Paper 3451-2015

DATU – Data Automated Transfer Utility

James Brittain, National Center for Health Statistics;

Robert Schwartz, Information Technology Services Office;

Centers for Disease Control and Prevention

ABSTRACT

The Centers for Disease Control and Prevention (CDC) went through a large migration from a mainframe to a Windows platform. This e-poster will highlight the Data Automated Transfer Utility (DATU) that was developed to migrate historic files between the two file systems using SAS® macros and SAS/CONNECT®. We will demonstrate how this program identifies the type of file, transfers the file appropriately, verifies the successful transfer, and provides the details in a Microsoft Excel® report. SAS/CONNECT code, special system options, and mainframe code will be shown.

In 2009 the CDC made the decision to retire a mainframe that had been used for years for primarily SAS analytic work and a few financial applications. The replacement platform for the SAS and analytic processing is a Windows® based SAS Grid® system referred to as the Consolidated Statistical Platform (CSP). The change from mainframe to Windows required the migration of approximately one hundred and eighty thousand files totaling approximately 16 terabytes. To minimize countless man hours and human error an automated solution was developed to accomplish this task. DATU was developed for users to migrate their files from the mainframe to the new Windows CSP or other Windows destinations.

Approximately 95% of the files on the CDC mainframe were one of three file types: SAS datasets, sequential text files, and Partitioned Data Sets (PDS) libraries. DATU dynamically determines the file type and uses the appropriate method to transfer the file to the assigned Windows destination. Variations of files are detected and handled appropriately. File variations include multiple SAS versions of SAS datasets and sequential files that contain binary values such as packed decimal fields. To mitigate the loss of numeric precision during the migration, SAS numeric variables are identified and promoted to account for architectural differences between mainframe and Windows platforms.

To aid users in verifying the accuracy of the file transfer, the program compares file information of the source and destination files. When a SAS file is downloaded, a Proc Contents is run on both files and the Proc Contents output is compared. For sequential text files, a check sum is generated for both files and the check sum file is compared. A PDS file transfer creates a list of the members in the PDS and destination Windows folder, and the file lists are compared.

The development of this program and the file migration was a daunting task. This paper will share some of the lessons learned along the way and our method of implementation.

INTRODUCTION

In 2009, the National Centers for Disease Control and Prevention evaluated the use of mainframe computing technology in its computing environment. CDC had been running SAS in the mainframe environment since the early 1980's. Initially, the SAS System was only available on the mainframe at CDC. By 2009, usage patterns for SAS software had changed. Many data managers and analysts were running SAS either on Windows workstations or in a terminal server environment. Certain applications performed SAS processing on dedicated Windows servers. Usage of the mainframe platform for analytic processing had diminished to two primary groups which worked with significant quantities of data containing Personal Identifying Information (PII). These groups continued to use the mainframe, primarily to take advantage of the data isolation that the mainframe architecture provided.

The evaluation of the role of mainframe computing at CDC was undertaken for a number of reasons. The existing mainframe system had been sized to support the financial systems of CDC and two other

operating divisions in the Department of Health and Human Services. Those financial systems had all been migrated from the mainframe environment to web-based systems. The operating cost of the mainframe system was approximately 4.4 million dollars per year. Even after turning off the software components used by the financial systems, the operating cost for the mainframe system was still roughly 3.2 million dollars per year. Also, the mainframe hardware was aging, and would reach end-of-life in 2012, at which point the cost for hardware maintenance would increase. Purchasing a new mainframe system was estimated to cost 2 – 3 million dollars. In the current budgetary climate, purchasing a new mainframe system was not possible. Going forward, it was decided to pursue an alternative to mainframe computing.

CDC developed a server-based computing cluster for SAS processing using the SAS/Grid Computing product. Development of this system started in 2009. At the same time, the project team evaluated the volume of data stored on the mainframe system. Mainframe datasets were considered to be candidates for migration if they were associated with user or corporate accounts known to be used for scientific work, if they were allocated using standard parameters for SAS data libraries, if they were allocated as sequential data files, or if they were mainframe partitioned datasets (PDS). Very few CDC mainframe users wrote and compiled programs which would use partitioned datasets as load libraries, so this case was ignored. Based on these criteria, it was determined that the number of datasets that could potentially require migration was over 180,000. When we determined the volume of data that potentially needed to be transferred, we realized that manually transferring these datasets was not practical, and that an automated approach to batch file transfer would be needed. The DATU utility was developed to meet the following objectives:

1. Facilitate transferring mainframe datasets in batches rather than individually.
2. Enable data stewards with little or no current mainframe experience to transfer mainframe data to network locations.
3. Use a simple set of control directives to define a batch transfer session.
4. Convert mainframe SAS data libraries from several SAS versions to the current V9 native engine format.

DATU

The Data Automated Transfer Utility (DATU) was developed to meet the need for a tool that would automatically process a list of datasets, identifying their type and performing the steps appropriate to transfer their contents from the mainframe Hierarchical File System to network shares in a Microsoft Windows environment. Transferring files from mainframe to Windows systems involves copying and translating from EBCDIC to ASCII character-encoding scheme. The program had to be designed to conditionally process 3 main file types identified for transfer; SAS data libraries, text files, and PDSs. For each file type a verification process was used to provide the user some feedback about the success or failure of the file transfer. After files are verified by the user as transfer complete and correct the file's mainframe management class was changed to PERMMIG (Permanent Migration) indicating that the file transfer was complete. This allowed reports to be run to track the status of the migration project over time.

The mainframe system used automated file migration to manage space. There was a limited amount of direct access disk space for files being currently used. After a file had not been accessed for a period of time it was compressed (Migration level 1). After a file was in Migration level 1 status and not accessed for a set period of time it was then moved to tape (Migration level 2). With the prospect of many users migrating files to a Windows environment we needed to find a way to do this without running the system out of direct access disk space when multiple users recalled files. The DATU program was designed to loop through files and transfer them sequentially; recalling a file from mainframe migration, translating and copying the file to Windows, and finally returning the file to mainframe migration level 2 freeing up space.

We strived to give the user the options needed to transfer the files in the method of their choosing. Some users preferred to keep the mainframe dataset name (DSN) and just transfer files to a Windows location. Others preferred to use the (.) dots in the mainframe DSN to create a Windows folder structure.

HIGH LEVEL PROCESS

The DATU program is controlled by a set of SAS macro variables. These variables define the high-level qualifiers of the collection of mainframe datasets to be transferred in a run, the path to the top-level folder in the Windows environment where the files will be transferred, and several parameters that provide user control of various aspects of the transfer process. Users are provided with a template SAS program which they customize for a given run of DATU, specifying source, target and control parameters. This SAS program then invokes the main DATU processing routine.

The process of downloading a batch of mainframe files begins by retrieving a list of all datasets beginning with the top-level qualifier specified. The user then was presented with a list of datasets to select / deselect to be included in the transfer batch. A single DATU transfer run can process any number of datasets, from one up to the total number with a single account as the top-level qualifier. Once the list of download candidates has been returned by the mainframe system; DATU evaluates each one, determining if it meets the criteria for one of the three file types described above. DATU then performs the download process appropriate to each file type. Files that do not meet the criteria for one of the three eligible file types or that are specifically excluded from processing are skipped.

PROCESSING OF TEXT DATA SETS

A significant number of mainframe datasets were not SAS data libraries. The majority of these were sequential flat files containing raw input data that had been entered manually by off-site key entry contractors, or that had been transmitted electronically to CDC over the Internet. Those that were unmodified from this original state consisted entirely of printable EBCDIC characters. Since our design goals for DATU included automating the transfer process as much as possible, we had decided to convert text datasets from the EBCDIC character set to the ASCII character set when they were transferred. We also wanted the transfer process to validate that the data had been transferred accurately. This goal raised the question of how to perform validation on files that had been translated during the transfer. We realized that we would have to develop a validation routine that would produce identical results when applied to the ASCII and EBCDIC versions of the same file.

We began development with two implementations of our platform-independent validation algorithm. One of these is run on the mainframe to analyze the source data file. The second implementation is run in the Windows environment on the transferred copy of the file. An inspection of a sample of datasets showed that the contents consisted of mostly alphanumeric characters, but if a file contained survey data, the comment fields could contain punctuation and national symbols. We defined a character variable (countChars) containing all of the printable characters that we determined might appear in a text data file. Each character in the string occupied the same position in the mainframe version of the string and in the Windows version. As a result, a call to the SAS INDEXC function to search for a particular character returns the same index value when run on the mainframe and in the Windows environment. The INDEXC character search function is the basis for our file comparison algorithm.

We elected to process each line of the input file separately. This was done because some of the datasets on the mainframe were quite large, and we decided that checking a sample of lines in the file would provide a good balance between the need to validate the transfer and the amount of time necessary to perform the validation. The process reads a line of the file into a buffer. Each character in the record is tested using the INDEXC function to determine if that character is in the countChars string. If it is found, the index of where it was found is added to an accumulator variable. If the contents of a character position in the input record is not found in the countChars string, that character position is re-read using the PIB1. INFORMAT, the contents of that byte in the record is accumulated in an accumulator variable for binary (non-printable) character codes, and a flag is set indicating that binary data were found. The two accumulator variables are written to a SAS dataset which contains one observation for each line tested.

During our testing of the initial process, we found a larger number of files containing mixed text and binary content than we expected. Consultation with groups that owned these files determined that in some projects files containing key entry data would be sent to separate contractors for specific statistical pre-processing. In many cases the pre-processing involved weighting calculations. To preserve numerical

precision, the weights were appended to the original records in floating point binary representation. Based on this finding, we realized that translating these files to ASCII mode would corrupt the floating point data. DATU would need to support transferring these datasets in binary mode.

We decided to transfer all text files once in text mode. This produces a version of the file in which the original text record content can be read using standard SAS INFORMATS. Files containing binary data are transferred a second time in binary mode. This produces a version of the file that will be identical to the version on the mainframe, but will require the use of EBCDIC formats in SAS to read. This also added a complication to the validation algorithm. Files that were only transferred in TEXT mode must be validated using the ASCII version of the validation routine. However, the files that contained mixed content must be validated using the EBCDIC character encoding. So we developed a third version of the validation routine, in which the character codes are defined using hexadecimal literal constants containing the EBCDIC character codes. This version of the routine produces a validation results dataset identical to the one produced by the mainframe version of the routine.

Table 1 illustrates a sample of Characters in the countChars strings, with Character Encodings and SAS INDEXC function Return Values.

Character	space	A	B		Z	0	1		9	!	@		a	b		z
ASCII Code	32	65	66		90	48	49		57	33	64		97	98		122
EBCDIC Code	40	C1	C2		E9	F0	F1		F9	5A	7C		81	82		A9
Index	1	2	3		27	28	29		37	38	39		71	72		96

Table 1. Sample of Characters in the countChars strings, with Character Encodings and INDEXC Return Values

With the three versions of the validation routine functioning properly, the following process is used to transfer a sequential mainframe file:

1. Run the mainframe version of the validation routine on the source file, either processing the entire file or the sample of records selected by the user. The routine downloads the validation results dataset.
2. The DOWNLOAD procedure is used to download the source file in TEXT mode.
3. If the file contains both text and binary content, it is downloaded a second time in BINARY mode.
4. The ASCII version of the validation routine is run on the target file if it only contains text data. Otherwise the EBCDIC version of the validation routine is run on the copy that was downloaded in BINARY mode.
5. PROC COMPARE is used to determine if the two validation datasets are equal. The results of the comparison are logged.

Our experience with this algorithm is that it only found the source and target files differed if communication between the mainframe and the Windows client performing the transfer was interrupted while the transfer was in process. The algorithm identified cases in which a batch of mainframe files should be re-run, and is considered to be a success.

PROCESSING OF PARTITIONED DATASETS

Mainframe partitioned datasets were used by SAS users to store SAS program code and Job Control Language (JCL) files. These consisted entirely of printable text characters, so the content evaluation performed for sequential text datasets was not necessary. We found one characteristic or feature of PDS members that we felt DATU should be able to address for users. PDS members only have an 8 character base name. They do not have a file extension as files in the Windows operating environment do. This can be inconvenient as several automation features, such as opening a SAS session by double-clicking a SAS program file, are defined based on the file extension. We addressed this in DATU by allowing the user the option of renaming downloaded PDS member files to add an extension of their choice.

The process for transferring PDS members consists of the following steps:

1. Create the target folder for the PDS member files, based on control directives specified by the user.
2. Download all of the members from the PDS. The DOWNLOAD procedure automatically creates separate files for each member.
3. Capture a directory listing of the files in the target folder and compare this listing to the PDS members on the mainframe to verify that all members were transferred.
4. Optionally rename the downloaded PDS member files to add a file extension.

PROCESSING SAS DATA LIBRARIES

Prior to downloading a mainframe file, DATU must determine the content type of each file being downloaded. For SAS data libraries, this type determination must also address the type of engine used to create the library. Our experience was that Base SAS could work with libraries written with the following older engines, but that the DOWNLOAD procedure would not transfer them:

- SASIO500
- V5
- V5TAPE
- V6
- V6TAPE

We incorporated a pre-conversion step which is run when the detected engine is one of those listed above to enable DATU to successfully download these libraries,

When migrating SAS files from mainframe to Windows there is a difference in the numeric length precision. SAS provides options in the DOWNLOAD procedure code to handle this which we set conditionally: V6TRANSPORT & EXTENDSN=YES.

Table 2 illustrates the difference in numeric precision.

Variable Length	Largest Integer z/OS	Largest Integer Windows/UNIX
2	256	N/A
3	65,536	8,192
4	16,777,216	2,097,152
5	4,294,967,296	536,870,912
6	1,099,511,627,776	137,438,953,472
7	281,474,946,710,656	35,184,372,088,832
8 (default)	72,057,594,037,927,936	72,057,594,037,927,936

Table 2. Largest Integer That Can Be Safely Stored in a Given SAS Length

SAS library identification consists of a series of steps:

1. Attempt to assign a libname to the file using the LIBNAME function.
2. Read the first 8 bytes of the file, looking for the tag #SASLAAS. This tag is part of the header of SAS libraries produced by several engines. If the tag #SASLAAS is not there, the file is processed as a text data file.
3. Determine whether one of the older engines created the library by examining the DSORG and RECFM allocation parameters and querying the SASHELP.VLIBNAM dictionary table.
4. If the library was created with one of the older engines, use the COPY procedure to copy the members to a temporary library in V9 engine format. Update the library metadata so DATU will download the temporary library.

USER INPUT SAS MACRO VARIABLES

TESTFLAG (“TEST” | “LIVE”):

Prefix for the batch filename. A later process to mark files complete (PERMMIG) only considered LIVE batches.

ROOT

High level qualifier(s) of the mainframe dataset(s) the user wanted to transfer by DATU.
I.e. To select a group of mainframe data sets, assign the **ROOT** macro variable to: "**ABC1"**, **"ABC1.LEVEL1"**, or **"ABC1.LEVEL1.LEVEL2"** to select all datasets that start with the same root. (Examples 1-3)

WINROOT

User designated Universal Naming Convention (UNC) path in Windows where the dataset(s) selected for transfer from the mainframe were to be transferred. (Examples 1-3)

"****" (backslash) at the end of the UNC path assigned to the macro variable **WINROOT** designated all the data sets being transferred will be placed in that folder.

"**.**" (dot) at the end of the UNC path assigned to the macro variable **WINROOT** designates data sets being transferred to have the last part of the UNC path assigned to **WINROOT** as the beginning of the filename or sub-folder name.

FILE_STRUCTURE ("FOLDERS" | "MAINFRAME"):

The user determined whether data sets were migrated to the Windows system in a UNC path folder structure or maintained the conventional mainframe data set names. (Examples 1-3)

FOLDERS - DATU will convert the periods in the mainframe data set name into a UNC path folder and sub-folder structure.

MAINFRAME - DATU will maintain the MAINFRAME data set name which consists of nodes separated by periods as the file name in Windows.

CHECK_NTH_REC

Specifies a multiple of the records you want to check when creating the Check Sum for text files.

1 – Checks every record
0 – Checks the first record only and gets a record count
Other values will skip records and check a percent of the file; speeding up the processing (5=20% / 10=10% / 20=5% / 100/1%)

TEXTEXT ("*file_extention*" | *blank*)

User specified extension (i.e. ***.TXT*** or ***.DAT***) or Blank. DATU appended the specified extension to all of the text files transferred. (Example 3-5)

PDSEXT ("*file_extention*" | *blank*)

User specified extension (i.e. ***.SAS*** or ***.TXT***) or Blank. DATU appended the specified extension to all of the PDS member files transferred. (Example 3)

Examples of using the "File Naming" macro variable settings:

Sample Input Mainframe Files:

ABC1.PROJXYZ.FILEA.FINAL (Text File)

ABC1.PROJXYZ.WORKFILE.DATA (SAS Library)
- DATA1 (SAS Dataset)
- DATA2 (SAS Dataset)

ABC1.PROJXYZ.PROG.LIB (PDS Program Library)
- PROG1.sas (SAS Program)
- PROG2.sas (SAS Program)

Example 1:

To maintain the same mainframe data set name for the data set that is being transferred by DATU, the user assigned the end of the UNC path value of the macro variable **WINROOT** the same as the value of the macro variable **ROOT** and selected **FILE_STRUCTURE=MAINFRAME**:

%LET ROOT = **ABC1** ;

%LET WINROOT = \\cdc.gov\CSP_Share1\ZOS_Archive**ABC1.** ;

%LET FILE_STRUCTURE = **MAINFRAME** ;

Output Windows files:

\\cdc.gov\CSP_Share1\ZOS_Archive\ABC1.PROJXYZ.FILEA.FINAL
\\cdc.gov\CSP_Share1\ZOS_Archive\ABC1.PROJXYZ.WORKFILE.DATA
DATA1.sas7bdat
DATA2.sas7bdat
\\cdc.gov\CSP_Share1\ZOS_Archive\ABC1.PROJXYZ.PROG.LIB
PROG1
PROG2

Example 2:
Maintain the same name for the dataset that is being transferred by DATU; but change the (.) "dots" to (/) folder structure **FILE_STRUCTURE=FOLDERS**.

%LET ROOT = **ABC1** ;

%LET WINROOT = \\cdc.gov\CSP_Share1\ZOS_Archive**ABC1** ;

%LET FILE_STRUCTURE = **FOLDERS** ;

Output Windows files:

\\cdc.gov\CSP_Share1\ZOS_Archive\ABC1\PROJXYZ\FILEA\FINAL
\\cdc.gov\CSP_Share1\ZOS_Archive\ABC1\PROJXYZ\WORKFILE\DATA
DATA1.sas7bdat
DATA2.sas7bdat
\\cdc.gov\CSP_Share1\ZOS_Archive\ABC1\PROJXYZ\PROG\LIB
PROG1
PROG2

Example 3:
Several files belong to the same project but are mixed in an account with a lot of other files that belong to different projects. In this example we will select just those files and replace the account name and project name with a more descriptive Windows name and create folders from the dots.

%LET ROOT = **ABC1.PROJXYZ** ;

%LET WINROOT = \\cdc.gov\CSP_Share1\ZOS_Archive**XYZ_project** ;

%LET FILE_STRUCTURE = **FOLDERS** ;

Output Windows files:

\\cdc.gov\CSP_Share1\ZOS_Archive\XYZ_project\FILEA\FINAL.**dat**
\\cdc.gov\CSP_Share1\ZOS_Archive\XYZ _project\WORKFILE\DATA
DATA1.sas7bdat
DATA2.sas7bdat
\\cdc.gov\CSP_Share1\ZOS_Archive\XYZ_project\PROG\LIB
PROG1**.sas**
PROG2**.sas**

Example 4:
(Based on example 3) Add (.DAT) to the end of each TEXT file.

%LET TXTEXT = **.dat** ;

Output Windows files:

\\cdc.gov\CSP_Share1\ZOS_Archive\XYZ_project\FILEA\FINAL**.dat**

Example 5:
(Based on example 4) If file FINAL.DAT contained NON-ASCII translatable characters a second file would be downloaded using BINARY mode FTP with an extension (.BINARY) appended to the end.

Output Windows files:

\\cdc.gov\CSP_Share1\ZOS_Archive\XYZ_project\FILEA\FINAL**.dat**
\\cdc.gov\CSP_Share1\ZOS_Archive\XYZ_project\FILEA\FINAL**.dat.binary**

CONCLUSION

The utility met all of the design objectives listed above. DATU was used to transfer approximately 16 TB of data from the mainframe data center to network shares. From a programming perspective, DATU implements a sophisticated communication process between SAS sessions in the Windows environment and the mainframe, which enables it to evaluate source files and select appropriate actions based on the file type and SAS library version. The DATU utility was instrumental in CDC being able to shut down the mainframe system.

REFERENCES

Vyverman, Koen. 2000. “Using SAS Data Exchange to Pour SAS Data into.” http://www.sascommunity.org/seugi/SEUGI2000/vyverman_dynamicdata.pdf.

SAS® 9.4 Language Reference: Concepts, Fourth Edition “Numerical Accuracy in SAS Software - Largest Integer That Can Be Safely Stored in a Given Length.” http://support.sas.com/documentation/cdl/en/lrcon/67885/HTML/default/viewer.htm#p0ji1unv6thm0dn1gp4t01a1u0g6.htm.

CONTACT INFORMATION

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

James Brittain
National Center for Health Statistics - CDC
jbrittain@cdc.gov

Robert Schwartz
Information Technology Services Office - CDC
rschwartz@cdc.gov