

Using SAS System and Dataset Options to Control Your Data

David Franklin, Independent Consultant, New Hampshire, USA

ABSTRACT

Adding control to your data is becoming an important task. A lot is at risk when data becomes compromised, and depending on the event, regulators may become involved. This paper looks at a way that SAS provides of controlling who has access to the datasets for modification, and also looks at two alternatives for tracking changes, first through an audit trail and second using SAS dataset "versioning".

INTRODUCTION

Since SAS version 8 was first released the ability to do an audit trail on a SAS dataset has been available to the SAS user. With this new feature and the existing SAS options to control viewing and modification of data, SAS provides the user to have control over their most valuable asset, their data. While there are other applications out there that will track changes of SAS datasets, the features provided by SAS are robust enough for users to at least consider SAS as a possible answer to keeping track of changes to their data.

It must be noted from the onset that the idea of adding an audit trail to a SAS dataset is to track changes and modifications -- rebuilding the dataset deletes any existing audit trail.

SAS OPTIONS

There are four SAS Options that are very good for controlling access to a SAS dataset:

- ALTER= password to alter structure
- READ= password to read
- WRITE= password to modify the data
- ENCRYPT= password to encrypt the data

These passwords can be set when the dataset is first created. While there are the four options, in practice on a controlled network only two of these options are used, ALTER and WRITE, as to put READ and ENCRYPT passwords on a SAS dataset will require all users who want to view the data to have these passwords. Enabling only the ALTER and WRITE password will restrict altering the structure of the dataset, e.g. adding or deleting columns, and also restrict who can modify, add or delete data. The following example demonstrates adding ALTER and WRITE passwords to a SAS dataset that has information on medical monitors:

```
data trial.crew (alter=Dads write=Army label='Wilmington-on-Sea');
  infile cards;
  input name $ 1-17 rank $ 19-36 sn $ 39-43;
cards;
George Mainwaring Captain          61885
Arthur Wilson      Sergeant         50796
Jack Jones         Lance-Corporal  96822
Joe Walker         Private          52925
Frank Pike         Private          57467
James Frazer       Chief Petty Officer 07973
Charles Godfrey    Private          45322
;
run;
```

Running a CONTENTS procedure will produce the following output indicating that the dataset has been protected with an ALTER and WRITE password:

The CONTENTS Procedure

Data Set Name	TRIAL.CREW	Observations	7
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	Monday, October 13, 2008 02:52:45 PM	Observation Length	40
Last Modified	Monday, October 13, 2008 02:52:45 PM	Deleted Observations	0
Protection	WRITE/ALTER	Compressed	NO
Data Set Type		Sorted	NO
Label	Wilmington-on-Sea		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	name	Char	17
2	rank	Char	18
3	sn	Char	5

With the ALTER and WRITE passwords in place only those who are privy to these would be able to alter the structure of the dataset or add, delete or modify the records.

GETTING THE AUDIT TRAIL STARTED IN A SAS DATASET

An audit trail is useful if the dataset is in a state where few if any changes are needed, or it is a dataset that is like a list that is constantly changing. It is not good for the case where the dataset will be rebuilt many times over the life of the dataset – in this case the user should consider versioning which is discussed later in this paper.

You start an audit trail using the AUDIT statement in the DATASETS procedure. The following initiates the audit trail to the dataset CREW:

```
proc datasets lib=trial;
  audit crew (alter=Dads);
  initiate;
  user_var reason $100;
quit;
run;
```

Note that a password had to be entered to alter the dataset. A variable REASON has also been created using the USER_VAR statement so that it is possible to put a note to a record that has been changed. The output from a CONTENTS procedure call will produce a similar output as before but with four extra lines added at the bottom of the header indicating that the dataset does have an audit trail, but it does not contain any of the variables used to store the audit information:

The CONTENTS Procedure

Data Set Name	TRIAL.CREW	Observations	7
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	Monday, October 13, 2008 02:52:45 PM	Observation Length	40
Last Modified	Monday, October 13, 2008 02:56:26 PM	Deleted Observations	0
Protection	WRITE/ALTER	Compressed	NO
Data Set Type		Sorted	NO
Label	Wilmington-on-Sea		
Audit	Active		
Audit Before Image	YES		
Audit Admin Image	YES		
Audit Error Image	YES		
Audit Data Image	YES		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	name	Char	17
2	rank	Char	18
3	sn	Char	5

To get the audit trail variables for the dataset the TYPE=AUDIT option needs to be used in the CONTENTS procedure call, as shown below:

```
proc contents data=trial.crew (type=audit);
run;
```

with the following output:

The CONTENTS Procedure

Data Set Name	TRIAL.CREW.AUDIT	Observations	0
Member Type	AUDIT	Variables	10
Engine	V9	Indexes	0
Created	Monday, October 13, 2008 02:56:26 PM	Observation Length	206
Last Modified	Monday, October 13, 2008 02:56:26 PM	Deleted Observations	0
Protection	WRITE/ALTER	Compressed	NO
Data Set Type	AUDIT	Sorted	NO
Label	Wilmington-on-Sea		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
5	_ATDATETIME_	Num	8	DATETIME19.
10	_ATMESSAGE_	Char	8	
6	_ATOBSNO_	Num	8	
9	_ATOPCODE_	Char	2	
7	_ATRETURNCODE_	Num	8	
8	_ATUSERID_	Char	32	
1	name	Char	17	
2	rank	Char	18	
4	reason	Char	100	
3	sn	Char	5	

Descriptions of the variables created for the audit trail are listed below:

Variable	Description
ATDATETIME	Date/time of modification
ATUSERID	User ID of person making modification
ATOBSNO	Observation number of record being modified (NOTE: if the option REUSE=YES is on then value will always be '0')
ATRETURNCODE	Success/Failure return code (blank if successful, error code from SAS Log if unsuccessful)
ATMESSAGE	Stores the SAS log message from SAS when the modification occurred
ATOPCODE	Modification code. Valid return codes are: DA=Record added DD=Record deleted DR=Copy of record before update DW=Copy of record after update EA=Record added failure ED=Record deleted failure EW=Record update failure.
REASON	User created variable for purpose of noting a reason.

It is possible to suspend, resume and terminate the audit trail for a dataset using the following code examples:

```

proc datasets lib=trial;
*Suspend an audit trail;
  audit crew (alter=Dads); suspend;
*Resume an audit trail;
  audit crew (alter=Dads); resume;
*Terminate an audit trail;
  audit crew (alter=Dads); terminate;
quit;
run;

```

While it is possible to do these actions on an audit trail within a SAS dataset these are not encouraged since it will invalidate the integrity of the dataset.

TIME TO MODIFY SOME DATA WHEN AN AUDIT TRAIL IS ACTIVE

In the Introduction it was indicated that if a SAS Dataset was rebuilt then audit trail information pertaining to that dataset will be lost so some particular strategies have to be adopted when editing data in the SAS dataset.

The three tasks that are generally done in a SAS dataset are:

- modify an existing observation
- add a new observation
- delete an existing observation

Whenever a DATA step SET statement is used the dataset is rebuilt so using this statement in the code to modify the dataset should be highly discouraged. However the SQL procedure with UPDATE, INSERT and DELETE statements is one way of modifying the data while keeping the audit trail. The following examples show how to update the dataset while keeping the audit trail:

```

pproc sql;

*Correct record;
update trial.crew (write='Army')
  set rank='Private',
      reason='Corrected rank, ref. memo 2008-10-13'
  where sn='07973';

*Delete record - update first to add reason;
update trial.crew (write='Army')
  set reason='Disappeared, ref. memo 2008-10-13'
  where sn='52925';
delete
  from trial.crew (write='Army')
  where sn='52925';

*Add new record;
insert into trial.crew (write='Army')
  set name='Cheeseman',
      rank='Private, W.C.',
      sn='98072',
      reason='New member, ref. memo 2008-10-13';
quit;
run;

```

After the amendments the dataset has the following data:

Obs	name	rank	sn
1	George Mainwaring	Captain	61885
2	Arthur Wilson	Sergeant	50796
3	Jack Jones	Lance-Corporal	96822
5	Frank Pike	Private	57467
6	James Frazer	Private	07973
7	Charles Godfrey	Private	45322
8	Cheeseman	Private, W.C.	98072

To list the audit data the option TYPE=AUDIT must be used, as shown in the following SAS code and output:

```
proc print data=trial.crew (type=audit);
  title1 'Audit Trail After Modifying CREW Dataset';
run;
```

```

                                Audit Trail After Modifying CREW Dataset

Obs      name      rank      sn      reason
1 James Frazer Chief Petty Office 07973
2 James Frazer Private 07973 Corrected rank, ref. memo 2008-10-13
3 Joe Walker Private 52925
4 Joe Walker Private 52925 Disappeared, ref. memo 2008-10-13
5 Joe Walker Private 52925
6 Cheeseman Private, W.C. 98072 New member, ref. memo 2008-10-13

Obs      _ATDATETIME_ _ATOBSNO_ _ATRETURNCODE_ _ATUSERID_ _ATOPCODE_ _ATMESSAGE_
1 13OCT2008:15:12:30 6 . dfrankli DR
2 13OCT2008:15:12:30 6 . dfrankli DW
3 13OCT2008:15:12:30 4 . dfrankli DR
4 13OCT2008:15:12:30 4 . dfrankli DW
5 13OCT2008:15:12:30 4 . dfrankli DD
6 13OCT2008:15:12:31 8 . dfrankli DA

```

Note that in the listing from the audit trail it recorded that the record for SN 61885 was changed, the record for SN 52925 was deleted (it was actually modified first to get a reason for deletion in the audit trail) and a new record for SN 98072 was added. Also added was the UserID of the person who added it and the date/time it occurred.

ANOTHER WAY TO TRACK CHANGES, DATASET VERSIONING

As mentioned earlier the audit information is lost when a SAS dataset is rebuilt using a statement like SET in the DATA step. However it is sometimes necessary to keep versions of a SAS dataset. This can be achieved using the SAS DATASETS procedure with the COPY statement and placing that copy into another directory or placing it on another media.

SAS has another feature that is available if “versions” of a dataset have to be made instead of using the audit trail facilities. To set a SAS dataset up for “versions” the GENMAX= option is invoked, as shown in the example below:

```
data trial.crew (alter=Dads write=Army label='Wilmington-on-Sea');
  infile cards;
  input name $ 1-17 rank $ 19-36 sn $ 39-43;
cards;
George Mainwaring Captain 61885
Arthur Wilson Sergeant 50796
Jack Jones Lance-Corporal 96822
Joe Walker Private 52925
Frank Pike Private 57467
James Frazer Chief Petty Officer 07973
Charles Godfrey Private 45322
;
run;
```

The GENMAX=2 statement allows for three versions of the dataset to exist at any one time with number 0 as the current version, 2 as the most recent version, and 1 as the oldest version. A maximum of 999 versions, excluding the latest version, can be set for a dataset with the oldest version “dropping off” if the new version is created and the number of versions exceed that allowable.

To access a particular version of a dataset the option GENNUM= is used and can be used both in a direct and relative reference, as the following examples show:

```
*Print current version of dataset;
proc print data=trial.crew;
run;

*Print previous version of dataset;
proc print data=trial.crew (gennum=-1);
run;
```

Using this same technique it is possible to use the COMPARE procedure and compare versions of a dataset using the following code:

```
proc compare base=trial.crew data=trial.crew (gennum=-1);  
run;
```

Versions of a SAS dataset can be deleted using the DELETE statement in the DATASETS procedure, as the following example shows:

```
proc datasets library=trial;  
  *Deletes all versions except current version;  
  delete crew (gennum=hist);  
  *Deletes current version and moves previous version to current;  
  delete crew;  
  *Deletes all versions of dataset;  
  delete crew (gennum=all);  
quit;  
run;
```

The main disadvantage of using this method is that no note is available as to why a version was needed or any record as to why an individual observation was changed. However changes can be quickly seen using tools like the COMPARE procedure.

CONCLUSION

With careful planning it is possible to use SAS for tracking and securing SAS datasets. The methods looked at here are good and can track changes using their own features but it is important to know that with each method there are limitations.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David Franklin
16 Roberts Road
Litchfield, NH 03052
Tel/Fax 603/216-2232
Email 100316.3451@compuserve.com
<http://ourworld.compuserve.com/homepages/dfranklinuk>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.