

Energizing End Users with a Slice of SAS® and a Cup of Java™

Randy Curnutt, Solutions Plus, Inc., Indianapolis, IN

Michael Pell, Solutions Plus, Inc., Indianapolis, IN

John LaBore, Eli Lilly And Company, Indianapolis, IN

ABSTRACT

Many corporate information systems have evolved to integrate a diverse mixture of hardware platforms. Although commonplace, the mainframe has been joined by a plethora of UNIX and NT servers as well as an army of personal computers often connected by Local Area Networks (LAN) or Wide Area Networks (WAN). Further, software tools in this environment are diverse. Users now wish to analyze their mainframe data with PC and UNIX tools in addition to using mainframe tools. It is often necessary to migrate data to new platforms because some tools require local data access. Users may become intimidated and stymied by the prospect of finding an easy and reliable method for moving mainframe data to remote platforms. We explore how the flexibility of SAS system architecture, combined with Java programming capabilities, provides instant access in an easy-to-use tool for point-and-click data migration and conversion (OS/390 SAS, DB2 views, Oracle views) to a PC or UNIX/NT servers. We also discuss methods to automatically convert the resulting dataset to the desired format (SAS dataset, SAS transport file, Excel, CSV, tab delimited, space delimited, or dBASE) on the target platform. Our tool implements a verification process that ensures integrity of transferred data. Additionally, meta-data is available before and after the transfer.

INTRODUCTION

BUSINESS PROBLEM

The DATAAccess Tool facilitates clinical data analysis by providing an easy way to transport and convert SAS data on the OS/390 mainframe to the platforms and formats desired for analysis. Instead of learning mainframe or communications programming commands, or waiting for a systems analyst to move the data for them, the user is free to focus on analyzing the data.

CONSTRAINTS

The budget for the project was limited to purchasing a Java integrated development environment and leveraging other existing tools already in house, such as PC SAS. The application had to adhere to all existing security policies across all platforms.

GOALS

The goal of the DATAAccess Tool was to provide an intuitive graphical user interface that gave non-technical users a simple and repeatable process for copying (and converting) data from any platform to any supported target platform. Data integrity was of the utmost importance, especially since this tool may be used to create SAS Transport files that are submitted to the regulatory agency for review. The DATAAccess Tool facilitates the verification process to ensure the transferred data is accurate. Since the tool brought together so many disparate technologies and platforms, it was imperative that it go through a rigorous

validation process to ensure that it functioned as desired. It was also important for the tool to provide instantaneous transfers since the users had previously experienced lengthy delays waiting on technical analysts to move and convert the data for them.

The DATAAccess tool allows the user to select test or production data, provides a list of research project codes for the selected environment. It then queries OS/390 to build a list of SAS libraries that the user is permitted to access for the selected research project code. Then a user can select a library and request that the tool identify the datasets available within that library (this generates another query to the OS/390). To transfer a dataset, the user selects a target host, a directory location, and a file format, then clicks a "Transfer" button.

ENVIRONMENT

Our environment is very diverse, both from a user perspective as well as from a technological standpoint. The end-users span a very broad base; from non-technical to very technical. This user base includes statisticians, systems analysts, data analysts, research scientists, and physicians; these staff are located at both US and overseas locations.

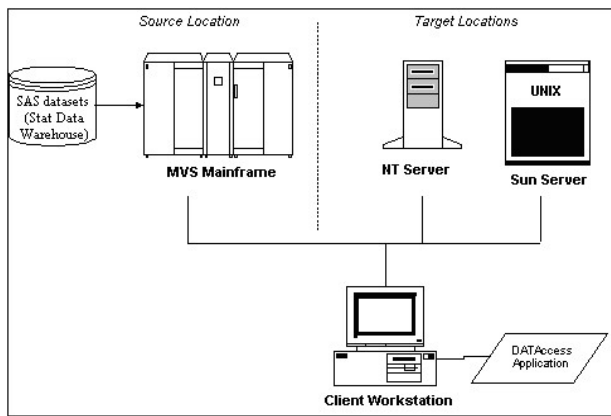
The hardware encompasses an OS/390 mainframe, Sun Solaris systems, Windows NT servers and Windows based client machines.

The software environment is comprised of a Java applet, Windows NT server, Windows 95/98/2000/NT clients, PC SAS 6.12, UNIX SAS 6.12 and OS/390 SAS 6.09, JCL, FTP. The data is a mixture of SAS, SAS Transport, DB2, Oracle, CSV, dBase, tab delimited text files, and Excel. Communications techniques utilize both SAS/Connect® and FTP. The DATAAccess Tool also generates and submits OS/390 JCL jobs, as well as dynamically generating SAS programs that run on the most appropriate platform via SAS/Connect (i.e., the Windows client, OS/390, UNIX, or NT Server).

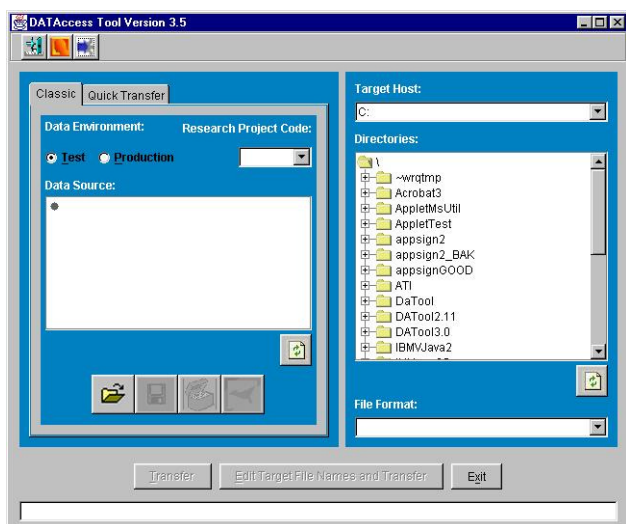
SOLUTION

SAS and Java provide the glue necessary to tie all of the various technologies together. Architecturally, the DATAAccess Tool is viewed as a pipe that connects a source to a target, and manages the transfer and conversion process. Realistically, the initial design was for the source to always be SAS data residing on an OS/390 platform, but we did not wish to architect to such a rigid standard. The basic idea is that it shouldn't matter what platform or data structure resides at either end of the pipe.

The following diagram depicts the physical architecture for the tool:



The user interface is developed with Java. The code that was developed can be deployed as either a stand-alone application or a Web based applet.



Based on the user's selections, Java is used to dynamically generate PC SAS, OS/390 SAS, UNIX SAS programs or OS/390 JCL jobs to accomplish the data transfer and conversion. The tool submits the generated program to the appropriate SAS engine depending on the client and server configuration. For example, if the user has PC SAS, then most programs generated by the application would leverage the local SAS software. If the user does not have PC SAS, then all generated programs are submitted to OS/390 SAS, which would make use of SAS/Connect as needed. If the user desires a transfer to the UNIX platform, this would always be accomplished by generating an OS/390 SAS program that runs SAS/Connect on the OS/390 to spawn a job on the desired UNIX host.

Generating SAS code for multiple platforms is relatively easy due to the fact that SAS code is highly portable across platforms.

We have included 3 code samples in this paper. The first is an example of a Java method to generate a SAS program. The output of the Java method is a String of text that is the SAS program. The second is an example of a SAS program that was created by Java, and the third is an example of a Java method that accepts the String of text, writes the String to a temporary file, then invokes PC SAS using the new file as a parameter to the sas.exe.

The following Java code sample is an example of using Java to

generate a SAS program:

```
/** Builds a PC-SAS program to transfer
 * the requested files.
 * The PC-SAS program will connect to the
 * MVS host using the
 * user's username and password (as
 * gathered from the Password dialog).
 */
```

```
public String buildPcSasProgram(String
sourceLib, String sourceDataset,
String targetHost, String targetDirectory,
String destinationFileFormat,
String targetFileName, String username,
String password,
String rowConstraints)
{
```

```
StringBuffer strBuf = new StringBuffer();
```

```
strBuf.append("%macro scon2Win(userId);
strBuf.append("=", pswrd=);\r\n");
strBuf.append("options comamid=tcp ");
strBuf.append("remote=mvs");
strBuf.append("set=tcptn3270 1;\r\n");
```

```
strBuf.append("signon "C:\\SAS\\");
strBuf.append("connect\\saslink\\");
strBuf.append("datcptso.scr\\");\r\n")
strBuf.append("%let userId=&userId; ");
strBuf.append("\r\n");
strBuf.append("%let pswrd=&pswrd;\r\n");
```

```
strBuf.append("rsubmit;\r\n");
```

```
strBuf.append("libname srcLib ");
strBuf.append(sourceLib);
strBuf.append(" DISP=SHR;\r\n\r\n");
```

```
strBuf.append("proc sql;\r\n");
strBuf.append("create table WORK.");
strBuf.append(sourceDataset);
strBuf.append(" as\r\n");
strBuf.append(" select * from srcLib.");
strBuf.append(sourceDataset);
strBuf.append("\r\n");
```

```
// Do the actual download.
strBuf.append("\r\n\r\n");
strBuf.append("proc download ");
strBuf.append("data=WORK.");
strBuf.append(aDataSetName);
strBuf.append("\r\n");
strBuf.append(" out=WORK.");
strBuf.append(aDataSetName);
strBuf.append(";\r\n run;\r\n");
(the rest of this method was not included)
```

Below is a simple example of what the output of the Java code above might look like follows (note: this is not showing the exact output of the method shown above). Also, to fit in the columns, the formatting had to be altered in a few places.

```
%macro scon2Win(userId=, pswrd=);
options comamid=tcp remote=mvs
set=tcptn3270 1;

signon "C:\SAS\connect\saslink\datcptso.scr";
%let userId=&userId;
%let pswrd=&pswrd;
libname targDir 'C:\temp\';
```

```

/* Beginning of LIB_ABC.sd2 transfer */

rsubmit;
libname srcLib 'LIB_ABC' DISP=SHR;

proc sql;
create table WORK.LIB_ABC as
  select *
  from srcLib. LIB_ABC;

proc download data=WORK.LIB_ABC
  out=targDir.LIB_ABC;
run;
/* validate the transferred data */
PROC upload data=targDir.LIB_ABC
out=WORK.xfer;
run;
proc compare base=WORK.LIB_ABC
  compare=WORK.xfer
  outstats=WORK.vallog;
run;
proc download data=WORK.vallog
  out=WORK.vallog;
run;
endrsubmit;
/* Now put the output of the Contents to the
Validation log file. */
proc contents data=targDir.LIB_ABC;
run;

PROC PRINTTO;
run;
RSUBMIT;
/* The following bit of code is used to
record this transfer on the OS/390 host
for reporting purposes. */
FILENAME SUB 'MYDIR.PDS.CNTL' DISP=OLD ;
DATA _NULL_ ; FILE SUB (DATMETR) ;
  PUT @1 "//MYJOB JOB (,8305,S,MYUSERID), "
  /
    @1 "//
'anytexthere',CLASS=A,PTY=10,MSGCLASS=T " /
    @1 "//STEP2 EXEC PGM=IEFBR14 " ;
RUN;
FILENAME SUB CLEAR;
X SUBMIT 'MYDIR.PDS.CNTL (DATMETR)' ;
RUN;
endrsubmit;

/* End of LIB_ABC.sd2 transfer */

signoff
"C:\SAS\connect\saslink\datcptso.scr";
%mend scon2Win;
%scon2Win (userId="MYID", pswrd="xxx");

```

The appendix contains an example of how Java can actually submit the generated SAS program to PC SAS.

A variety of SAS Procedures are used within the generated programs. For example, PROC SQL is used because it allows the tool to dynamically limit which rows and columns to include in the target dataset (per the user's specification). PROC DOWNLOAD is used to copy the dataset from a source to a target host, and PROC COMPARE is used when possible to assist with the verification that the transferred data is exactly the same as the source data. There are also cases where a SAS PROC was not available, such as for creating text files from the SAS datasets. Buchecker (1996) describes SAS macros that effectively automate the creation of flat files from SAS datasets.

SAS/Connect facilitates the communication with many different hardware platforms. This topic is dealt with in greater detail in our other paper titled, "Integrating SAS/Connect With Java".

When it is necessary to execute an OS/390 based program, the tool will generate a JCL job. It then uses the FTP protocol to connect to the OS/390 mainframe, send a command to alert the host that the next command should be auto-submitted to the JES2 job stream.

CONCLUSION

Some of the benefits of the DATAAccess Tool include:

- 15,000+ files transferred with no data corruption
- 35 teams have used the DATAAccess Tool
- Ease of use – minimal or no training required
- Improves the clinical analysis process by allowing the users more flexibility to choose the software tools that provide the analytical capabilities they need
- Provides an easy to use graphical user interface for transferring SAS datasets from OS/390 to the NT or Unix servers, or the local machine.
- Shields the user from the complexities of OS/390, JCL, SAS/Connect and FTP
- Provides data conversion to a format that is usable at the remote host (SAS, SAS Transport, Microsoft Excel 97, CSV text, Tab delimited text, space delimited text, and dBase)
- Displays which SAS data libraries exist on OS/390 for a given research project code
- Displays which SAS datasets are available within a selected SAS library
- Source data can be moved immediately, any time of the day, any day of the week
- Common tool used across the organization (and around the globe)
- Source data can be OS/390 SAS, DB2 views, and Oracle views
- Leverages capabilities of OS/390 and UNIX
- Allows user to specify subsetting criteria by columns and rows
- Provides a verification report that addresses the integrity of the transferred data and that can be submitted to a regulatory authority.
- Provides transfer progress monitoring (color-coded)
- Provides users instant access to SAS datasets on the platform they prefer and in the format they choose
- Adheres to all corporate security standards
- Provides metrics reports on the number of transfers, what data was transferred and where, who transferred the data, what formats were used, etc.
- Uses batch mode for multiple transfers

REFERENCES

Curnutt, R., Pell, M., LaBore, J., "Integrating SAS/Connect® with Java™", These Proceedings.

Buchecker, M. M. 1996. "%FLATFILE, and Make Your Life Easier," Proceedings of the Twenty-First Annual SAS Users Group International Conference, 21, 178-180.

SAS Institute Inc. (1996), *SAS Companion for the Microsoft Windows Environment*, Cary, NC: SAS Institute Inc.

SAS Institute, Inc. (1996), *SAS Companion for the MVS Environment, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1993), *SAS Companion for Unix Environments: Language, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1994), *SAS/Connect Software: Usage and Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990), *SAS Guide to Macro Processing, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1989), *SAS Guide to the SQL Procedure: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

TRADEMARK NOTICE

SAS is a registered trademark of the SAS Institute Inc, Cary, NC and other countries. Other brand and product names are registered trademarks or trademarks of their respective companies.

ABOUT THE AUTHORS

Randy Curnutt, Solutions Plus, Inc. (<http://www.sol-plus.com>)
Randy Curnutt is the president of Solutions Plus, Inc., a software consulting company that specializes in applying leading edge technologies in order to provide comprehensive solutions to its clients. He focuses on client/server solutions, especially object oriented technology, and relational database management systems. He has experience with Java, Smalltalk, Visual Basic, C, C++, Oracle, MS SQLServer, and numerous other development languages. Randy may be reached via email at

rcurnutt@sol-plus.com.

Michael Pell, Solutions Plus, Inc. (<http://www.sol-plus.com>)
Michael Pell is a consultant at Solutions Plus, Inc., a software consulting company that specializes in applying leading edge technologies in order to provide comprehensive solutions to its clients. He focuses on the analysis, design, and implementation of object oriented technology client/server solutions. Michael has 2-3 years of Java development experience, and spent 6 years as an IBM consultant prior to joining Solutions Plus, Inc. Michael may be reached via email at mjpell@sol-plus.com.

John LaBore, Eli Lilly And Company (<http://www.lilly.com>)
John LaBore is the SAS and JMP Coordinator for Eli Lilly and Company, a leading innovation-driven pharmaceutical corporation. He is responsible for supporting SAS and JMP use by Lilly staff worldwide. John has been a SAS software user for more than 20 years, and has authored numerous SAS technical papers for SUGI, PharmaSUG, SEUGI, and other SAS user group conferences.

CONTACT INFORMATION

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

Randy Curnutt
Solutions Plus, Inc.
10401 North Meridian, Suite 300
Indianapolis, IN 46217
(317) 848-3081
rcurnutt@sol-plus.com
<http://www.sol-plus.com>

Michael Pell
Solutions Plus, Inc.
10401 North Meridian, Suite 300
Indianapolis, IN 46217
(317) 848-3081
mjpell@sol-plus.com
<http://www.sol-plus.com>

John LaBore
Eli Lilly And Company
Lilly Corporate Center
Drop Code 6334
Indianapolis, IN 46285
(317) 277-6387
jml@lilly.com
<http://www.lilly.com>

APPENDIX

The generated PC-SAS program can be written to a temporary file, then PC-SAS can be run using the generated PC-SAS program (all done in the Java method as shown below).

```
/**
 * Writes the PC-SAS program to disk, then runs it.
 * @param sasProgramString String of the SAS program to be executed
 * @return int the return code returned by the executed SAS program
 */
public int runSasProgram(String sasProgramString) throws TransportFailedException
{
    int returnCode = 0;
    try
    {
        //Write the file to disk
        File sasProgramFile = new File("C:\\SAS\\GenProg.sas");
        BufferedReader buffReader = new BufferedReader(new StringReader(sasProgramString));
        PrintWriter sasProgOut = new PrintWriter(new FileOutputStream(sasProgramFile));
        printDISTreamOnPrintWriter(buffReader, sasProgOut);
        Runtime rt = Runtime.getRuntime();
        sasProgOut.close();

        //Now run the program using PC-SAS. This is equivalent to starting PC-SAS from a
        //DOS command line
        StringBuffer programSb = new StringBuffer("C:\\SAS");
        programSb.append("\\SAS.exe -NOSPLASH -ICON -SYSIN \\");
        programSb.append(sasProgramFile.getPath());
        programSb.append("\\");
        Process myProc = rt.exec(programSb.toString());
        returnCode = myProc.waitFor(); //wait here until SAS is done running
        SasProgramFile.delete(); //be a good neighbor and clean up
    }
    catch (Exception e)
    {
        throw new TransportFailedException("Error communicating with PC-SAS software.");
    }
    return returnCode;
}
```