Loading SAS® data in Informix: An Integrated Approach

M. Kumar Sagar
The Sagar Group, Inc., Framingham, MA

ABSTRACT

In a typical clinical trial, especially large and/or multicenter ones, there are many sources of data, including electronic data transfers from sites, central labs and CROs. While the data may come in many formats, the question arises: how to load these data in to appropriate databases. In this context, we have taken a holistic approach whereby SAS is used as the platform of choice to deal with these data transfers. SAS transport datasets coming in are converted to host SAS datasets, and required data transformations and manipulations are applied. Once the data is in the desired format, an output file is written and then, using SAS/ACCESS® for Informix, an Informix table is created and populated with the data from the SAS datasets. In this process, UNIX utilities are also used as an umbrella, to perform various tasks such as input verification, error checking, invoking SAS programs, cleaning the data, and to archive the associated files. This approach allows us to develop one-step utilities that are easy to run, easy to document and reduces the need to remember a plethora of steps and the need to follow these steps in a given order. Moreover, this reduces the complexity of the data transfer and loading process, allows us to set up the standards and provide users with validated tools and utilities. A case study of a utility developed at a large biotech company will illustrate the following:

- Utility Design and Specifications
- Usage of Operating System Utilities
- Usage of SAS/ACCESS for Informix to create/drop and populate tables
- Validation and Testing
- Documentation

INTRODUCTION

SAS/ACCESS is a family of products that allow access to a number of relational databases from within the SAS System for issuing queries, updating data in databases, reading data from databases, etc. However, not all products in this family are created equal. For example, SAS/ACCESS for Oracle® has both an interactive interface that allows the creation of access and view descriptors and data loads using PROC DBLOAD as well as the PROC SQL pass-through facility to create PROC SQL views and to pass native SQL statements to the databases. This is not true in case of SAS/ACCESS for Informix where currently only the PROC SQL pass-through facility is available and there is no automated facility to load data in an Informix database. It was in this context that we decided to utilize SAS/ACCESS to load data in an Informix database. This is our story.

BACKGROUND

For one of the single-center clinical trials being conducted, the data is being collected in two forms: CRF as well as electronic. CRF data goes through the usual process of double entry, arbitration, etc. The electronic data is transferred from the clinical center as SAS transport datasets. The site has a proprietary system that is not very flexible and can capture data only in a certain fashion. For example, all variables are captured as character data only, and hence, need to be broken down in three segments: original value, numerical value, and other characters such as =, <, >, L, H, etc. Additionally, all the dates need to be reformatted as well as flag variables need to be created to identify what portion of the date is incomplete, if any. Some datasets also need to be broken down into multiple database tables. A total of 6 SAS datasets were coming in which had to be processed, reformatted and loaded into 7 Informix tables. Finally, some variables had to be reformatted and renamed whereas some variables had to be created and populated with either null or calculated values, since existing in-house database utilities required it for various reasons; and it would reduce the complexity for study integrations for ISSs or ISEs.

THE UTILITY

Utility Design and Specifications:

Given this scenario, we decided to develop a utility that will provide a single command access to load the data from SAS transport datasets to the Informix

tables. The steps this utility had to perform are as follows:

1. Convert SAS transport datasets to SAS datasets.
2. Apply manipulations to the data in SAS datasets to yield data that can be loaded in the Informix tables.
3. Write an ASCII file that can be executed by SAS/ACCESS for Informix to load data in the database tables.
4. Drop the existing table(s).
5. Execute the file and load data.
6. Archive the program, log and lst along with the SAS datasets and other intermediate files.

The assumptions for developing the utility are as follows:

1. Data transmissions will always be complete. In other words, they will not be incremental. In this way, we can avoid the problems associated with data updates.
2. All the variables and datasets will be named according to a predefined convention so that we do not have to modify utility and SAS programs.
3. Files from the site will come in on a diskette and will be compressed using PKZIP® software. They will be named as:
4. YYMMDD.ZIP
5. where YYMMDD is the date of file creation.
6. The utility has to be validated, self documenting and self-archiving.

Given this information, we decided to exploit the capabilities of both the SAS System as well as the UNIX® operating platform as follows:

1. Use UNIX c-shell for inputs and error checking.
2. Invoke the appropriate SAS programs from within the shell utility, run them and provide the user with both UNIX and SAS errors/warnings, if any. Finally, archive the SAS datasets, log, lst and other intermediate files.

Utility Description:

LOADMDA

In the following discussion, capitalization is used to denote utilities or SAS programs, they aren't necessarily upper-case. LOADMDA, the c-shell utility, acts as an umbrella utility, to invoke and run a number of c-shell scripts, SAS programs and error checking routines. It is invoked at the command line by typing:

```
loadmda $1 $2
```

where

$1 = input SAS transport dataset name, and
$2 = date in mmddyy format.

Given these two inputs ($1 and $2), the c-shell utility does error checking to make sure that certain sub-directories (i.e. rawdata, sasdata, txtdata, outdata and sasprogs) used for archiving program logs and lsts and other files exists as well as $1 exists in the /rawdata sub-directory. If any of the above does not exist, the utility provides appropriate helpful message and terminates.

If no errors are found, then the utility continues whereby it writes a SAS program called loadmda.sas in the current directory, executes the SAS program, invokes another c-shell utility named CHECKLOG to search the loadmda.log for errors/warnings and prints any such on the screen. Then it invokes yet another c-shell utility called MVMDA to archive the SAS datasets, log, lst and other intermediate files. An abridged version of the LOADMDA utility appears in Figure 1.

LOADMDA.SAS

This SAS program is interactively written to the current directory by the LOADMDA utility every time it is invoked. Loadmda.sas contains a header with useful information such as date, time, user who invoked it, what is the purpose of this program, what other programs it calls, etc; followed by the libname definition as well as the macro variable for the date (i.e. $2 variable in LOADMDA) which will be inserted in all the tables as the data input date. Finally, appropriate %INCLUDEs are written to invoke the SAS programs needed to load the data. Figure 2 depicts a sample loadmda.sas program.

CHEM

This is a SAS program that is invoked by the LOADMDA.SAS, using %INCLUDE statement. This program reads the appropriate SAS dataset, renames and formats the variables, applies variable

transformation and manipulation macros contained in the MDAMAC file and also prints 2 listings, one pre and one post data processing. These listings help us validate the utility in the beginning and in continuous operation allow us to assure that utility works by randomly eyeballing the pre and post listings. Then the utility writes a text file (i.e.

```
#!/bin/csh
# LOADMDA Utility (K. Sagar)

echo "loadmda utility - load data in /dm"

# Error Trapping - test for files and directories
.....

# sample Help message if an error found

if ( $1 == '' ) then
   echo "You did not input source file."
endif

echo     " "
echo     "Usage: loadmda sourcefile date"
echo     "Example: loadmda X.OUT 121095"
echo     "In this example, X.OUT is a file in the
          /rawdata directory"
echo     "and /sasdata is a sub-directory in the
current  directory."
echo     "The date is input in MMDDYY format (6
          characters)."
echo     " "
echo     "EXITING . . ."
echo     " "
exit

# Set output filename and timestamp
set outfile = $cwd/loadmda.sas
set timestamp = "`date +'%d %B %Y %T'`"

# Write the loadmda.sas program
......

# Run the loadmda.sas program
/opt/sas611/sas $outfile

# Check the log and report errors, if any
checklog $outfile

# Move sas outputs and archive
/MDA/mvmda $1 $2 all

# Wrap up
echo     " "
echo     "Please check files loadmda.sas, loadmda.log
and      loadmda.lst in /sasprogs."
echo     "if any errors/warnings are found, Contact
          Kumar (x1891)."
```

Figure 1
abridged LOADMDA Utility

```
************************************************************;
* Written by /MDA/loadmda utility;
* Utility Author:          M. Kumar Sagar;
* Date:                    25 March 1996 12:49:28;
* User:                    selph;
* Purpose:                 Convert transport file to
                           SAS dataset and load data
                           into Informix tables;
* SAS Executable:          /dm/loadmda.sas;
* SAS Modules used:        /MDA/
                           chem, hem, coag, med,
                                    bmmorph, urine,
and vitals;
* Input Data:              /dm/rawdata/x.out;
* Output SasData Dir:      /dm/sasdata;
* Output TextData Dir:     /dm/txtdata;
* Output FinalData Dir:    /dm/outdata;
* Program Archive Dir:     /dm/sasprogs;
* NOTE:                    In above directories, all the
                           files are archived with
                           DATE appended at the end.
************************************************************;
options ... ;

libname indat xport '/dm/rawdata/x.out';

libname outdat '/dm/sasdata';

proc copy in = indat out = outdat;

%let date = input('032596',mmddyy8.);

%include        '/MDA/mdamac';
%include        '/MDA/chem';
%include        '/MDA/hem';
%include        '/MDA/coag';
%include        '/MDA/meds';
%include        '/MDA/bmmorph';
%include        '/MDA/urine';
%include        '/MDA/vitals';
run;
```

Figure 2
LOADMDA.SAS Program

chem.txt in this case) that contains statements executable by SAS/ACCESS for Informix. Then the program uses the SAS X statement to invoke SED®

utility to manipulate the data. This final step results in a text file named chem.out that is Informix loadable. Next, PROC SQL is invoked, a connection is made to the Informix database of interest, the table is dropped (if existing) and then the chem.out file is executed using %INCLUDE statement.

```
***       Process SERUM CHEMISTRY data;
***       rawdata from /sasdata/GE2_BC;
***       intermediate data in chem.txt;
***       final, uploadable data in chem.out;

data chem(keep = ... ); set outdat.ge2_bc;
          rename ....;  format ...;

proc print; title 'Listing 1 - Chem Data';
proc sort; by study patnum lsdt;

data chem; set chem; by study patnum lsdt;
          ***       convert char to num and flag;
          %breakvar(var = sodium);
          ***       initialize variables;
          %initall;

proc print; title 'Listing 2 -  Chem Data';
          footnote '-999 = Missing value';

data _null_; file 'chem.txt' new; set chem;
          by study patnum lsdt;
          length rec $100;
          ***       write the ASCII file;
          put 'execute ( insert into chem
          values (' ;
          rec = "'" || trim(study) || "',' ||
          put(patnum,8.) || ",'" ||
          put(lsdt,mmddyy10.) || "',' ||
          put(sodium,7.3) || ",'" ||   trim(sodiumq) || "','"
          || trim(sodiumx)  || "')) by informix;' ;
          put rec;

***       do the seds on the chem.txt file;
x "sed -e 's/-999.00/NULL/g' -e 's/-
          999/NULL/g' chem.txt >! chem.out";
x "echo 'Informix Loadable Output placed in file
          chem.out.'";

proc sql errorstop;
connect to informix(db = '//rosie/dm/');
***       comment following statement if table chem
          doesn't exist in  the db;
          execute (drop table chem) by informix;
          execute (create table chem
                    (study char(8), patnum integer,
                    lsdt date, sodium decimal(8,3),
                    sodiumq char(2), sodiumx char(8))
                    ) by informix;

%include 'chem.out';
quit;
```

Figure 3
Abridged CHEM Program

```
#!/bin/csh -f

echo Compressing and Moving.....

# rawdata file
if ($3 == 'rawdata') then
          echo $1 to rawdata/$1_out_$2.Z
          mv rawdata/$1 rawdata/$1_out_$2
          compress rawdata/$1_out_$2
          ls -l rawdata/$1_out_$2.Z

# sasdata files
else if ($3 == 'sasdata') then
          echo     "sasdata/*.ssd01 files to
                    sasdata/*_$2.Z"
          mv sasdata/ge2_bc.ssd01
          sasdata/ge2_bc_$2.ssd01
          compress sasdata/ge2_bc_$2.ssd01
          ls -l sasdata/ge2_bc_$2.ssd01.Z

# txtdata files
else if ($3 == 'txtdata') then
          echo "txt files to txtdata/*_txt_$2.Z"
          mv chem.txt txtdata/chem_txt_$2
          compress txtdata/chem_txt_$2
          ls -l txtdata/chem_txt_$2.Z

# outdata files
...

# SAS Executables, .log, and .lst files
...

else if ($3 == 'all') then
          mvmda $1 $2 rawdata
          mvmda $1 $2 sasdata
          mvmda $1 $2 txtdata
          mvmda $1 $2 outdata
          mvmda $1 $2 loadmda

else
          echo Please specify one of the following:
          echo "   gemmddyy.out"
          echo "   sasdata "
          echo "   txtdata "
```

```
        echo "   outdata "
        echo "   loadmda (i.e. loadmda.sas,
                 loadmda.log, loadmda.lst"
        echo "   all (to archive all [all of the
                 above options])"
        exit
endif
```

Figure 4
Abridged MVMDA Utility

MVMDA

Finally, this is the c-shell utility that archives all of the files in appropriate directories, as follows:

/rawdata	incoming SAS transport data file
/sasdata	processed host SAS datasets
/txtdata	txt files
/outdata	Informix loadable data files
/sasprogs	loadmda.sas, .log and .lst

As can be seen from Figure 4, MVMDA can be invoked as:

```
        mvmda $1
```

where

        $1 = type of file

Validation and Documentation

The validation of this utility was done by means of hardcopy comparisons. As can be seen from Figure 3, program chem prints out listings - one for the raw data and the other for the final data. Next, once the data is loaded in to the database, we generate a listing of the data from the database. That becomes listing 3. Each one of these listings are compared to each other to assure that only the intended data modifications take place. We also archive electronic copies of these documents as well as all the programs, log, lst and other files used as part of this validation. Finally, whenever any component of this utility undergoes a code modification, depending on the type of change, we do a module retesting and revalidation.

In terms of documentation, we created a document that identifies the utility requirements and specifications, data transformation and manipulations necessary, validation and documentation requirements, and a user manual. In fact, this document has been so successful that we are in the process of taking it a step further by trying to come up with a department wide Electronic Data Transfers handling document.

CONCLUSION

Based on our experience with this utility development process, here are some of the lessons we have learned:

1. Keep your programs simple and modularized. The more modularized they are, the easier it is to adapt them to ever changing needs of Clinical Trials Management.

2. UNIX platform has a host of powerful utilities that can help make the task of Electronic Data Transfers a smoother one.

3. Work closely with the end-users to identify the requirements of a utility such as this and anticipate to what other likely uses such a utility or a program can be put.

REFERENCES

ACKNOWLEDGMENTS

I would like to thank Tom Selph for participating in the development phase of this utility as well as for reviewing this paper. I would also like to express the gratitude I owe my wife, Sejal, for her encouragement and support, and for proof reading and reviewing this paper.

AUTHOR CONTACT

M. Kumar Sagar
The Sagar Group, Inc.
35 Queens Way, #7

Framingham, MA 01701

(508) 788 6936 Tel/Fax