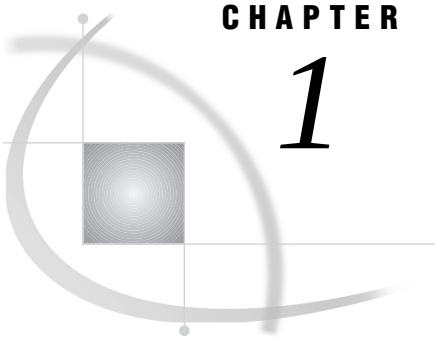


CHAPTER

1

What Is SAS/CONNECT Software?

Introduction

SAS/CONNECT software is a SAS-to-SAS client/server toolset. SAS/CONNECT provides users and applications developers the ability to manage, access, and process data in a distributed environment by enabling you to

- direct processing to a remote data source and get results back locally.
- develop local graphical user interfaces that process remote data sources.
- transfer disk copies of data.
- build integrated client/server applications.
- develop store-and-forward applications.
- write object-oriented applications that span platform boundaries.
- perform client/server based task scheduling.

This allows you to create a distributed environment that best utilizes your data and hardware resources to satisfy your enterprise needs.

Using SAS/CONNECT in a Client/Server Environment

The client/server environment of SAS/CONNECT gives you access to files, hardware resources, and SAS software on various remote hosts to use with a SAS session on the local host.

You can connect to multiple remote SAS sessions, process applications, and access data in any of the remote sessions or in your local SAS session. Processing and data access can be performed asynchronously on a remote host as you continue processing on your local host. This processing can be performed on a periodic basis using the SAS/CONNECT agent scheduler.

Having the ability to spread processing among remote resources, SAS/CONNECT also provides the ability to design and develop applications that communicate directly or indirectly by using message queues. Also, SAS/AF developers are no longer limited to accessing objects only within their local environments. They can now distribute selected portions of their encapsulated object frameworks across remote session boundaries.

The client/server capabilities of SAS/CONNECT also enable you to combine data from two seemingly incompatible systems into one data set. For example, you can access data in an ORACLE database on one system and in a DB/2 database on another system and combine them on your local host.

With all SAS/CONNECT's capabilities, you are able to use your computing resources to their best advantage by distributing SAS processing to the most appropriate machine.

SAS/CONNECT provides several services to enable the development of distributed applications:

Compute Services
: give you access to all of the computing resources on your network by enabling you to direct the execution of SAS programs to a remote host. The results and any output generated by the remote execution is returned to the local SAS session. For short running tasks, remote submits can be processed serially. This means that control is regained after the remote processing is complete. For longer running tasks, remote submits can be processed asynchronously so you can immediately continue local processing.

Data Transfer Services
: provide a method for moving a copy of the data from one machine to another machine.

Remote Library Services
: provide transparent access to remote data. The data resides in remote libraries and moves the data through the network as the local execution requests it.

Messaging Services
: give you the ability to design and implement applications that communicate by sending data in messages. The message structure is completely user-defined. Messaging Services can be further divided into *direct* and *indirect messaging* services. Direct messaging is used to pass data between two active applications. For example, direct messaging might be used when a remote application needs data before processing can continue. Indirect messaging enables programs to communicate indirectly by placing messages on queues. Therefore, the different programs can run independently of each other.

Remote Objecting Services
: give you the ability to process Frame objects between two SAS/CONNECT sessions. Remote objecting extends the messaging capability by allowing SAS/AF developers to distribute selected portions of their encapsulated object frameworks remotely.

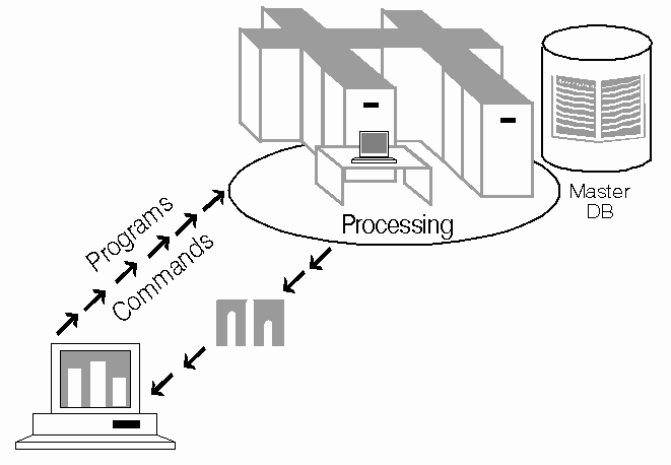
Agent Scheduling Services
: give you the ability to execute a series of SAS statements periodically or on a more dynamic, on-demand basis. Agent scheduling can also be used to specify the completion notification, to spawn additional jobs based on conditions, and to produce alerts and reports.

Each set of services has a well defined set of benefits. By matching these characteristics to the characteristics of your data and your application needs, you can determine how best to combine these services to write the most efficient distributed application. Most often, the best implementation of a distributed application incorporates a combination of these services.

Compute Services That Use RSUBMIT in the SAS Client/Server Environment

Compute services take advantage of remote computing resources (hardware, software, and data) to execute an application more efficiently by maximizing the use of all computing resources. As Figure 1.1 on page 5 illustrates, these services enable you to move some or all portions of an application's processing to a remote machine.

Figure 1.1 Model of Compute Services That Use RSUBMIT Processing



You can:

- take advantage of remote hardware resources.
- utilize software available in the remote environment.
- interface with existing mainframe and other legacy systems, for example, by building a single SAS program that contains statements that run locally and statements that execute on multiple remote legacy hosts.
- perform remote tasks in the background (asynchronously) while processing locally.
- execute against the remote copy of the data.
- use RSUBMIT to remote-submit macro steps to the remote host, and then pass return code information about the remote process to the local SAS session.
- execute graphics programs on the remote host, and display the graphics locally by using the graphics capabilities of the local workstation, plotter, or printer.

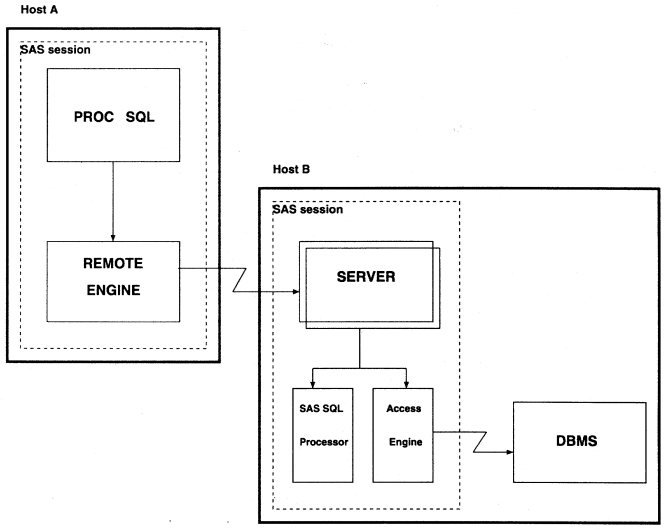
The results of the remote processing are then returned to the local machine. This is useful when the remote machine has hardware or software resources available to perform the task at hand more efficiently. It can also be preferable if the amount of data to be processed is too large to be moved to the local machine or if the data are updated too frequently for a local static copy to be useful.

In addition, compute services allow you to submit statements to be executed asynchronously by a remote host. Once submitted, control is returned to the local host to allow you to immediately continue local processing. This gives you the ability to increase productivity by processing in parallel with both a local and a remote SAS session. Results from the remote asynchronous processing can be retrieved at your request.

Compute Services That Use Remote SQL Pass-Through in the Client/Server Environment

Remote SQL Pass-Through (RSPT) gives you control of where SQL processing occurs. As Figure 1.2 on page 6 shows, RSPT allows you to pass SQL statements to a remote SAS SQL processor by passing them through a remote SAS server. You can also use RSPT to pass SQL statements to a remote DBMS by passing them through a remote SAS server and a REMOTE access engine that supports pass-through.

Figure 1.2 Remote SQL Pass-Through Services



With RSPT you can:

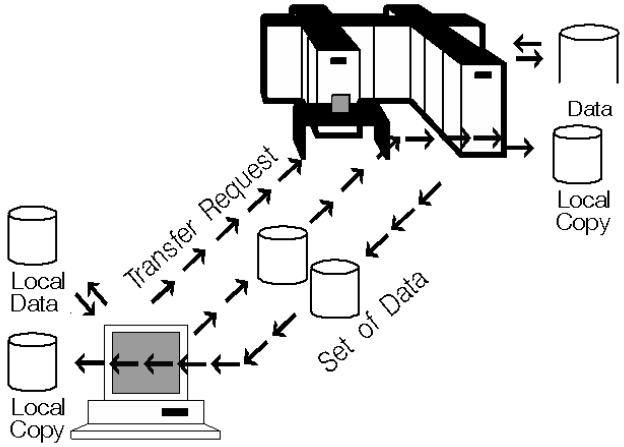
- pass SQL statements to a remote DBMS to select data or execute statements to modify, manipulate, and manage data. This includes creating DBMS views.
- pass SQL statements to SAS SQL to select data or execute statements to modify, manipulate, and manage data. This includes creating SAS SQL views.

You can invoke RSPT through PROC SQL statements that are passed to the remote server for execution in the server SAS session, or you can store SQL pass-through statements in local SQL views.

Data Transfer Services in the SAS Client/Server Environment

As Figure 1.3 on page 7 illustrates, data transfer services provide a method for moving a copy of the data from one machine to another machine. Subsequent local processing takes place against the local copy of the data without generating further network traffic until you decide to update the copy of the data with another transfer.

Figure 1.3 Model of Data Transfer Services Processing



Data is transferred with the UPLOAD and DOWNLOAD procedures. You can transfer SAS data sets, SAS catalogs, MDDB, FDB, DMDB, SQL views, entire SAS data libraries, and external files.

Note: External files can be transferred in either text or binary format. △

The data transfer capabilities enable you to

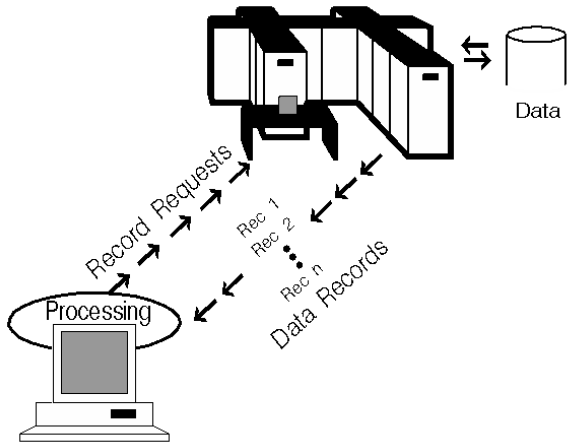
- automate both data or application distribution and centralized data collection.
- increase the robustness of your decision support environment by keeping a local copy of your data, which is insulated from network failure.
- transfer multiple SAS files in a single step by using the INLIB= and OUTLIB= options. This capability enables you to transfer an entire library or selected members of a library in a single PROC UPLOAD or PROC DOWNLOAD step.
- specify certain entries in a catalog or certain members in a library that should be transferred by using the SELECT and EXCLUDE statements.
- name a specific translation table to be used during a data set or a catalog transfer.
- use WHERE processing for dynamic data subsetting and SAS data set options when transferring individual SAS data sets.
- replicate certain data set attributes when you transfer a data set.
- back up local files to a remote host.
- back up remote files to a local host.
- transfer collections of files (such as a partitioned data set, a MACLIB, or a directory) between local and remote hosts.
- distribute files from one workstation to others by uploading to a remote host and then downloading to other workstations that need the files.
- move SAS files between releases of the SAS System as well as across hosts.
- transfer catalog entries that contain graphics output by using a simple one-step process.

Remote Library Services in the SAS Client/Server Environment

Remote library services (RLS) provides transparent access to remote data. The data resides in remote libraries, and RLS moves the data through the network as the local execution requests it. As Figure 1.4 on page 8 illustrates, a copy of the data is not

written to the local file system, and the data must pass through the network on any subsequent use by the local processing system.

Figure 1.4 Model of RLS Processing



RLS gives you:

- transparent access to remote data.
- access to current data because no local copy is made.
- a reduction of disk space consumption because multiple copies of the data are not created.
- the ability to run a local graphical user interface and process remote data.

Remote library services are enabled in the client/server environment by a *REMOTE engine* that executes in the local or client SAS session and a *server* that executes in the remote or server SAS session. The SAS procedures and DATA steps that execute in the local session pass requests to access remote SAS files to the REMOTE engine. The REMOTE engine communicates the requests for data to the server. The server then administers the requests to SAS files on behalf of the local (client) SAS session.

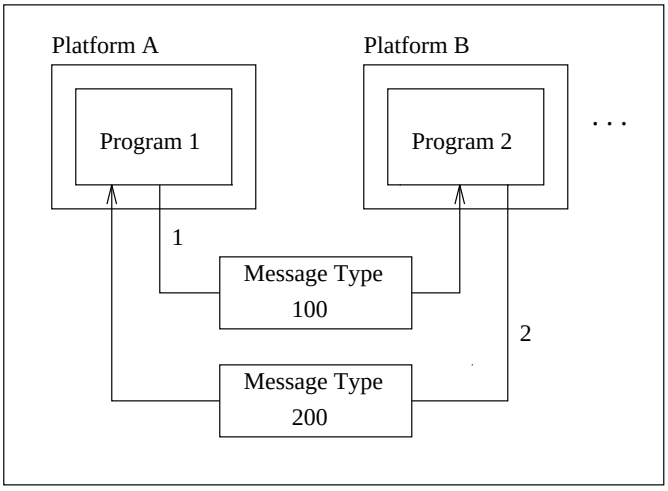
Direct Messaging Services in the SAS Client/Server Environment

Messaging allows applications to communicate by sending each other data in messages. Any action can be taken upon receipt of a message, and acknowledgments can be returned to the sender if appropriate.

The direct-messaging facility allows basic and flexible message construction, transmission, and notification services that span operating system and hardware boundaries across the enterprise. Messages are free-form. Their structure, which is defined by the applications developer, may range from a simple collection of variables to complex hierarchies of SCL lists. Additionally, messages may include one or more attachments that can take the form of SAS data sets or filtered subsets, catalogs or catalog entries, MDDB, FDB, DMDB, SQL views, and external files.

Each message contains a message type field. This field is used to define the set of message types that are meaningful to a particular program. When a program receives a message with a known message type, it knows the layout of the data that is contained in the message body and can take the appropriate action based on the values of the data. Figure 1.5 on page 9 illustrates the basic structure of direct-messaging.

Figure 1.5 Basic Structure of Direct-Messaging



Direct messaging services can be used to

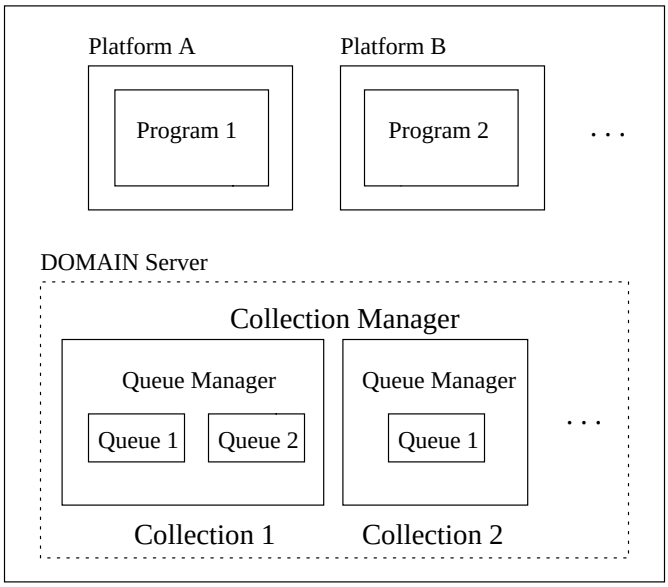
- develop and deploy multi-tiered distributed applications.
- separate and centralize business and data access to the server portion of the application.
- develop single and/or multi-user server applications.
- segment your logic into individual distributed programs.
- execute individual programs of an application on the best host that meets your data and resource requirements.
- isolate message transmissions to only those programs that require the information.
- allow for a simple, direct exchange of data.

Indirect Messaging Services in the SAS Client/Server Environment

SAS indirect messaging enables programs to communicate indirectly by placing messages on queues in storage. Therefore, the pieces of your application can run independently of each other, can run at different speeds and times, and can run without a direct connection between them. Figure 1.6 on page 10 illustrates the basic structure of indirect-messaging.

Indirect messaging provides a basic and logical means of communication. Programs communicate indirectly by delivering messages to queues and by fetching from or browsing messages in queues. The message queues are administered by a queue manager. The queue manager is a server process that is responsible for allocating the queues, maintaining access information for each of the queues, and administering the messages that belong to each queue. Queues can be designated as permanent, which means that the queue manager is responsible for storing the messages sent to this type of queue and for maintaining their persistence until the messages are fetched.

Figure 1.6 Basic Structure of Indirect-Messaging



Indirect messaging services provide:

- the ability to design the communicating programs of your distributed application to run *independently* of each other with respect to time.
- a messaging-queue facility that is integrated with the SAS System and completely portable.
- access to the message queue interface through SCL programs, the SAS DATA step, and the SAS macro facility.
- a way for each program to be completely removed from the interface of any other program by using indirect communication through message queues.
- the ability to execute different logic based on message receipt.
- a scheduling technique so programs can be run on different platforms or at different times without affecting other programs in the application.
- reduced network maintenance in the event of a network failure by minimizing the number of active connections, because queue-based programs never require a direct connection.
- applications developers with the ability to focus on the business needs for the application rather than the details of the network.

How SAS/CONNECT Works with the SAS System

SAS/CONNECT enables both direct and indirect connections between two or more SAS sessions. Typically, each of these sessions is running on a different host. After the connection is made, you have access to the services and resources available to both sessions.

When you initiate a connection from a local SAS session to a remote host, you invoke the SAS System on the remote host. This type of connection is required when you use remote compute services or remote library services, perform data transfers, or use remote objects. Connections are not required for messaging services or agent scheduling.

The terms *remote* and *local* refer to how you interact with a SAS session. These terms are not related to the physical location of the host device. The SAS session in

which you issue local commands is the local host. The SAS session to which you direct your remote commands is the remote host. In addition, the term *host* is used to describe any computer and operating system on which you can run SAS software.

The remote SAS session is not directly accessible. However, you can think of it as running in a special mode. You can access it only through the SAS/CONNECT link from your local host. All output and log messages produced by the remote session are generally displayed in the local session.

To make the connection, SAS/CONNECT uses a communications access method. There are a number of different access methods supported by SAS/CONNECT. Before you can use SAS/CONNECT, you must configure your hosts for a supported communications access method. Table 1.1 on page 11 identifies the communications access methods that you can use to make a connection between a local and a remote host.

Table 1.1 SAS/CONNECT: Valid Access Methods between Hosts

Remote Hosts	Local Hosts								
	CMS	OpenVMS	OS/2	OS/390	UNIX	Win NT Win 95 Win 98	Win 32s (V6 only)	Macintosh (V6 only)	VSE (V6 only)
CMS	APPC TCP/IP	TCP/IP TELNET	APPC EHLLAPI TCP/IP	APPC TCP/IP	APPC TCP/IP TELNET	APPC EHLLAPI TCP/IP TELNET	APPC CPIC EHLLAPI TCP/IP TELNET	TCP/IP	APPC
OpenVMS	TCP/IP	TCP/IP TELNET DECnet	TCP/IP TELNET	TCP/IP	TCP/IP TELNET	TCP/IP TELNET DECnet	TCP/IP TELNET DECnet	TCP/IP	none
OS/2	APPC TCP/IP	TCP/IP	APPC TCP/IP NetBIOS	APPC TCP/IP	APPC TCP/IP	APPC TCP/IP NetBIOS	APPC TCP/IP NetBIOS CPIC	TCP/IP	APPC
OS/390	APPC TCP/IP	TCP/IP TELNET	APPC TCP/IP TELNET EHLLAPI	APPC TCP/IP	APPC TCP/IP TELNET	APPC TCP/IP TELNET EHLLAPI	APPC TCP/IP TELNET EHLLAPI CPIC	TCP/IP	APPC
UNIX	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP	none

Remote Hosts	Local Hosts								
	CMS	OpenVMS	OS/2	OS/390	UNIX	Win NT Win 95 Win 98	Win 32s (V6 only)	Macintosh (V6 only)	VSE (V6 only)
Win NT Win 95 Win 98	APPC TCP/IP	TCP/IP DECnet	APPC TCP/IP NetBIOS	APPC TCP/IP	APPC TCP/IP	APPC TCP/IP DECnet NetBIOS	APPC TCP/IP DECnet NetBIOS CPIC	TCP/IP	APPC
VSE (V6 only)	APPC	none	APPC EHLLAPI	APPC	APPC	APPC EHLLAPI	APPC EHLLAPI CPIC	none	APPC

To initiate and terminate a connection, you invoke the SIGNON and SIGNOFF commands, respectively. Based on the access method that you use, these commands execute a script or connect you directly to the remote host. A *script* is an external file on the local system that contains special SAS statements that control the connection. You can use one of the sample scripts that is provided by SAS Institute, a script that is provided by your computing installation, or a script that you write on your own.

You can use SAS/CONNECT in interactive line mode and non-interactive mode, as well as in the SAS windowing environment. Noninteractive mode gives you a way to

- perform daily or nightly automated backups.
- initiate transaction processing to a master database at a specified time each day.
- centralize and automate data and report distribution to workstations in a network.
- centralize and automate data collection from workstations in a network.