CHAPTER 1

SAS/CONNECT: Definitions and Services

SAS/CONNECT Terminology

SAS/CONNECT

SAS/CONNECT software is a SAS client/server toolset that provides scalability through parallel SAS processing. By providing the ability to manage, access, and process data in a distributed and parallel environment, SAS/CONNECT enables users and applications developers to do the following:

- achieve SAS interoperability across architectures and SAS releases
  - directly process a remote data source and get results back locally
  - transfer disk copies of data
  - develop local graphical user interfaces that process remote data sources
- develop scalable SAS solutions
  - run multiple independent processes asynchronously and coordinate the results from each task execution in a client SAS session

  - □ scale up to fully use the capabilities of symmetric multiprocessing (SMP) hardware, and scale out to fully use the features of distributed processors
  - □ use pipeline processing (TCP/IP ports) to run multiple dependent processes asynchronously
  - □ collect the resources of multiple computers that work in parallel, which produces a powerful, yet inexpensive processing solution

- □ manage distributed resources
  - □ 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

The Client/Server Relationship

SAS/CONNECT links a SAS client session to a SAS server session. The terms SAS/CONNECT *client* and *server* depict a relationship between two SAS sessions.

The client session is the initial SAS session that creates and manages one or more server sessions. The server sessions can run either on the same computer as the client (for example, an SMP computer) or on a remote computer across a network.

Single-User Server

SAS/CONNECT provides the following single-user server functionality for Remote Library Services (RLS):

- □ provides transparent access to remote data
- □ gives single-user access to a dedicated server
- □ enables full, unrestricted access to DBMS data via a SAS/ACCESS engine
- □ enables you to connect to the server by using a SIGNON statement and a LIBNAME statement that specifies the REMOTE engine

```
SIGNON server-ID;
LIBNAME libref REMOTE 'datalib' SERVER=server-ID;
```

  The LIBNAME statement implicitly starts the single-user server.

Multi-User Server

SAS/SHARE provides the following multi-user server functionality for Remote Library Services (RLS):

- □ gives concurrent, multi-user access to a server

  Note: The ability to access DMBS data through a multi-user server is controlled by a specific SAS/ACCESS engine. △

- □ is explicitly started and controlled by a system administrator

```
PROC SERVER server=server-ID;
```

- □ enables you to connect to the server by using a LIBNAME statement that specifies the REMOTE engine

```
LIBNAME libref REMOTE 'datalib' SERVER=server-ID;
```

The LIBNAME statement causes a connection to a pre-existing server.

Communications Access Method

A *communications access method* is the interface between SAS/CONNECT and the network protocol that you use to connect two SAS sessions. You must specify a communications access method for SAS/CONNECT.

TCP/IP is the supported access method on all SAS 9.2 operating environments. The XMS access method is used to connect client and server sessions that both run under z/OS.

Before any meaningful work can be accomplished between a client and a server, the access method must be configured in the client and the server environments. For details, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE*.

Encryption Providers

Encryption providers include the SAS products and third-party strategies for protecting data and credentials (user IDs and passwords) that are exchanged in a SAS/CONNECT client/server environment. All these providers use proven, industry-standard encryption algorithms for data protection.

Here are the encryption providers that SAS/CONNECT can use:

SAS Proprietary	is a fixed encoding algorithm that is included with Base SAS software. It requires no additional SAS product licenses. The SAS proprietary algorithm is strong enough to protect your data from casual viewing. SASProprietary provides a medium level of security.
SAS/SECURE	is an add-on product that provides encryption and data integrity algorithms in addition to the SASProprietary algorithm. SAS/SECURE requires a license, and it must be installed on each computer that runs a client and a server that will use the encryption algorithms. Although SAS/SECURE increases data security, it cannot completely prevent unauthorized access to your data.
Secure Sockets Layer (SSL)	is a protocol that provides network security and privacy. Developed by Netscape Communications, SSL uses encryption algorithms that include RC2, RC4, DES, TripleDES, and MD5. In addition to providing encryption services, SSL performs client and server authentication, and it uses message authentication codes to ensure data integrity.
Secure Shell (SSH)	is a protocol that enables users to access a remote computer via a secure connection. SSH is available through various commercial products and as freeware. OpenSSH is a free version of the SSH protocol suite of network connectivity tools. Although SAS software does not include a programming interface to SSH functionality, SAS does support the tunneling feature of SSH that enables a SAS client to make an encrypted connection to a SAS server. Port forwarding is another term for tunneling. The SSH client and SSH server act as agents between the SAS client and the SAS server, tunneling information via the SAS client's port to the SAS server's port.

For details about these encryption providers, see *Encryption in SAS*.

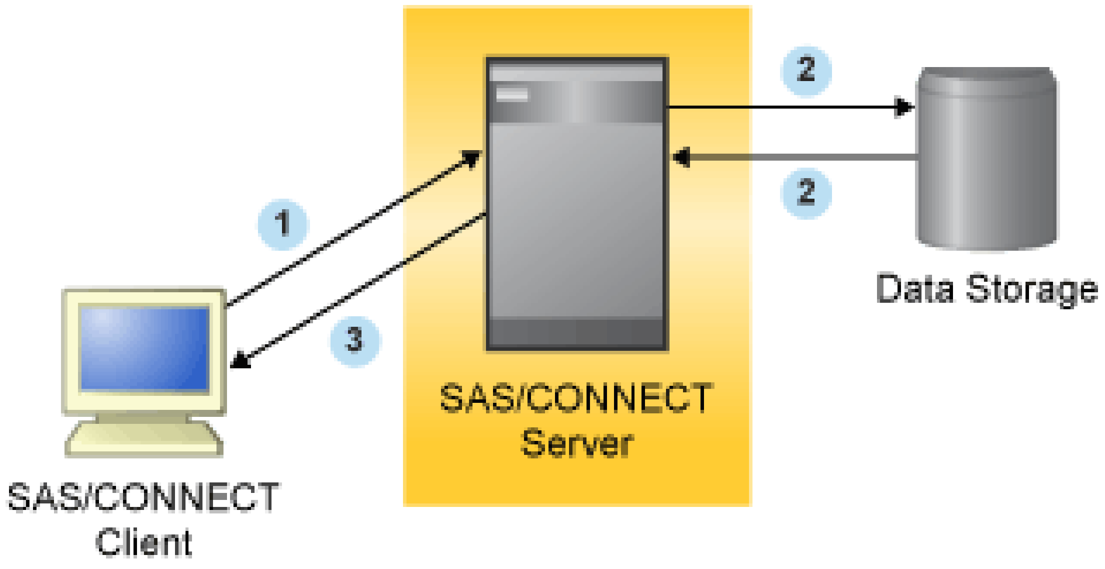
Programming Services

Compute Services and MP CONNECT

Compute Services That Use RSUBMIT

Compute Services provides access to all of the computing resources on your network by enabling you to direct the execution of SAS programs to one or more server sessions. The results and any output that is generated by the remote execution are returned to the client session. For short-running tasks, remote submits can be processed synchronously. This means that control is returned after the remote processing is complete. For longer-running tasks, remote submits can be processed asynchronously. This means that control is returned immediately, and you can continue local processing or remote processing to another server session.

Figure 1.1 Model of Compute Services



❶ The SAS/CONNECT client sends SAS statements to the server session.
❷ The SAS statements execute in the SAS/CONNECT server session using remote data.
❸ Results are sent back to the client session.

Note: Asynchronous Compute Services is commonly referred to as MP (Multi-Process) CONNECT. △

The following figure shows that these services enable you to move some or all portions of an application's processing to a remote computer.

Compute Services enables you to do the following:

- □ achieve scalability for your SAS applications

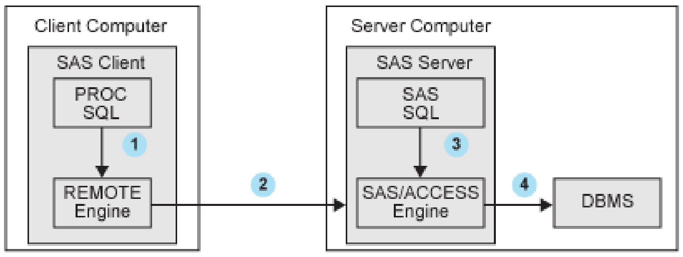
  - □ perform remote tasks in the background (asynchronously) while processing locally
  - □ run multiple SAS processes asynchronously and coordinate the results from each task execution in your client SAS session
  - □ use pipeline processing to overlap execution of multiple dependent SAS DATA steps or procedures
  - □ use processors on an SMP computer (which is referred to as “scaling up”) and using idle processors across a network (which is referred to as “scaling out”)

- □ access remote resources
  - □ take advantage of server hardware and software resources
  - □ access 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 computers)
  - □ execute against the remote copy of the data
  - □ submit macro steps remotely to the server, and then pass return code information about the server process to the client
  - □ execute graphics programs on the server and display the graphics locally by using the graphics capabilities of the local workstation, plotter, or printer

Compute Services That Use Remote SQL Pass-Through

Remote SQL Pass-Through (RSPT) gives you control of where SQL processing occurs. RSPT enables 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



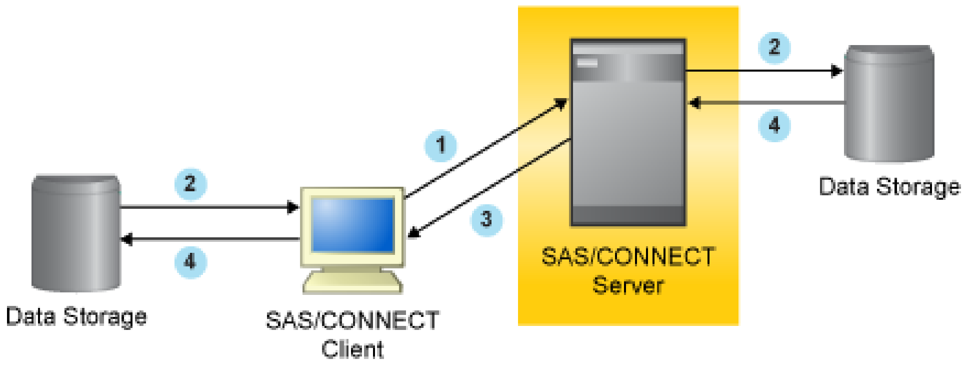
❶ The SAS client uses a REMOTE engine to pass SQL statements to a server session.
❷ The SQL statements are passed to the server session.
❸ The SQL statements are passed to SAS SQL to select data or to execute statements in order to modify, manipulate, and manage data. This includes creating SAS SQL views.
❹ The SQL statements are passed to a remote DBMS to select data or to execute statements in order to modify, manipulate, and manage data. This includes creating DBMS views.

You can invoke RSPT by using 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

Data Transfer Services enables you to move a copy of the data from one computer to another computer. The data is translated between computer architectures and SAS version formats, as necessary.

Figure 1.3 Model of Data Transfer Services (UPLOAD and DOWNLOAD)



❶ The SAS/CONNECT client requests an upload of data records to the SAS/CONNECT server session for processing.

❷ Data is copied from the client disk and is written to the server disk for processing.

❸ The SAS/CONNECT client requests the download of data records from the server to the client for processing.

❹ Data is copied from the server disk and is written to the client disk for processing.

Data is transferred using the UPLOAD and DOWNLOAD procedures. You can transfer SAS data sets, SAS catalogs, MDDB, 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 do the following:

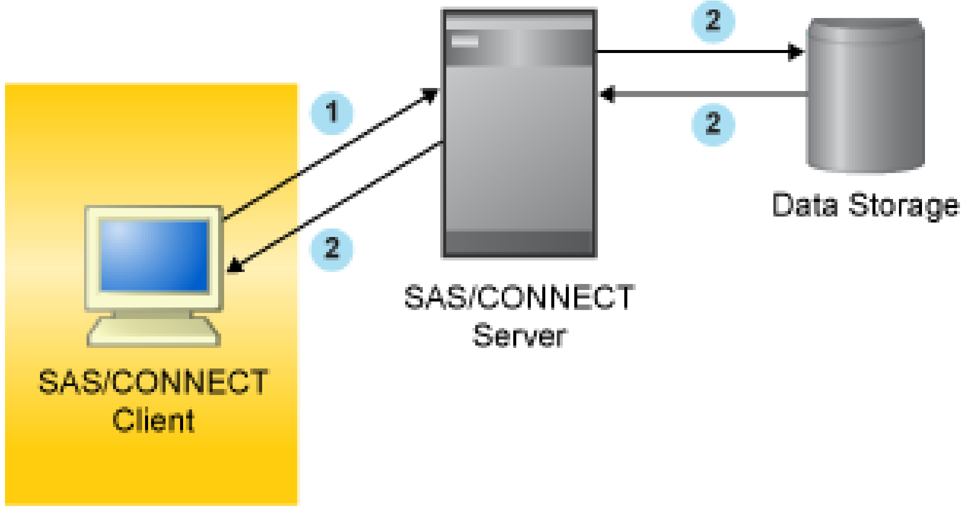
- customize data transfers
  - transfer multiple SAS files in a single step by using the INLIB= and OUTLIB= options. This capability enables you to transfer an entire library or selected members of a library in a single PROC UPLOAD or PROC DOWNLOAD step.
  - transfer collections of files (such as a partitioned data set, a MACLIB, or a directory) between a client and a server.
  - use WHERE processing for dynamic data subsetting and SAS data set options when transferring individual SAS data sets.
  - transfer catalog entries that contain graphics output by using a simple one-step process.
- protect data
  - increase the robustness of your decision support environment by keeping a local copy of your data, which is insulated from network failure.
  - back up local files to a server.
- manage data distribution
  - automate both data or application distribution and centralized data collection.

- □ distribute files from one workstation by uploading to a server and downloading to other workstations that need the files.
- □ move SAS files between releases of SAS as well as across operating environments.

Remote Library Services

Remote Library Services (RLS) provides transparent access to SAS data that is located on a remote computer. The data resides in server libraries, and RLS moves the data through the network as client processing requests it. The data must again pass through the network on any subsequent use by the client session. As the following figure shows, a copy of the data is not written to the client file system.

Figure 1.4 Model of RLS Processing



❶ The SAS/CONNECT client session requests records from the SAS/CONNECT server session or the client requests that records be written to the server.

❷ Data records are written to the SAS/CONNECT server session or are sent to the SAS/CONNECT client session for processing.

The SAS procedures and DATA steps that run in the SAS/CONNECT client session request access via the REMOTE engine to SAS files that are located on a SAS/CONNECT server. The REMOTE engine communicates the requests for data to the server. The server administers the requests to access SAS files on behalf of the client.

RLS provides the following:

- □ transparent access to SAS data that is located on a remote computer
- □ access to current SAS data because no client 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 SAS data that is located on a remote computer

Administering Logging for SAS/CONNECT

About the SAS Logging Facility

The SAS/CONNECT server and the SAS/CONNECT spawner use the SAS logging facility as the standard debugging tool in a SAS Foundation environment and in a SAS Intelligence Platform deployment. To make the logging facility functional, you must configure its properties in a logging configuration file. After you configure the file, you can easily enable the logging facility by specifying the -LOGCONFIGLOC system option in the SAS invocation.

Here are the primary components that are defined in the configuration file:

Loggers
: specify the objects that are used to create log events for a specific aspect of an application. A predefined set of loggers corresponds to the supported components such as Root, Audit, Admin, App, IOM, and Perf.

Appenders
: specify the output destinations for the log events. Examples include the FileAppender, RollingFileAppender, DBAppender, and ARMAppender. A level of severity is also associated with the log event. Examples are trace, debug, info, warn, error, and fatal.

Pattern Layouts
: specify the formats of the error messages that are associated with the log event.

For complete details about the component of the SAS logging facility, see *SAS Logging: Configuration and Programming Reference*.

Logging Configuration File

Here is a typical configuration file that defines the logging components:

Example Code 1.1 Typical Log Configuration File for SAS/CONNECT

```
<?xml version="1.0"  ?>
<log4sas:configuration xmlns:log4sas="http://www.sas.com/rnd/Log4SAS/" debug="true"> 1
     <appender name="LOG" class="FileAppender" > 2
     <param name="File" value="c:\v9\spawner.log" />
     <layout>
        <param name="ConversionPattern" value="%d %-5p [%t] %c (%F:%L) - %m" /> 3
     </layout>
     <param name="threshold" value="all" />
     </appender>
 <root> 4
    <appender-ref ref="LOG" />
    <level value="all" />
 </root>
</log4sas:configuration>
```

1 DEBUG="TRUE" indicates that debugging is enabled.

2 CLASS="FileAppender" indicates that the log events are written to the file path c:\v9\spawner.log.

3 The ConversionPattern parameter specifies a pattern layout that formats log messages. It identifies the type of data, the order of the data, and the format of the data that is generated in a log event and is delivered as output. In this example, the date and time, the log level, the thread ID, and the logger constitute the log event.

4 The root logger controls the entire SAS log event and is at the highest level in the logger hierarchy. If any other loggers are included in the logging configuration file, they are located beneath the ROOT logger in the hierarchy. All other loggers inherit the specified appender and threshold value of the root logger.

Invocation of the Logging Facility

The SAS logging facility is started in a SAS invocation. Here is a Windows example:

```
sas -logconfigloc winlog.xml
```

The -LOGCONFIGLOC option is used to specify the location of the logging configuration file named winlog.xml, which is used to initialize the SAS logging facility. The file specification that defines the location of the logging configuration file must be a valid filename or a path and filename for your operating environment.

Triggers for Log Events

Log events are triggered for SAS/CONNECT under these circumstances:

- server sign-on via the SIGNON statement and the SAS/CONNECT spawner invocation
- the beginning of the RSUBMIT statement and the occurrence of the ENDRSUBMIT statement
- server sign-off via the SIGNOFF statement and the SAS/CONNECT spawner termination

Note: SAS/CONNECT sign-on to and sign-off from a grid session is also supported. For details, see *Grid Computing in SAS*. △

Performance (such as response time, throughput, and availability) can also be measured for SAS transactions such as a DATA step or a SAS procedure in a SAS/CONNECT application by using the product SAS Application Response Measurement (ARM). To enable ARM, you would insert ARM macros into the SAS/CONNECT application. For details about implementing ARM in a SAS/CONNECT application, see *SAS Interface to Application Response Measurement (ARM): Reference* and *SAS Logging: Configuration and Programming Reference*.

Example of a Log Event

The data and the format of the log event are defined in the conversion pattern that is specified in the configuration file. Here is an example of a log event:

```
  2008-06-25-10:24:22,234; WARN; 3; Appender.File; (yn14.sas.c:149);
Numeric maximum was larger than 8, am setting to 8.
```

Accessibility Features in SAS Products

For information about accessibility for any of the products mentioned in this book, see the documentation for that product. If you have questions or concerns about the accessibility of SAS products, send e-mail to accessibility@sas.com.