AUDIT TRAIL PLUG-INS FOR SAS® SOFTWARE APPLICATIONS

Clark Roberts, Decision Analytics
Deborah Testa, PhD., Qualex Consulting Services
Russell Holmes, Synteract, Inc.

ABSTRACT

Secure access and change tracking are two important factors to consider when developing software applications today. A combination of SAS and operating system security measures can be implemented to easily secure a database. Tracking changes in SAS, on the other hand, has always required custom programming, a costly task. SAS Version 7 will have some ability to track changes, but may not be enough to suit the needs in some applications. SynAudit™ was developed with the goal of providing Plug-n-Play modules that could be easily integrated into any SAS application to track which records and fields were changed, who made the changes and when they were made. It also provides push button access to the change history of the record currently being edited, as well as, dataset level rollback capabilities The paper will discuss the steps required to implement SynAudit into an application and provide a simple example for demonstration.

INTRODUCTION

In the past 10 years several authors in the SAS Observations® magazine, SAS Views® and SAS Users Group International Conference Proceedings have discussed ways to provide SAS software systems with audit trail capabilities (Boyle, 1993; Haske, 1997; Rinehart, 1989; Rosenberg, 1998; SAS Institute, 1998; and Witko, 1996). Synteract Inc. developed SynAudit as a tool for tracking changes to database records. It uses SAS data sets to store information regarding the fields that were changed, who made the changes, when and why they were made and the system module that was used to initiate the changes.

SynAudit can be applied to standalone SAS/FSEDIT systems or to multi-user systems executed in a SAS/SHARE® environment. Access to the change history tables for viewing or printing is enabled using the SAS VIEWTABLE facility from within the FSEDIT session, as well as externally through a SAS/AF® selection list application. SynAudit currently runs under version 6.12 of the SAS System in Windows® 3.1, 95, 98 and NT, and a prototype system was implemented in HP-UX.

Although SAS Institute has long indicated that audit capabilities for SAS data sets would be added sometime in the future (Clifford, 1991), the implementation in SAS Version 7 has not lived up to expectations. Unlike the audit functionality built into SAS Version 7, audit trail information in SynAudit is not lost when the data set is modified. SynAudit does not rely on the observation number to link audit information to the underlying data set like the native SAS Version 7 implementation. Instead it creates a utility variable named _AUDID_ to facilitate the linkage and is thus impervious to sorts or merges that would change the observation number and in turn break the linkage.

Since SynAudit is not part of the native SAS data set structure, it does require additional programming to maintain audit information integrity for changes made outside the FSEDIT environment. However, modules are included with SynAudit that provide some assistance in maintaining the audit datasets.

IMPLEMENTATION

SynAudit is a Plug-n-Play application intended to extend the functionality of traditional FSP or AF data entry applications. It consists of a set of SAS macros and SAS System Component Language (SCL) modules that are 'plugged' into new or existing SAS/FSP programs, enabling the maintenance of a history of changes to SAS data sets. After installation of the SynAudit SAS catalog, the first step is to create a new SAS table (data set), the audit table, in the change history directory for each data set in the system for which changes will be tracked. The audit table will mirror the contents (fields) of the primary data set with the addition of the following fields:

_AUDOPER:	The system userid of the person logged on when the changes were made
_AUDTRDT:	The date and time the changes were made
_AUDTTYP:	The audit transaction type, Initial, Add, Change or Delete
_AUDRTYP:	Indicates whether the audit record pertains to the record values before (Old) or after the change (New).
_AUDSRCE:	The system module that initiated the change.
_AUDNOTE:	The reason the change was made. Optionally entered by the user.

A field named _AUDID_ will be added to both the original and audit data sets, that is used to link the corresponding records. An index will be created in the audit table for _AUDID_, and other audit related variables, to improve system performance when retrieving audit history information. The _AUDID_ field is the only change required to the structure of the data set being audited.

A table, named AUDIDTAG, is maintained that contains the current highest value for each data set's _AUDID_ variable. When a new record is added, this value is incremented by one and assigned to the _AUDID_ variables in the base and audit data sets. A SAS macro, *AUDITBLD.SOURCE* in the SynAudit catalog, is provided to assist in the creation of the AUDIDTAG data set and the audit tables. A copy of this macro is displayed in exhibit 2.

The next step is to 'plug' the SynAudit modules into the FSEDIT SCL programs. It is truly Plug-n-Play for applications using the SAS/FSP FSEDIT procedure. With FSEDIT, users can modify individual records in a SAS data set and SynAudit provides the add on tools to capture these changes. To implement SynAudit in an FSEDIT SCL program, a call to the SYNAUDIT macro is placed at the end of the FSEDIT SCL program. The arguments to this macro are defined by the application developer for the purpose setting up and initializing the SynAudit environment at compile time.

The SYNAUDIT macro then monitors user actions, categorizing them by type (add, change, delete) and writing data about these actions to a specialized SAS data set. This monitoring and recording is entirely unobtrusive to the user, who simply uses the FSEDIT application as usual.

The SYNAUDIT macro must be compiled and available when the FSEDIT SCL program is compiled. This can be accomplished by putting it in the SASMACR catalog and specifying the MSTORED and SASMSTORE= SAS system options. The SYNAUDIT.MACRO entry can also be copied to the WORK.SASMACR catalog during system startup. The latter is the approach used in the SynAudit demonstration system (refer to the listing in exhibit 1).

The purpose of the macro is to evaluate changes made to the record and call the appropriate SynAudit SCL method that will record the changes. Statement specific LINK statements need to be added to the standard FSEDIT sections (FSEINIT, INIT, MAIN, TERM, and FSETERM) in the FSEDIT SCL program that will tie the application to sections in the included macro. A template, named *AUDITINC.SOURCE*, is included in the SynAudit catalog that can be used as a basis for building the FSEDIT SCL program. It requires only a few modifications to become a fully operational SCL program.

Change history information can be accessed for viewing or printing through a set of included SAS/AF programs that allow the user to select the SynAudit table to view. The audit history for the current record on the SAS FSEDIT screen can be accessed by issuing the HIStory command on the command line or by selecting the File -> History option from the pull down menus (if the *AUDPMENU.PMENU* pmenus are used). Another option requires the user to enter a reason for the change. This function is provided by the *SYN_NOTE.FRAME* entry.

Once implemented, the SynAudit facilities can be accessed by assigning the appropriate SAS LIBNAMES and initializing a set of SynAudit specific global macro variables. An example of the required SAS statements can

be found in exhibit 1, which contains the SAS autoexec program that starts the SynAudit demonstration system used in the EXAMPLE section later in the paper.

COMPONENTS

The SynAudit SAS/AF catalog consists of several modules used in the construction and execution of the audit trail function. These can be categorized as SAS/FSEDIT components, SAS/AF SCL modules, SAS/AF utilities and SAS macros. These modules rely on the settings of specific global macro variables, assignment of required SAS libnames, and the adherence to variable naming conventions that avoid the use of SynAudit reserved names.

SAS/FSEDIT COMPONENTS

AUDITINC.SOURCE: A template for a SAS FSEDIT SCL program consisting of all the standard FSEDIT sections (FSEINIT, INIT, MAIN, TERM and FSETERM) and containing links to SCL sections defined in the *SYNAUDIT* SCL macro that drive the audit function. A call to the SYNAUDIT macro is contained at the end of the template. Only a few minor modifications are required to the macro call to make the template fully functional in a different application. A listing of this code can be found in exhibit 3.

SYNAUDIT.MACRO: A compiled macro containing SAS FSEDIT SCL sections that are referenced from the standard SAS FSEDIT SCL sections, providing the calls to the *SYNAUDIT* SCL method that ultimately handles the audit trail transactions. It is provided without source and requires no modification.

SAS/AF METHODS

SYNAUDIT.SCL: This program contains SCL methods that provide the low -level transaction processing for the audit trail implementation. One method maintains the AUDIDTAG table and retrieves _AUDID_ variable values for new records, while the others provide the mechanism for adding information to each of the audit data sets. It is provided without source and requires no modification.

SAS/AF UTILITIES

AUDPMENU.PMENU: A set of pull down menus that provide basic SAS/FSEDIT command access and access to the HISTORY command for viewing the audit trail history. Source is provided with this module in *AUDPMENU.SOURCE* for individual customization if desired.

AUDSHOW.FRAME: A complete SAS/AF frame entry, including the corresponding SCL, that can be added to a SAS/AF system to provide viewing access to the SynAudit audit data sets. This module requires no modification, but is provided with source so that the developer can customize it, if desired.

REPLIST.FRAME: A SAS/AF frame entry that provides a selection list for SynAudit reports. This module requires no modification, but is provided with source so that the developer can customize it, if desired.

SYN_NOTE.FRAME: A SAS/AF frame entry that provides the data entry functionality to enter a reason for the change being made to the data set. This function can be turned on and off for the entire SynAudit implementation by setting the *AUDNOTE* global macro variable to 0 or 1 (off/on). The text that is displayed in the frame can be set in the NOTETITL global macro variable. This module requires no modification and is distributed without SCL source.

BASE SAS MODULES AND MACROS

AUDITBLD.SOURCE: A SAS macro that can be tailored to build the basic SynAudit audit trail infrastructure initially. This includes the creation of the corresponding audit data sets, the AUDIDTAG data set and the addition of the _AUDID_ variable to the data sets being audited. For existing data sets, it will also assign initial values to the audit variables and initialize the AUDIDTAG data set with the highest value of the _AUDID_ variable for each data set. The listing of this module, which was modified for the SynAudit demonstration system, is located in exhibit 3.

_ACTVTY.MACRO: A SAS macro that produces an activity report on changes made to a selected data sets.

_AUDPRN.MACRO: A SAS macro that produces a formatted print of an audit data set.

GETAUDID.MACRO: A SAS macro that provides a tool for managing the _AUDID_ variables in the AUDIDTAG data set from base SAS programs.

SYNAUDUP.MACRO: A SAS macro that is used to update an audit data set from within a base SAS program where changes are made to the base data set.

REQUIRED GLOBAL MACRO VARIABLES:

AUDITPW: Contains the SAS data set password used on the audit datasets. *AUDITPW* is the default macro variable name used by the *SYNAUDIT.MACRO* module. If another macro variable is used to contain the password for the audit data sets then that name must be explicitly specified in the *SYNAUDIT* macro call.

AUDNOTE: Indicates whether or not the user should be asked to enter a reason for changes made to a data set. A value of 1 indicates the user should.

MAXTRIES: Used in applications where SAS/SHARE is used for concurrent access to data sets. It specifies the number of times that SynAudit should try to lock an audit data set so that an audit record can be written to it, before it aborts the write operation. 1000 is probably a reasonable value to use since SynAudit only requires access to an audit data set for a short amount of time to either write out a new record or make a copy for viewing, any more than that number of tries would indicate a problem, possibly someone accessing the data set outside the current application.

NOTETITL: The text that is displayed in the *SYN_NOTE.FRAME* entry when the audit note function is active (SYNNOTE macro variable equal to 1).

SYNAUDIT: Indicates whether the audit trail function is on (1) or off (0) for an application. For SynAudit to function this macro variable must be set to 1. If it is not, then no audit information will be tracked even if SynAudit functionality was coded into the FSEDIT programs. Conversely, the SynAudit functionality can be temporarily immobilized by setting this macro variable to 0.

USERID: Contains the logon ID for the current user. This is the default macro variable name used by the *SYNAUDIT.MACRO* module. If another macro variable is used to contain the user ID, then that name must be explicitly specified in the *SYNAUDIT* macro call.

REQUIRED SYNAUDIT LIBNAMES:

The only additional SAS libnames required for SynAudit are for the directories that contain the audit data sets and the SynAudit catalog.

SYNAUDIT RESERVED VARIABLE NAMES:

_AUDOPER: The system userid of the person logged on when the changes were made

_AUDTRDT: The date and time the changes were made.

_AUDTTYP: The audit transaction type, Initial, Add, Change or Delete

_AUDRTYP: Indicates whether the audit record pertains to the record values before (Old) or after the change (New).

_AUDSRCE: The system module that initiated the change.

_AUDNOTE: The reason the change was made.

AUDID: A tag that links records in the base data sets to corresponding records in the audit data sets.

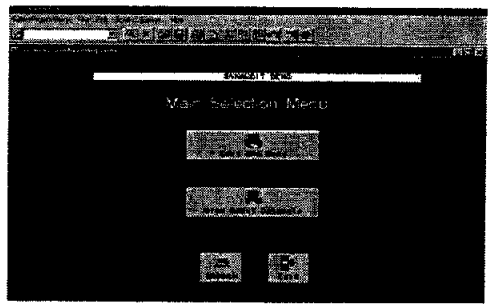
RESERVED DATASET NAMES:

AUDIDTAG: The name of the dataset in the audit data set directory that contains the current *highest* value for each data set's _AUDID_ variable.

EXAMPLE

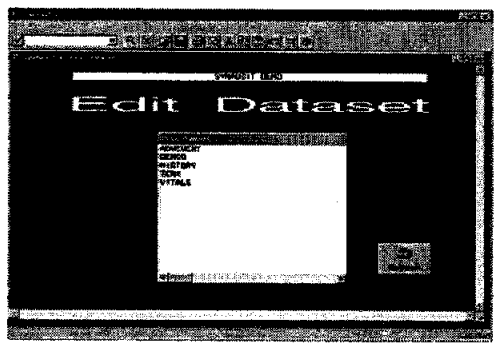
We developed a simplified SAS/AF application consisting of a main menu and selection lists for choosing a data set to edit and an audit data set to view in order to demonstrate the functionality of SynAudit. An abbreviated version of a clinical trial demographics data set will be used as an example of the SynAudit process.

The SAS autoexec file for the example, *SynAudit Demo.sas* shown in Exhibit 1, sets the required SynAudit macro variables, assigns libnames, performs other functions required by SynAudit and initiates a SAS/AF session by executing the main menu frame for the example. Selecting the SynAudit Demo icon from the desktop executes the *SynAudit Demo.sas* program and displays the SynDemo Main Selection Menu:

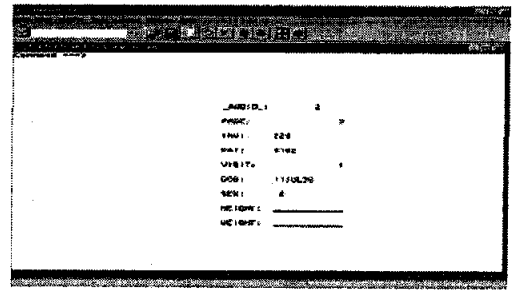


To initially setup the example data sets for the SynAudit environment, the developer would select the GoBack icon from the SynAudit Demo Main menu, which exits the SAS/AF session and returns to the Program Editor window. The AUDITBLD.SOURCE routine from the SynAudit catalog was modified (refer to exhibit 2) and executed to prepare the clinical data sets and build the initial audit data sets and the AUDIDTAG data set. The AUDIDTAG data set contains the highest value for the _AUDID_ variable for each of the audit data sets.

The next step for the developer is to add the required SynAudit SCL code to each of the clinical data sets. For the example only the FSEDIT screen for the DEMOG data set will be built. Return to the SynAudit Demo Main Menu by issuing the AF command from the command line and select the Edit SAS Data set icon. The Edit Data set selection list is displayed:



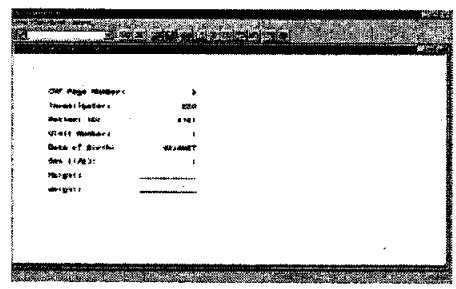
From the Edit Data set selection list, select DEMOG and the default FSEDIT screen for the DEMOG data set is displayed.



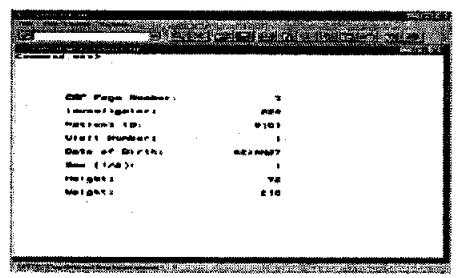
Entering MOD on the command line will open the FSEDIT menu selection screen. Select item 2, Screen Modification and Field Identification, and erase the line that contains the _AUDID_ variable since it is not a variable defined in the original data set. At this time the screen can be further customized for appearance. During the normal exit procedure for this screen answer UNWANTED when requested to put the cursor on the _AUDID_ field. If other screen modifications were made then the new locations affected variables will have to be identified at this time.

Next select item 3, Edit Program Statements and Compile, the program edit screen will be empty. Select File -> Open -> Read object and type **std.synaudit.auditinc.source** in the *Enter object name to copy* box. This brings in the SynAudit FSEDIT template, refer to exhibit 3, which contains all the SCL required to implement a SynAudit audit trail for the data set. Items that require modification are denoted by a string of lower case x's, i.e. xxxxxxxx. These include some information in the header and comment blocks, the *WNAME* element, and the SynAudit macro call at the end of the SCL. The modified AUDITINC.SOURCE for the DEMOG data set is displayed in exhibit 4. If the modifications were entered correctly the SCL program should compile and the DEMOG data set is now ready to use with audit trail functionality.

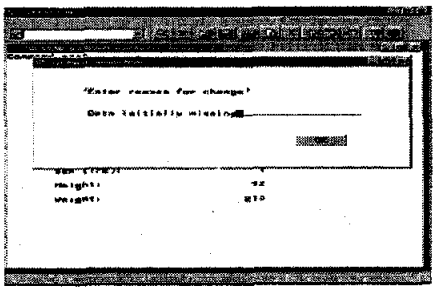
Return to the data entry screen for the DEMOG data set by issuing an *END* command. The revised screen now appears as:



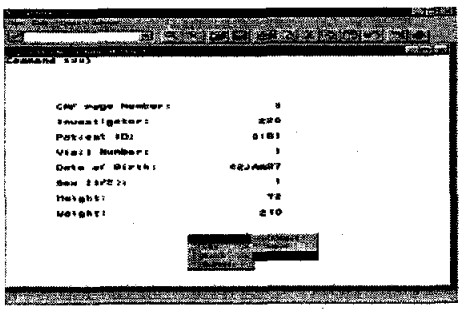
To test the audit functionality, modify a record by entering values for HEIGHT and WEIGHT.



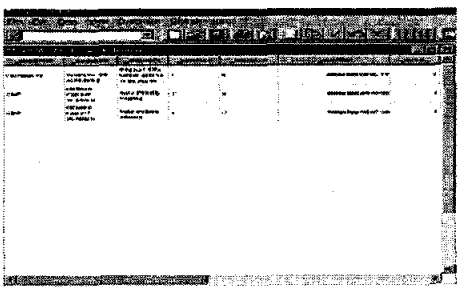
Now select File -> Save to save the observation. If the AUDNOTE function is turned on, the following screen is displayed so a reason for the change can be entered *into the* audit data set.



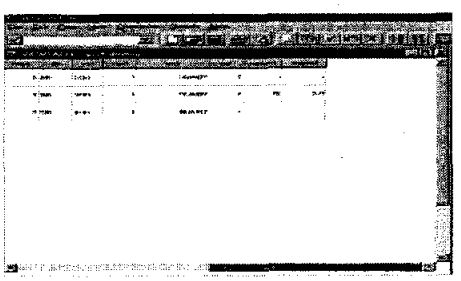
Once the observation has been saved, the change history for the observation, including the modifications just made, can be viewed. From the pull down menus, select File -> History:



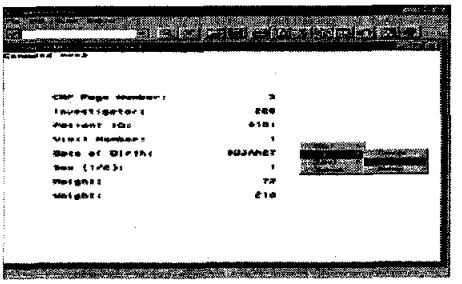
A VIEWTABLE window containing a working copy of the observations in the DEMOG audit data set that correspond to the current DEMOG clinical data set is displayed.



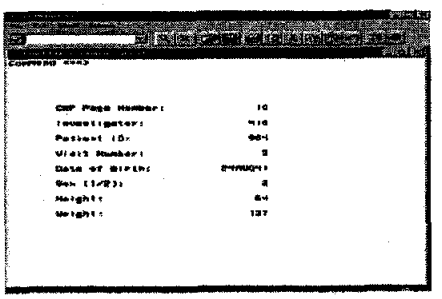
Scroll across to view the entire record. Note that there are now 3 entries for this record, the initial audit setup record and a before and after modification record.



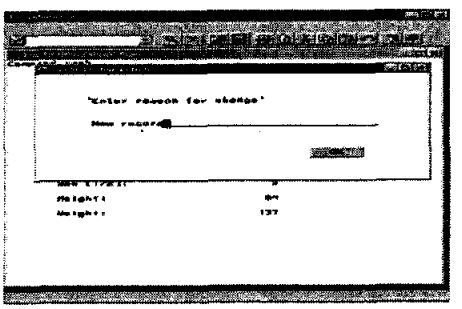
Close the VIEWTABLE window and return to the DEMOG data entry screen. Next we'll add a new observation to the data set. Select Edit -> Add from the pull down menus:



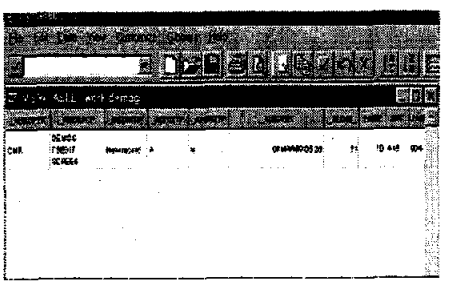
An empty DEMOG data set record will be displayed. Enter data into some of the fields.



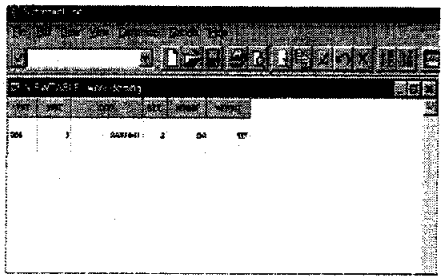
Select File -> Save from the pull down menus and the SynAudit Change Note Window is displayed. Enter a reason for adding the new record,



Now select File -> History from the pull down menus and a VIEWTABLE window containing a working copy of the observations in the DEMOG audit data set that correspond to the current DEMOG clinical data set record is displayed.

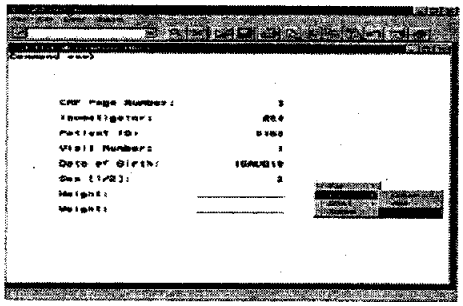


Scroll across to view the rest of the record.

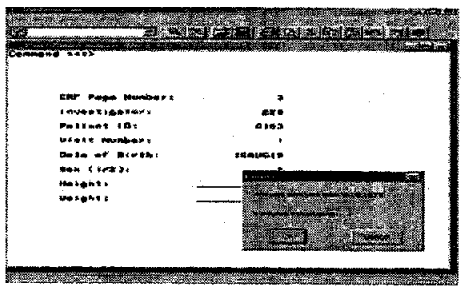


There is only one record in the DEMOG audit data set because it was a newly added record as indicated by the AUDRTYP, AUDTTYP and AUDTRDT variables. Also a unique _AUDID_ value has been added that corresponds to the one in the record added to the base DEMOG data set.

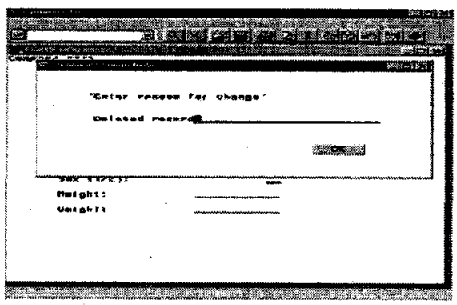
Close the VIEWTABLE window and return to the DEMOG data entry screen. Next we'll delete an observation from the data set. Move to Observation number 3. Select Edit -> Delete from the pull down menus.



Reply yes to the 'Are you sure you want to delete the selected observation' message.



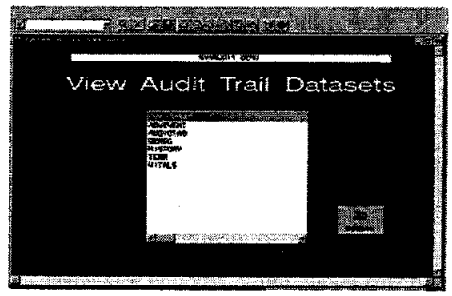
The record is now empty. Select File -> Save from the pull down menus. Enter a reason for the deletion.



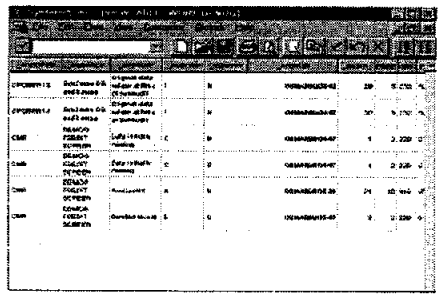
The previous observation is now displayed. Since the deleted observation is no longer in the data set its audit trail can no longer be accessed from within the FSEDIT session. To view audit trail information for deleted records, you must invoke the View Audit Data Sets icon from the Main Selection Menu.

Select File -> GoBack from the pull down menus to exit the DEMOG data entry screen. Select the GoBack icon from the Edit Data Set menu to return to the Main Selection Menu.

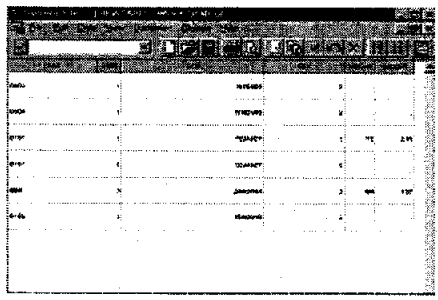
Now select the View Audit Data sets icon and the View Audit Trail Data sets window is displayed.



Select DEMOG from the selection list and a VIEWTABLE window that contains a working copy of the entire DEMOG audit data set is displayed. Scroll to the bottom to view the most recent additions, or build a VIEWTABLE WHERE clause to access the desired record.



Scroll across to view the rest of the fields in each record.



Note there are four records at the end of the DEMOG audit data set that represent the changes made to observation 1, the addition of a new observation and the deletion of observation 3.

Exit the VIEWTABLE window, select the GoBack icon in the View Audit Trail Data Sets window, then the Exit icon in the SynAudit Demo Main menu to close the application. The audit trail functionality for the DEMOG data set is now operational. The same set of steps could be applied to the other data sets to provide them with audit capability.

The following code excerpt demonstrates how the audit trail is maintained when changes to a dataset are made in a Base SAS program. The example SAS program makes a mass change to field in the history data set, then calls the SYNAUDUP macro to update the changes in the corresponding audit data set.

```
data demodata.history;
     set demodata.history;
     if inv = '226' then inv = '415';
run;

%synaudup(history,
          dslib    = demodata,
          oper     = CROBERTS,
          source   = update.sas,
          note     = Investigator change
         );
```

FUTURE DEVELOPMENT CONSIDERATIONS

A conversion to version 7 of the SAS system is underway and plans to port it to MVS, VM, VMS and UNIX platforms are under consideration.

Modifications are currently under development that will allow SynAudit to be integrated into systems using SAS/AF Form Data and Table Data classes, and the VIEWTABLE object.

Other enhancements under consideration include data set level data recovery and roll-back and observation level roll-back through the history viewing object.

SUMMARY

The primary benefits of providing a system with audit trail functionality are the ability to track the who, what, when, where and why of modifications to a database, and provide an inclusive backup of the database that will allow it to be recreated for any point in time. This functionality can mean the difference in withstanding regulatory and client audits.

SynAudit provides an easily implemented solution for implementing audit trail capabilities to data entry systems based on the SAS/FSP FSEDIT product. It works equally well with native SAS data sets and with relational database tables accessed through SAS views to which the user has update privileges. Once the SynAudit catalog is installed audit trail functionality can be added to a new or existing system in a matter of minutes with a minimum of additional programming.

Unlike the audit function native to SAS Version 7, audit trail information maintained using SynAudit is not lost when the data set is sorted or otherwise re-arranged.

EXHIBITS

Because of the restrictions on the number of pages that can be published in the WUSS proceedings, the exhibits have been omitted. A complete copy of the paper, including the exhibits, can be downloaded from Synteract's web site at www.synteract.com .

REFERENCES AND RELATED READINGS

Boyle, Jennifer L. and Cathy L. Brinsfield (1993), "Beyond Journaling: Data Validation with Dual Data Entry," *Observations: The Technical Journal for SAS Software Users*, 2(3), pp. 4-12.

Boyle, Jennifer L. and Cathy L. Brinsfield (1993), "Beyond Journaling: Rollback, Recovery and Reporting," *Observations: The Technical Journal for SAS Software Users*, 2(3), pp. 6-12.

Clifford, William D. (1991), "Database Features Extend The Scope of SAS/SHARE Software," *Proceedings of the Sixteenth Annual SAS Users Group International Conference*, 16, pp. 335-339.

Haske, Carl R., Ph.D. (1997), "Using SAS/AF for Managing Clinical Data," *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*, 22, pp. 557-562.

Rinehart, John M. (1989), "Techniques for Database Update and Auditing in SAS Software Information Systems," *Proceedings of the Fourteenth Annual SAS Users Group International Conference*, 14, pp.103-112.

Rosenberg, Martin J., PhD. (1998), "Using the Data Warehouse Model to Streamline and Accelerate New Drug and Medical Device Development," *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*, 23, pp. 527-531.

SAS Institute, Inc. (1998), *Advanced SCL Applications (Release 6.07) Course Notes*, Cary, NC, SAS Institute Inc.

Testa, Deborah, Ph.D., Clark Roberts and Russell Holmes (1999), "SynAudit: A Plug-N-Play Audit Trail Tool For SAS/FSEDIT Applications," *Proceedings of the PharmaSUG '99 Conference*, pp. 76-86.

Witko, Tyler (1996), "A Plug and Play Data Entry Audit Trail System," *Proceedings of the Twenty-First Annual SAS Users Group International Conference*, 21, pp. 668-671.

ACKNOWLEDGEMENTS

Microsoft, MS, Windows, Windows 3.1, Windows 95, Windows 98 and Windows NT are registered trademarks of Microsoft Corporation.

SAS, SAS/AF, SAS/FSP, SAS/SHARE, SAS Views and SAS Observations magazine are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

SynAudit is a trademark of Synteract Incorporated.

CONTACT INFORMATION

For inquires or further information the author can be contacted at the following addresses:

Clark Roberts, Principal Consultant
Decision Analytics, A SAS Quality Partner
5663 Balboa Avenue, Suite 400
San Diego, California 92111
Voice: (619) 565-9998
Fax: (619) 565-9627
Cell/Pager: (619) 925-4964
e-mail: dacmr@san.rr.com

Deborah A. Testa, Ph.D.
Seven of Nine Systems, A SAS Quality Partner
Studio City, California
Voice: (818) 980-5901
Fax: (818) 761-6167
e-mail: dtesta@sevenofninesystems.com
URL: www.sevenofninesystems.com
www.qlx.com

Russell Holmes, Vice President
Synteract, Inc., A SAS Quality Partner
187 Calle Magdalena, Suite 100
Encinitas, California
Voice: (760) 634-1002
Fax: (760) 634-2170
e-mail: rholmes@synteract.com
URL: www.synteract.com

QUALITY
PARTNER
SAS INSTITUTE INC.