Paper: 241-2007

Process Your SAS® Datasets Anyway, Anyhow, Anywhere You Choose With SAS/Connect

Michael A. Raithel, Westat, Rockville, MD

Abstract

One of the challenges of an organization that has multiple computing platforms is accessing SAS data sets stored on other operating systems. For example, maybe you do most of your work on SAS for Windows, but have to process SAS data sets stored on a corporate Linux server. Or, perhaps you normally work with SAS on a UNIX server, but need to access SAS data sets stored on a Z/os mainframe server. In such cases, the old method of processing data stored on a different server was to create a SAS transport file on the remote SAS server, FTP it to your local server, then create a SAS data set from the transport file. This method resulted in a local copy of the SAS data set that was converted to the correct machine format of the local SAS server. SAS/Connect software eliminates this process and allows you to either submit SAS programs to execute on the remote server, or to process SAS data sets located on the remote server from your local SAS session.

This paper discusses the major features of SAS/Connect software that facilitate the processing of SAS data sets located on foreign operating systems. It outlines three of the most salient features of SAS/Connect software: Compute Services, Data Transfer Services, and Remote Library Services. These features allow SAS data sets to be transferred to and from another operating system, and to be read from a SAS program running on another operating system, respectively. Then, this paper provides multiple examples of how you can use SAS/Connect to process remote SAS data sets locally, or submit your SAS programs to run on a remote computer.

After reading this paper, you will have practical knowledge that allows you to process your SAS data sets anyway, anyhow, anywhere you choose using SAS/Connect software.

A Brief Overview of SAS/Connect Software

SAS/Connect software provides three distinct areas of functionality that are known as ***Services***. First, it provides ***Compute Services***, which allow you to execute SAS programs on a remote server. Second, SAS/Connect provides **Data Transfer Services**, which you can use to transfer SAS files between servers. Finally, SAS/Connect features ***Remote Library Services***, which give you the ability to access data stored on remote servers as if it was stored locally. For these services to be available, you must have SAS/Connect installed on both of the computers that are to communicate.

(In this paper, we will assume that you are going to communicate between SAS running on a Windows workstation and SAS running on a remote server with a different operating system. However, the same basic principles apply to using other computers as the client).

Here is a summary of what the three aforementioned SAS/Connect services provide:

Compute Services

Compute Services are designed so that you can submit your SAS programs to run on a remote server from your own desktop. With Compute Services, you can submit a SAS program to execute on the remote server in batch mode. When the batch program completes, the SAS Log and List files are written back to the PC directory where the batch program resides. You can also submit a SAS program to execute on the server from a PC SAS session that you have up and running. When the program completes, the SAS Log and List files are written back to the Log and Output windows of your active PC SAS session.

There are two types of Compute Services:

1. **Compute Services with RSUBMIT**. This Compute service remote submits SAS program code that is bracketed between the RSUBMIT and the ENDRSUBMIT commands. The RSUBMIT command directs SAS/Connect to submit the following SAS program statements to a remote server. SAS/Connect remote submits every line of SAS code that it finds, until it encounters the ENDRSUBMIT command. Any valid SAS programming statement may reside between the commands and be submitted to the remote server. RSUBMIT/ENDRSUBMIT can be used in either batch or interactive mode.

2. **Compute Services with Remote SQL Pass-Through (RSPT).** This Compute service allows you to pass SAS SQL statements to the remote server for processing. The SQL can process SAS tables (data sets) that reside on the remote server, or they can access a DBMS via a SAS/ACCESS engine such as SAS/ACCESS to ORACLE. RSPT can be used in either batch or interactive mode.

Data Transfer Services

This service provides two SAS procedures for moving data between the desktop and the remote server. PROC UPLOAD is used to transfer SAS files from a network or desktop directory to a directory on a remote server. PROC DOWNLOAD is used to transfer SAS

files from a directory on a remote server to a desktop directory or network directory. When transferring data sets, both PROCs convert the data to the receiving server's correct machine representation, so that the data is immediately usable by the SAS System.

Remote Library Services
This service furnishes you with the ability to allocate a SAS data set on a remote server via a LIBNAME statement in your local PC SAS program. By using Remote Library Services, you can create, delete, edit, update, read, or process SAS observations in SAS data sets on the remote server as if the data were locally available. You can use SAS Explorer in the SAS Display Manager to browse, edit, delete, etc. SAS data sets and other SAS files located on the remote server.

Establishing a Connection to a Remote Server

Before you can employ any SAS/Connect service, you must connect to the remote server that you intend to use. The connection is initiated by specifying the *comamid* and *remote* SAS System options, and executing the *signon* command from within a SAS program. The SAS program can be run interactively in PC SAS or run in batch mode. In either case, the SAS System will request your remote server logon id and password. When you have entered both correctly, your connection to the remote server will be established.

The *comamid* SAS System option identifies the communication access method that will be used to communicate between the remote host and your PC. The value "TCP" stands for the TCP/IP protocol and should always be specified. The *remote*[1] SAS System option specifies the remote server that you intend to connect to. This should be set to the server id of another remote server that has SAS/Connect installed, such as SASB, UNIX1, etc.

The *signon* command directs the SAS System to sign-on to a remote server using a specified logon script. The logon script actually does the systems work of establishing the connection between your PC and the remote server. SAS/Connect comes with a dozen logon scripts that can be found in the "***!sasroot\connect\saslink***" directory. Some of the logon scripts are:

tcpunix.scr	-	To connect to UNIX and Linux servers
tcpwin.scr	-	To connect to Windows servers
tcptso.scr	-	To connect to z/OS servers

You only have to invoke the *comamid* and *remote* options, and execute the *signon* command once when you are executing SAS programs from an interactive PC SAS session. When you execute the *signon* command, it establishes your link to the remote server and keeps that link open until you either terminate the SAS session or you terminate the link via the *signoff* command.

Here is an example of the SAS statements needed to establish a connection to the SASB server:

```
/*****************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASB */
/*****************************************************************/
options comamid=TCP remote=SASB;

/*************************************************/
/* Signon to SASB using a Linux script           */
/* User is prompted for Linux login and password */
/*************************************************/
signon "C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpunix.scr";
```

In the example, above, the comamid option specifies that TCP/IP is to be used as the communication access method. The *remote* SAS System option specifies that a connection to the SASB server is desired. The *signon* command is followed by the specific path of the logon script for TCP/IP between a PC server and a Linux server. The options and signon command, specified above, should always be used when establishing a connection to the SASB server.

Here is an example of the SAS statements needed to establish a connection to the SASA Linux server:

```
/*****************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASA */
/*****************************************************************/
options comamid=TCP remote=SASA;

*************************************************/
/* Signon to SASA using a Linux script           */
/* User is prompted for Linux login and password */
/*************************************************/
signon "C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpunix.scr";
```

[1] This option is formally known as *connectremote*. However, it is more commonly known by its more popular alias, *remote*. Two other aliases for this option are: *cremote* and *process*. This paper uses the more common term: *remote*.

The example above mirrors the previous example except that the remote option is set to **SASA**. This change is *absolutely necessary* to connect to the SASA Linux server.

Examples of Exploiting SAS/Connect Software

This section presents examples of SAS programs that use SAS/Connect to process local data on remote servers. You can use these examples to better understand the syntax and the functionality of SAS/Connect software. If you are viewing this document in electronic form, you can cut and paste the examples into your active SAS session and tailor them to your own needs. All of the examples assume that you are connecting from SAS on a Windows workstation to SAS on a Linux server via SAS/Connect software.

1. Uploading a SAS Data Set to a Linux Server and Processing it There

This is an example of using both ***Data Transfer Services*** and ***Compute Services***. In the example PROC UPLOAD is used to copy a SAS data set from a PC directory to a directory on the SASB Linux server. Accompanying SAS program statements are remote submitted to process the data set. The program summarizes the data on the Linux server and then uses PROC DOWNLOAD to store the resulting SAS data set in the PC directory. Before the data are downloaded, the DATASETS procedure is used to delete the transported SAS data set from the Linux directory since it is no longer needed. The downloaded SAS data set is printed from the local PC SAS session.

```
/*****************************************************************/
/* Example 1. Upload data set to Linux and process it there.     */
/*****************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASB */
/*****************************************************************/
options comamid=TCP remote=SASB;

/*********************************************/
/* Signon to SASB using a Linux script       */
/* User is prompted for Linux login and password*/
/*********************************************/
signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpunix.scr';

/*****************************************************/
/*  Allocate a PC directory that contains SAS data sets*/
/*****************************************************/
libname PCLIB 'C:\WINDOWS\Temp';

/*****************************************************/
/* RSUBMIT this block of code to run on the SASB server*/
/*****************************************************/
RSUBMIT;

  /*****************************************************/
  /* Allocate a Linux  directory to contain SAS data sets*/
  /*****************************************************/
  libname LINUXLIB "/home/marsyst/wessug";

  /*****************************************************/
  /* Move SAS data set from desktop to Linux for processing*/
  /*****************************************************/
  proc upload data=PCLIB.prdsale out=LINUXLIB.prdsale;
  run;

  /*****************************************************/
  /* Summarize data on the Linux Server                */
  /*****************************************************/
  proc summary nway data=LINUXLIB.prdsale;
       class country division prodtype;
       var actual predict;
  output out=summ1(drop=_type_ rename=(_freq_=Months)) sum=;
  run;

  /*************************************************/
  /* Delete the data set from the Linux Server     */
  /*************************************************/
  proc datasets library=LINUXLIB;
       delete prdsale;
```

```
  run;

  /*****************************************************/
  /* Move summarized SAS data set from Linux to desktop */
  /*****************************************************/
  proc download data=summ1 out=PCLIB.summ1;
  run;

/*************************************************************/
/* End of block of SAS code that runs on the SASB server*/
/*************************************************************/
ENDRSUBMIT;
signoff;

  /*******************************/
  /* Print out the results.      */
  /*******************************/
ods listing close;
ods html file="Linuxrpt.html";

proc print data=PCLIB.summ1 noobs label;
      by country;
      id country;
      sum Months actual predict;
title1 'Actual and Predicted Sales Data';
title2 'Summarized by Country, Division and Product Type';
run;

ods html close;
ods listing;
```

For this first example, we will look at the SAS Log in detail:

```
1    /*********************************************************************/
2    /* Example 1. Upload data set to Linux and process it there.       */
3    /*********************************************************************/
4    /* Inform SAS/Connect that we are using TCP/IP to connect to SASB */
5    /*********************************************************************/
6    options comamid=TCP remote=SASB;
```

In the first part of the log, below, the first three NOTE:'s, highlighted in red by the author, let you know that SAS is logging onto the remote server. You will *always* see these messages in your log when you have successfully logged onto a remote server.

The next several NOTE:'s, highlighted in blue by the author, report that a SAS session was successfully initiated on the remote server. The two most important notes are the first, stating that a SAS/Connect conversation was established, and the last, stating that the remote signon to the server is complete. Once you see these notes, you can be assured that you have successfully logged onto your remote server and have SAS at your disposal running on it.

```
10   /*********************************************/
11   /* Signon to SASB using a Linux script         */
12   /* User is prompted for Linux login and password*/
13   /*********************************************/
14   signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpLinux.scr';
NOTE: Remote signon to SASB commencing (SAS Release 9.01.01M3P020206).
NOTE: Script file 'tcpLinux.scr' entered.
NOTE: Logged onto Linux... Starting remote SAS now.
NOTE: SAS/Connect conversation established.
NOTE: Copyright (c) 2002-2003 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) 9.1 (TS1M3)
      Licensed to WESTAT, Site 0000427l955
NOTE: This session is executing on the Linux 2.4.21-47.0.1.EL platform.

NOTE: SAS 9.1.3 Service Pack 4
```

```
NOTE: SAS initialization used:
      real time           0.18 seconds
      cpu time            0.02 seconds

NOTE: Remote signon to SASB complete.
```

Even though we have a SAS/Connect conversation established between servers, options statements, DATA steps and PROC steps run in the program will continue to be executed on the Windows workstation. So, the LIBNAME statement allocates a SAS data library on the Windows workstation.

```
16
17   /*******************************************************/
18   /*  Allocate a PC directory that contains SAS data sets*/
19   /*******************************************************/
20   libname PCLIB 'C:\WINDOWS\Temp';
NOTE: Libref PCLIB was successfully assigned as follows:
      Engine:        V9
      Physical Name: C:\WINDOWS\Temp
```

The next section remote submits a block of SAS code to be run on the Linux server. It is initiated with an RSUBMIT statement and terminated with an ENDRSUBMIT statement. You will see a note in the log acknowledging that the remote submit is commencing.

```
21
22
23   /*******************************************************/
24   /* RSUBMIT this block of code to run on the SASB server*/
25   /*******************************************************/
26   RSUBMIT;
NOTE: Remote submit to SASB commencing.
```

Now that statements are being sent to the Linux server, you can use a LIBNAME to allocate a SAS data library there. Below, the LIBREF *LINUXLIB* is assigned to the /home/marsyst/wessug Linux directory.

```
1      /*******************************************************/
2      /* Allocate a Linux  directory to contain SAS data sets*/
3      /*******************************************************/
4      libname LINUXLIB "/home/marsyst/wessug";
NOTE: Libref LINUXLIB was successfully assigned as follows:
      Engine:        V9
      Physical Name: /home/marsyst/wessug
```

This next section uses the UPLOAD Procedure to upload the PRDSALE SAS data set from a SAS data library on the Windows workstation to a SAS data library on the Linux server. Data Transfer Services recognizes that PCLIB is a LIBREF on the Windows workstation and transfers the prdsale.sas7bdat SAS data set from that library to the Linux library identified by the LINUXLIB LIBREF. The data are converted to the correct byte representation for a Linux server during the transfer. The NOTE:'s provide information on the particulars of the upload of the PRDSALE SAS data set.

```
6
7      /*******************************************************/
8      /* Move SAS data set from desktop to Linux for processing*/
9      /*******************************************************/
10     proc upload data=PCLIB.prdsale out=LINUXLIB.prdsale;
11     run;

NOTE: Upload in progress from data=PCLIB.PRDSALE to out=LINUXLIB.PRDSALE
NOTE: 138240 bytes were transferred at 1772312 bytes/second.
NOTE: The data set PCLIB.PRDSALE has 1440 observations and 10 variables.
NOTE: Uploaded 1440 observations of 10 variables.
NOTE: The data set LINUXLIB.PRDSALE has 1440 observations and 10 variables.
NOTE: PROCEDURE UPLOAD used (Total process time):
      real time           0.13 seconds
      cpu time            0.01 seconds
```

The uploaded SAS data set is summarized on the Linux server, taking advantage of its greater processing power. Once the summary data set is created, the original PRDSALE SAS data set is no longer needed. So, the DATASETS Procedure is executed to remove it from the Linux SAS data library. The WORK.SUM1 SAS data set created by the SUMMARY Procedure will cease to exist when the remote SAS session is terminated at the end of the SAS program.

```
13
14     /*******************************************************/
15     /* Summarize data on the Linux Server                  */
16     /*******************************************************/
17     proc summary nway data=LINUXLIB.prdsale;
18          class country division prodtype;
19          var actual predict;
20     output out=summ1(drop=_type_ rename=(_freq_=Months)) sum=;
21     run;
NOTE: There were 1440 observations read from the data set LINUXLIB.PRDSALE.
NOTE: The data set WORK.SUMM1 has 12 observations and 6 variables.
NOTE: PROCEDURE SUMMARY used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds

23
24     /*************************************************/
25     /* Delete the data set from the Linux Server         */
26     /*************************************************/
27     proc datasets library=LINUXLIB;
                                      Directory
                          Libref             LINUXLIB
                          Engine             V9
                          Physical Name      /home/marsyst/wessug
                          File Name          /home/marsyst/wessug
                          Inode Number       648982
                          Access Permission  rwxrwxr-x
                          Owner Name         marsyst
                          File Size (bytes)  4096

                              Member     File
                   #  Name     Type       Size  Last Modified
                   1  PRDSALE  DATA     155648  29Dec06:11:26:58
                   2  SHOES    DATA      49152  02Nov06:14:42:36
28          delete prdsale;
29     run;

NOTE: Deleting LINUXLIB.PRDSALE (memtype=DATA).
31
32     /*************************************************/
33     /* Move summarized SAS data set from Linux to desktop */
34     /*************************************************/

NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

The DOWNLOAD Procedure is executed to download the summarized SAS data set from the remote Linux server to SAS on the desktop. The NOTE:'s report the outcome of the download.

```
35     proc download data=summ1 out=PCLIB.summ1;
36     run;
NOTE: Download in progress from data=WORK.SUMM1 to out=PCLIB.SUMM1
NOTE: 672 bytes were transferred at 10839 bytes/second.
NOTE: The data set PCLIB.SUMM1 has 12 observations and 6 variables.
NOTE: Downloaded 12 observations of 6 variables.
NOTE: The data set WORK.SUMM1 has 12 observations and 6 variables.
```

```
NOTE: PROCEDURE DOWNLOAD used (Total process time):
      real time           0.09 seconds
      cpu time            0.00 seconds
```

Below, the NOTE: highlighted in red indicates that the SAS program encountered the ENDRSUBMIT statement; causing the completion of the remote submit. The SIGNOFF statement causes the link that was established between the Windows workstation and the Linux server to be terminated. The two NOTE:'s highlighted in blue show that this has happened.

```
37
38
39   /*************************************************/
40   /* End of block of SAS code that runs on the SASB server*/
41   /*************************************************/
NOTE: Remote submit to SASB complete.
27   signoff;
NOTE: Remote signoff from SASB commencing.
NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           0.57 seconds
      cpu time            0.05 seconds

NOTE: Remote signoff from SASB complete.
```

The final PRINT Procedure is executed by SAS on the desktop against the summary SAS data set that was downloaded from the Linux server.

```
31     /*****************************/
32     /* Print out the results.      */
33     /*****************************/
34   ods listing close;
35   ods html file="Linuxrpt.html";
NOTE: Writing HTML Body file: Linuxrpt.html
36
37   proc print data=PCLIB.summ1 noobs label;
38       by country;
39       id country;
40       sum Months actual predict;
41   title1 'Actual and Predicted Sales Data';
42   title2 'Summarized by Country, Division and Product Type';
43   run;

NOTE: There were 12 observations read from the data set PCLIB.SUMM1.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.21 seconds
      cpu time            0.04 seconds

45   ods html close;
46   ods listing;
```

2. Processing a SAS Data Set that Resides on a Remote Server

This example uses ***Compute Services*** to allocate a SAS data library that already exists on a Linux server, to summarize a SAS data set within that library (on the Linux server), and then to print the results back to the desktop.

```
/*****************************************************************/
/* Example 2. Process a SAS data set that resides on Linux       */
/*****************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASB */
/*****************************************************************/
options comamid=TCP remote=SASB;

/*********************************************/
/* Signon to SASB using a Linux script       */
/* User is prompted for Linux login and password*/
```

```
/*********************************************/
signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpLinux.scr';

/*****************************************************/
/* RSUBMIT this block of code to run on the SASB server */
/*****************************************************/
RSUBMIT;

   options nofmterr nodate;

  /*************************************************************/
  /* Allocate the Linux directory that contains SAS data sets */
  /*************************************************************/
  libname LINUXLIB "/home/marsyst/wessug";

  /*************************************************/
  /* Summarize data on the Linux Server             */
  /*************************************************/
  proc summary nway data=LINUXLIB.shoes;
       class region product;
       var sales stores;
  output out=LINUXLIB.prdsale(drop=_type_ _freq_) sum=;
  run;

  /*********************************/
  /* Print out the results.       */
  /*********************************/
ods listing close;
ods html file="Linuxrpt.html";

proc print data=LINUXLIB.prdsale noobs label;
             by region;
             id region;
  title1 'Shoe Data';
  title2 'Summarized on the SASB Linux Server';
  run;

ods html close;
ods listing;

/*****************************************************/
/* End of block of code submitted to the SASB server  */
/*****************************************************/
ENDRSUBMIT;

signoff;
```

In this example, the entire program is sandwiched between the RSUBMIT/ENDRSUBMIT block, meaning that it will execute on the Linux server. Once it begins executing, it allocates the /home/marsyst/wessug SAS data library on the Linux server. It summarizes the SHOES SAS data set in that SAS data library and creates a report. The report file, LINUXRPT.HTML, will reside on the Linux root directory and not be sent to the Output window of the Windows SAS session. This is a good way to create reports that are to exist on a remote server.

3. Processing a SAS Data Set that Resides on a Linux Server From Your Desktop

This example uses ***Remote Library Services*** to allocate a SAS data library that already exists on a Linux server. Then, a PC SAS program is used to summarize a SAS data set within that library and print the results. Note that all of the processing is done on the desktop, and not on the Linux server. The key to doing this lies in using the ***SERVER=*** keyword on the libname statement executed on the desktop.

```
/*****************************************************************/
/* Example 3. Process Linux SAS dataset from desktop.            */
/*****************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASB */
/*****************************************************************/
options comamid=TCP remote=SASB;

/***********************************************/
/* Signon to SASB using a Linux script         */
/* User is prompted for Linux login and password*/
/***********************************************/
signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpLinux.scr';

/*************************************************************/
/* Allocate the Linux directory that contains SAS data sets */
/*************************************************************/
libname LINUXLIB "/home/marsyst/wessug" SERVER=SASB;

options nofmterr;

  /***********************************************/
  /* Summarize data on the Linux Server          */
  /***********************************************/
  proc summary nway data=LINUXLIB.shoes;
       class region product;
       var sales stores;
  output out=LINUXLIB.sumshoes(drop=_type_ _freq_) sum=;
  run;

  /*********************************/
  /* Print out the results.       */
  /*********************************/
ods listing close;
ods html file="Linuxrpt.html";

proc print data=LINUXLIB.sumshoes noobs label;
             by region;
             id region;
  title1 'Shoe Data';
  title2 'Summarized on the SASB Linux Server';
  run;

ods html close;
ods listing;

  /*********************************/
  /* Delete obsolete SAS data set.*/
  /*********************************/
  proc datasets library=LINUXLIB;
       delete sumshoes;
  quit;

signoff;
```

The first thing that that you should notice is that there are no RSUBMIT/ENDRSUBMIT statements in the program. That means that all of the processing takes place in SAS for Windows. The SERVER= option on the LIBNAME statement specifies the server that **Remote Library Services** is to connect to in order to find the directory specified as the LIBREF. Once the connection is made between the servers the LIBNAME statement executes and produces the following log entry:

```
144  /*********************************************************/
145  /* Allocate the Linux directory that contains SAS data sets */
146  /*********************************************************/
147  libname LINUXLIB "/home/marsyst/wessug" SERVER=SASB;
NOTE: Libref LINUXLIB was successfully assigned as follows:
      Engine:        REMOTE
      Physical Name: /home/marsyst/wessug
```

The second thing that you should notice is that this program does not make efficient use of the remote server. It drags the observations from the SHOES Linux SAS data set across the network to the Windows workstation, summarizes them there, and then sends the summarized observations back across the network to reside in the SUMSHOES SAS data set on the Linux server. Usually, the remote server is more powerful than the desktop, and one uses SAS/Connect to process large SAS data sets there and then has the result sets sent to the desktop. However, the point of this example is that you *can* process SAS data sets that reside on remote servers with SAS on your local server if you need to.

4. Process a SAS Data Set Created on a Linux Server

In this example, ***Remote Library Services*** is used to allocate a SAS data library on a Linux server using the SLIBREF option of the LIBNAME statement. For this to work, a SAS data library on the Linux server must first be allocated with a LIBNAME statement in an RSUBMIT/ENDRSUBMIT block sent to execute on the Linux server. Then, a LIBNAME statement can be executed on the Windows server with the SLIBREF option to allocate the remote SAS data library with a local LIBREF. Subsequent SAS statements running on Windows can use the local LIBREF to read data stored on the Linux server.

In this example, some SAS code is sent to the Linux server to execute in an RSUBMIT/ENDRSUBMIT block. The SAS code includes a LIBNAME statement that allocates a Linux SAS data library and a SUMMARY Procedure that summarizes data to the Linux SAS session's WORK SAS data library. After the RSUBMIT/ENDRSUBMIT block, a LIBNAME statement allocates the Linux server's WORK data library to the local (Windows) SAS session using the SLIBREF option. The Linux server's WORK SAS data library has the local LIBREF of "LINUXLIB" in the Windows Sas session. (Naming it "WORK" would confuse the Windows SAS session, since a local WORK SAS data library already exists). When the PRINT Procedure executes, ***Remote Library Services*** transfers the observations from the Linux server to the PRINT Procedure on the Windows server.

```
/*****************************************************************/
/* Example 4. Process a SAS data set created on a Linux server.    */
/*****************************************************************/
/* Inform SAS/CONNECT that we are using TCP/IP to connect to SASB */
/*****************************************************************/
options nodate nonumber;
options comamid=TCP remote=SASB;

/***********************************************/
/* Signon to SASB using a LINUX script         */
/* User is prompted for LINUX login and password*/
/***********************************************/
signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpLinux.scr';

/*********************************************************/
/* RSUBMIT this block of code to run on the SASB server */
/*********************************************************/
RSUBMIT;

   options nofmterr nodate;

  /*********************************************************/
  /* Allocate the LINUX directory that contains SAS data sets */
  /*********************************************************/
  libname LINUXLIB "/home/marsyst/wessug";


  /***********************************************/
  /* Summarize data on the LINUX Server             */
  /***********************************************/
  proc summary nway data=LINUXLIB.shoes;
       class region product;
       var sales stores;
  output out=WORK.prdsale(drop=_type_ _freq_) sum=;
  run;

/*****************************************************/
/* End of block of code submitted to the SASB server */
/*****************************************************/
ENDRSUBMIT;

/*********************************************************/
/* Allocate the LINUXLIB LIBREF used in the RSUBMIT block. */
/*********************************************************/
```

```
libname LINUXLIB slibref=WORK server=SASB;

  /*********************************************************/
  /* Print the summary data set residing on the Linux server */
  /*********************************************************/
ods listing close;
ods html file="linuxrpt.html";

proc print data=LINUXLIB.prdsale noobs label;
             by region;
             id region;
  title1 'Shoe Data';
  title2 'Summarized on the SASB LINUX Server';
  title3 'Read by a LIBNAME on the Windows Server';
  run;

ods html close;
ods listing;

  /*******************************/
  /* End the SAS/Connect session. */
  /*******************************/
signoff;
```

Note that you can allocate any SAS data library available to the Linux SAS session after the RSUBMIT block has executed. This includes libraries specifically allocated on the Linux server via LIBNAME statements as well as common libraries such as SASHELP, SASUSER, and MAPS.

5. Executing a SAS Program Stored on a Linux Server

This example uses ***Compute Services*** to execute a SAS *program* that is stored on a Linux server. In the example, the SAS program and the data that it will process *already exist* in a Linux directory. The SAS code submitted to the Linux server uses the **X** SAS statement to submit commands to native Linux. Two commands are submitted. The first command changes directories from the users root directory to the directory that contains the SAS program. The second command executes the SAS program.

When SAS programs are executed from a remote server's directory, the SAS Log and the SAS List files are written to that directory. They *are not* written to the log and output windows of your local SAS session. Consequently, unless you logon to the remote server and check the SAS Log and list files, you will not be able to ascertain the success of your SAS program.

```
/*****************************************************************/
/* Example 5. Executing a SAS program residing on a Linux server.  */
/*****************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASB */
/*****************************************************************/
options comamid=TCP remote=SASB;

/*********************************************/
/* Signon to SASB using a UNIX script         */
/* User is prompted for UNIX login and password*/
/*********************************************/
signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpLinux.scr';

/*********************************************************/
/* RSUBMIT this block of code to run on the SASB server  */
/*********************************************************/
RSUBMIT;

   /*************************************************/
   /* Get to the Directory with the SAS programs        */
   /*************************************************/
   X 'cd /home/marsyst/wessug';

   /*************************************************/
   /* Submit a SAS program that resides on the Linux server  */
   /*************************************************/
    X 'bsas Size_The_Data.sas';
```

```
/*****************************************************/
/* End of block of code submitted to the SASB server */
/*****************************************************/
ENDRSUBMIT;
signoff;
```

In this example, both the SAS Log and List files *could* be returned to the local SAS session by simply %INCLUDE-ing the SAS program in the RSUBMIT block:

```
%include(/home/marsyst/wessug/Size_The_Data.sas);
```

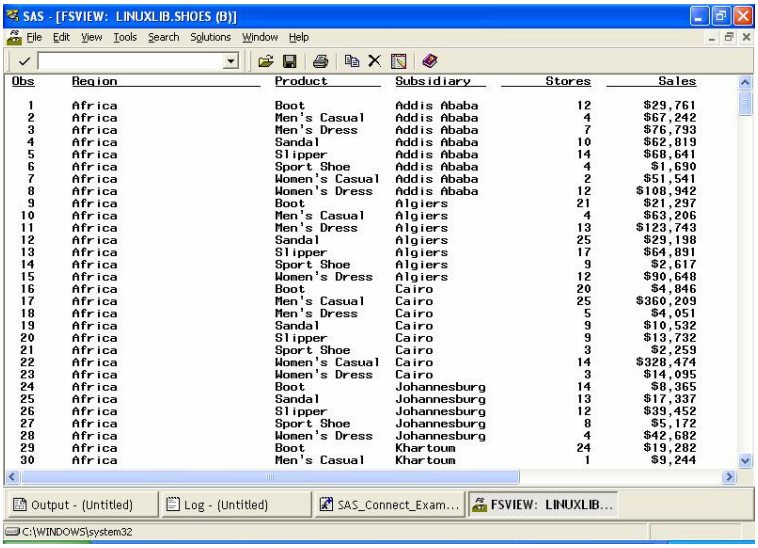
Since the program would be executed within the remote SAS session—not outside of it via the X statement—the SAS Log and List files would be available to the local SAS session just as if it the entire SAS program had actually been sandwiched between the RSUBMIT and ENDRSUBMIT statements.

6. Viewing a SAS Data Set that Resides on a Remote Server

This example uses ***Remote Library Services*** to view the observations of a SAS data set that resides on a Linux server. When the SAS code in the example is executed, you can interactively view the observations via PROC FSVIEW. Note that you can easily substitute PROC FSEDIT or PROC FSBROWSE in the SAS code, below.

```
/*****************************************************************/
/* Example 6. View a SAS data set that resides on a Linux server.  */
/*****************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASB. */
/*****************************************************************/
options comamid=TCP remote=SASB;

/*********************************************/
/* Signon to SASB using a Linux script       */
/* User is prompted for Linux login and password*/
/*********************************************/
signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpLinux.scr';

/*************************************************************/
/* Allocate the Linux directory that contains SAS data sets */
/*************************************************************/
libname LINUXLIB "/home/marsyst/wessug" SERVER=SASB;

/*************************************************/
/* FSVIEW a SAS data set in the allocated directory */
/*************************************************/
proc fsview data=LINUXLIB.shoes;
run;

signoff;
```

As with Example 3, above, this program uses the SERVER= option on the LIBNAME statement to specify the name of the remote server that **Remote Library Services** is to connect to in order to find the directory specified in the LIBREF. After the SAS/Connect conversation is successfully established, the FSVIEW Procedure reads observations from the SHOES SAS data set in the /home/marsyst/wessug SAS data library on the SASB server and transfers them across the network to the Windows SAS session. This results in FSVIEW being opened on the Windows workstation:



Obs	Region	Product	Subsidiary	Stores	Sales
1	Africa	Boot	Addis Ababa	12	$29,761
2	Africa	Men's Casual	Addis Ababa	4	$67,242
3	Africa	Men's Dress	Addis Ababa	7	$76,793
4	Africa	Sandal	Addis Ababa	10	$62,819
5	Africa	Slipper	Addis Ababa	14	$68,641
6	Africa	Sport Shoe	Addis Ababa	4	$1,690
7	Africa	Women's Casual	Addis Ababa	2	$51,541
8	Africa	Women's Dress	Addis Ababa	12	$108,942
9	Africa	Boot	Algiers	21	$21,297
10	Africa	Men's Casual	Algiers	4	$63,206
11	Africa	Men's Dress	Algiers	13	$123,743
12	Africa	Sandal	Algiers	25	$29,198
13	Africa	Slipper	Algiers	17	$64,891
14	Africa	Sport Shoe	Algiers	9	$2,617
15	Africa	Women's Dress	Algiers	12	$90,648
16	Africa	Boot	Cairo	20	$4,846
17	Africa	Men's Casual	Cairo	25	$360,209
18	Africa	Men's Dress	Cairo	5	$4,051
19	Africa	Sandal	Cairo	9	$10,532
20	Africa	Slipper	Cairo	9	$13,732
21	Africa	Sport Shoe	Cairo	3	$2,259
22	Africa	Women's Casual	Cairo	14	$328,474
23	Africa	Women's Dress	Cairo	3	$14,095
24	Africa	Boot	Johannesburg	14	$8,365
25	Africa	Sandal	Johannesburg	13	$17,337
26	Africa	Slipper	Johannesburg	12	$39,452
27	Africa	Sport Shoe	Johannesburg	8	$5,172
28	Africa	Women's Dress	Johannesburg	4	$42,682
29	Africa	Boot	Khartoum	24	$19,282
30	Africa	Men's Casual	Khartoum	1	$9,244

This can be a very helpful tool, because it allows you to review data stored in SAS data sets on your remote servers.

7. Creating a CONTENTS Listing for a SAS Data Library on a Linux Server

This example uses ***Remote Library Services*** to create a CONTENTS procedure listing of a SAS data library that exists on a Linux server. The CONTENTS listing is printed in the Output window of your Windows SAS session. This technique can be very handy in giving you a look at the characteristics of SAS data sets stored on remote servers.

```
/*********************************************************************/
/* Example 7. PROC CONTENTS of SAS data set on a Linux server.     */
/*********************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASB */
/*********************************************************************/
options comamid=TCP remote=SASB;

/*********************************************/
/* Signon to SASB using a Linux script          */
/* User is prompted for Linux login and password */
/*********************************************/
signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpLinux.scr';

/*************************************************************/
/* Allocate the Linux directory that contains SAS data sets */
/*************************************************************/
libname LINUXLIB "/home/marsyst/wessug" SERVER=SASB;

/*****************************************/
/* Create the CONTENTS procedure listing   */
/*****************************************/
ods listing close;
ods html file="contents.html";

proc contents data=LINUXLIB._all_ details;
run;

ods html close;
ods listing;

signoff;
```

This example saves the output in an HTML file. Here is what the Directory segment of the CONTENTS Procedure output would look like if printed, instead:

The CONTENTS Procedure

Directory

Libref	LINUXLIB
Engine	REMOTE
Physical Name	/home/marsyst/wessug
Accessed through server	SASB
Server's libref	LINUXLIB
Server's engine	V9
Server's SAS release	9.01.01M3P02022006 (105)
Server's host type	Linux - i686
Server's versus user's data representation	DIFFERENT
Views interpreted in server's execution	YES
File Name	/home/marsyst/wessug
Inode Number	648982
Access Permission	rwxrwxr-x
Owner Name	marsyst
File Size (bytes)	4096

#	Name	Member Type	Obs, Entries or Indexes	Vars	Label
1	PRDSALE	DATA	80	4	
2	SHOES	DATA	395	7	Fictitious Shoe Company Data

You can see that the /home/marsyst/wessug SAS data library on the SASB Linux server contains two SAS data sets. The rest of the CONTENTS Procedure output would give you detailed information on both data sets.

Note that several of the metrics, *Engine*, *Physical Name, Accessed through server, Server's host type, File Name, Inode Number*, and *Access Permission* are all indicative of the SAS data library being on a Linux server.

8. Processing Tables with Remote SQL Pass-Through (RSPT)

This example uses ***Compute Services*** to pass SQL statements to the Linux server for execution. The CONNECT statement identifies the remote server that the SQL statements will be passed to. The first SELECT statement instructs the SAS System to select everything that is passed back to it from the remote server by the query. The parenthesis contains the query that will be processed on the remote server. Note that the entire query that is to be passed to the remote server *must* be contained within the parenthesis. The DISCONNECT statement ends the connection to the SAS SQL processor in the remote server's SAS session.

```
/*****************************************************************/
/* Example 8. Process Tables on Linux with Remote Pass-Through.    */
/*****************************************************************/
/* Inform SAS/Connect that we are using TCP/IP to connect to SASB */
/*****************************************************************/
options nodate nonumber;
options comamid=TCP remote=SASB;

/*********************************************/
/* Signon to SASB using a LINUX script         */
/* User is prompted for LINUX login and password*/
/*********************************************/
signon 'C:\SAS\Foundation.9.1.3\sas\connect\saslink\tcpLinux.scr';

/*****************************************************/
/* Allocate a LINUX  directory to contain SAS data sets*/
/*****************************************************/
libname LINUXLIB "/home/marsyst/wessug" server=SASB;

/*****************************************************/
/* SQL to summarize data by GENDER.                    */
/*****************************************************/
ods listing close;
ods html file="sqlrept.html";
```

```
proc sql;

connect to remote(server=SASB);

select * from connection to remote
(select region,
        sum(sales) as sumsales format=dollar20.2,
        sum(inventory) as suminvty format=dollar20.2
        from LINUXLIB.shoes
        group by region
        order by region);

disconnect from remote;

quit;

ods html close;
ods listing;

signoff;
```

Note that it was necessary to use the SERVER= option on the LIBNAME statement to allocate the remote SAS data library where the SHOES SAS table resided before attempting to use the LINUXLIB libref in the SQL Procedure.

Conclusions

SAS/Connect software provides a simple way for you to access and process SAS data stored on your organization's various computer platforms. SAS/Connect provides three main types of services:

- Compute Services, which allow you to execute SAS programs on remote servers
- Data Transfer Services, which allow you upload and download SAS files between servers
- Remote Library Services, which allow access to data stored on remote servers as if the data were stored locally.

SAS/Connect must be installed on all of the servers that you intend to communicate with. You begin a SAS/Connect session by first specifying the COMAMID and REMOTE options in an OPTIONS statement, and then executing the SIGNON statement specifying a SAS/Connect script. Afterward, what you do is entirely up to you.

This paper presented eight examples of some of the basic things that can be done with SAS/Connect software. You can use them as springboards for your own efforts. In doing so, you will be able to process your SAS datasets anyway, anyhow, anywhere you choose with SAS/Connect software.

Disclaimer

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

References

SAS Institute Inc. 2006. **SAS OnlineDoc® 9.1.3.** Cary, NC: SAS Institute Inc.

Acknowledgements

I would like to thank Westat Vice President Mike Rhoads for his review and comments on the draft of this paper.

Contact Information

Please feel free to contact me if you have any questions or comments about this paper. You can reach me at:

Michael A. Raithel
Westat
1650 Research Boulevard
Room RW4523
Rockville, Maryland 20850

michaelraithel@westat.com