Paper 152-31

SAS®9 Migration: How to Develop an Automated Process Using Perl

Xiaoming Liang, Fred Hutchinson Cancer Research Center, Seattle, WA

ABSTRACT

Migration process usually presents challenges when multiple source locations are involved. Identifying those source locations manually could be very labor intensive since SAS files may exist in any sub-directories. Migrating SAS files from those locations is also very time consuming due to the need of code modification to reflect the change of source location each time. This paper is intended to introduce an automated migration process, which complements SAS with Perl scripts to traverse the directory tree to identify and extract source locations, and to update macro variables used in SAS migration script. When migrating SAS files from 32-bit to 64-bit environment, PROC CPORT/CIMPORT is used to migrate catalog files, while PROC MIGRATE is used to migrate datasets.

INTRODUCTION

Migration of SAS files to a new version can be very simple if all of your source files are in one location. However, in reality, it's usually a complicate process as you may deal with multiple protocols, with multiple studies from each protocol, and multiple types (e.g. primary, secondary) of datasets from each study. Going through each directory and sub-directory to search for SAS files for migration is very labor intensive. In Perl, the *file::find* module exports the *find* function, which can perform recursive searches through a directory tree (Wall et al. 2000) Using this method could efficiently extract the SAS file locations and prepare for the migration automation.

PROC MIGRATE procedure is recommended by SAS Institute for SAS 9 Migration due to its benefit of retaining audit trails and integrity constraints (Olson & Wiehle, 2004). However, that method requires Remote Library Services (RLS) when migrating catalogs from 32-bit to 64-bit environment. The RLS needs to be completely turned off for library reference update whenever the source location changes. Therefore, this method is suitable for datasets migration, but not for catalog files migration when multiple source locations are involved. The convention method, CPOR/CIMPORT, however, doesn't require the RLS, and thus is chosen to migrate catalog files in the automation process.

This paper introduces the automated migration process in 32-bit Solaris environment. The CPORT/CIMPORT procedures are used to migrate catalog files, while the MIGRATE procedure is used to migrate the datasets. The automation process is accomplished by complementing SAS with Perl scripts to traverse the source directory tree, to identify and extract source locations, and to update macro variables used in SAS migration scripts.

MIGRATION AUTOMATION

The purpose of this project is to migrate all SAS 8 files (datasets and catalogs) in the */trials* directory tree. To avoid inadvertently running against the source installation, the target environment is setup to be identical to, but physically isolated from the source (SAS migration community). One can read access the source */trials* from the target environment via */main/trials*. The *traverse.pl* script is executed in the source environment to extract the source locations, while the other migration scripts are executed in the target environment to migrate SAS files from the source to the target.

The automated migration process described in this paper involves 6 steps, with the 1st two steps for preparation, and the remaining 4 steps for migration automation.

1. Traverse the directory tree to identify source locations
2. Count source locations
3. Call migration scripts
4. Extract individual source location
5. Migrate catalog files from the source location
6. Migrate datasets from the source location

IDENTIFY SOURCE LOCATIONS

The traverse.pl script is developed to traverse the */trials* directory tree to search all of SAS 8 files. The program call and scripts are shown below:

```
      perl traverse.pl  /trials  > file_list.txt
1     use File::Find;
2     my  $start_dir = @ARGV;
3     my %path;
4     find(\&wanted, $start_dir);
5     map {print $_} keys %path;
6     sub wanted {
7     $path{"$File::Find::dir\n"}++ if /\.sas7bdat$|\.sas7bcat$/;
8     }
```

The root directory and the external file name are provided through the command line. Line 1 of the script *traverse.pl* instructs the Perl compiler to import the *File::Find* module. The module exports the *find* function, which will be used on line 4. Line 2 defines a scalar variable *$start_dir*. This variable holds the first argument */trials* passed in the command line through the standard array *@ARGV*. Line 3 defines a hash *%path*. A hash is composed of a set of keys and a set of values, with each key corresponding to an individual value within the hash. The find function (on line 4) calls a subroutine *wanted*, and passes the root directory to the sub. The sub uses a regular expression to look for SAS files ending with *sas7bdat* (dataset) or *sas7bcat* (catalog). When found, a hash element will be created, with the key containing the file location, and indexed by an integer value. The map function (line 5) is used to read every key of the *%path* and write to the external file. The *$_* is the default scalar, which holds the individual key from the *%path* hash. At the end, the external file looks like the following.

```
/trials/microbe/p050/s082/data
/trials/microbe/p050/s082/qdata
/trials/microbe/p050/reporting/progress
/trials/microbe/p050/analysis/smc/data
/trials/perinatal/p012/s029/data
/trials/perinatal/p012/s030/data
… …
```

COUNT SOURCE LOCATIONS

The script *count.pl* is used to count the records in the source location file *file_list.txt*, and pass the counts to SAS migration scripts through macro statements. Line 1 defines a scalar variable *$count*. Line 2 opens an external file *file_list.txt* for input, while line 3 opens another external file *file_count.txt* for output. The file handles (*FILE, OUTFILE*) are used in the open function to refer to the external files. The input operator <> (line 4) instructs Perl to bring in data from input file. The counter (*$count*) counts the number of observations in the input file. The printer function (line 7) writes a SAS macro statement to the output file, with the total number of records assigned to the macro variable *&total_file*. Both external files are closed on Lines 8 and 9.

```
1     my $count=0;
2     open(FILE, "< file_list.txt");
3     open(OUTFILE, "> file_count.txt");
4     while(<FILE>){
5     if ($_ gt " ") {++$count;}
6     }
7     print OUTFILE "%let total_file=$count;\n\n";
8     close OUTFILE;
9     close FILE;
```

CALL MIGRATION SCRIPTS

The *migrate.sas* (as shown below) program is called to launch the migration automation process. The *file_count.txt* file is included to use the macro variable *&total_file*. A *%do* loop is constructed to extract individual source location by the *shift_file.pl*, and migrate the catalogs and datasets from that location by SAS migration scripts. Details about the SAS migration scripts are illustrated in the later sections.

```
%macro migrate;
   %include "file_count.txt";
   %do i=1 %to &total_file;
      x "perl shift_file.pl ";
      x "/sas82/sas  cport_cat.sas";
      x "/sas9/sas cimport_cat.sas";
      x "/sas9/sas migrate_data.sas";
   %end;
%mend migrate;
```

```
%migrate;
```

EXTRACT INDIVIDUAL SOURCE LOCATION

The Perl script *shift_file.pl* is used to extract the source location and pass it to the migration script through the SAS macro statement. Lines 1 and 2 allow Perl to connect to the external files through file handles. Line 3 uses an array *@files* to hold records from the external file *file_list.txt*. Line 4 applies shift function to shift records (one at a time) from the array to a scalar *$out*. Line 5 creates the macro variables (*%source, %target*) using the information from the scalar *$out*. External files are closed on line 6 and 7. Lines 8 and 9 open *file_list.txt* for update using the remaining records in the array *@files*. Line 10 closes the file.

```
1     open (FILE, "< file_list.txt");
2     open (OUTFILE, "> source.txt");
3     my @files = <FILE>;
4     my $out = shift(@files);
5     print OUTFILE "%let source=/main$out;\n %let target=$out;";
6     close OUTFILE;
7     close FILE;
8     open (OUTFILE2, "> file_list.txt");
9     print OUTFILE2 @files;
10    close OUTFILE2;
```

MIGRATE CATALOG FILES FROM SOURCE LOCATION

Program *cport_cat.sas* is called to include the source location, and use PROC CPORT to export the catalog files from source into a transport file called *temp_file*. The second program *cimport_cat.sas* is called to include the target location, and apply PROC CIMPORT to restore the transport file to catalog files and put them into target location. While running the first program in SAS 8 and the second program in SAS 9, we can migrate version 8 catalog files from the source to the target.

```
%macro cport_cat;
   %include "source.txt";
   filename file1 "temp_file";
   libname lib1 "&source";
   proc cport library=lib1 file=file1 memtype=catalog;
   run;
%mend cport_cat;
%cport_cat;
```

```
%macro cimport_cat;
   %include "source.txt";
   x “rm –f &target/*.sas7bdat”;
   x “rm –f &target/*.sas7bcat”;
   x “rm –f &target/*.sas7baud”;
   filename file1 "temp_file";
   libname lib2 v9 "&target”;
   proc cimport library=lib2 infile=file1;
   run;
%mend cimport_cat;
%cimport_cat;
```

MIGRATE DATASET FILES FROM SOURCE LOCATION

Finally, program *migrate_data.sas* is used to migrate version 8 datasets from the source to the target. It first copies all files from the source location to a temporary directory *temp_dir*, and then removes all catalog files from the *temp_dir*, preparing for datasets migration. PROC MIGRATE procedure is then used to migrate all of the dataset files.

```
%macro migrate_data;
   x “rm –f temp_dir/*”;
   %include “source.txt”;
   x “cp &source/* temp_dir/”;
   x “rm –f temp_dir/*.sas7bcat”;
   libname lib1 “temp_dir”;
   libname lib2 v9 “&target”;
   proc migrate in=lib1 out=lib2 keepnodupkey;
```

```
   run;
%mend migrate_data;
%migrate_data;
```

CONCLUSION

This paper has shown how to use Perl scripts to complement SAS in automating the migration process. The find function provides an efficient way to search files across directory. Through the open, shift, and print functions, you can read the source location from the location file, and pass the information to the migration script through macro statements. The MIGRATE procedure can be used to migrate datasets so that the audit trails and integrity constrains can be retained, while the CPORT/CIMPORT procedures can be used to migrate catalogs from 32-bit to 64-bit environment without the RLS. When dealing with hundreds of protocols, this automated migration method can save days of tedious work and avoid human errors in manual migration process.

REFERENCE

1. Olson, D., Wiehle, D. 2004. *Proc Migrate: How to Migrate Your Data and Know You've Done It Right*. Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference. Paper 288-28.3.
2. SAS migration community. *http://support.sas.com/rnd/migration/execution/install_1.html*
3. Wall, L., Christiansen, T., and Orwant, J. 2000. *Programming Perl (3rd edition)*. Page 889. O'Reilly.

ACKNOWLEDGMENTS

The author would like to thank Geoff Snyder, Al Williams, Rita Lai, and specially Bryce Baril for their support and/or guidance throughout the project.

CONTACT INFORMATION

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

Xiaoming Liang
1100 Fairview Ave. N., LE-400
P. O. Box 19024
Seattle, WA 98109-1024
Work Phone: 206-667-6064
Fax: 206-667-6888
Email: xliang@scharp.org

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

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