Double-Key Data Entry/Verification in SAS® Software
Part of Downsizing a Clinical Data System to the OS/2® Operating System
Barry R. Cohen, Planning Data Systems

ABSTRACT

The pharmaceutical industry, which has typically used SAS software only for the data analysis phase of clinical data processing, now has a new opportunity to use SAS software for the data capture phase. The history of the use of SAS software by this industry is briefly reviewed, explaining why this new opportunity exists today. Next, a powerful double-key entry/verification system, which was built in SAS software to capitalize on this opportunity, is described. Finally, issues of porting this mainframe system to the OS/2 platform are addressed.

SAS SOFTWARE USE BY PHARMACEUTICAL INDUSTRY

Prior to Version 5, SAS software was primarily a statistical package with file management capabilities. Many advances were made with Version 5; however, the product still did not have certain features of a full-featured database management system (DBMS) that make the latter product attractive to pharmaceutical companies for clinical data capture and management. These include:

- A strong interactive programming environment;
- File sharing on many platforms;
- Built-in dictionary handling;
- Query tools for non-programmers;
- Application generator for non-programmers.

With Version 6, though, SAS software has moved further toward a full-featured database management system; specifically, all of the above features except built-in dictionary handling are now part of the product.

The pharmaceutical industry's historical response to this situation is no surprise: specifically, SAS software has been used primarily for data analysis, while other software tools have been used for data capture and data management.

Table 1 summarizes this historical response. Typically, there have been two multiple software platform solutions to clinical data processing. The first uses a database management system for both data capture and data management, and then uses SAS software for data analysis. If a double-key entry/verification system is used, it is written using the interactive programming tools resident in the database management system.

The second uses a dedicated entry package for data capture, a database management system for data management, and SAS software for data analysis. If a double-key entry/verification system is used, it is probably built into the dedicated entry package; custom programming would be less common here.

Table 1 also shows that some companies have used a single software platform "all SAS software" solution, although this has been much less common. If double-key entry/verification is part of this solution, it may be quite cumbersome because of the lack of interactive programming tools. Some may simply skip double-key in this solution, and verify their data through some other means.

But the "all SAS software" solution is more viable today. This is because the new interactive programming tools in Version 6 of SAS software make possible a sophisticated, interactive double-key entry/verification system in this environment. This is shown in the bottom row of Table 1.

A single software platform solution is attractive because there are certain costs to operating within a multiple software platform:

1. Multiple licenses: the need to purchase annual licenses for multiple software products;

2. Multiple skill sets: the need to maintain programmers and other staff in-house who are trained in use of the multiple software products;

3. Multiple file sets: the need to maintain a copy of your files for each software product,
### Historical

<table>
<thead>
<tr>
<th>Software Platforms</th>
<th>Data Capture Tool</th>
<th>Double-Key Verify</th>
<th>Data Management Tool</th>
<th>Data Analysis Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple</td>
<td>DBMS</td>
<td>Custom program (Interactive)</td>
<td>DBMS</td>
<td>SAS</td>
</tr>
<tr>
<td></td>
<td>entry package</td>
<td>Custom program or built-in</td>
<td>DBMS</td>
<td>SAS</td>
</tr>
<tr>
<td>Single</td>
<td>SAS</td>
<td>Custom program (Batch)</td>
<td>SAS</td>
<td>SAS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Platforms</th>
<th>Data Capture Tool</th>
<th>Double-Key Verify</th>
<th>Data Management Tool</th>
<th>Data Analysis Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>SAS</td>
<td>Custom program (Interactive)</td>
<td>SAS</td>
<td>SAS</td>
</tr>
</tbody>
</table>

including the logistics of space, naming, backup, archive, and purge rules for all these files, and including the difficulties of propagating late-breaking data updates from the original entry file through the final analysis file;

4. **Multiple hardware platforms**: in some instances, the need to maintain multiple hardware platforms because the platform required for the entry package and/or DBMS is different from the one used to run the SAS software.

It would be nice to operate within a single software platform where possible, and avoid these costs. But for who is this possible? For who is this appropriate? Who could benefit most?

Probably smaller organizations, those with smaller staffs who might find it difficult to maintain multiple skill sets in-house. Such organizations will definitely have programmers with SAS skill to support their analyses, but they would like to avoid having to carry other programming staff with other skills.

Also, those with smaller budgets might benefit. They might find the multiple licenses an issue, or the cost of multiple hardware platforms for the multiple software platforms an issue.

Also, those who are not already entrenched in an existing DBMS for clinical data processing might benefit. Such organizations probably do not need built-in dictionary capability because their CRF’s are not mature enough and modular enough to use a dictionary.

This solution might also fit for larger organizations in some instances, even if such groups are already heavily into a DBMS-based system. Consider the scenario where a large group has a satellite operation of some sort, and finds it logistically difficult to bring that group onto the mainstream system. They might need a more simple, lower-cost solution such as can be delivered in the "all SAS software" single platform solution.

Some readers might be concerned that SAS software is still not a full-featured DBMS, and thus is still not the way to go for clinical data processing. But the following features, being mentioned for Version 7 of SAS software, will further address this concern:

- dictionary capability;
- SAS/PH - Capture: a new member of the SAS/PH family;
- document image handling.

In short, the simplicity of a single software platform
solution can save costs and have important benefits for an organization. And the ability of SAS software to now support sophisticated data capture, such as through the double-key entry/verification system described below, now makes the "all SAS software" single software platform solution more viable than ever.

DATA ENTRY/VERIFICATION - BACKGROUND

Figure 1 presents the basic steps of a double-key system for verifying data. The data is keyed twice in the first step. Next, the two keyings are compared. When they match, the process returns to step 1 to key and re-key more data. When they do not match, a third step is engaged where the discrepancy is resolved with only the correct one of the two keyings remaining. Basically, there are three steps: key, compare, resolve.

Figure 1: Double-Key Entry Verification Process

![Diagram showing the double-key entry verification process]

There are several ways that double-key entry/verification can be implemented in an automated system. Following are four important attributes that characterize a good implementation of a double-key entry/verification system:

1. Use two separate keyers:

   The advantage of two keyers is clear. Some discrepancies between the first and second key are the result of keystroke errors while others are the result of misreading data items recorded on the case report form (CRF). Only the former error type can be picked up when the same person keys twice; hence, using two keyers will give you a more thoroughly verified data file.

2. Combine the three steps into one process:

   There is a particular advantage to combining the compare and resolve steps with the second-key step in a highly interactive manner; specifically, the data that must be used to resolve the discrepancy (i.e., the CRF) is on-hand just when it is needed. This is because second keyers have this CRF on their desk when they are notified of the problem, and it is open to the precise page and item being keyed.

   In contrast, other implementations may separate the comparison and resolution in time from the second keying. These systems will require, during the resolution, that CRF's that have already been closed and put aside now be retrieved and opened to the proper page. This is much less efficient.

   Note that some organizations do not want to combine the three steps of key-compare-resolve because they feel it requires that the second keyer be more skilled than the first one in order to do the resolutions. They prefer to leave resolution instead to a third, more skilled person. However, this problem does not occur when you carefully separate verification from validation, and only handle the former task here. Keyers are instructed to just enter what they see and pay no attention to validation issues. And during resolution, they need only determine which of the two keyed responses matches the one on the CRF, even if that response is invalid. This is a simple determination that does not require skills beyond a keyer's.

3. Maintain only one file:

   A good system will only maintain one copy of the data, even though it is keyed twice. It will do the second keying to a temporary file that is discarded following that keying.

4. Answer two questions easily, on demand:

   a. What is verified and what is not?
   b. What has changed since being verified, such that it must be re-verified?
Use Of Flags

All four of these attributes can probably most easily be delivered in an automated system by using field-level and record-level flags in the data sets to track verification status.

This can be demonstrated by considering the fourth attribute, that of answering the two questions. Without flags to track the verification status of each field, you would have to keep both the first- and second-key data sets to answer these questions. The answers would be obtained by re-running a compare of the two files any time the questions were asked. Of course, any items that miscompared would be "not verified", and those that did not miscompare would be "verified".

Note, though, that this approach to answering the questions would only be a reasonable one if you were updating both data sets all along, regardless of whether the erroneous keying was in the first-key data set (which is to be kept for analysis) or in the second-key data set (which is to be ultimately discarded).

But with flags embedded in the data sets, you can easily answer the two questions by just running a report of the flag variables. This is much easier than re-running the comparison of the two files, and, it allows you to not have to keep and update the second file.

Flags also facilitate combining the three steps into one interactive process, and using two keyers instead of one. This is primarily because the flags can be used to protect verified fields from access, and they can be set dynamically in an interactive process. Flags also facilitate entering and verifying partial records for this same reason.

Finally, these flags and the program code written to handle them also poised a system for growth. For example, the flags can be expanded to also support automated validation of the same data sets.

DATA ENTRY/VERIFICATION IN SAS SOFTWARE

I built my double-key entry/verification system in Version 6.06 of SAS software, using SAS/AF, SAS/FSP, and Screen Control Language (SCL) with both products. The implementation includes each of the four attributes discussed above. Following is an overview of the system.

The system is designed to work with regular SAS data sets and regular SAS/FSP screen sets. These can be built by any SAS programmer with knowledge of base SAS software and the SAS/FSP product. A minor additional effort is required on the programmer's part to add the flags to the data and screen sets. The system will work with absolutely any SAS data set you wish to define, as long as you properly add the flags.

The first-key entry is to the one and only data set defined by the SAS programmer. The screens will appear to the keyer just as they would if SAS/FSP were being used for entry without my system. The field-level verification status flags are present, but they are not visible on the screens. The entry/verification system's SCL code, which is included with each screen set, controls all entry and specifically protects already verified fields from entry.

Second-key verification is to a temporary work data set. The screens have the same look as they do during first-key. In fact, they are the same screens. The entry/verification system's SCL code controls all second entry and verification. It protects already verified fields from entry, and compares each data field's value with the same from the first keying.

The frequency of these comparisons is up to the user. Comparisons can be as frequent as immediately after each field is entered, or as infrequent as after a full screen's worth of fields have been entered. But the user cannot change from one screen to another without the fields on the first screen being compared by the system.

When there is no discrepancy between the keyings, the flag is updated and the next field (if any) is compared. Whenever a discrepancy is found between the two keyings, the resolution is immediate. A resolution window pops up over part of the SAS/FSP entry screen. It contains the first- and second-key values together, and allows the user to choose one or the other value, or to edit the values if neither is correct. The user interacts with this resolution window via function keys which make all choice selection fast. Finally, as fields are resolved, their flags are updated by the system.
POINTS FOR APPLICATION DEVELOPERS

Notice that I chose to do the entry and verification with SAS/FSP and the SCL code included with it, instead of with SAS/AF displays and program entries. I could have used either product. But I felt it was better to provide SAS/FSP for the SAS programmers who would define data and screen sets to use with the system on an on-going basis. This way, they could take advantage of the default screen sets that SAS/FSP provides for each data set defined. But the SCL code I wrote to control the key-compare-resolve process could have been written to work behind either SAS/FSP screens or SAS/AF displays. The choice of SAS/FSP was only to help the on-going programmers.

The field-level flags are one byte character variables. You might be wary of the impact that field flags will have on the size of data sets. The impact has been fairly small in my case. They tend to increase the size of my client's data sets by about 15 percent, on average. But their impact on the size of your data sets will vary with the nature of the data sets. Table 2 may help you to estimate the impact on the size of your data sets.

Table 2: Size Impact of One Byte Flags

<table>
<thead>
<tr>
<th>Data Set Variables</th>
<th>Data Set Size Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-byte character</td>
<td>100%</td>
</tr>
<tr>
<td>4-byte numeric</td>
<td>25%</td>
</tr>
<tr>
<td>8-byte numeric</td>
<td>12.5%</td>
</tr>
<tr>
<td>Many text fields</td>
<td>5% - 10%</td>
</tr>
</tbody>
</table>

One of the most challenging aspects of the design was to provide a generic system that would work with any data set and screen set that a SAS programmer might define. More specifically, the entry/verification system's SCL code that is to be included with each specific SAS/FSP screen set, and which controls the key-compare-resolve process, must be generic even though it must refer to different variables for each new data set defined.

Solving this problem was critical to a successful implementation. The solution uses a SAS/AF program that does the following:

- Read the new SAS data set defined by the programmer;
- Dynamically generate specific SCL code that references the specific variables of this data set;
- Read the system's generic SCL code for the key-compare-resolve process, and append this code to the dynamically generated specific SCL code, making one complete and custom program;
- Include this complete, custom SCL program with the specific SAS/FSP screens for the given data set.

When this process is done, the data set is ready for entry and verification within the system. The programmers who define the data sets and screen sets simply run this utility program which is part of the system. This way, programmers do not have to write any SCL code at all. They only have to define data and screen sets just as they would otherwise without my system.

ISSUES OF PORTING TO OS/2

The entry/verification system just described was written on the mainframe CMS platform, and is a central part of my client's full clinical data processing operation on that platform. Following completion of the entry/verification system, a decision was made to downsize the entire operation to the OS/2 - Novell® LAN platform. The entry/verification system was ported to OS/2 as part of this downsize project.

Many issues were addressed during this downsize project, some expected, and perhaps more unexpected. It is not possible in the space that remains for this paper to detail these issues. However, a brief overview of this experience is possible, with the intent of helping you gain a better sense of the full effort required to downsize a mainframe SAS application to OS/2. A more detailed discussion of downsizing SAS applications to OS/2 can be found in my other paper for this conference (Cohen, 1993).

My message in this overview is this:

Programmers often focus first on porting, or downloading, the actual application code to the new platform. This is the obvious view of the downsize project. However, there is much more, such that the effort to port the
application code does not come close to the effort to completely downsize the application.

A mainframe SAS application is likely to be comprised of the following:

- SAS/AF and/or SAS/FSP catalogs holding screens and SCL programs;
- SAS data sets;
- Base SAS programs (DATA steps and PROC steps).

The application would probably be ported, or downloaded, from the mainframe to the OS/2 platform in a manner such as follows:

1. Build SAS transport files for all SAS catalogs, and for a few data sets to be used for testing the code once it is downloaded;

2. Transfer the transport files, and the base SAS programs which are regular character files, over a network from the mainframe to a PC which is running OS/2;

3. Convert the transport files back to SAS System files, probably putting everything into temporary directories on your PC's local hard disk drive just for the purpose of testing the downloaded code;

4. Conduct a syntax check out, and then run the system with live data:

   - run PROC BUILD to re-compile the SAS/AF SCL code, just to make sure it is OK;
   - run PROC FSEDIT to re-compile the SAS/FSP SCL code, again, just to make sure it is OK;
   - change any existing LIBNAME and FILENAME statements to refer to files in their new locations in OS/2 directories.

If you do this you will probably find that the SCL code compiles syntax error free. And if you choose your temporary directories wisely and reference them properly with the required LIBNAME and FILENAME statements, you will also find that your application code works on the new platform, just as on the old one. And the whole process might take as little as half of a day.

At this point you will be quite happy and probably say something such as "Wow, I am impressed. I downloaded my whole application to OS/2 in four hours! That SAS Institute really knows what it is doing; and me, too!"

Well, the SAS Institute knows what it is doing, but if you think that you are nearly done, then you certainly do not. Let me introduce the kinds of issues you will soon face, which are not necessarily obvious, and which will make your full downsize project so much more than just downloading the code and changing a few LIBNAME and FILENAME statements. While many of these issues will not have a large time impact alone, they are certainly measurable, and when added together will substantially change the size of the project. These issues will be presented in three groups:

- User Interface Issues
- Data Handling Issues
- Other Issues

User Interface Issues

1. Color may very well be new to your application. Many mainframe terminals in use today do not use color. You will need time to review the default colors for background and text for all your displays on the PC, making sure they are consistent and meaningful.

2. The PC screen size is likely to be different than that of the mainframe terminals. Displays formatted for mainframe screens will skew to the upper left on many PC screens, especially the newer, larger PC's. You may need to reformat all your displays.

3. You may be downsizing into an existing LAN environment where the PC's to be used are already in place. These PC's may be different from each other, exhibiting different performance characteristics regarding color, font appearance, screen size, and CPU cycle speed. You may, thus, have to test your application on several PC's instead of just one.

4. Function key assignments for some SAS products (e.g., SAS/AF) are slightly different in the OS/2 environment, probably owing much to the fact that OS/2 has some reserved keys that were previously available on other platforms. You may have to re-train users, re-write custom
documentation, and possibly re-write SCL code that uses custom key assignments.

5. The OS/2 Desktop is a new user interface, and a very new place from which users will run SAS applications. You will have to design the OS/2 desktop, and probably have different versions for SAS application developers and SAS application users respectively. You may require OS/2 training for developers and users.

Data Handling Issues

1. OS/2 is a directory based environment while CMS and MVS, for example, are not. You will have to understand this new environment generally (if you do not already), then study exactly how your mainframe application relates files to logical disk volumes and SAS libraries, and then re-design this for the new platform.

2. Today, most OS/2 based downsizing is probably to a LAN environment instead of just a stand-alone PC. As such, your data sets will be shareable on a file server, even though this will often not be your intention. You cannot today rely on SAS software for OS/2 to handle this problem because it is still strictly a single-user product on OS/2. You will probably have to learn about directory, file, and userid rights as implemented on your file server in order to adequately address this issue.

Other Issues

1. File backup and restore are handled for you on the mainframe platform. This service is probably available to you on the OS/2-LAN platform, but not for stand-alone PC's. But even if it is available on the LAN, you will have to learn and evaluate its details, and possibly negotiate modifications, before you are through.

2. Many applications generate print files. Print file handling tends to be very platform specific, and as such, you will have to redesign how it is handled on your new platform.

CONCLUSIONS

There is a new opportunity for the pharmaceutical industry today to be all SAS software based for clinical data processing. This is largely because version 6 of SAS software is a strong interactive programming environment, which now allows development of sophisticated data capture systems. There are important benefits to being single software platform based. The most likely pharmaceutical groups to gain these benefits are smaller ones who are not DBMS based or dictionary based.

A double-key entry/verification system was successfully developed in SAS software, and it is now part of an all SAS software based clinical data operation. It includes the four important attributes for a double-key system discussed herein:

- Use two separate keyers;
- Combine the key-compare-resolve process;
- Keep only one keyed data set permanently;
- Easily answer questions about verification status, on demand.

The system was ported to OS/2 from CMS as part of downsizing the full clinical data processing operation. The port, or download, of the application code was fast, but the full downsizing of the operation was a much larger project. Many issues arose, and they could be placed into three groups: User Interface, Data Handling, and Other. In short, to download or port code is not to downsize an operation of which that code is a part.

REFERENCES


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

For more information about this paper contact:

Barry R. Cohen
Planning Data Systems
P.O. Box 666
Ardmore, PA 19003
215-649-8701