Conversion to Version 6 of the SAS® System: Experiences and Advice from a Management Information Systems Department

Jerry J. Hosking, SAS Institute Inc., Cary, NC
Clifford J. Schell, SAS Institute Inc., Cary, NC

ABSTRACT

Discussed here are experiences and advice gathered during initial conversion efforts to Version 6 of the SAS® System by the Management Information Systems Department at SAS Institute. Our organizational approach is explained and specific details of changes required and recommended for the conversion process are provided. Selected enhancements to applications using new functionality available through Version 6 are highlighted.

INTRODUCTION

Services of the Management Information Systems Department (MIS) may be requested from any of the more than 1800 employees of SAS Institute stationed at headquarters in Cary, regional offices in North America, or foreign offices around the world. This user population includes people with experience in the computing industry ranging from extreme novices to computing gurus. However, our primary targets for applications development are the tasks necessary to the everyday functioning of any large business. A sampling of applications in place covers the following functions:

- financial dealings of the company including billing, payroll, and investments
- production areas such as customer information, product development, control, and distribution
- sales from marketing analysis, customer contact, and customer support
- support services within the company including tracking of requests for services for building maintenance, computing resources, and inventory
- personnel activities including staff acquisition and reporting for internal needs and federal requirements
- access to a variety of data sources for information
- text processing.

Primary software and hardware platforms used for applications supported by MIS are a partitioned IBM® 3090 on which MVS/XA™ and VM/XA™ are hosts and a VAX™ 6000-450 running VMS™.

CONVERSION GOALS

In MIS, our number one programming-related goal in the conversion process is to avoid changes to existing source code. Our aim was to simply *flip a switch* (that is, change our invocation from Release 5.18 of the SAS System to Release 6.06 of the SAS System) and have no other changes to source code needed. The reasoning behind this goal is that minimizing the changes necessary to convert reduces the chance of introducing an error and keeps the effort necessary for the initial conversion lower.

Reducing the chance of introducing an error contributes to the achievement of our second goal: minimal disruption of our users, and therefore the company's normal work flow. Minimal disruptions also reduce the need for users to interact with the MIS staff, so that the conversion will go more smoothly.

The third conversion goal, providing additional functionality where needed in applications, is a minor goal during the initial conversion stages. However, in cases where functionality can be added without significantly reducing progress in converting, then changes were made during initial conversion. The limits placed on this goal are a result of the previous goals of requiring no changes (chances for errors) and minimizing disruption to users. It also contributes to our fourth goal, surviving the entire conversion process with good humor and no bloodshed.

Our survival goal is applied to finding techniques in which the normal work flow can be handled in addition to the additional challenge of initial conversion. To this end, users are kept informed of efforts being directed toward conversion of applications they use. These discussions help explain the trade-offs of spending additional efforts for *fix-ups* to applications using Version 5 of the SAS System against efforts spent on conversion. Providing information about the additional functionality they will gain during enhancement conversion helps persuade users of the benefits they will experience. Then informed users and MIS programming staff work together to limit maintenance requests to those absolutely essential to reflect changes in business functions.

ORGANIZATIONAL APPROACH TO CONVERSION

To meet our primary goals of no changes to code, minimal disruption to users, and surviving, we developed a two-step approach to converting. The steps are identified as *transparent* and *enhancement.*

Transparent Conversion

Transparent conversion, the initial stage of the conversion, attempts only to switch to Version 6 of the SAS System. It does not incorporate changes or enhancements to the application's functionality. However, it does include changes that are required to make the application run successfully (classified in this discussion as *required* changes) or to maintain the application's functionality so it is compatible with Version 5 (classified here as *compatibility* changes).

Enhancement Conversion

Enhancement conversion is the term applied to the second stage of conversion. This stage is the time for incorporating added functionality now available with new features in Version 6 of the SAS System. Although there is not a clear distinction when enhancement conversion ends and simply becomes maintenance of an application in Version 6, the beginning of enhancement conversion is the identification and implementation of functional changes. An example of such a change is a cross-field validation on a data entry screen rather than production of a hardcopy error report based on data entered.

Physical Conversion of SAS Data Libraries and Format Libraries

Separate from the conversion process applied to SAS source code is the physical conversion of SAS data libraries and format libraries. This part of the conversion is simply a technical process different from the creative process of converting source code. The timing of this physical conversion depends on the inter-relationships among data used in applications. Data in an application that are clearly independent of any other use in another application can be converted during the transparent stage of the conversion process. However, in the MIS Department, our data are rarely totally independent. We found it much more common for our data to be accessed by multiple separate applications. Therefore, conversion of data and format libraries is one of the last changes made. In fact, applications with all of their interactive and batch processing being conducted under Version 6 may still use Version 5 data.

However, it is necessary to physically convert catalogs early in the conversion process. For catalogs used in applications for SAS/FSP®, SAS/AF®, and SAS/GRAPH® software containing entries such as customized PROC FSEDIT screens, PROC FSLETTER forms and letters, SAS/AF catalog entries, and graphics, it is necessary to use the V5TOV6 procedure. The mechanics of the conversion are straightforward. You simply define the Version 5 catalog as input and the resulting Version 6 catalog as output with LIBNAME statements, and run PROC V5TOV6 to identify input and output.

One complication concerning physical conversion occurs when catalogs reside in a library that also contains data sets for the application. If the data will not be converted until later in the conversion process, the catalogs and data sets must be physically separated into different libraries. This could result in requiring coding changes in the application. Historically, we have maintained most of our catalogs and data sets separately, so impact from this was minimal. For sites at which catalogs and data sets are stored together, consider whether this application can have data converted to Version 6 early as well to avoid additional coding changes.

During this physical conversion, a plan for naming and identification of the versions of catalogs is important. We already had standard naming conversions for data libraries in place, so it seemed natural to simply append a 6 to data library names as they were converted. But we realized that the proliferation of a 6 into all data library names would become a burden. We chose to retain data library names as consistently as possible. On VM, this is a smaller problem because it is not difficult to have data libraries with the same name simply in different locations (mini disks) and then to point to the correct location by attaching to the appropriate mini disk. However, on MVS we could not duplicate names because all data sets were cataloged. This resulted in having to use different names for Version 6 data libraries during conversion and renaming at the time of the actual change to Version 6.

Transparent Conversion: Required Changes

Invocation of the SAS System Change the verb used for invoking Release 5.18 of the SAS System (SAS) to the one used to invoke Release 6.06 (SAS6) in batch jobs and interactive command procedures. Of course the names you actually use here will depend on those in use at your site.

Addition of the $LIBCLEAR option at SAS invocation The results when using system commands to free SAS data sets are slightly different after conversion to Version 6. However, using the $LIBCLEAR option will allow system commands (TSO FREE and CMS FILEDEF) to free a library reference without using a LIBNAME CLEAR statement. The definition of a required change is being stretched slightly to include this change. Strictly speaking, $LIBCLEAR is not required to make a converted application run successfully. However, for the transparent stage, the use of this option requires fewer changes than the replacement of system commands already in place with LIBNAME statements. Use the following code for batch processing:

VM	MVS
`EXEC SAS6 ($LIBCLEAR`	`EXEC SAS6,OPTIONS=$LIBCLEAR`

Use the following code for interactive processing:

VM	MVS
`SAS6 ($LIBCLEAR`	`SAS6 OPTIONS($LIBCLEAR`

The $LIBCLEAR option is needed only if system commands are still being used to reference SAS data sets. Changing to the use of LIBNAME statements is an alternative. The choice of which to use, $LIBCLEAR option or LIBNAME statements, is based on which method is least disruptive or requires fewest changes.

Addition of equal sign for system options at SAS System invocation (VM only) The syntax for use of system options in which values are specified when invoking the SAS System has changed to require an equal sign. For example, when invoking the Version 5 SAS System with the LINESIZE option, use the following:

```
SAS (LINESIZE 72
```

To invoke Version 6 with the same option, use the following:

```
SAS6 (LINESIZE=72
```

Use of an equal sign for system options is valid under Version 5 so that change can begin being made prior to full-scale conversion efforts.

Change option name MAUTOSRC to MAUTOSOURCE (VM only) Two versions of the MAUTOSOURCE option were recognized in Version 5: the official name MAUTOSOURCE, and a shortened version MAUTOSRC. The abbreviated version is no longer recognized. However, MAUTOSOURCE is the default (unless changed by your site), so you may choose to simply remove any occurrence of MAUTOSRC rather than change it.

Change keyword for identifying autocall library to SAS System from AUTOS to SASAUTOS The original name is no longer recognized. This option can also be specified in a configuration file that may reduce the number of changes necessary.

Change references to SAS log and list files (MVS only) References to FT11F001 and FT12F001 must be changed to SASLOG and SASLIST, respectively. In addition, in batch they must be positioned immediately following the JCL EXEC statement and in the order in which they occur in the cataloged procedure used to invoke the SAS System. By default, this should be SASLOG and then SASLIST.

Replace use of CLEAR; statement with CMS VMFCLEAR; (VM only) The CLEAR; statement, used to clear the terminal display, is no longer valid. Use of CMS VMFCLEAR; is valid under Version 5 so that change can begin being made prior to full-scale conversion efforts.

Replace use of USERID function with &SYSJOBID (VM only) The USERID function is no longer valid. &SYSJOBID is also available in Version 5, so you can discontinue use of the USERID function prior to full-scale conversion efforts. For example, replace the following statement in the Version 5 application:

```
varname=USERID();
```

with the following in Version 6:

```
varname="&SYSJOBID";
```

Discontinue use of single alphabetic characters as menu options on catalog menu entries In SAS/AF menu entries, any single alphabetic characters used as menu options in catalog menu entries need to be changed to numbers or longer alphabetic strings. Use of single alphabetic characters can result in conflicts with SAS keywords or aliases for keywords.

Use the V5TOV6 procedure to convert specific catalogs As needed, perform the physical conversion of catalogs for applications being converted. See **Physical Conversion of SAS Data Libraries and Format Libraries** for further discussion.

Transparent Conversion: Compatibility Changes

Using the NOBORDER option By default, SAS/FSP and SAS/AF screens are enclosed by a border in Version 6 of the SAS System. If a screen in Version 5 uses the full width of the screen (79 for many terminals), when converted and displayed under Version 6, the full display you have designed may not be viewed without left/right scrolling. Use the NOBORDER option in the PROC statement to remove the border and permit the display of a width of 79 characters (or physical width of the screen), as in the following example:

```
PROC FSEDIT DATA=libref.dataset SCREEN=libref.screenname NOBORDER;
```

Use of the NOBORDER option causes the NR option (which controls the number of rows displayed per screen) to be ignored. If you use the NR option currently and want to use the NOBORDER option too, in order to set the number of rows displayed, use the modification menu for the specific procedure in SAS/FSP software to access the general parameter settings (Option 5), and set the number of rows. For catalog entries, set the number of rows using the general attribute (GATTR) panel.

Using the ENGINE= SAS system option Defining the default engine to be the Version 5 engine at invocation of the SAS System resolves other changes that might otherwise be needed for compatibility purposes during the transparent conversion stage. Engine specification on each LIBNAME statement for new data libraries is not necessary when a default is set. Therefore, it will not be necessary to specify a Version 5 engine prior to and during conversion, and then remove or change to a Version 6 engine when data has been converted. In addition, if you want to continue use of system commands instead of LIBNAME statements, using a default engine is helpful because system commands do not permit specification of an engine. Engine specification is not necessary for existing data sets, but must be done if you are creating new data sets under Version 6 and want them to remain Version 5.

Using key definitions to set catalog menu options to PF keys The ATTRIBUTE window for SAS/AF catalog menu entries no longer allows you to assign a menu option to a PF key. To define PF keys as menu options, change the key definitions for the SAS/AF menu display. This key definition can be stored in the same data library as the catalog and will be used when the catalog is called.

Avoiding data set corruption by using the AUTOSAVE parameter When a data set has not been closed properly during a previous session with the SAS System, the next time the data set is accessed, it appears corrupted to the software. To reduce the likelihood of this occurring, use the modification menu of the specific procedure in SAS/FSP software to access the general parameters and set AUTOSAVE to 1. This is particularly important for data sets that are accessed interactively during the day and then processed in batch at night. If during interactive/daytime access, the user does not exit gracefully from a data set, when the data set is called for use during batch processing, the SAS System will not be able to use the data set and batch processing can fail.

Setting PAGESIZE= and LINESIZE= options Default LINESIZE and PAGESIZE settings have been changed. If you do not specify settings for these two options, and the new settings of 76 and 28 for LINESIZE and PAGESIZE, respectively, are not appropriate for your application, you will need to set these to values to match your needs. These options can be set system wide by including them in a configuration file.

The required and compatibility changes for conversion from Version 5 to Version 6 are summarized in the table below.

Operating System	Version 5	Version 6
MVS and VM	SAS	SAS6
MVS and VM	n/a	add $LIBCLEAR at invocation
VM only	SAS (*option_name value*	SAS6 (*option_name*=*value*
VM only	SAS (MAUTOSRC	SAS6 (MAUTOSOURCE
MVS	SASAUTOS ('"*d.s.name*"')	SAS6 SASAUTOS ('"*d.s.name*"')
VM	hardcoded in SAS EXEC	specified in configuration file or at SAS invocation SAS6 (SASAUTOS='*name maclib*'
MVS only	FT11F001 FT12F001	SASLOG SASLIST
VM only	CLEAR;	CMS VMFCLEAR;
VM only	*varname*=USERID( );	*varname*="&SYSJOBID";
MVS and VM	single character menu options	numbers or multiple character menu options
MVS and VM	n/a	add NOBORDER as needed for SAS/FSP and SAS/AF software displays and set NR via general parameter and general attribute screens
MVS and VM	n/a	use ENGINE= option at invocation
MVS and VM	attribute screen for SAS/AF menu entries	key definition stored in same data library as catalog
MVS and VM	n/a	set AUTOSAVE
MVS and VM	n/a	set PAGESIZE= and LINESIZE= options

SELECTED CONVERSION ENHANCEMENTS

Once the initial transparent conversion to Version 6 has been accomplished, it is time to consider how certain features of Version 6 can be used to improve an existing application's performance and reliability.

This enhancement stage requires a careful, in-depth analysis of the existing application. For example, using certain features of Version 6 might not be considered appropriate due to the coding effort required and should wait until the application is being redesigned and rewritten at a later date.

This section of the paper will focus on the new features of Version 6 that can easily be integrated into existing applications during the conversion cycle. We selected those enhancements that would provide the maximum initial benefit to the end user resulting in a minimum amount of disorientation that would require extensive retraining.

Using Screen Control Language to Enhance Applications

The applications that were most in demand for enhanced conversions were those in which employees had direct customer contact. These have traditionally been those applications that require real-time information quoted to customers over the telephone. These employees had been working from multiple sessions that were used for both order entry and reference. Other groups were working from listings that had been generated on a nightly basis and were cumbersome to use. These approaches obviously tend to be inefficient and prone to error.

The addition of Screen Control Language (SCL) to the FSEDIT procedure was considered the most powerful Version 6 enhancement and was used heavily in many applications. This tool can easily provide the look-up capability to eliminate the use of multiple sessions or listings all together. In addition to the look-up capability, cross-field validation can also be easily accomplished with data entered on the screen or contained in one or more currently open files.

In previous versions, one of the few ways to look-up or validate data in a field was to design customized formats that contained valid data values for each field. Once a value was entered in the field by the user, its corresponding formatted value then appeared when the ENTER key was pressed. This served in a limited capacity as both a look-up and validation. Since data values were supplied based upon the data in the format, an invalid data message would result if the entered value was not contained in the format.

This approach has several inherent limitations. The format itself needs to be maintained and supported as would any data set. Next, the formats are typically updated overnight, so changes usually do not take effect until the following day. Finally, no cross-field capabilities exist using this approach, so a look-up or validation based on two or more data values is not possible.

Without SCL and given the limitations as outlined earlier, it can be seen that in a typical application a considerable amount of post-processing needs to be done to insure data integrity. Error and exception reports are typically produced in the evening on the day's data so that errors can be corrected. So, not only are people working on the current day's business, they are fixing the previous day's errors as well.

SCL enables the applications programmer to have several files open in addition to the one currently being edited. This permits the SCL program to retrieve *and* update data in these other files based upon values entered in the window. Information can be looked up and displayed in the window or be used for cross-field validation purposes.

This can provide great programming flexibility and allows for the development of sophisticated applications. By preventing bad data from being entered in the first place, virtually all post-processing and error reporting can be avoided. Applications that are truly WYSIWYG can be developed for the first time, using the interactive window to display how the final form will appear. In our applications, all the data that will appear on a customer's final form can be calculated and displayed directly on the window, eliminating the need for costly night production runs that will post-process daily data and generate the same forms.

Data from multiple data sets can now be linked together on the screen to conserve resources. Previously in FSEDIT applications, screen variables reserved space on a data set to contain information that may or may not exist for that particular observation. For example, imagine an application where comment lines exist for a particular observation on a secondary window. In this case, all of the comment variables have to be accounted for on the main data set whether they are used or not. This can lead to quite a waste of space in the resulting data set. Using SCL programming techniques on the FSEDIT window, the comment lines can dynamically be generated as separate observations in another data set and displayed when needed. They can either appear on another window or be put in their own pop-up window and displayed when needed.

SCL has its own debugging environment that can be used in both SAS/FSP and SAS/AF applications. This interactive debugger, not previously available in other SAS software programming environments, makes it much easier to determine problems in SCL code.

A factor to be considered in using SCL to code applications pertains to the use of macros. Since SCL is compiled, not interpretive, as are the rest of the SAS software programming environments, macro code is resolved at compile time. This means changes made to existing applications require that the FSEDIT window or AF program window be recompiled to see the effect of these changes. We use macros in source code heavily, so this became an important consideration for us, and deterred the conversion of some applications until a source management system could be developed to keep track of which screens and programs needed to be recompiled when each macro was changed.

Converting SAS/AF Applications to Use SCL

When a SAS/AF application is converted from Version 5 to Version 6 using the V5TOV6 procedure, enhancements to the resulting programs can be made. The resulting SAS/AF program, after conversion, simply takes the SAS software programming statements and submits them to the SAS System for execution. A great performance improvement can be realized if these submit blocks generated in the converted code are replaced by SCL programming statements. The compiled SCL code runs at a much faster rate than the interpretive code. If the application being converted will not be rewritten in the near future, it might be worthwhile to consider this approach.

CONCLUSIONS

After converting several large-scale systems to Version 6 of the SAS System, it has been our experience that the conversion process can be relatively painless. It requires forethought and planning to set up source, data set, and catalog libraries, but the rewards in eliminated processing and added functionality make the experience well worth the effort.

We believe that an approach that separates enhancements (rewriting) from the physical conversion is a method that allows efficient and ordered progression to Version 6 of the SAS System. It also allows you to schedule the addition of enhancements into the regular routine of changes to any production application. The use of new features, such as SCL, WHERE statements, indexes, compressed data sets, and pop-up windows, will greatly improve the functionality and response time for your users. The work you invest to incorporate new features will be rewarded by the satisfaction you gain from providing an application that is more durable, uses resources better, and is friendlier for the user. Your users will benefit, you will benefit, and your company will benefit.