Version 6 and Version 7: A Peaceful Co-Existence

Steve Beatrous and James Holman, SAS Institute Inc., Cary, NC

Abstract

Version 7 represents a major step forward for SAS Institute and is the first release of a new generation of SAS® software. Customers confronted with a new generation of software are concerned about the cost of moving their applications to that new release and are reluctant to move everything at once.

This paper reviews some of the reasons why customers want to move some or all of their applications to Version 7. We recognize that many sites will not want to move everything forward at the same time; therefore, this paper discusses techniques to make partial migrations (mixing Version 6 and Version 7 applications) and techniques for running critical systems in parallel (to facilitate comparing Version 6 to 7). Maintaining data libraries in the proper format (Version 6 versus 7) and cross version compatibility issues in a client/server environment are two of the concerns addressed.

Why Convert to Version 7?

Version 7 contains a number of features that were beyond the scope of Version 6. These are features that required major re-writes of the SAS system. The new Version 7 features provide the major motivation for moving to Version 7. While this paper does not go into detail about these features, the following is a list of the author's favorite new features:

1. **Output Delivery System (ODS):** You have many more options for the output created by SAS procedures, allowing you to:
    - Transform procedure output into a SAS data set
    - Render output as colorful HTML pages with embedded hyperlinks
    - Operate seamlessly with word processing software using RichText and/or Postscript files (experimental)
2. **Long Variable Names:** Allows for richer name space for tables(data sets) and columns (variables)
3. **SAS Explorer:** Provides a rich visual front end to the SAS System
4. **Asynchronous SAS/CONNECT® Program Submits:** Allows you to do work in the foreground while remote submits are being processed in the background
5. **Dynamic Libnames**: Gives transparent access to external databases with SAS/ACCESS® dynamic libname statements
6. **CEDA:** Allows access to SAS datasets created by multiple operating systems without having to go through a transport process or bring up a server on the machine that created the file
7. **Advanced Database Features**: Provides integrity constraints to ensure data conforms to user-defined rules and provides versioning to allow you to keep more than one copy of the same file

You would be motivated to convert to Version 7 if any of the previous features are important to you. Motivation, however is only part of the story. Version 7 is a first step in a new generation of SAS software. As any such first step, it is not completely evolved. You must consider the following when deciding to convert from Version 6 to Version 7:

1. The Version 7 product line is not complete. Some products such as Enterprise Miner and Data Warehouse Administrator are not part of the Version 7 offering.
2. For some jobs, Version 7 uses more memory and takes more CPU time than Version 6.

The Version 8 release (being demonstrated at this conference) is meant to complete the product line and address performance differences between Versions 6 and 7. If you depend on one of the products that are not yet available in Version 7 or if your applications are extremely performance sensitive then you may need to wait for Version 8. The content of this paper applies equally to those sites planning on converting from Version 6 to 7 or from Version 6 to 8.

Model for Converting to a New Release

In the past, converting from one release of a software product to another was like moving into a new house. The customer made a decision to move, set up elaborate systems to prepare for the move, and expected to be out of commission for a while after the move completed.

The software vendor usually supplied tools to assist in the process. For example, when Version 6 was introduced SAS Institute provided `PROC V5TOV6` to convert data and applications. In the best of all worlds the conversion tool would make all of the required changes to the user's files and source programs so that after the tool had run, the application could run in the new release. However, converting an application is often like moving furniture - the old stuff just doesn't work right in its new home.

The move to a new release was seen as a complete operation (all applications must migrate forward at the same time) that was irreversible. With this kind of model, it is no wonder that software users grimace every time a vendor introduces a new release.

SAS Institute understands how painful upgrades to software can be. One of the design goals of Version 7 was to facilitate a seamless transition from Version 6.

The Institute expects customers to evolve from Version 6 to 7 rather than do a massive conversion. We have worked hard so you will not have to go through the total and irreversible kind of conversion that you went through between Versions 5 and 6.

To achieve the goal of seamless transition, Versions 6 and 7 complement one another. Some of the Version 7 features that facilitate a complementary relationship are:

1. SAS automatically senses the format of a library, e.g. is it a Version 6 library or a Version 7 library?

SAS programs that access Version 6 libraries will (for the most part) run unchanged in Version 7.

2. Read, Write, and Update access to Version 6 SAS data files are supported.
3. Read access to Version 6 SAS catalogs and SQL views are supported.
4. You can mix Version 6 and 7 clients and servers. For example, a Version 7 client can process data from a Version 6 SAS/SHARE server (and vice versa).
5. Library and catalog concatenation (a new Version 7 feature) allows you to move some data forward into a V7 format while leaving other data in Version 6 format.

All of these features taken together mean that most[1] Version 6 applications can run unchanged in Version 7 and that it is possible to migrate part of an application to Version 7 while leaving other parts in Version 6. The conversion from Version 6 to Version 7 can be painless and does not have to be complete.

In later sections, two of the above features (concatenation and mixed release client/server) will be discussed in detail.

Library Concatenation

Library concatenation allows you to reference two or more SAS libraries with a single libref. A complete description of library concatenation may be found on the CD ROM SAS Language Reference: Dictionary, First Edition.

Library concatenation allows you to combine libraries that are processed by different engines. For example, suppose there are some files in a Version 6 library and some other files in a Version 7 library. Further, suppose that the application needs to process the collection of files in both libraries. The following syntax can be used to establish a single libref ("MYLIB") that combines the Version 6 and Version 7 libraries:

```
libname v6lib 'path-to-v6-library';
libname v7lib 'path-to-v7-library';
libname mylib (v7lib v6lib);
```

The preceding example allows you to leave some files used by an application in Version 6 format while converting others to a Version 7 format. (Note that converting a file from Version 6 format to Version 7 format is as simple as running a PROC COPY from a Version 6 libref to a Version 7 libref.)

SAS Catalog Concatenation

Catalog concatenation allows the combination of two or more catalogs into a single logical catalog. A complete description of catalog concatenation may be found on the CD ROM SAS Language Reference: Dictionary, First Edition.

In the above example a Version 6 and a Version 7 library were concatenated into a single library named MYLIB. All of the catalogs in the concatenation with the same catalog name will be logically combined.

[1] There are some exceptions where data must be converted to Version 7 format before the application can run in Version 7. These are discussed in Appendix 1 of this paper.

An Example: Library and Catalog Concatenation

Library and catalog concatenation provides a method of combining Version 6 and 7 libraries. You can then decide which parts of the library to upgrade and which parts should remain in a Version 6 format.

You would want to upgrade a data file, a data view, a catalog file, or a catalog entry for any one of the following reasons:

1. To exploit new Version 7 features. For example, you may want to have update access through an SQL view.
2. The Version 6 format is incompatible with your intended usage in Version 7 (see Appendix 1 of this paper for details). For example, suppose your application expects to be able to update FRAME entries in a catalog. Version 7 will only allow updates to catalog entries that are in Version 7 format[2].
3. To change or extend your application such that some parts of the application run only in Version 7.

You would not want to upgrade your data if either of the following were true.

1. You expect Version 6 and 7 clients to need access to the data and you do not want or you cannot have multiple copies of the data.
2. You are comparing Versions 6 and 7 and you need the data stored in a form that both releases can get to.

Effectively using library concatenation assumes that you start with a Version 6 library. You then decide which pieces of this library you need to convert to a Version 7 format (see reasons 1-3 above). You copy any SAS files to the Version 7 library with PROC COPY and you copy selected catalog entries with PROC CATALOG.

To illustrate how this might work for you, consider an example of a Version 6 library, which contains:

[2] It is important to note that the engine creating a SAS catalog determines its format (the actual file format). Furthermore, this format is different in Versions 6 and 7 of the SAS System.

On the other hand, the format of a SAS catalog entry is determined by the SAS program or application that created it and may or may not be forwards or backwards compatible.

V6LIB
FORMATS.CATALOG ONE.FORMAT TWO.FORMAT THREE.FORMAT
MYPROG.CATALOG A.FRAME B.FRAME C.FRAME PIC1.GRSEG PIC2.GRSEG
TABLE1.DATA
TABLE2.DATA

In the above example the user could initially run their application in Version 7 without changing a thing.

However, after things have run successfully in Version 7 the user wants to replace `FORMATS.ONE.FORMAT`, `MYPROG.B.FRAME`, and `TABLE2.DATA` with versions that exploit new features. For example, `PROC FORMAT` in Version 7 supports a `NOTSORTED` option that allows the user to list most likely values for a format first and therefore get better performance with long lists of format values. The user would create an overriding `ONE.FORMAT` in the catalog `V7LIB.FORMATS`.

Also, suppose you want to add Integrity Constraints and long variable names to `TABLE2`. In doing so, the user knows that (s)he still needs a Version 6 form of these upgraded files. However, the user does not want to have to create a Version 7 format of the unchanged data. The results are a Version 7 library that looks like this:

V7LIB
FORMATS.CATALOG ONE.FORMAT
MYPROG.CATALOG B.FRAME
TABLE2.DATA

The concatenated libref `MYLIB` would be set up as follows:

```
libname mylib (v7lib v6lib);
```

MYLIB `would` have the following contents:

MYLIB Libref
FORMATS.CATALOG ONE.FORMAT (from V7LIB) TWO.FORMAT (from V6LI B) THREE.FORMAT(from V6LIB)
MYPROG.CATALOG A.FRAME (from V6LIB) B.FRAME (from V7LIB) C.FRAME (from V6LI B) PIC1.GRSEG (from V6LIB) PIC2.GRSEG (from V6LIB)
TABLE1.DATA (from V6LI B)
TABLE2.DATA (from V7LIB)

Running Version 6 and Version 7 at the Same Time

There are going to be some sites that want to run their applications in Version 6 and Version 7. A cautious customer, for example, would experiment with Version 7 by running their applications in both releases. Another customer may be comfortable with moving some groups of users to the new release. For example, you may want to have your applications developers running Version 7, but your end users run Version 6. Finally, there will be customers in the business of delivering data in the form of SAS data files. These customers will want to deliver a Version 6 or a Version 7 form of the same file. The data producer, however, wants their Version 7 file not to be restricted to Version 6 features. For example, the data producer would want their Version 7 deliverables to have long descriptive variable names.

The SAS System offers some features to assist those customers who need to run Version 6 and Version 7 at the same time.

Writing Version 6 Compliant Code in Version 7

The SAS option `VALIDVARNAME=V6` forces all variable names to be eight or fewer characters. This option prevents a developer from writing SAS code that would fail to compile in Version 6.

For example,

```
data a; long_var_name=1; run;
```

compiles and runs in Version 7, but produces a syntax error in Version 6. If you perform the above DATA step after setting `VALIDVARNAME=V6`, the DATA step will fail to compile in Version 7.

The option may be set with the options statement:

```
options VALIDVARNAME=V6;
```

or in the SAS configuration file. For example, on a UNIX system,

```
-VALIDVARNAME V6
```

The VALIDVARNAME=V6 option setting is useful when you have developers working in Version 7 writing code that must compile in Versions 6 or Version 7.

Making a Version 6 Copy of a Version 7 File

There will be some customers who want to run Version 7 and fully exploit Version 7 features, but who have the need to create Version 6 data files to deliver to their clients. To do this, however, requires stripping out all of the Version 7 specific feature set from the file. For example, all variable names greater than eight characters would have to be renamed to fit the Version 6 eight-character limit. A SAS customer who is in the business of delivering SAS data sets will want to be able to easily deliver a Version 6 or Version 7 form of the file. Manually stripping out Version 7 specific features is tedious.

PROC COPY, when used with VALIDVARNAME=V6 (see above), will do this for you[3].

For example,

```
data employee;
     LastName = 'Smith';
     FirstName = 'John';
     EmployeeID = 50;
   run;

libname v6lib 'path';
options validvarname=v6;

proc copy in=work out=v6lib;
     select employee;
  run;
```

The resulting Version 6 data file, **V6LIB.EMPLOYEE** will contain the variables **LASTNAME**, **FIRSTNAM**, and **EMPLOYEE** and will be usable by any Version 6 or Version 7 application or SAS program.

Mixing Version 6 and Version 7 Client / Servers

Up to now this paper has been covering traditional SAS applications represented by one user and one process on one machine. The SAS System, however, has client/server extensions represented in products like SAS/SHARE® and SAS/CONNECT®.

When moving a client/server application forward, you have the choice of:

1. Moving some or all of the clients forward
2. Moving just the server forward or
3. Moving both client(s) and server forward

You may ask which of these three approaches makes the most sense. The answer to that question is going to be "it depends". The point is that the Version 7 system gives you the ability to mix and match Version 6 and Version 7 client/server scenarios. For example, one customer may wish to move their SAS/SHARE® clients to Version 7 so that the clients can take advantage of some of the new Version 7 client side report writing features (such as ODS). That same customer may see no advantage to moving their server forward.

A SAS/CONNECT® or SAS/SHARE® customer may be using a server on another machine. At your site it may be that one machine environment has Version 7 available before another. A customer may wish to move their client or server forward to Version 7 when Version 7 is available on their machine. You do not have to wait until Version 7 is available on all of the machines that make up your client/server application.

There are some restrictions to mixing Version 6 and Version 7 client and servers. See Appendix 2 for a set of summary tables detailing what you can and cannot do in a mixed setting.

Conclusions

Version 7 was built with features that enable a slow migration or a complete and total migration. Many applications will require no changes at all to run under Version 7, while others may require some minor adjustments (such as using PROC COPY to move your data from a Version 6 format to a Version 7 format).

The new features in Version 7 provide the primary motivation for moving forward. The move can be total or partial, but either way it should be relatively painless.

Appendix 1: Conversion Gotchas

For the most part, Version 6 applications can run unchanged in Version 7. There are some features, however, which require a customer to do something to their source or their data in order to run in Version 7. If you do not use any of these features, your transition from 6 to 7 should be seamless. If you do use some of these features, then this section tells what you need to do to convert from Version 6 to Version 7.

1. Problem: Version 6 Data Step Views cannot be read by Version 7. If your Version 6 application processes Data Step Views then it will not run in Version 7.
   Solution: Recreate your Data Step Views in a Version 7 library. Concatenate the Version 7 library with the Version 6 library that contains the rest of your data or migrate all of the Version 6 files to Version 7 format. Note that this will not be a problem in Version 8. Version 8 will be able to read Version 6 data step views.
2. Problem: Version 6 Catalogs cannot be updated by Version 7. If your application updates catalog entries then the application cannot run unchanged in Version 7.
   Solution: Copy the catalog entries that your application updates to a Version 7 catalog. Use catalog concatenation to combine this partial Version 7 catalog to the full Version 6 catalog or copy all of the required entries to a Version 7 catalog.
3. Problem: Version 7 cannot create data step or SQL views in a Version 6 library. If your application creates SQL or Data Step Views then it may not run unchanged in Version 7.
   Solution: Modify your application to use library concatenation to insert a Version 7 library in front of the Version 6 library.
4. Problem: Customizations stored in a Version 6 SASUSER.PROFILE are not available in Version 7.

[3] It is important to note that PROC COPY will be able to create a truncated name for variables that use the new Version 7 feature of long names. This truncation is based on the stem of the variable name and is guaranteed to be unique, though not as descriptive as the long name.

The SASUSER.PROFILE catalog is used to store customizations to the SAS System. These customizations, for the most part, relate to the look and feel of the SAS GUI. The Version 7 GUI is so different from the Version 6 GUI that there are no provisions for moving your SASUSER.PROFILE forward from Version 6 to 7.
Solution: Customize the Version 7 GUI after installation. Non-GUI data stored in a Version 6 SASUSER.PROFILE may be moved to a Version 7 catalog with a PROC COPY. The OUTPUT from PROC COPY may be used as the Version 7 SASUSER.PROFILE although the Version 6 GUI settings will be lost.

5. Problem: MVS, CMS, and VAX VMS customers cannot read 6.06 data sets in Version 7. If your application processes SAS data files in the 6.06 format then it will not run in Version 7.
Solution: Use Version 6 to reformat your data. A simple DATA step: **DATA MYLIB.A; SET MYLIB.A;** will reformat a 6.06 file to be compatible with Version 7.
6. Problem: MVS and CMS customers will have READ ONLY access to Version 5 data libraries. If your application requires updating or creating Version 5 data then it cannot be run in Version 7.
Solution: Use PROC COPY in Version 6 or 7 to migrate your Version 5 data to either Version 6 or Version 7 format.
7. Problem: MVS and CMS cannot be used to build AF FRAME, RESOURCE, or CLASS entries.
Solution: You can build the application on another platform (for example UNIX or Windows) and then import them to MVS or CMS.
8. Problem: On the AIX operating system, Version 7 SAS cannot read Version 6 Catalog files.
Solution: Run PROC CPORT in Version 6 to create a transport file. Version 7 PROC CIMPORT can read a Version 6 transport file and create a Version 7 Catalog.
9. Problem: HOSTFMT= dataset option on VMS has been replaced by the OUTREP= dataset option.
Solution: Change all occurrences of the HOSTFMT= VAX to OUTREP=VAX_VMS and all occurrences of HOSTFMT=ALPHA or HOSTFMT=AXP to OUTREP=ALPHA_VMS.

Appendix 2: Restrictions on mixed Version 6 and Version 7 Client/Server

The tables in this appendix use the following conventions:

R = Read Access

W= Write Access

U = Update Access

Depending on the SAS Data Library member that is being accessed, compatibility between Version 6 and Version 7 varies. For example, under SAS/SHARE® and SAS/CONNECT® Remote Library Services (RLS), SAS Data Files are compatible as follows:

	Version 6 Server		**Version 7 Server**	
	V6 Data File	V7 Data File	V6 Data File	V7 Data File
V6 Client	R/W/U		R/W/U	R/W/U†
V7 Client	R/W/U†		R/W/U†	R/W/U

† No Version 7 specific features are allowed

For SAS Data Views, compatibility varies, this time depending on the type of Data View - Data Step, SAS/ACCESS® and PROC SQL and the setting of the RMTVIEW option. Again, considering SAS/SHARE® and SAS/CONNECT® RLS, for Data Step and PROC SQL SAS Data Views,

	Version 6 Server		**Version 7 Server**	
	V6 View	V7 View	V6 View	V7 View
V6 Client	R/W/U		R/W/U†	R‡
V7 Client	R‡		R±	R/W/U

†`RMTVIEW=NO` has been set
‡`RMTVIEW=YES` has been set
± PROC SQL views only

while for SAS/ACCESS® views we have

	Version 6 Server		**Version 7 Server**	
	V6 View	V7 View	V6 View	V7 View
V6 Client	R/W/U		R/W/U	R/W/U
V7 Client	R/W/U		R/W/U	R/W/U

Finally, again from the standpoint of SAS/SHARE® and SAS/CONNECT® RLS, for SAS Catalogs we have:

	Version 6 Server		Version 7 Server	
	V6 Catalog	V7 Catalog	V6 Catalog	V7 Catalog
V6 Client	R/W/U		R	R/W/U†
V7 Client	R/W‡		R/W‡	R/W/U

† The catalog produced will be a Version 7 catalog with Version 6 data
‡ Limited writing is allowed so that PROC COPY can copy a Version 6 catalog to another Version 6 library

It is important to note, however, that there are two other facets to SAS/CONNECT® - Remote Compute Services (RCS) and Remote Data Transfer Services (RDTS). Compatibility of SAS Data Library members must take into account the version of the SAS System the local and the remote hosts are using.

For example, with SAS/CONNECT® RCS:

- A Version 6 local host can resubmit a SAS program that references either Version 6 or Version 7 data files to a Version 7 remote host with no complications.
- A Version 7 local host can remote submit a SAS program to a Version 6 remote host but the programs must not contain any references to Version 7 specific features.

Another example, this time with SAS Catalogs, again using SAS/CONNECT® RCS, compatibility varies as follows:

	Version 6 Remote		Version 7 Remote	
	V6 Catalog	V7 Catalog	V6 Catalog	V7 Catalog
V6 Local	R/W/U		R†	R/W/U‡
V7 Local	R/W‡		R/W†	R/W/U

† PROC UPLOAD cannot create a Version 6 catalog entry
‡ Relies on the entry's backwards compatibility code

Authors

Steve Beatrous
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919) 677-8000
sassmb@wnt.sas.com (Steve Beatrous)

James Holman
SAS Institute Inc.
SAS Campus Drive
Cary. NC 27513
(919) 677-8000
jaholm@wnt.sas.com (James Holman)