modified for:

- Application name
- Version control number
- Primary switchboard

These parameters are stored in the application's local environment list. This technique facilitates the development of generic templates. When the code for a template is written, global application information can be referenced in the environment list. Typical applications store additional data in the environment list, such as standard file structures for projects.

Listing 1 shows the environ method for a basic employee candidate survey application used by STAPROBE for

- Completing surveys on candidates
- Reporting, summarizing, and graphing results
- Modifying survey questions

```
ENVIRON:
Method;
  envlist=envlist('L');

  *** Application name,
    version, and prime menu slist ***;
  appname='STATPROBE CANDIDATE SURVEY';
  version='1.0A';
  appmenu='cssprime';

  rc=insertc(envlist,appname,-1,'APPNAME');
  rc=insertc(envlist,version,-1,'VERSION');
  rc=insertc(envlist,appmenu,-1,'APPMENU');
  rc=insertn(envlist,0,-1,'IN_ERROR');
Endmethod;
```

Listing 1. Environ method

Listing 2 shows statements in main.scl that reference the environment list after the environ method is invoked. The application title is passed to the login frame to display the application name (see figure 25 in the next section). The login system requires a standard user table with fields representing user id, user name, and password.

```
*** Login ***;
call display('login.frame',
  rc,getnitemc(envlist('L'),
    'APPNAME'));
```

Listing 2. Login section of main.scl

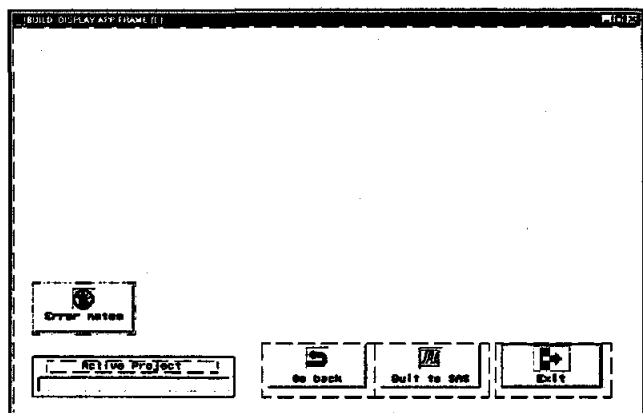


Figure 22. App.frame

The final statement in main.scl performs a call display to execute the application (see listing 3). The application primary switchboard, application name, and version are passed to the frame app.frame. This frame is a generic application node navigator. Figure 22 displays app.frame. There are basic controls for viewing error notes, navigation along the node tree, quitting to SAS display manager, and exiting the application. All these controls are standard to every application. There is a large blank area above the controls that is reserved to display switchboards.

```
*** Execute ***;
call display('app.frame',
  scan(screenname(),1,'.')||'|'.'||
  scan(screenname(),2,'.')||'|'.'||
  getnitemc(envlist('L'),
    'APPMENU')||'|'.slist',
  getnitemc(envlist('L'),
    'APPNAME')||'|'.v.'||
  getnitemc(envlist('L'),
    'VERSION'));
```

Listing 3. Application execution in main.scl

An additional level of standard functions are typically included in STATPROBE's applications. These include

- User administration
- Project administration
- Project selection
- Active project display

Figure 23 shows the primary switchboard of an application.

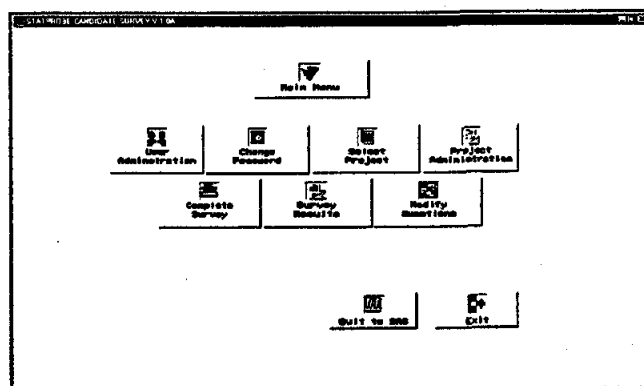


Figure 23. Primary switchboard

Figure 24 shows a second level switchboard. This switchboard displays after the "Survey Results" menu image icon object on the primary switchboard is selected.

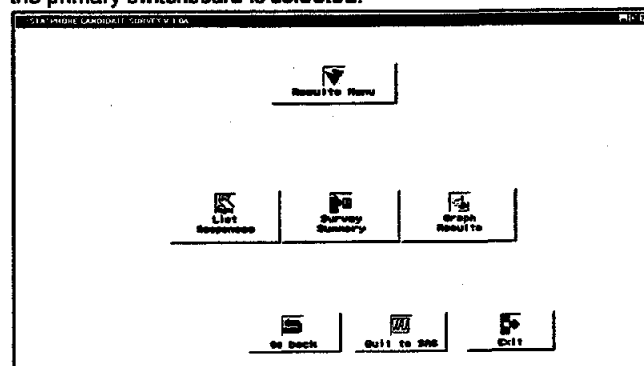


Figure 24. Second level switchboard

## REUSABLE TEMPLATES AND CLASSES

This section describes some of the template frames and classes that are usable in all applications. The previous section described the login procedure. The login frame is displayed in figure 25. The code for the INIT section of the frame is displayed in listing 4. Notice that the application name is a parameter in the code and is used to set the title of the login frame. If the variable appname is blank, then the title of the login frame will simply be "Login". Each application uses a standard structure for the system user table. This allows the login frame to be used by all applications. Also, if the login frame is ever changed or updated, it will automatically be updated for all applications.

Figure 25. Login screen with title

The code for the login frame also shows how the error handler works. Although the error handler is not a graphical component of applications, it consists of an SCL entry and is considered a template since it is usable in all applications. If the login frame fails to open the user table, the error handler is invoked. The error handler writes information to a list entry in the *errorlog* catalog for an application. It allows the developer to track user problems and debug the system.

```
INIT:
  *** Assume failure ***;
  rtn=0;

  *** Set application title ***;
  call send(_frame_,
    '_SET_TITLE_', 'Login '||appname);

  *** Make a list to hold the
    user id and set key ***;
  userlst=makelist();

  *** Open USER data set ***;
  user=open('dbfs.user');
  If not user then do;
    call method('ehandler',
      'errlog',screenname(),
      _status_,event(),
      sysmsg(),dsid,_self_);
    call notify('ok','_GRAY_');
  End;

Return;
```

Listing 4. INIT section of login.frame

Figure 26 shows the SELECTOR template. This template is typically called from a form editor to quickly locate a record and "pop" that record into the form. However, the selector can be used by procedure to select a data set record and perform an action on that record.

The data set that is being edited in the form is passed as a parameter to the selector frame. The selector frame displays the data set in a data table. When the user clicks on the desired record, the record number of the selected record is returned to the calling procedure. The calling procedure receives this information and performs the action on the record. Listing 5 shows the code for the selector template.

|    | NAME   | SEX | AGE | HEIGHT | WEIGHT |
|----|--------|-----|-----|--------|--------|
| 1  | Alice  | F   | 13  | 56.5   | 84     |
| 2  | Becka  | F   | 13  | 65.3   | 98     |
| 3  | Gail   | F   | 14  | 64.3   | 90     |
| 4  | Karen  | F   | 12  | 56.3   | 77     |
| 5  | Kathy  | F   | 12  | 59.8   | 84.5   |
| 6  | Mary   | F   | 15  | 66.5   | 112    |
| 7  | Sandy  | F   | 11  | 51.3   | 50.5   |
| 8  | Sharon | F   | 15  | 62.5   | 112.5  |
| 9  | Tammy  | F   | 14  | 62.8   | 102.5  |
| 10 | Alfred | M   | 14  | 69     | 112.5  |
| 11 | Duke   | M   | 14  | 63.5   | 102.5  |
| 12 | Guido  | M   | 15  | 67     | 133    |
| 13 | James  | M   | 12  | 57.3   | 83     |

Figure 26. Selector template

The INIT section of selector.scl establishes the data set for the data table. After any action occurs on the frame, the MAIN section gets the current row then halts immediately. This row number is returned to the calling procedure.

```
INIT:
  call send(_frame_, '_GET_WIDGET_',
    'tbl1',tblid);
  call send(tblid, '_SET_DATASET_',
    dataset);
  If keyname^='_BLANK_' then do;
    vals=makelist();
    rc=insertc(vals,keyval,-1,keyname);
    call send(tblid, '_SET_KEY_',
      rc,keyname,'EQ','_SCROLL_',vals);
    rc=dellist(vals);
  End;
  If row^=. then
    call send(tblid,
      '_GOTO_ABSOLUTE_ROW_',row);

  call send(_frame_,
    '_SET_TITLE_',dataset||
    ' Record Selection');

Return;

MAIN:
  call send(tblid,
    '_GET_CURRENT_ROW_NUMBER_',row);
  _STATUS_='H';

Return;
```

Listing 5. Code for selector template

A template that is reusable in many applications is an error log. The allows the system to maintain an error log to help in debugging and validation. If a system error occurs, a record is written to the error log. The user can post a note on this record describing the circumstances of the error. This helps the application developers track errors and maintain the system. Table 10 shows the structure of the error log data. This data is written to a SAS catalog and stored using SLIST structures.

| ATTRIBUTE | DESCRIPTION                  |
|-----------|------------------------------|
| Err_date  | Date of the error            |
| Err_time  | Time of the error            |
| User_id   | User identification          |
| Err_msg   | Error code                   |
| Err_src   | Source location of the error |
| Err_info  | Processing information       |
| User_msg  | User posted message          |

Table 10. Error log table

The class library provides tools for the user to construct, modify, copy, and view data. In addition, typical actions like appending, sorting, and printing data are handled by classes.

The filter object used in the frame shown in figure 5 is an example of a generic tool derived from the class library. When this object is instantiated, the developer specifies the action to be taken when filter criteria are complete. For example, the action might be browse data, print data, copy data, or some other database operation. The button labeled "Browse" in figure 5 is set dynamically to describe the action that is executed when the button is clicked.

Several other classes contribute to the operation of the system. Numerous standard command buttons, list movers (see figure 27), data navigation controls, and list managers are all classes used by many applications. The list mover object in figure 27 is an object from the *chooser* class. This class is used in many applications to select items from a list.

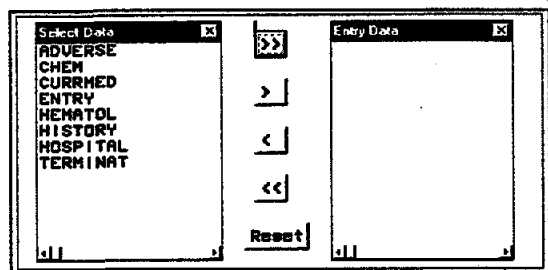


Figure 27. List Mover Object

## CONCLUSION

SAS/AF provides many classes for developing solid applications. More important is the ability to derive sub-classes. In particular, the composite class allows classes to be combined to create very complex classes that can be re-used in many applications.

STATPROBE has developed several database applications using SAS/AF classes. The current data management system is at version 3.0. The system was implemented on OS/2 and has been ported to Windows 95. New classes available in SAS 6.11 have immensely enhanced the usability of SAS/AF based applications.

This paper presented applications that use object-oriented programming techniques and an application development framework in SAS/AF. We discussed the use of classes and

templates that have been developed at STATPROBE. Using these classes and templates, a typical application can be prototyped in a couple of days and the user can be involved in the software design phase to a greater degree. A basic application can be fully developed and tested in less than one week. More complex applications typically take one month to two months to fully compose.

There are many templates that have been developed by STATPROBE, too numerous for the scope of this paper. Additional templates include:

- Data entry form
- Relational entry form
- Informational dialog
- Question dialog
- Chooser
- Filter
- Browser
- Application builder

This library of templates took two months of development time. As STATPROBE develops more applications, our library of templates continues to grow.

## REFERENCES

SAS Institute, Inc. (1993), *SAS/AF Software: FRAME Entry, Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute, Inc. (1994), *SAS Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

Haske, Carl R. (1995), *Using SAS/AF and Frame Entry to Access Data*, SUGI 20 Proceedings, Cary, NC SAS Institute Inc.

Haske, Carl R. (1995), *Developing SAS/AF® Applications for Reviewing Clinical Data*, Proceedings of MWSUG '95, 5-9, Cleveland, OH.

Haske, Carl R. (1996), *Taking Advantage of inheritance in Developing SAS/AF® Applications*, SUGI 21 Proceedings, Cary, NC SAS Institute Inc.

Haske, Carl R. (1996), *A Clinical Data Management System in SAS®*, SUGI 21 Proceedings, Cary, NC SAS Institute Inc.

Haske, Carl R. (1995), *Clinical Data Management Using SAS/AF®*, Proceedings of MWSUG '96, Minneapolis, MN.

Haske, Carl R. (1995), *SAS/AF® Application Development Using Templates*, Proceedings of MWSUG '96, Minneapolis, MN.

## ACKNOWLEDGMENTS

SAS and SAS/AF are registered trademarks of SAS Institute Inc. in the USA and other countries. IBM and OS/2 are registered trademarks of International Business Machines Corporation. Windows 95 is a registered trademark of Microsoft Corporation. ® indicates USA registration.

## AUTHORS ADDRESS

Carl R. Haske, Ph.D.  
 STATPROBE, Inc.  
 3885 Research Park Drive  
 Ann Arbor, MI 48108  
 (313) 769-5000 x115  
 E-Mail:  
 • 72700,1075@compuserve.com