Paper 288-28.3

Proc Migrate: How to Migrate Your Data and Know You've Done It Right!

Diane Olson, SAS Institute, Cary, NC
David Wiehle, SAS Institute, Cary, NC

ABSTRACT

Migrating your data to a new version of SAS presents challenges; depending on the attributes of your data libraries, your migration can be relatively simple or complex. Some issues to consider when adopting SAS 9 include the version of SAS in which your data currently resides, what member types exist in your libraries and whether you must move members from 32-bit libraries to 64-bit libraries. To address these issues, SAS 9.1 includes a new utility procedure, Proc Migrate. Proc Migrate streamlines the process of moving libraries forward to a new release. This paper introduces Proc Migrate and discusses pitfalls of migrating with the traditional methods - Proc Copy, Proc Cport/Proc Cimport and Proc Catalog. Base SAS also provides tools that can help you guarantee the content and attributes of your data after migration. We discuss how Proc Datasets, Proc Contents and Proc Compare are used in validation tools to validate the integrity of your migrated libraries.

GET THE LATEST INFO ON PROC MIGRATE

See the Migration Community link on www.support.sas.com to access the most up-to-date information on Proc Migrate and the validation tools.

WHY A NEW PROCEDURE?

SAS customers have had the need to upgrade from an earlier version of SAS to the current version virtually since the SAS System was created. That upgrade has traditionally been accomplished by using the COPY, CPORT/CIMPORT and CATALOG procedures (hereafter referred to as the conversion procedures). Each procedure has its strengths and weaknesses, but none of them were designed specifically to migrate a SAS library to a new release. The conversion procedures have many options that can be used when converting SAS members; when using them, you must consider whether or not to keep indexes, integrity constraints and attributes such as compression. Should you preserve the original creation date? Should you convert the entire catalog or just convert the "wrapper" and let each entry be converted as it is accessed? Should you copy the entire library or sacrifice your referential integrity constraints?

It is clear that these options are necessary for the multitude of reasons that the conversion procedures are used, but for the task of migration, something easier and more intuitive is needed. New for SAS 9.1, Proc Migrate provides a simple interface for migrating a SAS library from an earlier version to the current version. It also allows new functionality needed for migration that the other conversion procedures do not.

Not only does Proc Migrate simplify upgrading your library, it also takes a burden off the conversion procedures by allowing them to focus on their primary purposes. In short, the new procedure makes it simpler for you to get what you need during a migration and removes potential performance impact and unnecessary code complexity in other procedures.

DO I HAVE TO MIGRATE?

Do you have to migrate your libraries? This is the single most important question on the minds of customers moving to SAS 9. Under the Migration Community link on www.support.sas.com, you will find a Compatibility Calculator at http://support.sas.com/rnd/migration/planning/files/calculator/. Clicking answers to a few questions using this calculator will help you determine whether or not you need to migrate.

INTENDED USAGE OF PROC MIGRATE

Unlike other procedures, Proc Migrate is designed to process a library once each release. After the library is migrated to the current release, Proc Migrate will likely not be needed again for that library until the next release. The only intended usage of Proc Migrate is to move a Version 6.12 or later library forward to the current release; it is not designed to have the flexibility of one of the conversion procedures. For example, note that you are not allowed to migrate individual files from a library. This restriction is necessary for several reasons, the most important being the retention of referential integrity constraints and maintaining data set generation groups.

Proc Migrate is not designed to migrate libraries from one machine architecture to another. That functionality is already fully integrated into the aforementioned conversion procedures. Migrating libraries across machine architectures is strongly discouraged, as results are not guaranteed. Use Proc Migrate for that task at your own risk. The exception to this rule is when you need to migrate a 32-bit library to a 64-bit library for use with SAS 9; then Proc Migrate is just what you are looking for. See *Migrating 32-bit libraries to 64-bit libraries* later in this paper.

PROC MIGRATE SYNTAX

```
PROC MIGRATE IN=source-libref
             OUT=target-libref
<BUFSIZE=bufsize >
<MOVE> <KEEPNODUPKEY>
<SLIBREF=remote-libref-for-32-bit
       migration>;
RUN;
```

For the normal case, the only required arguments are the source and target librefs. The *source-libref* references a Version 6.12 or later library. The *target-libref* references a library using the current release; it is recommended that the target library be empty prior to migration.

BUFSIZE is used if you wish to use a particular buffer page size for creation of all the members of the target library. The default is the original buffer page size used to create the source library member.

The MOVE option is for customers who are short on storage space and want the source library's member to be deleted once the migration of that member to the target library is completed without errors. This is for space-constrained customers only; it is much more effective to validate the migration of the member before it is deleted (as discussed in the *Validation of Your Library's Migration* section below).

The KEEPNODUPKEY option specifies that data sets in the source library with the NODUPKEY sort assertion will migrate with the assertion. Because a data set created pre-SAS 9 may have the assertion set erroneously, the default is to migrate without the NODUPKEY sort assertion.

The SLIBREF option is only necessary for migrating 32-bit catalogs in your source library forward to 64-bit catalogs in your target library. Note that 32-bit catalogs are inaccessible to SAS 9; therefore they must be migrated in order to be accessed. Also note that these catalogs are migrated to hosts within the same operating system family. See *Migrating 32-bit Libraries to 64-bit Libraries* below for more details.

With Proc Migrate, you migrate from a source library to a target library; as with the conversion procedures, you must migrate to a different physical location from the source library. There is no provision for migration in-place. The source and target librefs must reference different physical locations. Also, the target library must be assigned to the current version's BASE or TAPE engine. This is best accomplished in SAS 9 by:

```
libname outlib BASE 'valid-path';
or
libname outlib TAPE 'valid path';
 *when the valid-path location
  contains sequential members;
```

ADVANTAGES OF PROC MIGRATE

Proc Migrate provides functionality necessary for migration that the conversion procedures do not. It is the goal of Proc Migrate to produce members in the target library that only differ from the source library because they are in the new SAS format; in all other ways, they are exactly the same. In order to attain that goal, Proc Migrate's results are different than you are accustomed to when using the conversion procedures.

For example, the conversion procedures clean up data sets, such that deleted observations are removed and disk space recovered. This restructuring has its advantages, but results in a data set that is not historically accurate if you are tracking changes through an audit trail. Because the conversion procedures remove deleted observations, they cannot copy the audit trail; the transactions noted in the audit trail would not match the observations in the copied data set due to removal of deleted observations. In contrast, Proc Migrate does migrate audit trails, as it will migrate all deleted observations. Of course, Proc Migrate retains the deleted state of any deleted observations. We refer to this as "retaining deleted observations".

With Proc Migrate, target library members retain the created and last modified date/time of the source library members, just as Proc Copy does when the DATECOPY option is used. Note these date/time values are the internal date/times reported in the Proc Contents attribute information, and not the external date/times controlled by the operating system and reported by Proc Datasets. See the DATECOPY documentation in *The DATASETS Procedure: COPY Statement* in the SAS OnlineDoc for more details. All integrity constraints and indexes are automatically retained in a migration. Generation data set groups are migrated. Compressed files remain compressed, and encrypted files remain

encrypted. Passwords are also retained, though you do not need to specify them at migration time. More concisely, all attributes of the source library's members are maintained in the migration.

In order to migrate an indexed data set, the data set is first migrated, and then the index is applied. Note that if errors occur while indexing a migrated data set, the data set will be migrated without the index; this will be noted by a warning in the SAS log. Conversely, if errors occur when applying integrity constraints or when migrating an audit trail, the associated data set will be deleted from the target library and an error will be written to the SAS log. The reason for this discrepancy in error behavior is best explained in terms of data integrity. A data set that is migrated without its index may lead to a performance issue, but its data integrity remains intact. However, a data set migrated without its integrity constraint or audit file constitutes a data integrity exposure.

Another advantage to Proc Migrate is its simplicity. When using Proc Migrate, there is no need to think of all the possible permutations of options available with the conversion procedures. Data attributes and auxiliary files such as indexes and audit trails are automatically migrated to the target library. The target library becomes a clone of the source library, in the format of the current release.

SUGGESTED USAGE

It is recommended that the target library be empty before your migration begins. This guarantees that the target library will not be a *mixed* library, i.e. one containing members created by different engines. Also, if a member of the same name currently exists in the target library, that member will not be migrated from the source library; an error will be printed to the log.

If errors are encountered when migrating a member of the source library, and the cause of the errors is eradicated, a second Proc Migrate invocation will result in migrating only those members not already in the target library. In other words, Proc Migrate does not overwrite existing members in the target library.

DETAILS ABOUT SPECIFIC MEMBER TYPES

Data Files

For data files, the MIGRATE procedure retains encryption, compression, deleted observations, all integrity constraints, created and modified date/times and migrates the audit trail. Indexes and integrity constraints are rebuilt on the member in the target library. Migrated data files take on the data representation and encoding attributes of the library when not using remote library services; when not using remote library services, data sets take on the attributes of the host on which the target library is located. Other attributes retain the values of the source library's member. See Table 1.1 below.

When migrating data files that have NODUPKEY asserted in the metadata, you will see a WARNING in the SAS log stating that the NODUPKEY assertion was removed on the target library's data set. Re-sort the migrated data file using the FORCE option in Proc Sort so that observations with duplicate keys are eliminated and the correct metadata is recorded. If you are certain that the data is currently correct, you may specify the KEEPNODUPKEY option to retain the NODUPKEY sort assertion on the migrated data.

Conversion Procedures	Proc Migrate
No deleted observations retained	Deleted observations retained
No audit trails copied	Audit trails migrated
General integrity constraints copied only when option specified, referential integrity constraints copied as well if entire library copied	All integrity constraints migrated
Created and last modified date/times retained with an option on some conversion procedures	Created and last modified date/times retained automatically

Table 1.1

Data Files with Audit Trails

When an audit trail is migrated, you will see an additional entry in the audit trail indicating that the data set was migrated. The audit trail is migrated in its present state. If the audit trail was suspended, the resultant audit trail is also suspended after the "MIGRATE" record is written. See Example 1.1.

```
Obs  reason_code      _ATMESSAGE_
 1   Added record
 2                     SUSPEND
 3                     MIGRATE
 4                     SUSPEND
```

Example 1.1 Proc Print of Suspended and Migrated Audit Trail (keep=reason_code _ATMESSAGE_)

Views

There are three categories of views to be considered: DATA step views, SQL views and

ACCESS engine views. Each uses an entirely different format and has different migration considerations.

SQL Views

SQL views retain data in a transport format; therefore there are no problems to be concerned about when migrating an SQL view.

DATA Step Views

In SAS 8.0, DATA step views gained the ability to store the source used to create the view. Using that source, DATA step views are migrated to current versions of SAS by automatically recompiling the source the first time the newly migrated DATA step views are accessed. In this way, the execution of the DATA step view will take advantage of fixes and upgrades available in the current release. DATA step views previous to SAS 8.0 or DATA step views not containing their source cannot be migrated. They must be recreated from the source.

ACCESS Engine Views

ACCESS views written with the Oracle, Sybase or DB2 engines will be migrated by making use of a new procedure specifically for converting these views. Proc Migrate will call this new procedure, Proc Cv2view, on your behalf. Views from Version 6.12 and on will be migrated. Please see the documentation for Proc Cv2view under *SAS/ACCESS for Relational Databases: Reference* in the SAS OnlineDoc for more information.

Catalogs

Proc Migrate makes use of Proc Cport and Proc Cimport to migrate catalogs. Those procedures are called on your behalf, which is why you may notice CPORT or CIMPORT notes written to the log by Proc Migrate. CPORT and CIMPORT restrictions apply; for example, catalogs in sequential libraries are not migrated. The result of using the CPORT/CIMPORT combination is that catalogs are fully converted to the current version. Proc Copy, for example, does not provide this functionality. Instead, it changes the outer "container" catalog to the new version, but leaves the entries as they were, to be converted as they are updated. Proc Migrate converts the entries, by way of CPORT/CIMPORT. You can use CPORT/CIMPORT to convert your catalogs outside the Proc Migrate environment, but Proc Migrate provides one-step migration for your entire library. Note that there are special concerns with catalogs when migrating from a 32-bit library. See *Migrating 32-bit libraries to 64-bit libraries* later in this paper.

MDDBs

Proc Migrate will migrate MDDBs to a SAS 9 library. Note that although Version 7 MDDBs will migrate to SAS 9 with no error messages, like all Version 7 MDDBs, it cannot be accessed with any version other than SAS 7.

Program Files

Stored compiled DATA step programs cannot be migrated; they must be recreated from the source. You will see an error in the log when attempting to migrate them.

Item Stores

There are no special concerns for the migration of item stores, unless you are migrating from a 32-bit library to a 64-bit library. Item stores cannot be migrated in that case because they cannot be migrated across machine architecture boundaries. For any migration with MOVE, note that item stores cannot be stored in a sequential library. This is important only if you wish to save a copy of your library to tape previous to using the MOVE option in Proc Migrate.

MIGRATING 32-BIT LIBRARIES TO 64-BIT LIBRARIES

If you were using Version 8 or previous versions of SAS on AIX, Solaris or HP/UX platforms, then you probably have 32-bit members in your libraries. If you were using Version 8 or previous versions of SAS on OpenVMS Alpha or OpenVMS VAX and have catalogs in your libraries, you have catalogs that need to be migrated; skip to *Special Considerations for Catalogs.* Other platforms do not have 32-bit to 64-bit migration considerations.

For AIX, Solaris or HP/UX, Version 8.2 SAS was available in either 32-bit or 64-bit. Previous to that release, all versions of SAS were 32-bit. If you have 32-bit data in your libraries, you may need to migrate. You can determine if your Version 8 data sets are 32-bit by using Proc Contents in SAS 9; look at the *Data Representation* field. If the data representation ends with "_32", the data set is in 32-bit format. If the data representation ends with "_64", then your data set is already 64-bit and may not require migration.

Migration is important to customers with 32-bit libraries because access to your 32-bit library is limited. With SAS 9, you can read and write 32-bit data sets. You can read SQL views, ACCESS views and MDDBs. If you need additional access to these members, you should migrate your library. Other members must be migrated for any access.

Proc Migrate will automatically migrate the members of the source library to 64-bit versions in the target library, removing the access limitations mentioned above. If you are using Remote Library Services (SAS/CONNECT or

SAS/SHARE), you must assign the source and/or target libnames through a SAS 9.1 server.

Special Considerations for Catalogs
For the affected platforms mentioned above, with the exception of AIX Version 6 catalogs, in order to migrate a 32-bit catalog to a 64-bit catalog you must have access to a Version 8 32-bit server and Remote Library Services (RLS) from SAS/SHARE or SAS/CONNECT. Your catalogs are read through the server and written in 64-bit data representation in the target library.

Assign a libref through a 32-bit Version 8 server which references the source library. Use this libref with the SLIBREF option to migrate your catalogs. See example 1.1.2 below.

```
/* share1 is a 32-bit version 8 server
   where mycats is assigned to the
   location of the source library */
libname mycats server=share1;
proc migrate in=source out=target
   slibref=mycats; run;
```

Example 1.1 Migrating Catalogs through a 32-bit Server

If you do not have either SAS/SHARE or SAS/CONNECT, you will need to migrate your catalogs by hand using CPORT on a 32-bit SAS invocation, and CIMPORT in the current version of SAS to place a migrated copy in target library.

For AIX Version 6 catalogs, you must migrate your catalogs by hand, using CPORT in Version 6 to create a transport file and using CIMPORT in SAS 9 to write the 64-bit catalogs to the target library. Other AIX library members can be migrated with Proc Migrate.

For more information, please see http://support.sas.com/rnd/migration/planning/files/foreign.html

VALIDATION OF YOUR LIBRARY'S MIGRATION

Since Proc Migrate is a new procedure, we needed to develop methods and tools to validate its behavior. The software validation motto, "Know your requirements, and know your data" is certainly familiar to most readers. This concept played a central role in defining the methodology we developed to perform this task. Keeping this concept in mind may help explain why we chose certain data checking methods over others. The validation macros discussed grew out of our validation choices and became the tool we used to validate our results. We realized that customers could also use it to ensure the correctness of their own data migration. This tool alone is not enough to provide that certainty, however. It must be used in conjunction with your knowledge of the contents of your data libraries.

PROC MIGRATE VALIDATION STRATEGY

The migration validation strategy can be conceptualized in two basic parts. First, you must determine the expected behavior of the migration of a particular library. Then, after the migration, you must prove that the migration produced those expected results. In other words, what members and member attributes were in the source library prior to the migration? What members and member attributes appear in the target library after the migration?

SOURCE LIBRARY MEMBERS

Unfortunately, knowing your data is not as simple as knowing what member types exist in the source library. Consider the hidden complexities of SAS data sets. Do your data sets have indexes? Integrity constraints? Password protection? Permanent formats? What are the lengths of your variables? Are your data sets labeled? Do your data sets have data representation and encoding values different from the default data representation and encoding of the operating system? In order to validate the migration, you must document all these details before the migration to determine whether or not the migrated data sets contain expected attributes.

Proc Contents can be used to produce all this information. However, besides documenting this information, it is also necessary to understand what attributes are expected to change when a member is migrated from the source to the target library. For example, since the data sets in the source library were created with a different engine than that of the target library, it's expected that the Proc Contents *Engine* value will differ between the source and target library. Certain differences from the source library are desired. Another example of this is the *Encoding* value in the Proc Contents output; this is a new field added in SAS 9; pre-SAS 9 data sets will display *Default* for that value. In the migrated member, you will see a different, more specific value.

PROVE PROC MIGRATE PRODUCED EXPECTED RESULTS

Customers who are familiar with Proc Compare will recognize the value of having Proc Compare perform the "heavy lifting" when comparing two data sets. However, early in our planning process, we determined that the output generated for validation must be reasonable. Whatever method we chose had to strike a balance between providing too little output to

adequately validate the results, and providing too much output to review in a reasonable amount of time. With the use of the proper options, Proc Compare produces very little output when there are no differences between two data sets, but it can produce an incredible amount of output if there are any differences. We decided that Proc Compare is best suited for situations in which data sets are expected to compare exactly. Unexpected differences will be immediately recognizable.

USING ODS TO VALIDATE PROC MIGRATE RESULTS

Consider this example of default Proc Contents output which displays the attributes of two data sets in the source and target libraries:

Data set in source library

Data Set Name	SOURCE.IC_TWO
Member Type	DATA
Engine	V8
Created	May 1, 1999
Last Modified	May 1, 1999
Protection	
Data Set Type	
Label	
Data Representation	WINDOWS
Encoding	Default

Data set in target library

Data Set Name	TARGET.IC_TWO
Member Type	DATA
Engine	BASE
Created	May 1, 1999
Last Modified	December 1, 2002
Protection	
Data Set Type	
Label	
Data Representation	WINDOWS
Encoding	wlatin1

A review of these two listings will quickly reveal differences in *Data Set Name, Last Modified* and *Encoding*. The *Encoding* difference in the example above is expected. (Note that the *Last Modified* time changes because of an integrity constraint being recreated after the data set was migrated.) Besides this attribute data, Proc Contents also reports variable information, engine/host data and index information.

If you used Proc Contents output to validate the results of Proc Migrate, the output generated from each data set in the source library would have to be manually compared to those in the target library. This would be a tedious but manageable task if you were comparing ten data sets in a library. But what if your library contained a hundred data sets? Or a thousand?

Fortunately, SAS offers an alternative to this tedious chore. The Output Delivery System (ODS) allows customers flexibility in choosing what output is produced by a SAS procedure and flexibility in choosing what that output contains. For example, by combining ODS output statements and a simple DATA step, you can direct Proc Contents output to a data set (similar to the OUT= option) and produce a listing of only those data set attributes that are different in the source and target libraries.

Attribute	source library
Data Set Name	SOURCE.IC_TWO
Encoding	Default
Last Modified	May 1, 1999

Attribute	target library
Data Set Name	TARGET.IC_TWO
Encoding	wlatin1
Last Modified	December 1, 2002

This output is considerably easier to review than the default Proc Contents output, but it can be refined even further. Because the libname is included in the data set name, *Data Set Name* will generate an insignificant difference. This data set attribute can be excluded from the comparison. Once the Proc Contents output is directed to a data set, you can use data set options to keep only those variables (i.e., data set attributes) that are deemed significant for the comparison.

Attribute	source library
Encoding	Default
Last Modified	May 1, 1999

Attribute	target library
Encoding	wlatin1
Last Modified	December 1, 2002

The SAS program that produced this output can be seen in Appendix 1.

INTRODUCTION TO THE MIGRATE VALIDATION TOOLS

We determined that customers are likely to have a large number of members in a source library, as well as having different member types. It seemed logical to automate the validation process using SAS Macro as much as possible. We felt it was important to limit the amount of required input for the Validation Tools.

There are two tools. Both are SAS programs. Neither of them is shipped with SAS, but they are both available for download via the Migration Community web page. The first tool, migrate_macros.sas, contains thirteen SAS macros designed to output the validation information previously described to the SAS output window. The second tool, migrate_template.sas, uses the SAS macros defined in migrate_macros.sas to streamline the migration and validation process. In its simplest form, migrate_template.sas is just seven lines of SAS code. Here's an example:

```
libname lib1 <engine> 'path 1';
libname lib2 base     'path 2';
libname ods           'path 3';

%before;
proc migrate in=lib1 out=lib2;run;
%after;

%checkem;
```

Before we discuss how to use the tools, we should explain the reason we separated the Proc Migrate step from the Validation Tools. We wanted to allow maximum flexibility in the Proc Migrate step. For example, customers who are migrating catalogs from a 32-bit library to a 64-bit library must add an SLIBREF option to their MIGRATE step that is not required for all customers. If we wrapped the Proc Migrate step into the same program that performed the validation, we would increase the amount of input required by the validation macros. By performing the Proc Migrate step separately from the validation macros, we can make the validation macros straightforward.

We also determined that since Proc Migrate was new, a certain amount of trial and error could occur as a normal part of the migration process. We wanted to design a solution that would prevent the need to restart the validation process from scratch should there be a need to rerun a portion of the migration. For example, if Proc Migrate encounters a SAS member in the target library with the same name and memtype as a SAS member in the source library, then Proc Migrate outputs an ERROR to the log and does not migrate the member. If the member in the target library is renamed or deleted, the Proc Migrate step could be repeated, and the member would be successfully migrated. Separating the Proc Migrate step from the Validation Tools allows customers in this situation to simply pick up the validation process where they left off.

How to Use the Validation Tools
Summary:

1. Copy the text of migrate_macros.sas into the SAS Program Editor.
2. Submit the code to make the macros available for the current SAS session.
3. Copy the text of migrate_template.sas into the SAS Program Editor.
4. Revise the three libname statements at the top of the program to match your migration situation.
5. Submit the code.

It's recommended that you begin a fresh SAS session before you run the Validation Tools. Copy the text of migrate_macros.sas from the Validation link on the Migration Community Webpage http://support.sas.com/rnd/migration/resources/procmigrate/validtools.html and paste it into the SAS Program Editor. If internet access is impossible for you, please contact your technical support representative for an alternative method to obtain the Validation Tools. Submit the code in the SAS Program Editor to compile the validation macros and make them available for the current SAS session. After the validation macros have been compiled, it is recommended that you clear the text in the SAS Log Window. This will make it easier to review any NOTES, WARNINGS or ERRORS produced by Proc Migrate.

Copy the text of migrate_template.sas from the Migration Community Webpage and paste it into your SAS Program Editor. Once you compile the macros either by copying the macro code into an interactive session and submitting the code, or by saving a copy of the validation macro program and using "%include", all that is required is to define three librefs: a source library, a target library, and a library to contain the ODS output data sets created by the validation macros.

```
libname LIB1 <engine> 'valid path
        referencing location of
        members to be migrated';

libname LIB2  BASE 'some other valid
        path referencing location
        to which members are to be
        migrated';

libname ODS BASE 'still another valid
        path referencing destination
        of ods output data sets';
```

The libnames LIB1, LIB2 and ODS are required by the validation tools. Note that the first libname

statement includes an optional engine assignment. It is necessary only if you have a *mixed* library, i.e. one containing members created using different versions of SAS, such as Version 6 and Version 8. In that case, the engine assignment in the libname statement specifies which version's members you wish to migrate. Also, please note the ODS library must be distinct from the LIB1 and LIB2 libraries.

SUGGESTED GLOBAL SAS OPTIONS

Migrate_template.sas includes two global options.

```
options nocenter formdlim='-';
```

These options were added to help simplify your review of the results of the Validation Tools in the SAS Output Window. The default output is left-justified. If you prefer your output to be centered, remove the NOCENTER option or submit "OPTIONS CENTER" before submitting the %before macro. The FORMDLIM option replaces page breaks with a dashed line in the SAS Output Window. We found this helpful in navigating through the output. If you wish to print a hard copy of this output, you may wish to run the Validation Tools once to review the results in the SAS Output Window, then remove the FORMDLIM option before running the tools again to produce the desired output.

Also, migrate_template.sas surrounds the validation macros with "OPTIONS NONOTES" to suppress NOTES to the SAS Log generated by %before, %after and %checkem. Suppressing these NOTES can make it easier to review any NOTES, WARNINGS and ERRORs produced by Proc Migrate.

%BEFORE: DOCUMENTING THE CONTENTS OF THE SOURCE LIBRARY BEFORE THE PROC MIGRATE

Once the libname statements in migrate_template.sas have been revised and submitted, the next step is to capture information about the source library before the migration. What members are in the source library, and what are their attributes?

The %before macro catalogs the contents of the source library before migration and creates global "memtype flag" macro variables which are used to drive validation output produced later in the template. No output is produced the SAS Output Window, but messages are output to the SAS Log Window indicating what %before did behind the scenes. This information appears in the log even if "options nonotes" has been specified.

Now that we've documented the contents of the source library, we can submit a Proc Migrate step. After that, we can begin validating the library migration.

%AFTER: DOCUMENTING THE RESULTS OF PROC MIGRATE, PART 1: WHAT MEMBERS WERE MIGRATED

After the members are migrated, %after performs a similar cataloguing function on the members in the target library as %before did for the members in the source library. It produces the first report in the SAS Output Window: A side-by-side comparison of the contents of the source library before migration with the contents of the target library after migration:

```
members of target library after
MIGRATE (relative to source lib)

name      MemType  result
FORMATS   CATALOG  OK
TESTCAT   CATALOG  OK
C_INDEX   DATA     not MIGRATED
DATA_SET  DATA     OK
MYSTORE   ITEMSTOR OK
DS_VIEW   VIEW     OK
SQL_VIEW  VIEW     not in source
                     library
```

In this example, five of the seven members present in the source library were migrated successfully to the target library (result="OK"). The last file on the list was "not in source library", presumably because it was in the target library before the migration and not in the source library. One file ("C_INDEX") was not migrated, perhaps because there was some problem during the migration. You should check the SAS log for errors to determine the reason that this file was not migrated.

%after produces a comparison of the members of the source library before the migration with the members of the source library after the migration. Note that although C_INDEX was not migrated for some reason, it remains the source library after migration. By default, Proc Migrate has no effect on the members in the source library. The following report is your proof of that fact.

```
members of source library before and
after MIGRATE (OK indicates member
was present in source lib before
and after MIGRATE)

name       MemType  result
FORMATS    CATALOG  OK
TESTCAT    CATALOG  OK
C_INDEX    DATA     OK
DATA_SET   DATA     OK
MYSTORE    ITEMSTOR OK
DS_VIEW    VIEW     OK
SQL_VIEW   VIEW     OK
```

For those customers who wish to use the MOVE option with Proc Migrate, the Validation Tools can produce validation output only about what members were migrated; validation of the data is not possible. Customers who choose to use the MOVE option when migrating source libraries should do so with great care. It is not possible to use %checkem or any of the memtype comparison macros discussed below unless the source files remain in the source library after the migration. In other words, using the MOVE option significantly limits the validation tools at your disposal.

DOCUMENTING THE RESULTS OF PROC MIGRATE, PART 2: ARE THE MEMBERS CORRECT?

The %checkem macro uses information captured by %before and %after to output validation information about the following:

Audit trails
Catalogs (metadata only)
Data sets
Indexes
SQL views

%checkem can handle data sets with indexes, integrity constraints, generations, and, most importantly, audit trails. SQL views are processed with a data step and the resulting data sets are validated with Proc Compare. It should be noted that so long as the %before and %after macros are run prior to running %checkem, customers do not have to input member names, or even have to know how many members are expected to be in the source or target libraries.

An example of data set validation output appears below. %checkem outputs three reports for data sets:

1. a side-by-side comparison of data set attributes
2. a side-by-side comparison of data set engine/host information
3. a Proc Compare of the data.

%CHECKEM DEFAULT DATA SET OUTPUT 1: DIFFERENCES IN DATA SET ATTRIBUTES

The first title line of each data set validation output contains the name of the data set being compared. A counter indicating the current data set number (of the total number of data sets in the source library) appears in the second title line. The third title line indicates what the report contains, and the fourth title line tells us that all other data attributes not displayed in the body of the report were the same. The last title line indicates which of the three reports is printed, and the name of the validation macro run by %checkem to produce the report.

%checkem sample data set output 1: differences in PROC CONTENTS header data

```
SAMPLE_DATA_SET
Number 2 of 12 data sets in source
    library
Differences in PROC CONTENTS header
    data
Note:  all other header data was the
    same
Number 1 of 3 reports for this data
    set (from checkdata macro)

attribute    source     target
Encoding     Default    wlatin1
```

The *Encoding* difference in the example above is expected.

%CHECKEM DEFAULT DATA SET OUTPUT 2: DIFFERENCES IN ENGINE/HOST INFORMATION

As in first example above, the differences in engine/host information in the second example are limited to expected differences, *Host Created* and *Release Created*. The *Host Created* differs between the source and target libraries because the MS Windows operating system with which the source library was created is different than the current operating system, although it is in the same operating system family. Please note that *File Name* was excluded from this comparison because we expect differences; the libname is a part of the *File Name* and is expected to change from the source library to the target library, so the attribute is purposefully excluded.

%checkem sample data set output 2: differences in PROC CONTENTS engine/host data

```
SAMPLE_DATA_SET
Number 2 of 12 data sets in source
   library
Differences in PROC CONTENTS
   engine/host information
Note:  all other engine/host
   information was the same
Number 2 of 3 reports for this data
   set (from checkdata macro)

attribute       source     target
Host Created     WIN_PRO    XP_PRO
Release Created  8.0202M0   9.0100A0
```

%CHECKEM DATA SET OUTPUT 3: PROC COMPARE OF DATA SET CONTENTS

The Proc Compare used by the Validation Tools makes use of two COMPARE options: BRIEFSUMMARY prints only a short comparison summary to the SAS Output Window, and LISTALL lists all variables and observations found in only one data set.

%checkem sample data set output 3: PROC COMPARE of data

```
SAMPLE_DATA_SET
Number 2 of 12 data sets in source
   library
PROC COMPARE of data
Number 3 of 3 reports for this data
   set (from checkdata macro)

The COMPARE Procedure
(Method=EXACT)
NOTE: No unequal values were found.
```

The basic goal of Proc Migrate is to move an existing library forward to the current release so that the data within each member is exactly the same. Certain attributes of each member change so that the member can take full advantage of the features of the new release. The example above shows how we prove we have achieved that goal.

The default behavior of %checkem is to output only the metadata (and data, if any) that is different in the source library and the target library. To output a comparison of all metadata and data for all memtypes, submit the following:

```
%checkem (showall=yes);
```

This produces output for data sets similar to the following:

%checkem(showall=yes) sample data set output 1: comparison of header data

```
SAMPLE_DATA_SET
Number 2 of 12 data sets in source
   library
PROC CONTENTS header data
Number 1 of 3 reports for this data
   set (from checkdata macro)

attribute          source  target
Compressed          NO       NO
Data Representation WINDOWS  WINDOWS
Data Set Type
Deleted Observations 0       0
Encoding            Default  wlatin1
Engine              V8       V9
Indexes              1       1
```

%checkem(showall=yes) sample data set output 2: comparison of engine/host data

```
SAMPLE_DATA_SET
Number 2 of 12 data sets in source
   library
PROC CONTENTS engine/host data
Number 2 of 3 reports for this data
   set (from checkdata macro)

attribute           source   target
Data Set Page Size   4096     4096
First Data Page      1        1
Host Created        WIN_PRO  XP_PRO
Index File Page Size 4096     4096
Max Obs per Page     126      126
```

%checkem(showall=yes) sample data set output 3: PROC COMPARE of data

```
SAMPLE_DATA_SET
Number 2 of 12 data sets in source
   library
PROC COMPARE of data
Number 3 of 3 reports for this data
   set (from checkdata macro)

The COMPARE Procedure
(Method=EXACT)
NOTE: No unequal values were found.
```

Submitting "%checkem(showall=yes)" produces all output for all members. You may wish to output "(showall=yes)" information for only one memtype. If this is the case, submit %before and %after in migrate_template.sas, and then submit whatever individual validation macros you wish.

```
libname lib1 <engine> 'path to source
                       library';
```

```
libname lib2  base    'path to target
                       library';
libname ods   base    'path to ODS
                       library';

%before;
proc migrate in=lib1 out=lib2;run;
%after;

%checkdata(showall=yes);
%checkcatalog;
%checkindex;
%checkaudit(showall=yes)
```

Refer to Table 2 below for a list of validation macro names. %checkdata is one of a number of macros used by %checkem to assist in the validation of the migration of individual SAS files. Please refer to Table 2 below to determine which macro can be used to validate SAS files of a particular memtype. Full documentation of each of these macros can be found in http://support.sas.com/rnd/migration/resources/procmigrate/validtools.html

Memtype or file	Validation tool
Catalog	%checkcatalog
Data set	%checkdata
Data set with an index	%checkdata and %checkindex
Data set with integrity constraint	%checkdata and %checkindex
Data set with an audit trail	%checkdata and %checkaudit
Generations data set	%checkdata
Itemstore	NONE
DATA step view	NONE
SQL view	%checkview
SAS/ACCESS view descriptor	NONE
MDDB	NONE
Program file	NONE

Table 2: Suggested Memtype Validation Tools

CONCLUSION

Proc Migrate is a new tool in the SAS utility procedure tool belt, enabling you to easily upgrade your SAS libraries to the latest version of SAS. Using the procedure in conjunction with the validation macros discussed will allow you to have confidence in your migrated library's data integrity.

CONTACT INFORMATION

If you have questions, feel free to contact the authors of this paper. For Proc Migrate questions, send email to Diane.Olson@sas.com. For validation macro questions, send email to David.Wiehle@sas.com.

Appendix 1 Example of ODS "Smart Compare" Program

```
***libname source is source library;
***libname target is target library;

ods output attributes=atr1(keep=label1
                           cvalue1
            where=(label1 not in (" ",
                   "Data Set Name")));
ods listing close;
proc contents data=source.ic_two;run;
ods listing;

data atr1(rename=(label1=attribute
                  cvalue1=source));
set atr1;
run;

proc sort data=atr1;by attribute;run;

ods output attributes=atr2(keep=label1
                           cvalue1
           where=(label1 not in (" ",
                  "Data Set Name")));
ods listing close;
proc contents data=destin.ic_two;run;
ods listing;

data atr2(rename=(label1=attribute
                  cvalue1=target));
set atr2;
run;

proc sort data=atr2;by attribute;run;

data atr;
merge atr1(in=in1) atr2(in=in2);
by attribute;
if in1 or in2;
if source ne target then output;
run;

proc print data=atr;run;
```