Paper 266-2009

Retirement of Legacy Clinical Systems and moving to SAS Drug Development, Much More than Just Moving the Data

Fred R. Forst, Eli Lilly and Company
John Standefer, SAS Institute

Abstract

Retirement of legacy applications on z/OS and Windows and subsequent migration of data to SDD (SAS Drug Development) is accomplished with a 100% SAS solution, SAS/CONNECT. Data spanning 20+ years, created under three operating systems (z/OS, Windows, and Unix), from 4 versions of SAS (V5, V6, V8, V9) resulting in many permutations of data formats to be migrated to SDD. Traditional SAS data, text files, CGM graphs, SAS transport files, Microsoft Office files were just a few of the numerous data types migrated. This paper describes the innovative, flexible, and automated approach to moving and reformatting all these data types with one SAS application. At its core is SAS/CONNECT, using the WebDAV (Web-Based Distributed Authoring and Versioning) protocol available with the libname and filename statements. A robust SAS/CONNECT application resulted, moving data directly from any source to SDD, regardless of the file type, version of SAS it was created with, or originating OS. Finally, a PROC COMPARE is used to automatically verify the source and SDD data are identical for SAS data members.

Background

Life will be so much easier now after retiring the mainframe and moving the data to SAS Drug Development (SDD), the SAS hosted ASP model for clinical research systems. From now on, it will be SAS's responsibility to provide and maintain the validated environment that is 21 CFR Part 11 Regulatory compliant.

There is only one catch: moving over three terabytes of clinical data from 25 years of clinical studies to the new SDD environment hosted in the data center on the SAS Campus in Cary, NC. But it is not simply 'moving data' because of the wide variety of source file types and the need to convert most of these to a Unix based SAS environment. The HTTPS:// SSL WebDAV protocol was the chosen method of moving data into SDD as it is a secure and supported method.

It is a known fact and a painful standard to bear for IT departments of life science companies to provide and maintain validated systems that are 21 CFR Part 11 Regulatory compliant. SAS Drug Development (SDD), is the only validated SAS solution for storing and managing clinical, scientific or any kind of research data. SDD provides an integrated system for managing, analyzing, reporting and reviewing clinical research information. All research content is versioned and audit trailed, sustaining a high level of confidence regarding the accuracy and integrity of research content. http://www.sas.com/industry/pharma/develop/fact.pdf

Lilly has relied on SAS for 30 years to bring new medications to patients more quickly, to provide evidence of drug safety and efficacy to regulatory bodies, to provide answers about the use of its products, to increase understanding of disease states and to improve its manufacturing processes as well as sales and marketing efforts.

At Lilly, more than 300 employees work in the statistics group in 10 cities across four continents. They rely on SAS in drug discovery, toxicology studies, drug formulation development, clinical trials, sales and marketing, and manufacturing. SAS complements their individual talents and skills for delivering fast, innovative results by providing them a comprehensive suite of tools to meet their needs. http://www.sas.com/success/elililly.html

Introduction

Over a time span of 25 years, a major SAS shop can accumulate a large volume of data as well as many different data types and formats. One thing that has probably always been true, due to human nature, when data is accumulated over time there is not much thought put into the possibility that one day you may have to move it to a different environment. As such, we were faced with moving over five terabytes consisting of several data types to SDD. The following list details all the different file types:

- 32-bit z/OS based data:
  - V5,V6,V8,V9 SAS data members
  - V5,V6,V8,V9 SAS catalogs

    - PDS text
    - PDS CGM graphics
    - PDS SAS V5 formats
    - 'Flat' text files
    - 'Flat' CGM files
    - Transport files created with PROC CPORT
    - Transport files created with the xport engine (e.g. via PROC COPY)
- Windows based data:
    - V6,V8,V9 SAS data members
    - V6,V8,V9 SAS catalogs
    - Text files (e.g. .sas, .html files)
    - SAS Transport files
    - Microsoft Office files (xls, ppt, doc etc)
    - Pictures (.jpeg, .gif etc)
- 32-bit Unix based data:
    - V6 and V8 SAS data members
    - V6 and V8 SAS catalogs

In case you're counting, that is 35 different permutations of data platforms/data formats. The challenge was to move the data to SDD (64-bit Solaris environment) with the following requirements:

1. A secure, encrypted protocol must be used.
2. Data should be moved only once. Avoid 'staging areas'.
3. SAS data should be verified that the source and destination files are identical.

Since data was being migrated to a different data representation (from 32-bit z/OS, 32-bit Solaris, or 32-bit Windows to 64-bit Solaris) and different encodings (from wlatin1 or EBCDIC to latin1), it was *clearly* not a case of 'just moving data'. Furthermore, creation of the source SAS data and SAS catalogs spanned 4 versions of SAS. A solid understanding of data encoding/transcoding, compatibility issues between different versions of SAS (especially cross-platform), SAS/CONNECT, and WebDAV capabilities and limitations were crucial.

In addition to legacy data being migrated to SDD, our migration application also needed the flexibility to migrate data to an archive storage system (EMC Centera®) within the company.

SAS/CONNECT offers the ability to move all types of data, not just SAS data, using PROC DOWNLOAD. Even PROC COPY in a single SAS session can only move/copy SAS data. The flexibility of PROC DOWNLOAD and the 'move-once' advantage is the reason SAS/CONNECT was chosen as the basis for the data migration project.

Where to Start?

The first step in the migration project was to create an inventory of all data to be migrated, the destination path/folder for each dataset, and the destination (SDD or local LAN). This information was kept in an Excel spreadsheet and a 'migration team', knowledgeable about the data, maintained it. This spreadsheet became the input source, metadata, of our migration application. Therefore, the spreadsheet had all the metadata needed:

Source	Dest	Source DSN	Dest DSN
MVS	SDD	prod.clinical.drug1.trial1	https://sdd.sas.com/webdav/clinical/drug1/trial1
MVS	SDD	prod.clinical.drug1.trial2	https://sdd.sas.com/webdav/clinical/drug1/trial2
MVS	LAN	prod.clinical.drug2.trial3	\\lilly.com\prod\clinical\drug2\trial3
MVS	LAN	prod.clinical.saspgms	\\lilly.com\prod\clinical\saspgms
Win	SDD	\\lilly.com\group2\documents	https://sdd.sas.com/webdav/group2/documents

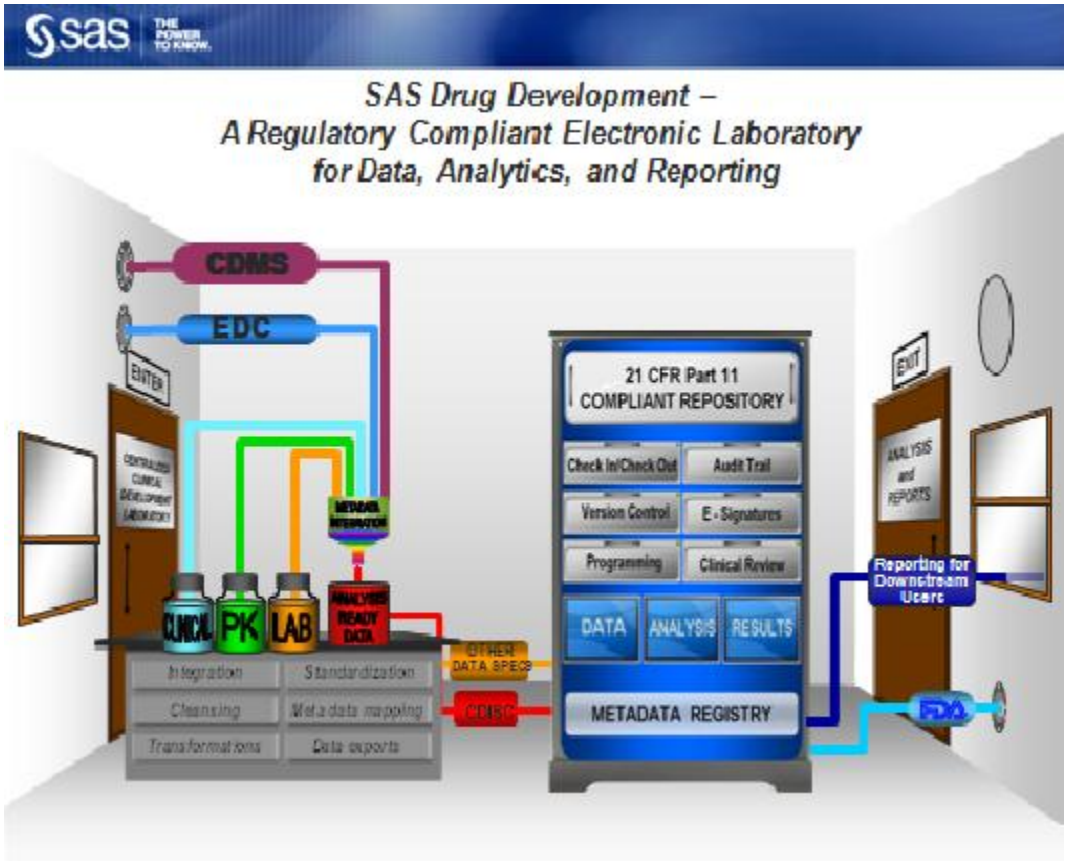


Figure 1 - High Level Schematic of SDD

SDD Configuration Overview

SDD is designed to meet the advanced information management requirements of life science research and to this end provides an integrated system for managing analyzing, reporting and reviewing clinical research information.

SDD provides access to content through three main interfaces:

1. The WebDAV client interface provides access to SDD by using the webdav `libname/filename` engine. This is the focus of our paper (see Figure2 below – SAS/Connect with WebDAV).
2. The embedded interface, designed to follow the Windows Explorer metaphor, provides access through a web browser using Web Folders.
3. The published APIs support linking SDD to other applications and systems. External applications can be built that provide customized management of SDD content.

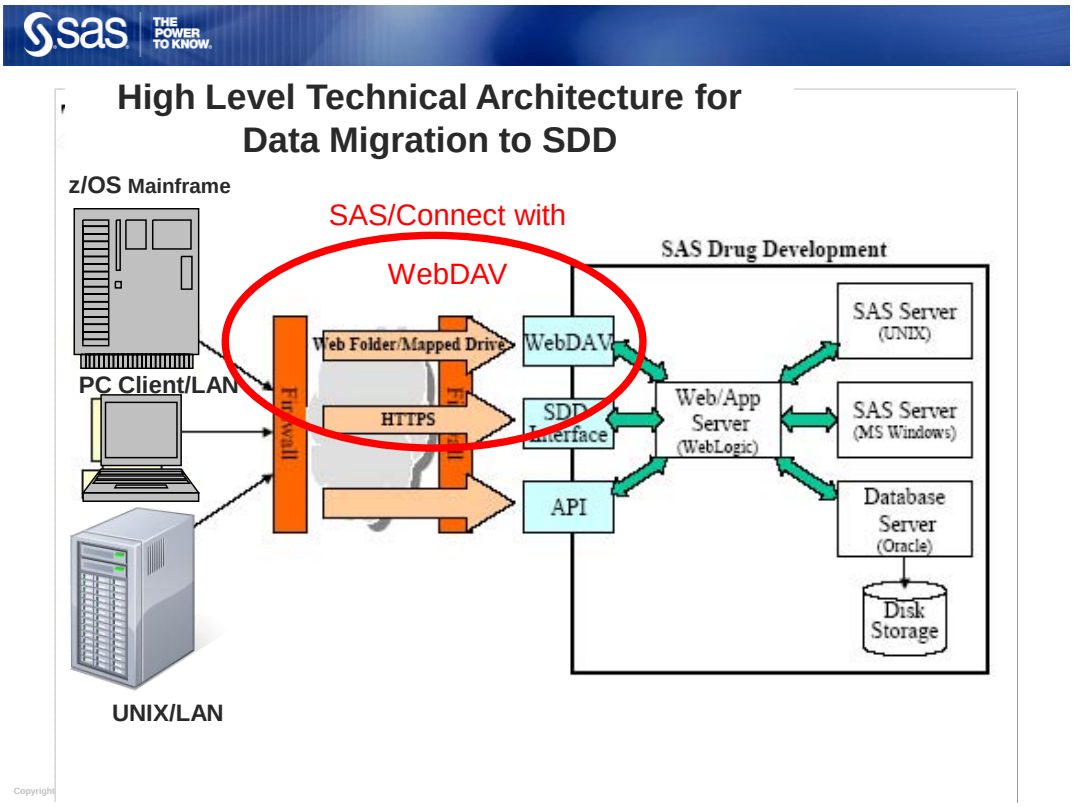


Figure 2

Generated documents and information are stored in Oracle using Xythos WebDAV to manage the content. SAS analytics are used on the data managed by the system.
The SDD architecture is complex. For the purpose of time we will focus our discussion on getting data *into* SDD from external sources.

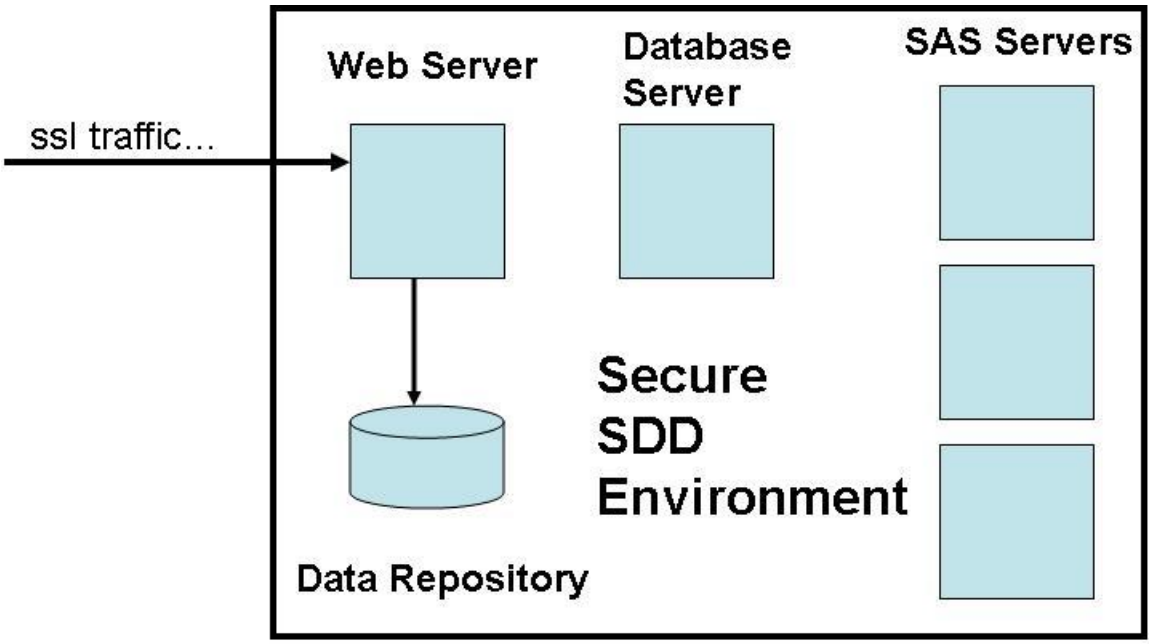
The only external pathway into SDD is via an https connection using the WebDAV protocol. This isn't limited to SAS' `libname/filename` statements. Other applications that can use

WebDAV, can access data in SDD. For example, a Windows Network Place could be established, defining SDD as a Network Place using WebDAV, and Windows Explorer ala web folders could be used to access data in SDD, just like you would use Windows Explorer to access any other folder.

Figure 3 below is a simplified diagram of an SDD configuration. Major components are

1. SAS server(s)
2. Web server(s)
3. Database/Oracle server(s)
4. A data repository

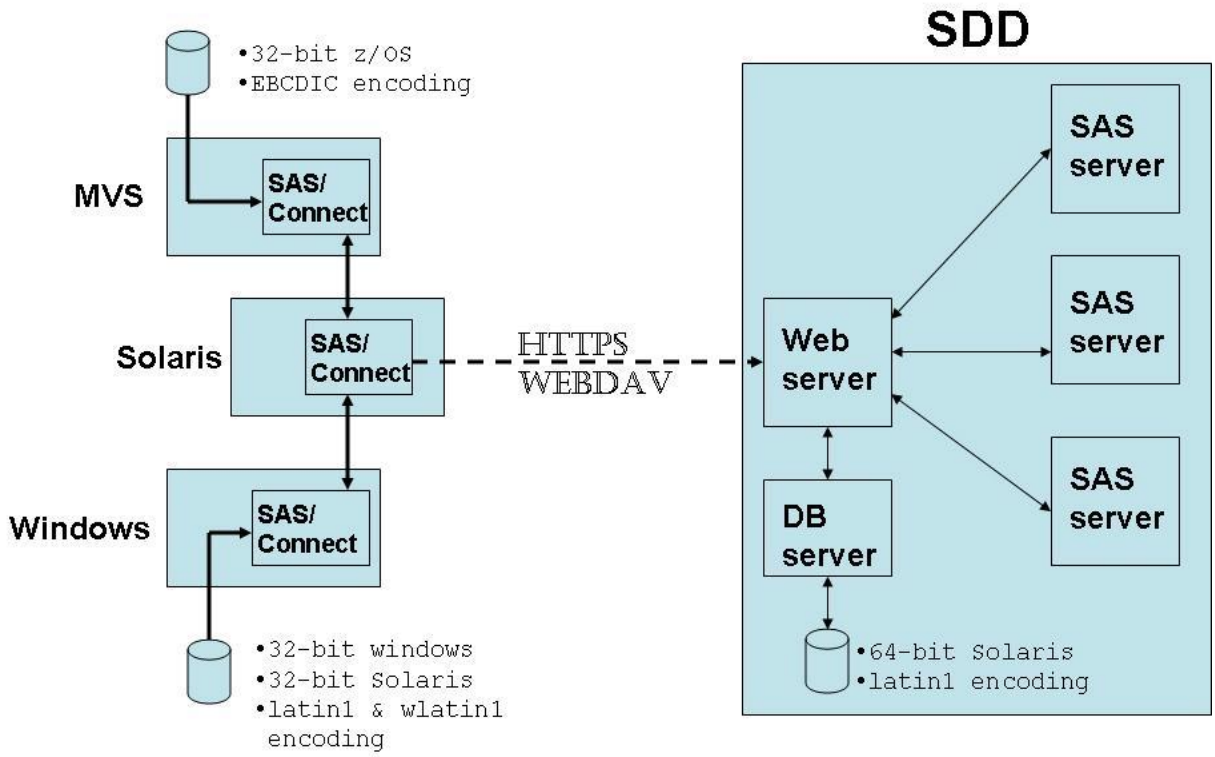
Figure 3



Migration Architecture

Figure 4 below shows the overall architecture of the data migration from our Legacy Systems (left hand side) to SDD on the right.

Figure 4



In the above diagram, the left hand side represents the source systems and data and the right hand side denotes SDD. A solaris SAS session spawns either a SAS/CONNECT session on MVS or a Windows SAS/CONNECT session, depending on the location of the source data. The Solaris SAS session is the session that is actually writing to SDD, via WebDAV, for a couple of reasons:

- MVS couldn't write directly to SDD since WebDAV isn't supported in SAS V9.1.3 on MVS.
- The resulting data being written to SDD will be in 64-bit Solaris data representation and latin1 encoding, which matches the SAS servers in SDD.

SAS/CONNECT handles all the transcoding issues when moving data cross-platform.

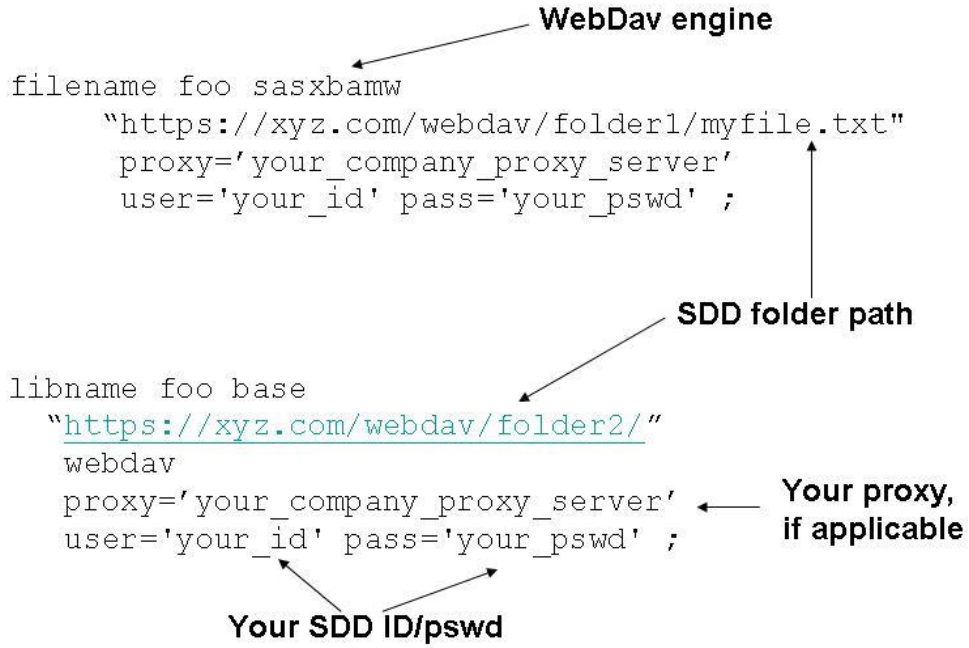
SAS `libname/filename` Support of WebDAV

With SAS V9, the webdav protocol is now supported with the `libname` and `filename` statements. In order to enable ssl, one must first do a few things:

- Be at SAS V9 SP4
- Install a SAS/SECURE module from SAS' website
- Install an ssl certificate. In Windows, this is done thru IE, and on Unix the ssl location is specified in a Unix SAS options statement.

Figure 5 below shows examples of the `libname` and `filename` statement syntax.

Figure 5



Note that the folder name syntax includes what looks like a sub-folder called `/webdav/`. This is not a folder name, but rather follows the same syntax as if you were creating a network place for the SDD site. In that case, you would create the network place

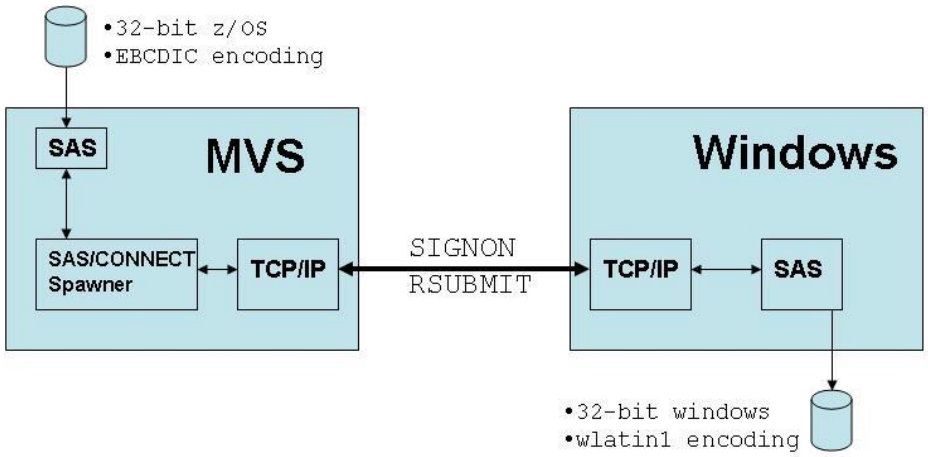
https://xyz.com/webdav

and, after that, you could access SDD the same way you access other folders, shares, or network places with Windows explorer.

PROC DOWNLOAD 101: Non-WebDAV

A typical usage of SAS/CONNECT might be to download SAS data from a mainframe to Windows using PC/SAS. Figure 6 below shows all the major players:

Figure 6



A PC/SAS session issues a `SIGNON` command, which uses the TCP/IP access method on both Windows (the local machine) and the mainframe (the remote machine). The SAS/CONNECT spawner on MVS is listening on a specific TCP/IP port for incoming requests. The spawner then spawns a SAS session on MVS and the connection is established. Subsequent SAS code is RSUBMITed from PC/SAS to MVS. For example, our simple download example code may look this way:

```
libname pc 'c:\sasdata\';
signon mvs.__port uid=my_mvs_id pw=my_pswd;
rsubmit;
  libname mvs '….mvs…SAS…lib…';
  proc download inlib=mvs outlib=pc;
  run;
endrsubmit;
signoff;
```

Everything between the `rsubmit` and `endrsubmit` block runs on the remote machine, MVS in this example. Since this example uses the `inlib=` and `outlib=` parameters, it is DOWNLOADing strictly SAS data. The `infile=` and `outfile=` parameters can be used for non-SAS data. Below is a pseudo example of DOWNLOADing files between 2 Windows machines. Note the use of wildcard specifications and the use of `filename` statements instead of `libname` statements:

```
filename pc 'C:\' ;
signon win.__5000 uid=_prompt_ pw=_prompt_;
rsubmit ;
  filename in '...LAN folder...';
  proc download infile=in('*.txt') outfile=pc;
  proc download infile=in('*.xls') outfile=pc binary;
  proc download infile=in('*.ppt') outfile=pc binary;
  proc download infile=in('*.doc') outfile=pc binary;
  run;
endrsubmit;
signoff;
```

The binary option is used to move an *exact* copy of the data. That is, no transcoding. The previous two examples brings out another key selling point for the usage of SAS/CONNECT via PROC DOWNLOAD to move all of our data: only 3 permutations of the PROC DOWNLOAD statement is needed to move *any* type of data:

```
proc download inlib=…  outlib=…;
```

This will move all SAS data (data members, catalogs, etc) and transcode to the proper (i.e. target) encoding and create data compatible with the target operating system.

```
proc download infile=…  outfile=…;
```

This will move non-SAS data (text files, .sas files etc) and transcode to the proper (i.e. target) encoding.

```
proc download infile=…  outfile=… binary;
```

This will move non-SAS data (MS Office files, pictures etc) in binary format and not transcode.

Data Encoding

Data encoding refers to the internal (i.e. binary) value used to store character data. The major categories are EBCDIC, ASCII, UTF8, and DBCS. There are many different encoding schemes, or code pages. For example, the value ‘Hello’ is stored as follows:

On MVS (EBCDIC-1047 encoding)

```
     H  e  l  l  o
hex=C8 85 93 93 96
binary=11001000 10000101 10010011 10010011 10010110
```

On Windows (wlatin1 encoding)

```
     H  e  l  l  o
hex=48 65 6C 6C 6F
binary=01001000 01100101 01101100 01101100 01101111
```

PROC DOWNLOAD automatically changes the encoding (i.e. transcodes) from the source encoding to the destination encoding.

PROC DOWNLOAD 201 – The Complexity Increases with WebDAV

As previously stated, there are only three variations of PROC DOWNLOAD syntax to move three types of data:

1. SAS data (data members and catalogs)
2. Text files
3. Files to be moved in binary (e.g. MS docs, jpeg files)

The destination, SDD, is a 64-bit unix based SAS V9 environment. For this reason, we initiate the download from a local 64-bit unix SAS V9 session. When DOWNLOADing the first data type (SAS data), the only difference in the code, compared with the MVS-to-Windows example above, is the syntax of the SDD/webdav libname statement. Note the addition of the PROC COMPARE as the last step:

```
libname sdd base
        "https://xyz.com/webdav/folderxyz"
        webdav USER='user_id'
        PW='your_pswd';
signon mvs.__port uid=my_mvs_id pw=my_pswd;
rsubmit;
  libname mvs '….mvs…SAS…lib…';
  proc download inlib=mvs outlib=sdd;
  run;
endrsubmit;

libname mvs server=mvs.__port;
proc compare data=mvs.abc compare=sdd.abc;
proc compare data=mvs.xyz compare=sdd.xyz;
signoff;
```

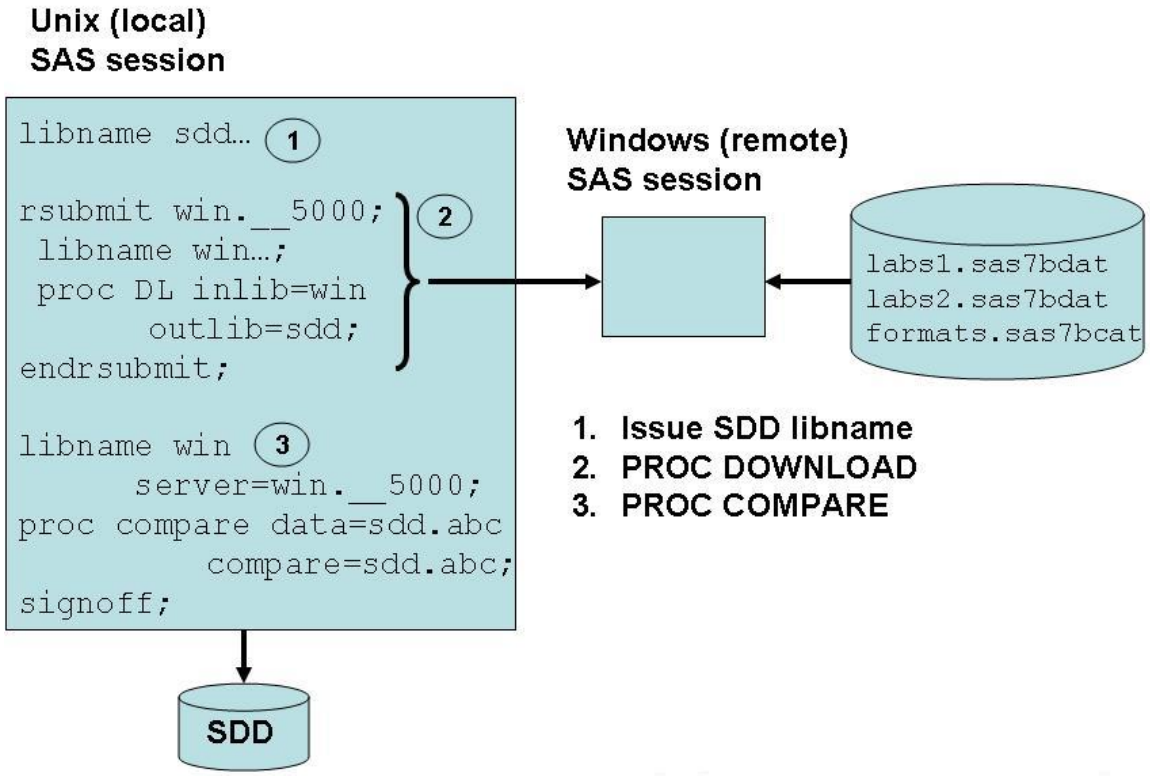
1 SDD libname statement

2 DOWNLOAD from remote to SDD

3 PROC COMPARE

Figure 7 below shows this process graphically.

Figure 7



Things get more complicated with the other two data types due to the fact there is no directory support or wildcard syntax when DOWNLOADing to a SDD folder. For example, if DOWNLOADing from a Windows folder to a SDD folder, the following code will fail:

```
filename sdd sasxbamw "https://xyz.com/webdav/folderabc"
         USER='your_id' Pass='your_pswd';
signon win.__5000 uid=_prompt_ pw=_prompt_;
rsubmit ;
  filename in '...LAN folder...';
  proc download infile=in('*.txt') outfile=sdd;
  proc download infile=in('*.xls') outfile=sdd binary;
  proc download infile=in('*.ppt') outfile=sdd binary;
  proc download infile=in('*.doc') outfile=sdd binary;
  run;
endrsubmit;
signoff;

ERROR: Bad outfile specification, outfile=SDD(total.xpt).
ERROR: Requested function is not supported.
ERROR: Download function terminated. The file created may be unusable.
```

The solution is to generate an individual filename for *each* file being moved. For example, if the input folder contained 4 files,

- my_stuff.txt
- costs.xls
- sugi28.ppt
- mypaper.doc

we would generate four `filename` statements. The code is shown below:

```
filename sdd1 sasxbamw "https://xyz.com/webdav/folderabc/my_stuff.txt"
         USER='your_id' Pass='your_pswd';
filename sdd2 sasxbamw "https://xyz.com/webdav/folderabc/costs.xls"
         USER='your_id' Pass='your_pswd';
filename sdd3 sasxbamw "https://xyz.com/webdav/folderabc/sugi28.ppt"
         USER='your_id' Pass='your_pswd';
filename sdd4 sasxbamw "https://xyz.com/webdav/folderabc/mypaper.doc"
         USER='your_id' Pass='your_pswd';

signon win.__5000 uid=abc pw=xyz;
rsubmit ;
  filename in '\\lan\folder1';
  proc download infile=in(my_stuff.txt) outfile=sdd1;
  proc download infile=in(costs.xls)    outfile=sdd2 binary;
  proc download infile=in(sugi28.ppt)   outfile=sdd3 binary;
  proc download infile=in(mypaper.doc)  outfile=sdd4 binary;
  run;
endrsubmit;
signoff;
```

The difficulty here is the fact that there could be thousands of files in a folder to move (and there were), making hard coding of the individual filename statements impossible. They must be dynamically generated with macro code as well as all the PROC DOWNLOAD statements. Another problem, since the above code is initiated in a unix SAS session, the names of the individual files to be moved are not initially known in the unix session. As such, code was needed to perform three tasks:

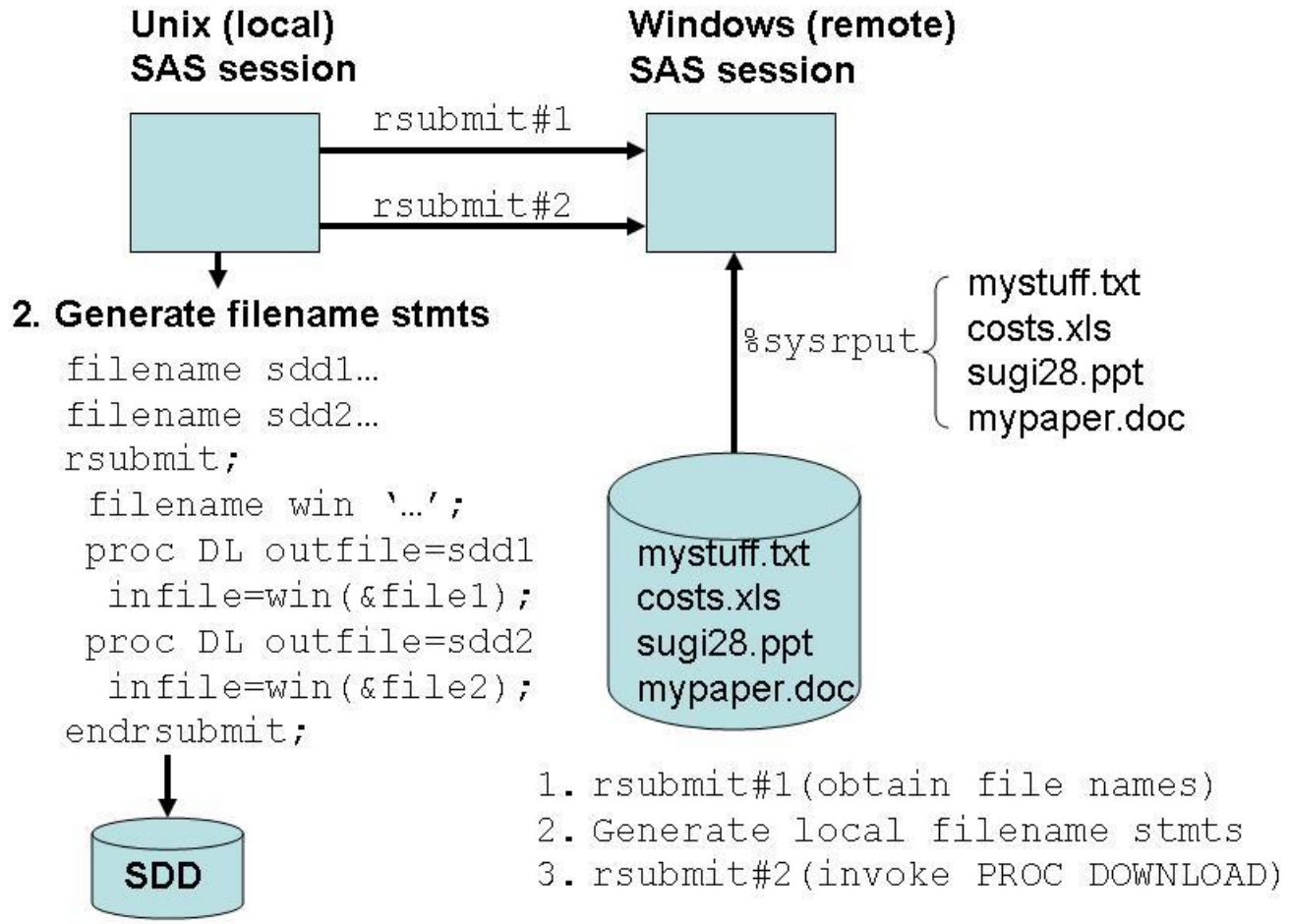
1. RSUBMIT to the remote session (MVS or Windows) and obtain file names and file count of the folder being moved.
2. Send, via `%sysrput,` these values back to the local session (unix) and issue a filename statement for each file.
3. RSUBMIT again to the remote session and generate the PROC DOWNLOAD statements to copy the data to SDD.

A dos `dir` command was piped in the Windows session to find all the file names and the number of files from folder `x`. These values were put into macro variables in the Windows session. Then, still in the Windows SAS session, the macro function `%sysrput` was used to 'send' or remote put the macro variables back to the Unix SAS session:

```
%do i=1 %to &files;
      %sysrput file&i=&&file&i;
%end;
  %sysrput files=&files;
```

At this point, the file count and file names are available in the Unix session and the `filename` statements are dynamically generated and executed. Figure 8 below shows this graphically:

Figure 8



The pseudo code would look like this:

```
signon win.__5000 uid=abc pw=xyz
rsubmit ;
 /* Step 1. Count files and obtain names in \\lan\folder1\ */
      %find_files(\\lan\folder1\);
 /* Step 2. Use sysrput to send file names back to Unix session */
      %sysrput…;
      %sysrput files=&files;
endrsubmit;

 /* Step 3. Do-loop to generate all filename stmnts in Unix session */
      %do i=1 %to &files;
        filename sdd&i sasxbamw
           "https://xyz.com/webdav/folderabc/&&file&i"
           USER='your_id' Pass='your_pswd';
      %end;
/* Step 4. RSUBMIT(again) to remote system to download all files*/

rsubmit ;
    filename in '\\lan\folder1';
  /* Do-loop to generate all PROC DOWNLOAD statements  */
      %do i=1 %to &files;
        PROC DOWNLOAD infile=in(&&file&i) outfile=sdd
            %if &type=binary %then %do;
                  binary;
            %end;
        run;
      %end;
endrsubmit;
signoff;
```

The pseudo code above would generate the same code as the hard-coded example.

Business Requirements

To make things even more complex, business requirements were:

- SDD file names must not contain blanks and be lowercase. For example, if the source file name was My Resume.doc, the SDD file name would be my_resume.doc.
- A PROC COMPARE was needed to verify that the source & destination SAS data are identical.
- A ‘history log’ must be dynamically updated as data is being moved. This contains source & destination folder names of all folders moved, along with return codes of tasks (DOWNLOAD & COMPARE RCs).

Diverse Data Sources

As mentioned, the source data existed on either MVS or Windows. Inherent differences in the data include

- There are no file extensions with MVS data. The type of data to be moved (SAS data, text, binary) must be derived from either dataset naming conventions, if they exist, or by examining contents of the data itself. For example, non-SAS data had to be opened to determine if it was a SAS transport file.
- MVS contains PDS' (Partitioned DataSet) and a PROC SOURCE must be executed to obtain the member names.
- Graphics, if stored in an MVS PDS or MVS 'flat' file, must be converted to a SAS catalog prior to moving to SDD.
- Version 5 SAS catalogs are not readable in a V9 session. They must be converted to a newer version of SAS using PROC V5TOV6 in a SAS V6 session.
- A windows folder can contain any type of data (SAS data, .txt, .doc, .sas files, etc). The same logic cannot be applied to MVS. A SAS lib on MVS contains only SAS data/catalogs.

The migration code had to dynamically handle these OS based differences.

Conclusions

Much more than 'just moving data', the final product performed many tasks, with the only input being the source and destination folder names. In general terms, the sequence of events was (for each folder being moved):

1. In a Unix SAS session (the local SAS session):
    a. Spawn either a MVS SAS session or a Windows SAS session, depending on the source folder location.
    b. Find the file names and file count in the input folder.
    c. Generate destination file names that are lowercase & contain no blanks.
    d. Dynamically generate SDD filename/libname statements.
2. In the MVS or Windows SAS session (the remote SAS session):
    a. Dynamically determine the type of data (SAS data, text data, binary data) and generate the appropriate PROC DOWNLOAD statements.
    b. Capture all return codes and pass back to the Unix session for a report.
    c. Update the history log of the data just moved.
3. In the Unix SAS session:
    a. PROC COMPARE the SAS data members, if any, just moved.
    b. Create summary report of each folder moved and all RCs.

Contact Information:

- Fred Forst, Eli Lilly and Company – frf@lilly.com
- John Standefer, SAS Institute –JohnStandefer@sas.com