Paper 43

Using SAS/CONNECT® Software in A Multi-Platform Environment

Cyndie Gareleck, RAND, Santa Monica, California
Rodger Madison, RAND, Santa Monica, California

ABSTRACT

This tutorial (1) provides an overview of client/server computing, (2) discusses client/server services available in SAS/CONNECT software, (3) gives instructions for installing and running the SAS spawner as an NT service, and (4) presents code from several applications that use SAS/CONNECT. The paper also discusses the use of scripts to initiate and end remote sessions, and potential benefits of using SAS/CONNECT software to improve program and cost efficiency. While the examples illustrate use on an NT system and a UNIX workstation, the concepts are applicable to other operating environments.

INTRODUCTION

SAS/CONNECT software is a SAS-to-SAS client/server toolset; it provides for a client/server environment and allows for direct and indirect connections between two or more SAS sessions. In general, the sessions are running on different host systems. With SAS/CONNECT software you can access data on hosts with differing architectures, use resources available on the network that would otherwise not be available to the local environment, and distribute processing over the network to the system most appropriate for the application. In other words, SAS/CONNECT software allows you to develop applications as efficiently and effectively as possible, given the resources of your organization.

A BRIEF OVERVIEW OF CLIENT/SERVER PROCESSING

Several years ago, most computing environments consisted of mainframes connected to dumb terminals. Over the years, personal computers (PCs) started to take the place of the dumb terminals, but the processing was still done by the mainframe with the capability of the PC largely ignored. However, with so much computing power unused, people began considering ways to share some of the processing demands between the mainframe and the PC.

The term “client/server” was first used in reference to PCs on a network. These networks were based on file sharing architectures, where the server downloads files from the shared location to the desktop and the data are then used by desktop applications. File sharing architectures are limited by the amount of shared usage, update needs, and the volume of data to be transferred.

Because of the limitations of file sharing architectures, more advanced client/server architecture emerged. Client/server technology is a computing architecture that splits the workload between two or more computers on a network; each function of an application is located on the computer most capable of performing that particular activity. Client/server architecture also allows for multiple users. For example, if two users are attempting to update the same record at the same time, the update must be reflected instantly on the other user's screen. Simply downloading files from the server to the client is not true client/server architecture. Advantages of true client/server architecture include scalability, ease of maintenance, efficient resource utilization, and flexibility.

Two-Tier Client/Server Architecture

Figure 1 details a two-tier architecture where a client talks directly to a server, with no intervening server. With two-tier client/server architectures, the client-based, local process is the front end that sends a message to a server process requesting the performance of a task. It is typically located in the user's desktop environment. As a rule, client programs manage the user-interface portion of the application, validate data entered by the user, and transmit requests to server programs.

The server process fulfills the client request by performing the requested task(s). The server is generally a more powerful machine and is capable of servicing multiple clients.

Server programs generally issue responses to client requests, receive requests from client programs, execute data retrieval and updates, and manage data integrity. Processing management is split between the local and server environments.

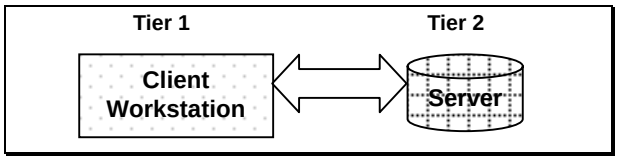

Figure 1: Two-Tier Architecture

The problem with this architecture is that the application logic may be distributed to dozens of client systems. This makes application maintenance very difficult and expensive. Also, the client environment becomes considerably more complex and thus more unstable.

Three-Tier Client/Server Architectures

Figure 2 shows a three-tier architecture. In the three-tier architecture, a middle tier, or agent, is added between the client environment and the server environment. The middle tier provides functions such as message queuing, intelligent agent services, translation, application logic execution, and database staging.

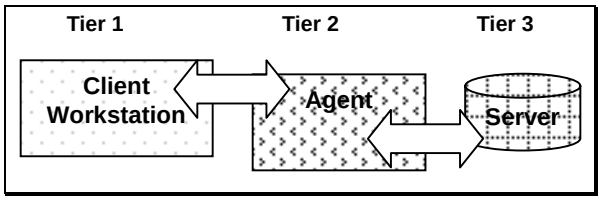

Figure 2: Three-Tier Architecture

The three-tier client/server architecture has been noted to improve scalability and increase flexibility when compared to the two-tier approach.

A limitation with three-tier architecture is that the development environment is purportedly more intricate and difficult to use than two-tier applications.

COMMUNICATIONS ACCESS METHODS

To use SAS/CONNECT software you must specify a communications access method to connect a local and remote SAS session on differing platforms. A communications access method is an interface between SAS Software and the network protocol you select to connect the local and remote host(s). The method you select is dependent on the host platforms and the available network protocols. SAS/CONNECT software supports ten operating platforms and nine access methods, but not all methods are available on all hosts. In most cases the method you select must be available on both the remote and local hosts. The protocol gateway service of a SAS DOMAIN server can be used to allow two SAS sessions on different hosts to communicate without a common communications access method. To use the protocol gateway services of the DOMAIN server, you must run a dedicated SAS session either in an OS/2 or in a Windows NT environment. The DOMAIN server will be discussed in more detail later in

the paper.

Communications methods are subdivided into terminal-based communications and program-to-program communications. Terminal-based methods allow two processes to interact over a network using a terminal interface. They are inexpensive and typically available. They are somewhat limited in that they do not support all services (e.g., remote library services), nor do they support some of the more advanced, multi-tier services. TELNET and ASYNC are examples of commonly used terminal-based methods.

Program-to-program communications methods utilize network protocols to allow two processes to interact over a network. They are often preferred to terminal-based methods for a variety of reasons including larger packet sizes, binary transmission, and asynchronous message notification. TCP/IP[1] is the most frequently used method in this group. NetBIOS and APPC are less commonly used methods.

TCP/IP is the only access method available on NT systems and on all UNIX platforms[2]; the remainder of this paper assumes use of the TCP/IP communications access method.

SAS DOMAIN SERVER

The SAS DOMAIN server, which has been enhanced in Version 7, delivers protocol-independent messaging and eliminates the need for licensing, configuring, and supporting multiple protocol stacks in a given environment. The DOMAIN server provides four services: collection management, queue management, agent scheduling, and protocol gateway. The DOMAIN server is initiated with PROC DOMAIN.

SERVICES PROVIDED BY SAS/CONNECT SOFTWARE

SAS/CONNECT software provides a client/server environment within which local SAS sessions have access to remote files and remote hardware and software resources via one or more remote SAS sessions.

In order to use the services of SAS/CONNECT software, it must be installed on each host, although the systems need not be the same type nor do they need to be running the same operating system.

The client/server services provided by SAS/CONNECT software include:

- **Messaging Services**: provide the ability to develop and deploy multi-tiered applications that communicate by sending data in messages
- **Remote Objecting Services**: provide the ability to process Frame objects between two SAS/Connect sessions
- **Agent Scheduling Services**: provide the ability to schedule executions of SAS statements on demand or periodically
- **Compute Services**: provide access to all available network resources from a local SAS session
- **Remote Data Services**: provide access to data anywhere on the network.

Depending on the needs of the application and available resources, these services can be used alone or in conjunction with each other.

MESSAGING SERVICES

First introduced in the maintenance release of SAS 6.12, messaging services provide the ability to develop and deploy multi-tiered applications that communicate by sending data in messages. The TCP/IP communication method is the only method that supports messaging.

There are two types of messaging services, direct and indirect.

Direct: Direct messaging is the simplest form of messaging. In this model, messages pass data directly between two applications. In direct messaging both the client and server systems must be active simultaneously.

An SCL interface to direct messaging allows you to develop SAS/AF® and FRAME applications that can communicate through a basic interface.

Indirect: Indirect messaging is more complex than direct messaging. Indirect messaging is accomplished with SAS message queues. Programs communicate indirectly by delivering messages to queues and by fetching from or browsing messages in queues. There are two interfaces available for using SAS message queues: an SCL interface and use through a SAS DATA step or a SAS macro.

With indirect messaging, the programs do not have to be running simultaneously in order to communicate. This minimizes the number of active connections. It also allows increased independence between the programs with respect to time.

A collection manager and a queue manager manage message queues. The collection manager manages groups or collections of queues. The queue manager allocates queues, maintains access information, and administers messages in each queue.

Both the collection manager and the queue manager are services of the SAS DOMAIN server. The DOMAIN procedure is used to start a DOMAIN server; the following code starts a DOMAIN server to initialize the collection manager.

```
libname domain ".";
  proc domain collection id=/shr4;
run;
```

The ADMIN procedure is used after the DOMAIN server has been started to perform general DOMAIN management and queue administration. The DOMAIN server remains active until stopped with PROC ADMIN.

REMOTE OBJECTING SERVICES

New to Version 7, remote objecting services is essentially a messaging service for object-oriented applications. Remote object services give SAS/AF developers the ability to process Frame objects between two SAS/CONNECT sessions.

AGENT SCHEDULING SERVICES

Agent scheduling services allow you to schedule SAS statements. They are used in conjunction with compute and messaging services to provide task management across the network. There are four types of agent scheduling services:

- Distributed agent processing: used to efficiently distribute work on a network
- Periodic agent processing: used to schedule tasks at a specific time or date or periodically
- Conditional agent processing: based on specific set of criteria
- Parallel agent processing: agents spawn additional agents which can be processed in parallel with each other

Agent scheduling services can be used together or separately to maximize effectiveness and efficiency of application. In order to use these services a DOMAIN server must be started with the agent option specified:

```
libname domain ".";
  proc domain agent;
run;
```

COMPUTE SERVICES

Compute services provide access to all available network resources from a local SAS session. They are provided by the RSUBMIT command or statement and with Remote SQL

---

[1] TCP/IP stands for Transmission Control Protocol/Internet Protocol. TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network. Multiple vendors on multiple platforms support it.

[2] APPC is also available on NT but only some UNIX platforms (HP-US, Solaris, and AIX).

Pass-Through (RSPT). RSPT allows you to pass SQL statements either to a remote SAS SQL processor or to a supported data base management system.

When using compute services, the local host sends a request to the remote computer; the remote host processes the request and returns the log, list, and any other output such as graphs to the local host.

Compute services are best utilized when the remote system has hardware or software resources unavailable to the local host; the data files are frequently updated; one wants to balance CPU utilization between local and remote resources; and it is not practical to transfer the files[3].

On the downside, the use of compute services increases CPU utilization on the remote host, and network traffic is increased when results are returned to the local host. Depending on the nature of the data being returned, this could severely affect network traffic.

REMOTE DATA SERVICES

Remote data services provide access to network data. Remote data services consist of

- Data Transfer Services: transfer data libraries between local and remote hosts
- Remote Library Services (RLS): provide read/write access to remote data

Remote Data Services: Data Transfer Services

Data transfer services allow you to transfer data libraries between local and remote hosts, without regard to system architecture and operating system. Data transfer services include the Cross-Environment Data Access (CEDA) facility, PROC UPLOAD, and PROC DOWNLOAD.

New to Version 7, CEDA takes advantage of the Version 7 file structure to minimize data conversion between different host formats and reduce steps required by other transfer methods. With CEDA, you can transfer Version 7 data files between hosts or NFS mount a directory from another host and automatically access your Version 7 SAS data files—no translation is required. However, there are some limitations to this facility:

- the hosts must both be directory-based[4]
- update access is not available
- the WHERE expression optimization with an index is not available
- only version 7 data files can be transferred—views and utility files are not supported

In addition, the number of observations that can be transferred between two heterogeneous data sets may be limited to 2^{31-1}. This is because the number of observations in a data set is stored in a long numeric type, and the number of bytes used to represent long numeric type varies between hosts. When this variance occurs[5], the result is either an inability to open the data set or an inability to add observations to the file.

If the restrictions to CEDA make it impracticable to use, PROC UPLOAD and PROC DOWNLOAD may be reasonable alternatives. These procedures allow you to transfer data, including SAS libraries and external files, between a local and remote host. The systems need not have the same internal data representations nor must they be running the same operating system. In addition, as of Version 7, a wildcard character (*) may be used when specifying a data set name.

In a typical scenario, the local host sends the remote host a request to transfer a copy of a file. The remote host fulfills the request and transfers the file to the local host in the appropriate format without any loss of precision. The file is then available on the local host for further processing, with no further impact on network traffic or the remote host.

Because data transfer services result in multiple copies of the data, data transfer is generally not a very feasible solution when the remote data are updated frequently. There may also be situations where security considerations prevent you from maintaining more than one copy of the data. Additionally, you might not want to transfer large files if it will negatively impact network traffic. On the other hand, data transfer is often an efficient way to offload work from a remote host or to maintain copies of data for backup purposes.

Remote Data Services: Remote Library Services

Using a remote engine that executes in the local SAS session and a single or multi-user server that executes in the remote session, RLS provide read/write access to remote data. The local SAS session requests records from the remote session, which then transfers the records to the local session for processing. The cycle continues until all processing is complete.

Since it is not necessary to specifically request a file transfer, RLS provides almost transparent access whether the data are processed interactively or in batch.

RLS is a better choice than data transfer services if it is advantageous to maintain only one copy of the data set. However, since data are constantly moving over the network it is generally not a practical solution for large data sets. In addition, if the remote and local hosts have different internal data representations, some loss of precision may occur with numeric data.

RLS is not available with terminal-based communications methods.

Comparing Compute and Remote Data Services

Compute services and remote data services offer some of the same functionality and are often considered interchangeable. However, there are situations where one may be more efficient than another. Table 1 summarizes when and where these services are most efficiently utilized.

Table 1: Modes of Service and Efficient Uses

	Compute Services	Data Transfer Services	Remote Library Services
Use remote hardware and software resources	✔		
Require multi-user access	✔[a]		✔
Minimize remote CPU use		✔	
Minimize ongoing network traffic		✔	✔
Access remote large data files	✔		
Access remote small data files			✔
Back up files to remote location		✔	
Maintain only one copy of data	✔		✔
Access frequently updated data	✔		✔
Need multiple passes of the data		✔	
Transparent access to data		✔[b]	✔
Subset remote data for local processing			✔
Use local GUI application with remote data			✔
Reduce overall use of disk space	✔		✔

[a] RSPT only

[b] CEDA only

When these services are utilized, there will always be impact on the network. The severity of the impact can be controlled

---

[3] E.g., too big, not enough network bandwidth, security considerations.

[4] Neither CMS nor bound libraries under MVS/390 are supported.

[5] E.g., when a 4-byte long host opens or attempts to add an 8-byte long data set containing greater than 2^{31-1} observations or an 8-byte long host opens or attempts to add a 4-byte long data set containing greater than 2^{31-1} observations.

somewhat through efficient use of the services but it can never be eliminated. It is important to consider network impact and how much can be tolerated when developing your client/server applications.

RUNNING SAS/CONNECT SOFTWARE

SAS/CONNECT sessions are initiated with a signon command. This command invokes a script that controls the connection to the remote host. The spawner is a program that listens for client requests for connection to a remote host.

USING THE SAS SPAWNER ON AN NT SYSTEM

Before remote connections can be made to an NT system, the SAS spawner must be installed and running as an NT service.

SAS Institute maintains a copy of the current spawner program for Windows on its Web site. It is recommended that you download the most current version of the spawner for installation on your system. To download the spawner:

1) Point your Web browser to http://www.sas.com/techsup/download/connect
2) Download the file winspawn.zip into directory c:\sas\connect\sasexe (assuming you installed SAS in c:\sas)
3) Unzip the file. It will expand to two files, spawner.exe and read.me

Installing the SAS Spawner as an NT Service

There are a large number of options available to use when installing the SAS spawner. The most commonly used options are

- **-INSTALL**: This option must be the first option specified when installing the spawner. This option will install the spawner as an NT service—userids and passwords will be checked against the NT domain.
- **-COMAMID**: This option specifies the access method to use. The value depends on the operating system where the spawner is running.
- **-SECURITY**: This option tells the spawner program to use the Windows NT security subsystem[6]. This option is **highly recommended**. If this option is not chosen, users will be able to access all files on the remote system as a root user.
- **-AUTHSERVER**: This option names the domain used for userid authentication. It can be either the local NT domain or the name of a domain accessible through a network.
- **-TELNET**: The spawner listens on port 23 by default. The TELNET option specifies an alternate port. This option would be used to avoid conflict with another program, such as a telnet daemon, listening on port 23.

To allow you and others to use SAS on your NT machine via SAS/CONNECT, you must first give each person the privilege of logging on as a batch job. Instructions are provided below. Note: Before beginning this procedure, you must be logged on as local administrator.

1. Do Start - Programs - Administrative Tools (Common).
2. Select User Manager.
   2.1. Select Policies; User Rights.
   2.2. Check the Show Advanced User Rights box.
   2.3. Scroll down to Log on as a batch job. Click Add.
   2.4. In the List of Names From box, select the appropriate choice.
   2.5. Click Show Users to display names.
   2.6. Click the names you want to add, including your own.
   2.7. After selecting the names, click Add, then OK. The User Rights Policy box returns. The selected names should now appear in the Grant to: box.
   2.8. Click OK in the User Rights Policy box.
   2.9. Close the User Manager.
3. Open a command prompt window. Click on the start menu. Click on command prompt.

2) Change to the !SASROOT[7] directory. From the !SASROOT directory issue the following command:

   C:\sas> C:\sas\connect\sasexe\spawner –install –comamid tcp –security –authserver *domainame*

Before installing the spawner, we suggest you read *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software,* chapter 13.

Starting, Stopping, and Deleting the SAS Spawner as an NT Service

You can start, stop, and delete the spawner using the command as shown below. You may also start and stop the spawner from the NT services control panel.

Starting the spawner: Before connecting to the spawner, you must start it. To start the spawner,

1) Open a command prompt window. Click on the start menu. Click on command prompt.
2) Enter the following commands from the command prompt:

   C:\> net start "SAS job spawner" <CR>

While the spawner will automatically load when you restart the machine, it must be manually started, using the net start command above, before it will accept connections. The spawner will remain resident on your system as an NT service until it is removed.

Stopping the spawner: To stop the spawner,

1) Open a command prompt window. Click on the start menu. Click on command prompt.
2) Issue the following command:

   C:\> net stop "SAS job spawner" <CR>

Removing the spawner: To remove the spawner,

1) Open a command prompt window. Click on the start menu. Click on command prompt.
2) Issue the following command:

   C:\> net stop "SAS job spawner" <CR>

   C:\> \sas\connect\sasexe\spawner –delete <CR>

USING SCRIPTS TO CONNECT TO A REMOTE NT OR UNIX SYSTEM

Scripts are used to manage logging into the remote SAS/CONNECT host. Script files are written using a specialized set of SAS statements called script statements. Scripts perform three basic functions:

1) Invoke the SAS system on the remote host using the SAS command, setting proper communications options for the remote SAS session. The script sets the COMAMID= and DMR system options on the remote host.
2) Listen for responses from the remote host and determine when the remote host is ready for communication with the local SAS session.
3) Look for responses from the remote host and send appropriate answers back to the local host.

A script can additionally perform several other functions including prompting the local user for information such a userid, password, and account information that controls access to the remote host. Scripts can be written to evaluate responses from the remote host and choose a different response conditional on the response. This latter capability is useful for error checking and allows a single script to handle

---

[6] This option is only valid on Windows NT systems.

[7] We are assuming SAS is installed in C:\SAS.

different types of connections.

SAS Institute provides a series of sample scripts for connecting to a range of remote hosts. These scripts can be used as is; however, modifications are often necessary because the scripts are somewhat generic and cannot properly respond to every type of remote host. Typically, an Institute-supplied script must be modified to allow it to handle the login conversation with the remote host. For example, the script for establishing a TCP/IP connection to a UNIX system, tcpunx.scr, checks for a variety of possible responses:

```
unx_log:
   waitfor 'Hello>' : unxspawn /*- Unix spawner prompt-*/
         , '$'              /*-- a common prompt character      --*/
         , '>'              /*-- another common prompt character --*/
         , '%'              /*-- another common prompt character --*/
         , '}'              /*-- another common prompt character --*/
         , 'Login incorrect'     : nouser
         , 'Enter terminal type'  : unx_term
         , 'TERM'                : unx_term
         , 30 seconds            : timeout
         ;
```

Since UNIX allows each user to set the prompt character, this statement will not work in every instance. In addition, the login conversation on a system may include a prompt for an account or other identifiers. The script must be programmed to anticipate and respond to different situations.

Script Examples

The first example shows a portion of the SAS-supplied script tcpunx.scr. The next two examples show modifications to this script. In both examples modified sections are highlighted.

```
/*----------------UNIX LOGON-------------*/
input 'Userid?';
   type LF;
   waitfor 'Password', 30 seconds : nolog;
   input nodisplay 'Password?';
   type LF;
```

Example 2 shows a script modified to allow for a connection to a RAND Solaris system.

```
/*----------------UNIX LOGON----------------------------------------*/
input 'Userid?';
   type LF;
   waitfor 'Password', 30 seconds : nolog;
   input nodisplay 'Password?';
   type LF;
waitfor ':'
/*-- Solaris sends this before the terminal prompt --*/
         , 30 seconds            : timeout
         ;
waitfor 'Terminal' : unx_term  /* Generic terminal prompt */
        , 'TERM'               : unx_term
        , 30 seconds           : timeout  ;
```

Example 3 shows changes made to allow for account and terminal type information that is requested by a RAND Sun OS system at login time.

```
/*----------------UNIX LOGON----------------------------------------*/
input 'Userid?';

   type LF;
   waitfor 'Password', 30 seconds : nolog;
   input nodisplay 'Password?';
   type LF;
   waitfor 'Account:', 30 seconds : nolog;
   input 'Account?' ;
   type LF;
   waitfor 'Terminal', 30 seconds : nolog;
   type 'tty' ;
   type LF;
```

USING SAS/CONNECT SOFTWARE: EXAMPLES

Three examples of compute services and remote data services are provided below.

Example 1 demonstrates updating a remote SAS file using a local GUI. We use PROC FSEDIT to update interactively a remote SAS file from a local system. The local host is a PII-400 running NT 4.0, SP3, and the remote host is a Sun workstation running Solaris 2.6.

```
/*Edit Remote Data with local FSEDIT session*/
/*Assign options and signon*/
/*Script=tcp.solaris.scr*/

options comamid=tcp options connectremote=wolf;

signon 'c:\programfiles\sas\connect\saslink\tcp.solaris.scr';

/*Assign remote library=wolf*/

libname rhost '/wolf/a/cyndie/saseval/700' server=wolf;

/*Edit remote data 10 obs at a time*/
/Use where clause to subset data*/

proc fsedit data=rhost.v700
        (tobsno=10);
        where employ='no';
run;
```

While this method allows remote editing without the need to interact with a second operating environment, the downside is that when the remote and local machines have different architectures, access to catalogs such as FSEDIT screens is not supported. CEDA could be used to browse the data using a custom screen, but updating is not supported when the architecture of the hosts differs.

Example 2 shows use of compute services to convert a remote Excel file to a local SAS data set. The local host is a Sun workstation running Solaris 2.6, and the remote host is a PII-400 running NT 4.0, SP3.

```
/*Assign local library on UNIX system*/
libname lhost   v7   '.' ;

/*Specify remote host (NT) name*/
%let mynode=gareleck-c;

/*Signon*/

options comamid=tcp connectremote=mynode;

signon  '/wolf/a/cyndie/sugi24/tcpwnt.scr';

/*Assign remote library on NT system*/

rsubmit;

libname rhost  v7  'd:\saseval\connect';

/*Convert remote Excel file to remote SAS view with
PROC ACCESS*/

proc access dbms=xls;
   create sasuser.cyndie.access;
     path=  'd:\saseval\software.xls';
       getnames=yes;
   list all;
      create sasuser.cyndie.view;
      select all;
   list view;
run;

endrsubmit;
```

```
/*Create local permanent data from remote view with RSPT */
/*Specify server for RSPT*/
libname rhost2 'd:\saseval\connect' server=mynode;
proc sql;
  connect to remote(server=mynode);
   create table lhost.soft as
 select *
  from connection to remote
  (select * from sasuser.cyndie ) ;
quit;

signoff;
```

Example 3 shows use of data transfer services to combine data from two remote hosts with data from a third, local host. Data is downloaded from two remote hosts, a SAS file is created from a local Excel file, and three data sets are combined on the local host. A format library is also downloaded. The local host system is a PII-400 running NT 4.0, SP3, and the remote host is a Sun workstation running Solaris 2.6.

```
/*Assign local library*/
libname lhost   v7  'd:\saseval';

/*Set TCP option for remote signons*/
options comamid=tcp;

/*Signon host 1*/
options connectremote=wolf;
signon 'c:\programfiles\sas\connect\saslink\tcp.solaris.scr';

/*Signon host 2*/
options connectremote=didabal;
signon 'c:\programfiles\sas\connect\saslink\tcp.solaris.scr';

/*Download from host 1-includes format library*/
submit wolf;
libname rhost1 '/wolf/a/cyndie/sugi24';
  proc download inlib=rhost1
    outlib=lhost mt=all;
    exclude soft b  / memtype=data  ;
  run;
endrsubmit;

/*Download from Host 2*/
rsubmit didabal;
libname rhost2 '/didabal/a/share';
  proc download data=rhost2.b out=b;
run;
endrsubmit;

/*Signoff host 1 and host 2*/
signoff wolf;
signoff didabal;

/*Convert Excel file to SAS view*/
proc access dbms=xls;
  create sasuser.xls.access;
   path=  'd:\saseval\c.xls';
   getnames=yes;
   list all;
 create sasuser.xls.view;
  select all;
  list view;
run;

/*Combine Data from Two remote hosts and PC*/
/*Assign libname for newly downloaded format library*/
libname library 'd:\saseval';
data lhost.all;
  set a  b  sasuser.xls;
  format var1 yn. ;
run;
```

VERSION 7 ENHANCEMENTS

Version 7 of SAS/CONNECT software contains many new features and enhancements. Some, such as the DOMAIN server, the CEDA facility, messaging services, remote objecting services, agent scheduling services, and use of a wildcard character for transferring binary files, have been discussed elsewhere in this paper. Other enhancements to SAS/CONNECT software include

- Support for the Microsoft security support provider interface (SSPI), which enables a Windows NT user to be transparently authenticated on another Windows NT machine.
- Support for general Version 7 enhancements
- Data encryption[8] to guarantee the security of data sent across a network
- Asynchronous remote processing, which means that control is returned immediately so you can continue processing on the local host
- New translation algorithm to prevent unnecessary translation from occurring
- Expanded list of filetypes that can be transferred with the memtype option in PROC UPLOAD and PROC DOWNLOAD
- Additional options in PROC UPLOAD and PROC DOWNLOAD to help identify the files that will be transferred, the translation tables to use, and the transfer technique

VERSION 7 COMPATIBILITY WITH VERSION 6

If your organization will be running Version 7 concurrently with Version 6, you should be aware of compatibility issues between the two versions. For example:

- Version 7 can read and write Version 6 files but the opposite is not true
- Features new to Version 7 are not available in Version 6 files
- Long variable names included in Version 7 files will be truncated if the files are converted to Version 6
- Some features in Version 7 files cannot be converted to Version 6 and any attempt to convert the files will fail

We strongly recommend that you refer to the Version 7 documentation before using SAS/CONNECT in a mixed environment.

CONCLUSION

The SAS System provides a complete and flexible set of tools to implement a client/server based application. There are a number of issues to consider when selecting an appropriate client/server architecture. These include amount of data to be accessed or processed; location of data; number of users; type of processing; patterns of usage; location of hardware and software resources; business strategic planning, potential growth on the number of applications, cost, and the current and future computational environment.

---

[8] You can use the SAS proprietary encryption services on all platforms; they are provided by SAS, are free of charge, and require no additional software license. You must license SAS/SECURE software in order to use the encryption services RSA BSAFE Toolkit or the Microsoft CryptoAPI. In additon, to use the Microsoft CryptoAPI you must install either Microsoft Base Cryptographic Service Provider, which supports weak encryption, or the Microsoft Enhanced Cryptographic Service Provider, which supports strong encryption.

ACKNOWLEDGMENTS

We would like to acknowledge the assistance of RAND for providing us the resources and time to develop this paper. We would also like to thank Eric Nilson for his insightful comments on client/server architecture, and Christine Taylor for her editing assistance.

REFERENCES

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

Edelstein, Herb,(1994), *Unraveling Client/Server Architecture. DBMS 7*, 34(7).

SAS Institute Inc. (1996), *Changes and Enhancements for SAS/CONNECT and SAS/SHARE Software, Preliminary Documentation Release 6.11, TS040*, pp. 13-17, Cary, NC:SAS Institute, Inc.

SAS Institute Inc. (1997), *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software, Version 6, First Edition,* Cary, NC:SAS Institute, Inc.

Beatrous, Steve and Clifford, Billy (1998), "Sometimes You Get What You Want: SAS I/O Enhancements for Version 7," *Proceedings of the Twenty-third Annual SAS Users Group International Conference*

CONTACT INFORMATION

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

Cyndie Gareleck
RAND
1700 Main Street, PO Box 2138
Santa Monica, CA 90401-2138
310 393-0411, x7815
cyndie@rand.org

Rodger Madison
RAND
1700 Main Street, PO Box 2138
Santa Monica, CA 90401-2138
310 393-0411, x7616
rodger@rand.org