THE POWER TO KNOW.

SAS® 9.3 Drivers for ODBC

User's Guide

Second Edition

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® 9.3 Drivers for ODBC: User's Guide, Second Edition*. Cary, NC: SAS Institute Inc.

SAS® 9.3 Drivers for ODBC: User's Guide, Second Edition

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, December 2011

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents

What's New in SAS Drivers for ODBC 9.3

Overview

In the first maintenance release for SAS 9.3, SAS provides the following enhancements to the SAS Drivers for ODBC:

- new 64-bit driver
- Userid/Password Override and SAS/SHARE server password support
- simplified syntax for specifying TCP/IP communication for local servers and SAS/SHARE servers
- location of SPD Server libraries is changed
- disable _0 override parsing option enabled by default
- new DQUOTE=ANSI SQL option

New 64-Bit Driver

SAS provides a 64-bit version of the driver so that 64-bit ODBC-compliant applications can use the driver in native mode. The 32-bit and 64-bit drivers are installed at the same time for Windows x64 operating environments.

The SAS 9.3 Drivers for ODBC are available for download from **http://support.sas.com/demosdownloads/setupcat.jsp?cat=ODBC+Drivers**.

The 64-bit driver does not support connections to the SAS Scalable Performance Data (SPD) Server.

Userid/Password Override and SAS/SHARE Server Password Support

In SAS 9.2 and the initial release of SAS 9.3, the SAS Drivers for ODBC supported using either the SAS/SHARE server password or the Userid/Password Override feature. In this release, the SAS Drivers for ODBC support using both features at the same time.

Simplified Syntax for TCP/IP Communication

In previous releases of the SAS Drivers for ODBC, it was necessary to edit the TCP/IP services file for the client machine with a service name. The service name was used to define the server name, and the driver used the service name to look up the TCP/IP port number to use for communication with the SAS server. In the SAS 9.3 release, a simplified syntax of two underscores and the port number are used for both local and SAS/SHARE server access. This enhancement avoids the need to edit the TCP/IP services file. However, the legacy behavior remains in the driver for sites that already have server names defined in the TCP/IP services file.

Location of the SPD Server Libraries

In previous releases of the SAS Drivers for ODBC, when a connection was made to an SPD Server, the driver searched for the spds.dll library file in a shared files location. With this release of the SAS Drivers for ODBC, the driver searches for the spds.dll file in the driver installation directory. For more information, see "Setting Up a Connection to SPD Server" on page 20.

Disable _0 Override Parsing Option Change

The Disable _0 Override Parsing option was not enabled in previous version of the SAS Drivers for ODBC and caused the driver to strip off a trailing _0 from variable names when the variable name is longer than eight bytes. Enabling this option can be helpful when accessing SAS Version 6 servers or version 6 data sets, but often causes errors with newer SAS versions. In this release, the option is enabled by default to avoid parsing errors with SAS servers at version 7 and newer.

New DQUOTE=ANSI SQL Option

The DQUOTE=ANSI SQL option appears on the **General** tab of the SAS ODBC Driver Configuration dialog box. When this option is enabled, values that are enclosed in double quotation marks are treated as variables by the SAS server that runs the query, rather than as a string. This option enables you to use DBMS names and other names that are not normally permissible in SAS as table names, column names, and aliases.

Recommended Reading

- *SAS/ACCESS for Relational Databases: Reference*
- *SAS Companion for Windows*
- *SAS Language Reference: Concepts*
- *SAS/SHARE User's Guide*

For a complete list of SAS publications, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

Chapter 1

SAS Drivers for ODBC

Overview: SAS Drivers for ODBC

This document describes the SAS 9.3 Drivers for ODBC. If you are using another version of the driver, refer to **http://support.sas.com/documentation/onlinedoc/odbc/index.html** for the version-specific documentation.

This document is intended for three audiences:

- users who use the SAS Drivers for ODBC to access data that is stored on their own computers[1]
- system administrators who use the SAS Drivers for ODBC to enable multiple users to access shared data on a remote machine
- application programmers and others who need information about how the SAS Drivers for ODBC is implemented

1 For more information about what types of data you can access, see "Types of Data Accessed with the SAS Drivers for ODBC" on page 4.

This document assumes that you are familiar with your Windows operating environment and that you know how to use a mouse and keyboard to perform common Windows tasks.

What Is ODBC?

ODBC stands for Open Database Connectivity. ODBC is an interface standard that provides a common application programming interface (API) to access databases. Most software that runs in the Windows operating environment follows this standard for data access so that you can access data created by a variety of software applications.

ODBC functionality is provided by three main components:

- the client application
- the ODBC Driver Manager
- the ODBC driver

The following figure displays the components of ODBC functionality. The ODBC Driver Manager, which was developed by Microsoft, manages the interaction between a client application and one or more ODBC drivers.

ODBC Version and Conformance

The SAS Drivers for ODBC are ODBC 3.5 version drivers. In addition to an ODBC version value, the Microsoft ODBC specification defines three levels of conformance for ODBC functions: Core, Level 1, and Level 2. The drivers conform to the Core interface level. However, there are some functions in the Core interface level that the drivers do not implement, or implement in a way that is different from the way a DBMS driver might implement the same function. The functions, and the behavior, are identified in "Support for and Implementation of ODBC Functions" on page 38.

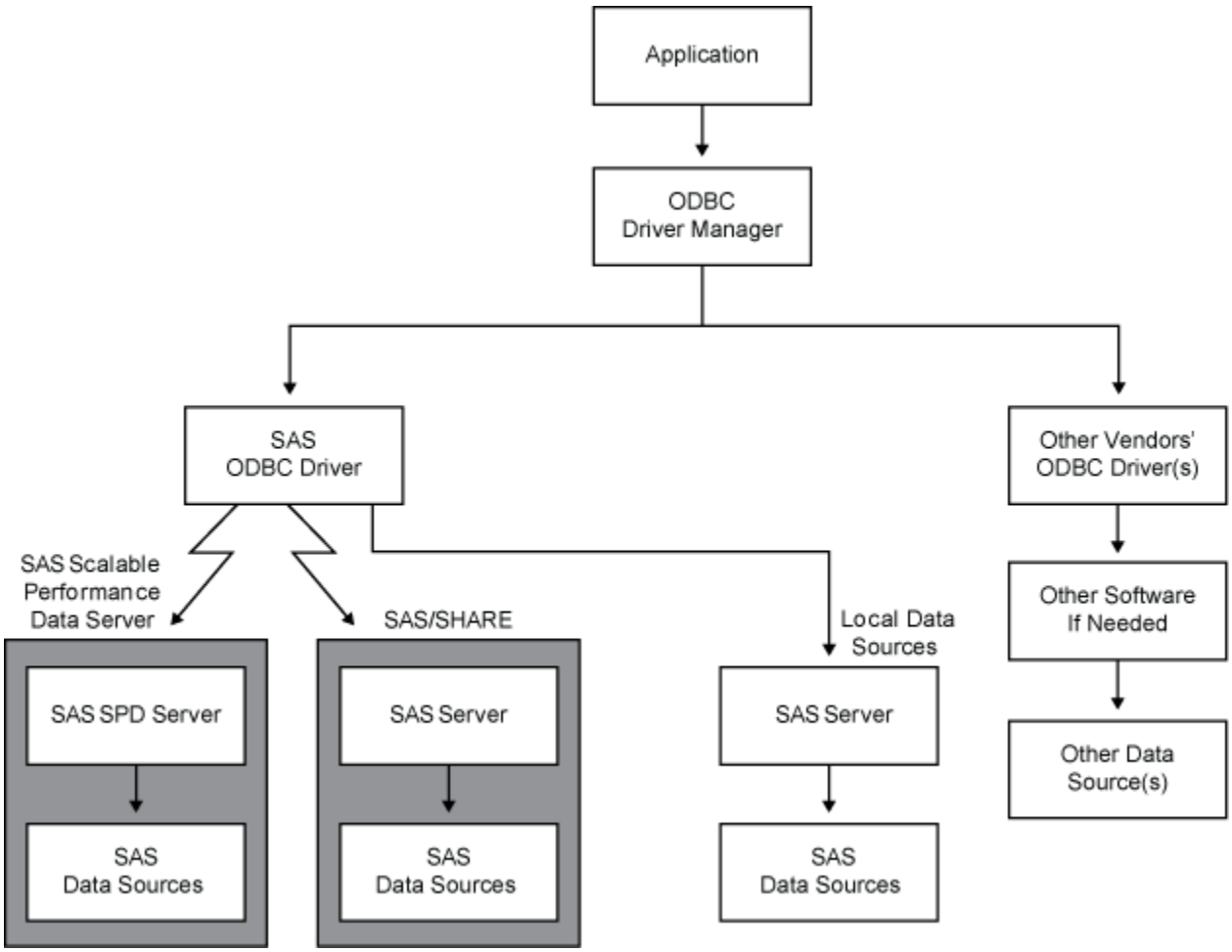
What Are the SAS Drivers for ODBC?

The SAS Drivers for ODBC are implementations of the ODBC interface standard that enable you to access, manipulate, and update SAS data sources from applications that are ODBC compliant. SAS provides a 64-bit driver and a 32-bit driver.

As the following figure shows, a driver uses a SAS server to access data from a SAS data source. The SAS server can be a local and minimized SAS session, a remote SAS/SHARE server, or a SAS Scalable Performance Data Server. (For more information, see "SAS Servers" on page 6.) If you use other ODBC drivers (such as an ODBC driver for Oracle or SQL Server) to access other data sources, those ODBC drivers might require additional software components.

Note: To access ODBC data sources from SAS (the opposite of what the SAS Drivers for ODBC enable you to accomplish), you must license SAS/ACCESS Interface to ODBC software. For more information, see *SAS/ACCESS for Relational Databases: Reference*.

Figure 1.1 *Components of ODBC Functionality*



Information about the 64-bit Driver

In 64-bit operating environments, both the 64-bit driver and the 32-bit driver are installed. The default installation locations are shown in the following table:

Table 1.1 *Default Installation Locations in 64-bit Operating Environments*

Driver Version	Pathname
64-bit	`C:\Program Files\SASHome\SASDriversforODBC\9.3\`
32-bit	`C:\Program Files (x86)\SASHome\SASDriversforODBC\9.3\`

Be aware that some 32-bit ODBC applications can run in compatibility mode in 64-bit operating environments. In this case, the 32-bit ODBC application needs to use the 32-bit driver, and you need to define each data source name (DSN) with the 32-bit ODBC

Data Source Administrator. In a 64-bit operating environment, the 32-bit ODBC Data Source Administrator is started from `C:\Windows\SysWOW64\odbcad32.exe`.

In addition, when either driver is used to access local data, the driver attempts to start SAS, and then connect to the SAS session. The 64-bit driver can start 64-bit SAS only, and the 32-bit driver can start 32-bit SAS only. If you must use the 64-bit driver to connect to a local 32-bit version of SAS, then start SAS in server mode manually before using the client application. For more information, see "Starting a SAS ODBC Server" on page 32.

The 64-bit driver does not support connections to SPD Server.

Types of Data Accessed with the SAS Drivers for ODBC

In the previous figure and throughout this document, the term SAS data sources is used to describe data sources that you have defined in the ODBC Data Source Administrator. These can include SAS data sets, flat files, and VSAM files, as well as data from many database management systems (DBMSs) through the use of SAS/ACCESS software. (For more information, see "SAS Data Sets" on page 5.)

If your PC is connected to a TCP/IP network, you can access both local data sources and remote data sources. (For more information about software requirements, see "What Software Do I Need?" on page 8.) Local data is data that you access through a SAS server on your local machine. The data can be stored either on your computer's hard drive or on a network file system, such as a NetWare Windows NT file server, or a Windows shared folder, that makes the physical location of the data transparent to applications. Remote data is data that you access through a SAS/SHARE server or SPD Server that runs on another (remote) machine.

The ability to use the SAS Drivers for ODBC with SAS/ACCESS software as a gateway to DBMS data is particularly useful in the following situations:

- When the DBMS vendor does not offer an ODBC driver. In this situation, you can use the SAS Drivers for ODBC with SAS/ACCESS software on the SAS server to connect to the DBMS. SAS/ACCESS software to the DBMS must be licensed.
- You do not have a software license for the necessary software (either an ODBC driver for the DBMS or DBMS network access software) on your client machine.
- You want to merge DBMS data with other data.

Currently, SAS/ACCESS software is available for the following systems:

- ADABAS
- CA-IDMS
- DB2 under UNIX, Windows, and z/OS
- HP Neoview
- IMS
- Informix
- SQL Server
- MySQL
- Netezza

- ODBC
- OLE-DB
- Oracle
- PC Files
- Sybase
- SYSTEM 2000
- Teradata

For more information about the individual SAS/ACCESS interfaces, see *SAS/ACCESS for Relational Databases: Reference*.

Understanding SAS

SAS Data Sets

A SAS data set is a file structured in a format that SAS can access. The physical object contains the following elements:

- data values that are stored in tabular form
- a descriptor portion that defines the types of data to SAS

The physical locations of the data values and the descriptor are not necessarily contiguous.

SAS data sets have two forms: data files and data views. SAS data files are relational tables with columns (or variables) and rows (or observations). The SAS data file structure can have many of the characteristics of a DBMS, including indexing, compression, and password protection.

SAS data views are definitions or descriptions of data that resides elsewhere. SAS data views enable you to use SAS to access many different data sources, including flat files, VSAM files, and DBMS structures, as well as native SAS data files. A data view eliminates the need to know anything about the structure of the data or the software that created it. Data views need very little storage because they contain no data. They access the most current data from their defined sources because they collect the actual data values only when they are called.

You can use data views to define subsets of larger structures, or supersets of data that have been enhanced with calculated values. You can create SAS data views that combine views of dissimilar data sources. For example, you can combine a view of a relational DB2 table with a view of a SAS data file, a view of hierarchical IMS data, or even a view from a PC-based dBASE file.

You create SAS data views in three ways:

- with the DATA step (DATA step views)
- with the SQL procedure (PROC SQL views)
- with SAS/ACCESS software (SAS/ACCESS views)

You can use SAS/ACCESS software to work directly with DBMS tables, such as DB2 and Oracle, as if they were SAS data sets and data views by using the SAS/ACCESS LIBNAME statement. For more information, see *SAS Language Reference: Concepts*.

These data views have some variation:

DATA step views
: describe data from one or more sources, including flat files, VSAM files, and SAS data sets (either data files or other data views). You cannot use a DATA step view to update the view's underlying data because DATA step views only read other files.

  For more information about how to create and use DATA step views, see "DATA Step Views," in *SAS Language Reference: Concepts*.

PROC SQL views
: define either a subset or a superset of data from one or more SAS data sets. These data sets can be data files or data views, and can include data sets composed of DBMS data that are created with the SAS/ACCESS LIBNAME statement, and data views that are created with the SQL pass-through facility to access DBMS data. You can also create PROC SQL views of DBMS data by using an embedded LIBNAME statement.

  For example, an SQL procedure can combine data from PROC SQL views, DATA step views, and SAS/ACCESS views with data in a SAS data file. You cannot use a PROC SQL view to update the view's underlying files or tables. However, with some restrictions, you can use the UPDATE, DELETE, and INSERT statements in the SQL procedure to update data that is described by SAS/ACCESS views.

  For more information about the SQL pass-through facility, see "Overview: SQL Procedure," in *Base SAS Procedures Guide*.

Unicode UTF-8 Support

The SAS Drivers for ODBC support data sets with UTF-8 encoding.

SAS Libraries

SAS data sets are contained in libraries. Each SAS library has two names: a physical name and a logical name (libref). The physical name of the library fully identifies the directory or operating system data structure that contains the data sets. Therefore, the physical name must conform to the rules for naming files within your operating system.

You use the libref to identify a group of data sets (files or views) within SAS. The libref is a temporary name that you associate with the physical name of the SAS library during each SAS job or session. After the libref is assigned, you can read, create, or update files in the library. A libref is valid only for the current SAS job or session and you can reference it repeatedly within that job or session.

You can use SAS/ACCESS software to associate a SAS libref with a DBMS database, schema, server, or group of tables and views, such as a DB2 database or group of Oracle tables and views. For more information about using SAS/ACCESS software, see *SAS/ACCESS for Relational Databases: Reference*.

For more information about SAS libraries, see "SAS Libraries," in *SAS Language Reference: Concepts*.

SAS Servers

To access your SAS data sources, the SAS Drivers for ODBC use a SAS server. A SAS server is a SAS procedure (either PROC SERVER or PROC ODBCSERV) that runs in its own SAS session. It accepts input and output requests from other SAS sessions and

from the driver on behalf of the applications that are ODBC compliant. While the server is running, the SAS session does not accept input from the keyboard.

The type of server that the driver uses depends on whether you are accessing local data or remote data:

local data
: The driver uses a SAS ODBC server to access your data. If you do not already have a SAS session running on your computer, the driver starts a SAS session and executes PROC ODBCSERV, which automatically starts the SAS ODBC server when you connect to your local data source. For more information, see "Understanding Access to Local Data Sources" on page 32. If you have a SAS session running on your computer (but not a SAS ODBC server), then you must either start the SAS ODBC server manually or end the SAS session before you connect to your local data sources.

remote data
: The driver uses a SAS/SHARE server or a SAS Scalable Performance Data (SPD) Server to access remote data. SAS/SHARE software or SPD Server must be licensed on the remote host. The driver requires TCP/IP software that is included with your operating system to communicate with either type of SAS server. For a SAS/SHARE server, your server administrator uses PROC SERVER to start the server on the remote host. For more information, see "Understanding Access to SAS/SHARE Data Sources" on page 33.

The SAS Drivers for ODBC can communicate with SAS/SHARE servers and SPD Servers. You can interchange SAS data and SPD Server data by using the LIBNAME statement engine option in either SAS or SPD Server.

SAS Scalable Performance Data (SPD) Server

The SPD Server uses the latest parallel processing methods and data server capabilities to efficiently access large volumes of data and to serve large numbers of concurrent users. The SPD Server provides efficient data access for hundreds of users across multiple processors. The driver can be configured for a direct connection to an SPD Server. ODBC connections to SPD SNET are not supported with SAS 9 and later versions of the SAS Drivers for ODBC.

The SPD Server allows access to SAS data for intensive processing (queries and sorts) on the host server machine. It organizes and processes SAS data to take advantage of parallel processors on specific host servers.

You must have the SPD Server licensed on your client machine. Then, you can interchange SAS data and SPD Server data by using the LIBNAME statement engine option either in SAS or on the SPD Server. For more information, see *SAS Scalable Performance Data Server: User's Guide*.

SAS Terminology

Software products often include similar components or constructs that are known by different names. For the ODBC standard and SAS, the following correspondences exist:

ODBC term	SAS Term
owner	library name (libref)
table	data set

ODBC term	SAS Term
qualifier	SAS data does not use a qualifier

Therefore, if your application that is ODBC compliant asks you to specify the owner for a SAS library, you should specify the libref. If the application asks for a table name, you should specify the name of the SAS data set. If a qualifier is requested, you can usually leave the field blank.

What Software Do I Need?

The SAS Drivers for ODBC run on 32-bit and 64-bit versions of Windows XP, Windows 2003, Windows 2008, and Windows Vista. If the operating system is a server form of Microsoft Windows such as Windows Server 2003, then SAS/SHARE software must be licensed even if the data source is local data. Other software requirements depend on your hardware configuration and on the data sources that you want to access, as shown in the following table:

Table 1.2 *Software Requirements*

Data Source	Configuration	Requirements
local SAS data only	Stand-alone PC	Base SAS software*
local SAS data and local DBMS data	Stand-alone PC	Base SAS software* SAS/ACCESS interface for each DBMS
remote SAS data	PC on a network	On the client PC, the SAS Drivers for ODBC is required.** On the remote host: • Base SAS software • SAS/SHARE software
remote SAS data and remote DBMS data	PC on a network	On the client PC, the SAS Drivers for ODBC is required.** On the remote host: • Base SAS software • SAS/SHARE software • SAS/ACCESS interface for each DBMS

* For local data access, Base SAS software includes the SAS Drivers for ODBC, but you install the driver separately. If the operating system is a server form, such as Windows 2003 Server, then SAS/SHARE software must be licensed for accessing local data.

** For remote data access, install the SAS Drivers for ODBC on your client PC. The SAS license for the remote host must include SAS/SHARE software.

Accessibility Features in the SAS Drivers for ODBC

The SAS ODBC Drivers for ODBC include accessibility and compatibility features that improve usability of the product for users with disabilities. These features are related to accessibility standards for electronic information technology adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. If you have questions or concerns about the accessibility of SAS products, send e-mail to accessibility@sas.com.

Chapter 2

Defining Your Data Sources

Introduction to Defining Data Sources

After you install the SAS Drivers for ODBC, you must provide information about the data sources that you want to access. This chapter explains how to provide the information.

From the SAS ODBC Driver Configuration dialog boxes, you can select **Help** at any time to obtain information about the active dialog box.

Accessing the SAS ODBC Driver Configuration Dialog Boxes

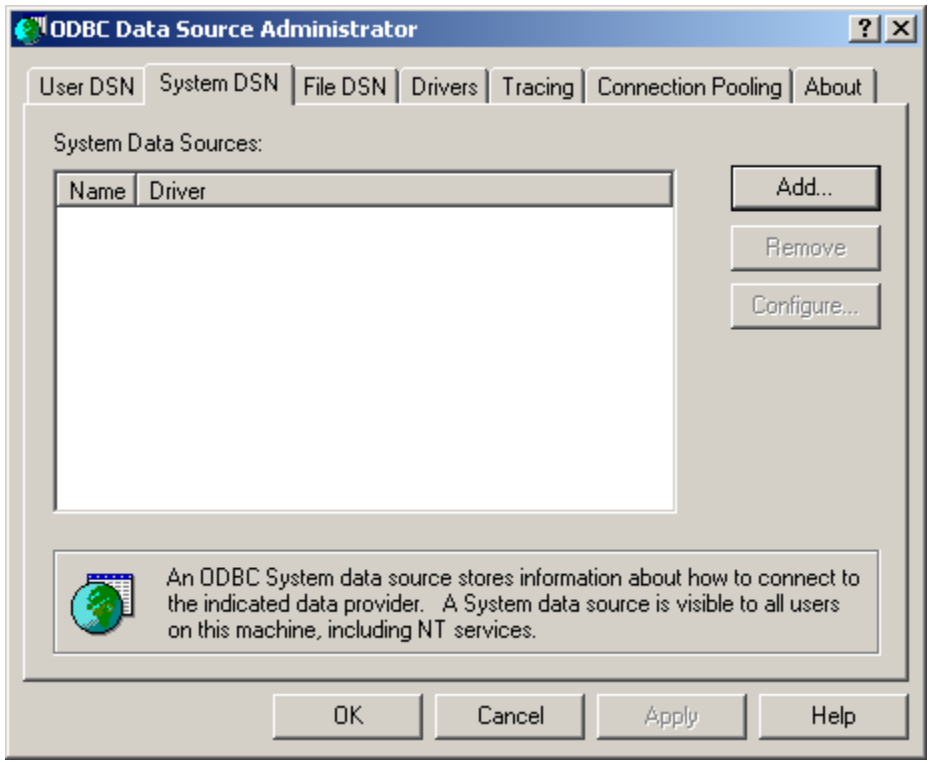
To access the SAS ODBC Driver Configuration dialog boxes, complete these steps:

1. Access the Windows Control Panel by selecting **Start** ⇨ **Settings** ⇨ **Control Panel**.

   Find the ODBC Data Sources or ODBC Administrator icon. This icon might be located in the Control Panel group, an ODBC group, or in the Administrative Tools group. If you have installed a package of other ODBC drivers, this icon might be in a group that is associated with that package.

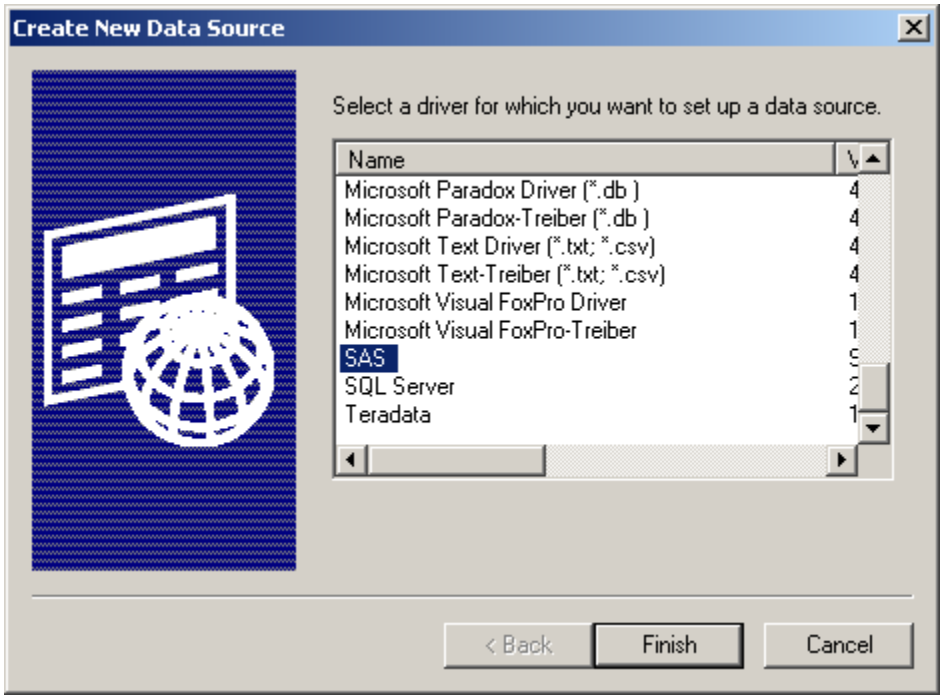
2. Double-click the icon. Depending on the version of ODBC Administrator that you have installed, you might see a slight variation in the following dialog box:

   Display 2.1 *ODBC Data Source Administrator Dialog Box*



3. To add a new data source, select either the **User DSN** tab or the **System DSN** tab, and click **Add**. The Create New Data Source dialog box appears.

Display 2.2 *Create New Data Source Dialog Box*

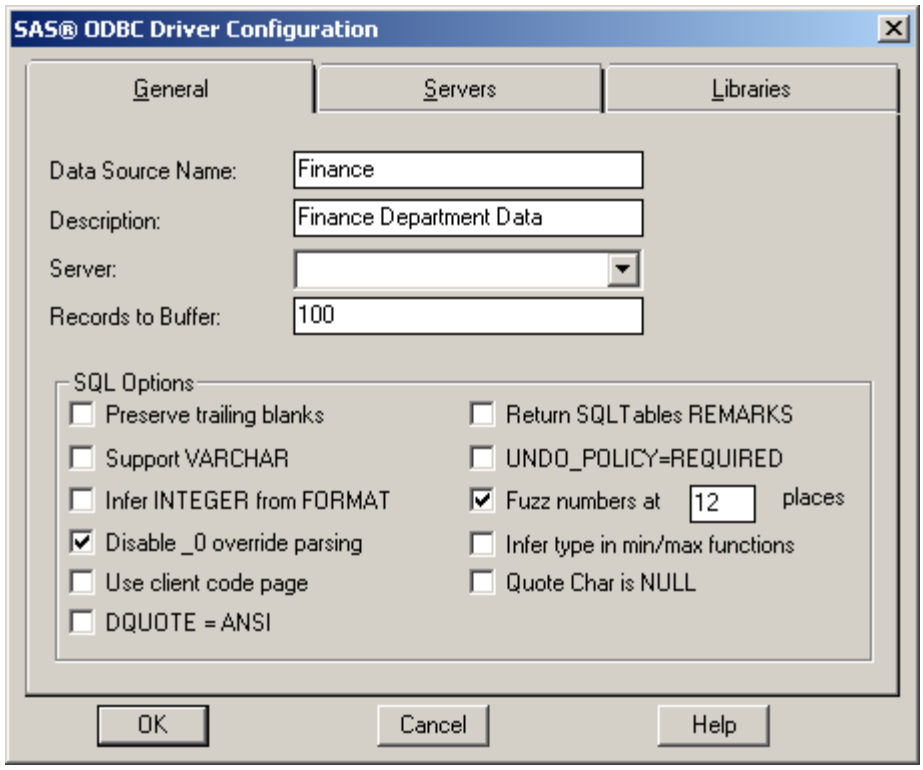


4. Scroll down the list of drivers (if necessary) and select **SAS**. Click **Finish**.

   Note: Select **SAS** even if you are going to use a SAS data view to access data that is not SAS.

The SAS ODBC Driver Configuration dialog box appears.

Display 2.3 *SAS ODBC Driver Configuration Dialog Box*



At the top of this dialog box are three tabs: **General**, **Servers**, and **Libraries**. On each tab, provide information about the data source that you want to access.

For more information about connecting to data sources, see the following topics:

- “Setting Up a Connection to Local Data” on page 14
- “Setting Up a Connection to SAS/SHARE Server” on page 17
- “Setting Up a Connection to SPD Server” on page 20

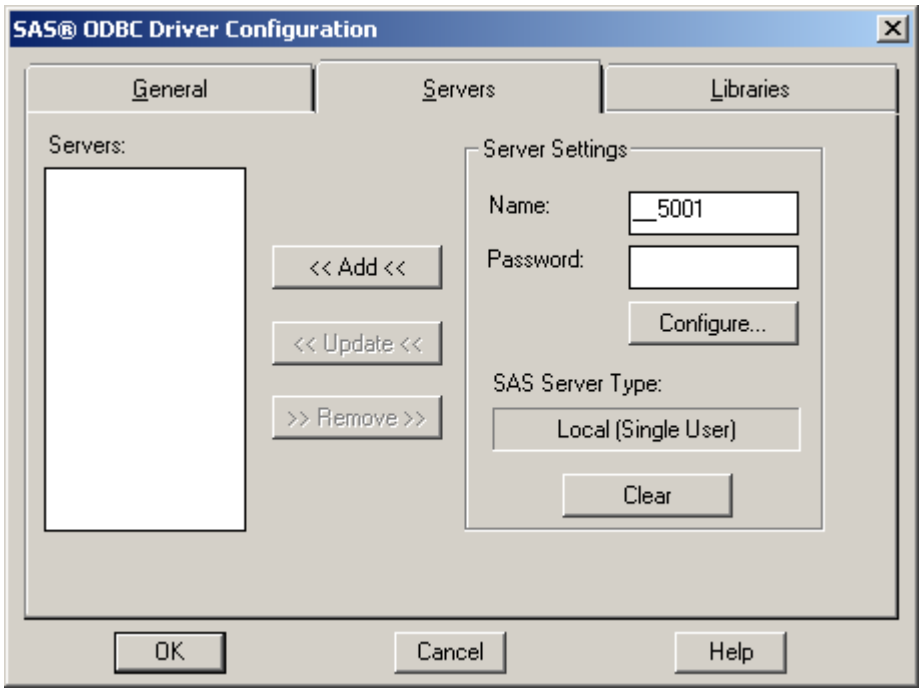
Setting Up a Connection to Local Data

Use the instructions in this section to create a Data Source Name (DSN) for accessing data on the local workstation. When you access local data from an ODBC-compliant application, the driver starts a SAS ODBC server, unless a SAS ODBC server is already running. The driver then connects to the SAS ODBC server and provides access to the libraries that are associated with the DSN.

Define the Server

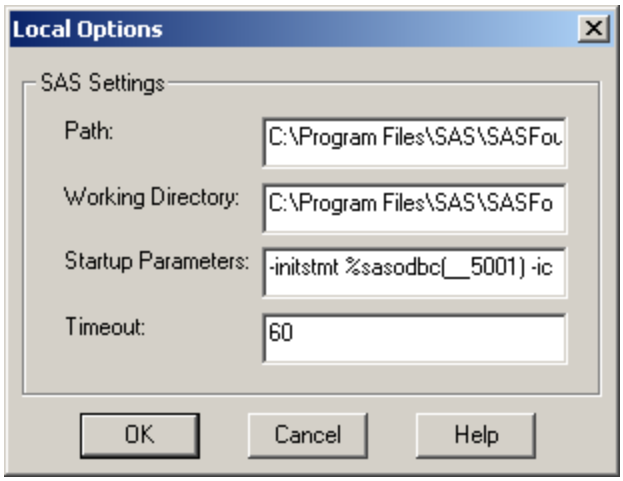
1. Access the SAS ODBC Driver Configuration dialog box.
2. Click the **Servers** tab.

In the **Name** field, enter two underscores and the port number that you want the SAS ODBC server to use. For example, `__5001`. The **SAS Server Type** field indicates `Local (Single User)`.



3. Click **Configure**.

The Local Options dialog box appears.



Each field in the Local Options dialog box contains default values that you can change by typing over them.

Path
specifies the fully qualified pathname for the SAS executable file (SAS.EXE) that you use to start a SAS session. The default path is `C:\Program Files\SASHome\SASFoundation\9.3\sas.exe`.[1]. If this field is left blank, then no attempt is made to start a SAS ODBC server when you connect to your data source.

1 The driver queries the registry for the current version of SAS and the value of DefaultRoot. This value is concatenated with **\sas.exe** to create the default path. The default working directory is the value of DefaultRoot.

Working Directory
specifies the fully qualified pathname for the directory that you want to use as the SAS working directory. This directory is usually where your SAS program files and documents are located. The default path is **`C:\Program Files\SASHome\SASFoundation\9.3`**.

Startup Parameters
specifies the parameters that are used to invoke SAS. The default values are initialization statement (**`-initstmt`**), which executes a SAS macro (**`%SASODBC`**), which invokes the SAS ODBC server. The **`__5001`** value is only an example. It is a SAS macro parameter whose value is taken from the name that you specified in the **Name** field of the **Servers** tab. The value can be either a service name, such as **`srv1`**, or it can be a port number prefixed with two underscores, such as **`__5001`**. The value cannot be just a port number, such as **`5001`**. The **`-icon`** option specifies that the SAS ODBC server should be invoked as a minimized SAS session because no interaction with the server is required. The **`-nosplash`** option (not shown) specifies that the SAS session is invoked without displaying the SAS logo and copyright information.

The **`%SASODBC`** macro is provided with SAS and is found in **`!SASROOT\core\sasmacros\sasodbc.sas`**[1]. The sasodbc.sas file executes PROC ODBCSERV.

The sasodbc.sas file can be modified to add SAS options or SAS statements, such as the LIBNAME statements mentioned in "Defining Libraries at Server Start-Up Time" on page 28. In addition, you can add options to PROC ODBCSERV. The available options are the same as those for PROC SERVER. For more information, see *SAS/SHARE User's Guide*.

An additional option, **`LOG=QUERY`**, is relevant for servers that are used by the driver. This option causes the server to log SQL queries. (By default, the server logs update and output operations, but not queries.) This option is useful when you need to see the queries that the server receives from an ODBC client application.

If your SAS session is installed on a network drive and is shared by multiple users, then you probably do not want individual users to modify the sasodbc.sas file. Instead, a user can make a copy of the file and store it in a personal library. In this case, the user must add the **`-sasautos`** option either to the **Startup Parameters** field or to the local config.sas file to indicate the pathname for the library, as in the following example: **`-sasautos c:\programs\sas`**

For more information about SAS system options and SAS statements, see *SAS Companion for Windows*.

Timeout
specifies, in seconds, how long to wait for the SAS ODBC server to start and to register itself. The default is 60 seconds.

4. Click **OK** to return to the **Servers** tab.

   Important! Click **Add** to save the server definition.

Define the Library

5. Click the **Libraries** tab. Define a library for each data library that you want to access with this DSN.

1 !SASROOT is a logical name for the directory in which you install SAS. For more information, see *SAS Companion for Windows*.

Name
enter a name for an existing physical SAS library that you want to access. (If you are familiar with SAS, this field corresponds to the libref in the SAS LIBNAME statement.) The name can be up to eight characters. The first character must be a letter or an underscore. Subsequent characters can be letters, numeric digits, or underscores. Blank spaces and special characters are not allowed. For example, you might use the name **`cost`** to designate a library of cost accounting data. The SAS library can include SAS data files, SAS data views, or both.

Note: If you use an ODBC application that exports databases using one-level names, then you need to define a library called **`user`**.

Host File
enter the physical name of the library. This must be a valid pathname for the local workstation. For example, **`c:\data\costacct`** and **`\\acctsrv\customers`** are valid pathnames.

Description
provide a description of the library to remind yourself or other users what the library contains. Providing this value is optional.

Engine
enter the name of the SAS engine that is required for writing to and reading from this library. This setting is necessary only if you do not want to use the **`V9`** engine that is the default for SAS 9.3. For information about other engines that might be available, see the description of the LIBNAME statement in the *SAS Companion for Windows*. Providing this value is optional.

Options
enter options for the library that you are defining, such as **`ACCESS=READONLY`**.

6. Click **Add** to save your library information. The library name is added to the list of libraries on the left.

Define the Data Source Name

7. Click the **General** tab.
8. Provide a name in the **Data Source Name** field. Use the **Server** menu to select the correct server for the DSN. For more information about SQL options, see "SQL Options on the General Tab" on page 24.

Setting Up a Connection to SAS/SHARE Server

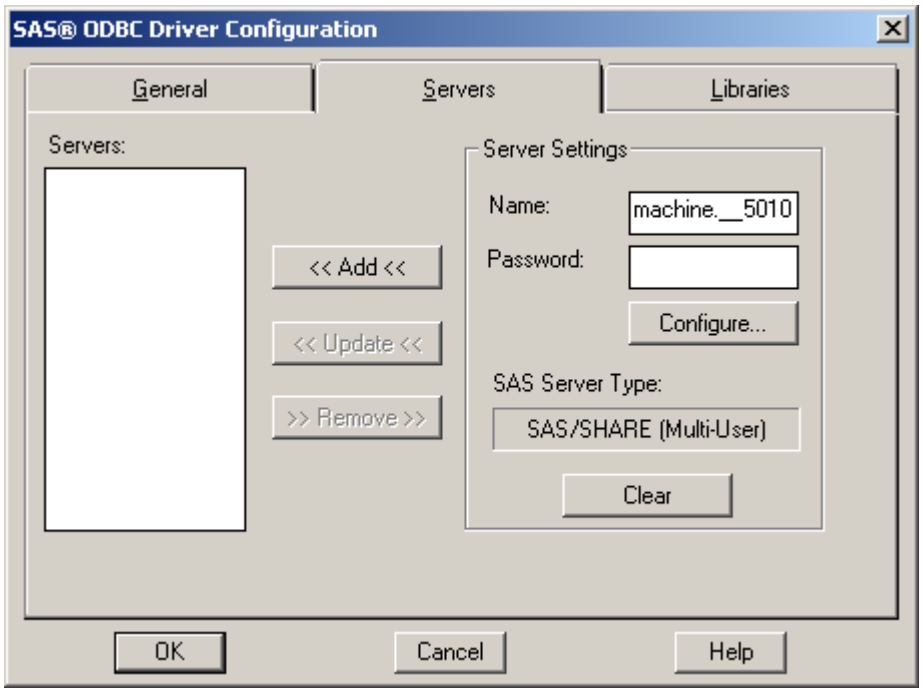
The SAS Drivers for ODBC use a TCP/IP network connection to communicate with a SAS/SHARE server. Use the instructions in this section to create a DSN for accessing data on a SAS/SHARE server.

Define the Server

1. Access the SAS ODBC Driver Configuration dialog box.
2. Click the **Servers** tab. In the **Name** field, enter a two-part name such as **`machine.__5010`**. The **SAS Server Type** field indicates **`SAS/SHARE (Multi-User)`**.

The driver interprets the first part of the name, **`machine`**, as the host name of the SAS/SHARE server and the second part, **`__5001`**, as the network port that the

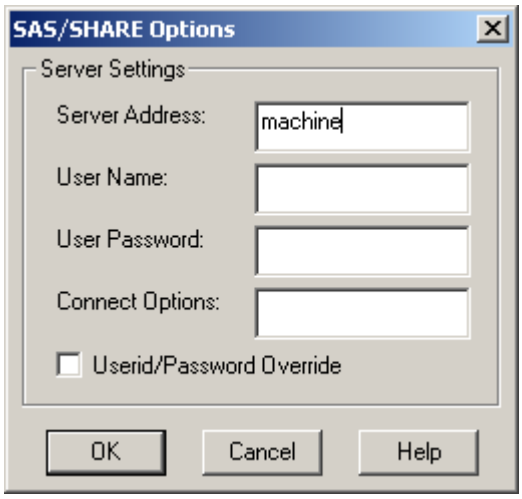
SAS/SHARE uses for TCP/IP communication. The second part of the name is two underscores and then the network port number. Contact your SAS/SHARE administrator for the network port number if you do not know it.



3. If the SAS/SHARE server is password protected, then enter the password in the **Password** field. The password should be the same password that was specified for the **`UAPW=`** option in PROC SERVER.
4. Click **Configure**.

The SAS/SHARE Options dialog box appears.

Display 2.4 *SAS/SHARE Options Dialog Box*



Provide the requested information:

Server Address
is automatically filled with the alias for the TCP/IP network machine name that you specified in the **`Name`** field of the **Servers** tab. In a complex networking environment, you might need to provide a fully qualified domain name address for the server (for example, **`machine.example.com`**).

User Name
is your user ID on the system where the server is running. This field is required if the server is running in secured mode. Otherwise, it is ignored.

User Password
is your password on the system where the server is running. If you provide a **User Name** without a **User Password**, then you are prompted for a password at connection time. The driver encrypts the password before storing the encrypted value in the Windows registry.

Connect Options
Leave this field blank for a connection to a SAS/SHARE server.

Userid/Password Override
requests that the UID keyword and PWD keyword be used in the ODBC client application. The driver passes the value of the PWD keyword as the user login password, and the value of the UID keyword as the user ID. For more information about using this option, see "Userid/Password Override" on page 33.

5. Click **OK** to return to the **Servers** tab.

   Important! Click **Add** to save the server definition.

Define the Libary

6. Click the **Libraries** tab.
7. Define a library for each data library that you want to access with this DSN. In addition to the libraries that you define, you have access to any libraries that are predefined on the SAS/SHARE server and that you have permission to access. Contact your SAS/SHARE administrator for information about libraries that are predefined on the SAS/SHARE server.

   Name
   enter a name for an existing physical SAS library that you want to access. (If you are familiar with SAS, this field corresponds to the libref in the SAS LIBNAME statement.) The name can be up to eight characters. The first character must be a letter or an underscore. Subsequent characters can be letters, numeric digits, or underscores. Blank spaces and special characters are not allowed. For example, you might use the name **`cost`** to designate a library of cost accounting data. The SAS library can include SAS data files, SAS data views, or both.

   For more information, see "Defining Libraries at Server Start-Up Time" on page 28.

   Note: If you use an ODBC application that exports databases using one-level names, then you need to define a library called **`user`**.

   Host File
   enter the physical name of the library. This must be a valid pathname for the machine that is hosting the SAS/SHARE server. For example, **`e:\data`** and **`\\acctsrv\customers`** are valid pathnames.

   Description
   provide a description of the library to remind yourself or other users what the library contains. Providing this value is optional.

   Engine
   enter the name of the SAS engine that is required for writing to and reading from this library. This setting is necessary only if you do not want to use the **`V9`** engine that is the default for SAS 9.3. For information about other engines that might be

available, see the description of the LIBNAME statement in the *SAS Companion for Windows*. Providing this value is optional.

Options
enter options for the library that you are defining, such as **`ACCESS=READONLY`**.

8. Click **Add** to save your library information. The library name is added to the list of libraries on the left.

Define the Data Source Name

9. Click the **General** tab.
10. Provide a name in the **Data Source Name** field. Use the **Server** menu to select the correct server for the DSN. For more information about SQL options, see "SQL Options on the General Tab" on page 24.

Setting Up a Connection to SPD Server

The SAS Drivers for ODBC use a TCP/IP network connection and a SPD Server library–spds.dll–to communicate with the SPD Server. Contact your SPD Server administrator to get the spds.dll file from the **`lib`** directory in the SPD Server distribution. Then, place the spds.dll file in **`C:\Program Files\SASHome\SASDriversforODBC\9.3`**.

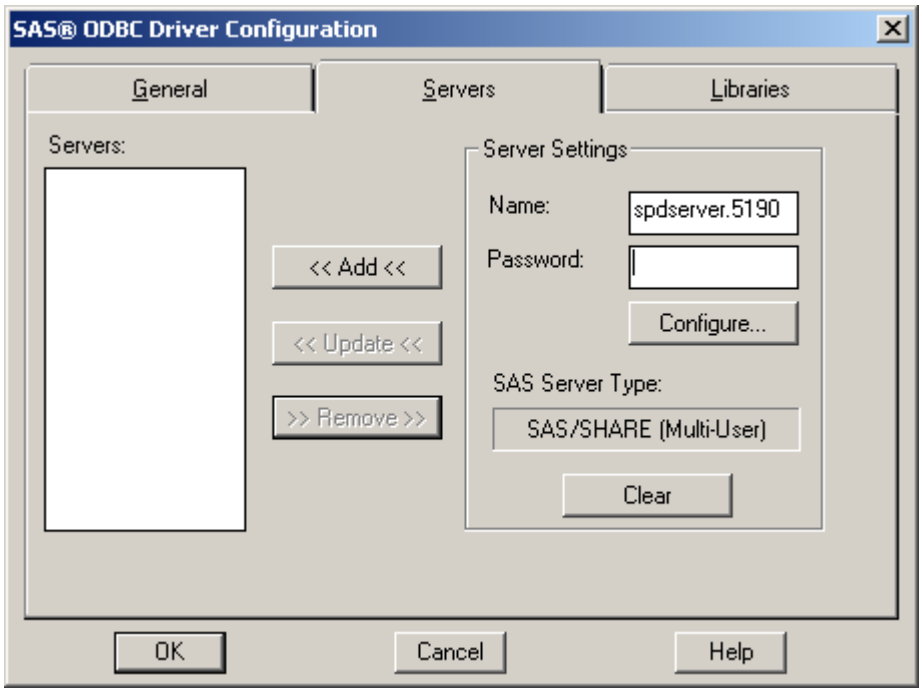
Note: The 64-bit driver does not support connections to SPD Server.

Use the instructions in this section to create a Data Source Name (DSN) for accessing data on an SPD Server.

Define the Server

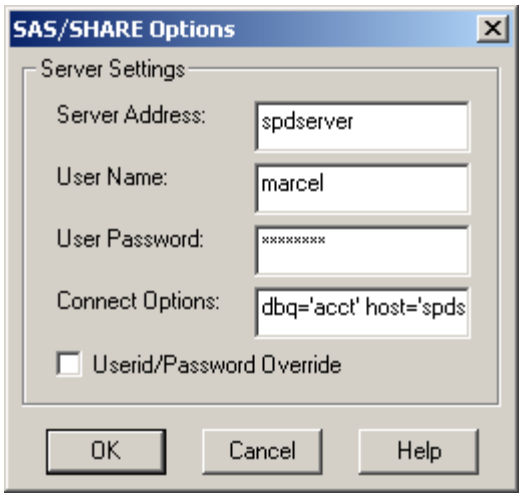
1. Access the SAS ODBC Driver Configuration dialog box.
2. Click the **Servers** tab. In the Name field, provide a two-part name such as **`spdserver.5190`**. The **SAS Server Type** field indicates **`SAS/SHARE (Multi-User)`** even though this is a connection to an SPD Server.

The first part of the name is the host name of the SPD Server. The second part of the name is the port number of the SPD Server Name Server.



3. Do not enter a value in the **Password** field. This field is not used for connections to SPD Server.
4. Click **Configure**.

The SAS/SHARE Options dialog box appears.



Provide the requested information:

Server Address
is automatically filled with the alias for the TCP/IP network machine name.

User Name
is your user ID for the SPD Server.

User Password
is your password on the SPD Server. If you provide a **`User Name`** without a **`User Password`**, then you are prompted for a password at connection time.

The driver encrypts the password before storing the encrypted value in the Windows registry.

Connect Options
provide the DBQ, HOST, and SERV parameters for the SPD Server. For example, **`DBQ='acct' HOST='spdserver.example.com' SERV='5190'`**.

`DBQ` is the SPD Server LIBNAME domain.

`HOST` is the name of the machine that is hosting the SPD Server.

`SERV` is the port number of the SPD Server Name Server.

Userid/Password Override
requests that the UID keyword and PWD keyword be used in the ODBC client application. The driver passes the value of the PWD keyword as the user login password, and the value of the UID keyword as the user ID. For more information about using this option, see "Userid/Password Override" on page 33.

5. Click **OK** to return to the **Servers** tab.

   Important! Click **Add** to save the server definition.

Define the Library

6. Click the **Libraries** tab.
7. Define a library for each data library that you want to access with this DSN.

   Note: Add a library for the DBQ= domain that you provided in the connect options. This library is called the primary LIBNAME domain. Also, add additional libraries that you want to access. These libraries are called secondary LIBNAME domains.

   Name
   enter the name of a LIBNAME domain that is defined on the SPD Server

   Host File
   enter the name of the LIBNAME domain again. This field is not used for a connection to SPD Server because the storage for the data is controlled by the SPD Server.

   Description
   provide a description of the library to remind yourself or other users what the library contains. Providing this value is optional.

   Engine
   enter **`spdseng`**.

   Options
   use the DBQ= option to identify the LIBNAME domain. For example, **`DBQ='acct'`**. Enclose the LIBNAME domain in single quotation marks.

Define the Data Source Name

8. Click the **General** tab.
9. Provide a name in the **Data Source Name** field. Use the **Server** menu to select the correct server for the DSN. For more information about SQL options, see "SQL Options on the General Tab" on page 24.

TCP/IP Services File

Understanding the TCP/IP Services File

The TCP/IP services file contains information about available services on the machine, including the name, port number, protocol, and any aliases for each service. For the SAS 9.3 release of the drivers, you do not need to create entries in the TCP/IP services file as long as you define local servers with the double-underscore syntax. For example, a local server can be defined with the name **__5001** to start a local SAS ODBC server on port 5001. A SAS/SHARE server can be defined with the name **machine.__5010** to connect with a SAS/SHARE server running on port 5010 of a host that is named machine.

For releases of the SAS Drivers for ODBC before SAS 9.3, an entry in the TCP/IP services file is necessary to associate a SAS server (service name) with a port number and protocol used by that service. For the SAS 9.3 release of the drivers, creating an entry in the TCP/IP services file is necessary only if you prefer not to use the double-underscore syntax.

The location of the TCP/IP services file varies on different platforms. Common locations for the TCP/IP services file are the following:

Windows
C:\WINDOWS\SYSTEM32\DRIVERS\ETC\SERVICES

UNIX
/etc/services

Entries in the TCP/IP services file have the following general form:

<official service name> <port number/protocol name> <aliases> # <comments>

Note: For a connection to a SAS/SHARE server, you must update the TCP/IP services file on both the server and client machine if the two-part name uses a service name, such as **machine.shr2**. If the SAS/SHARE administrator starts the SAS/SHARE server with the **SERVER=__*port-number*** syntax, then only the client machine must define the SAS server with the same **__*port-number*** value, or the client machine can add an entry to the TCP/IP services file.

Editing the TCP/IP Services File

To configure your TCP/IP services file for use with the drivers, you must add an entry to the services file for each SAS server (either local or remote) that you have configured using the ODBC Data Source Administrator dialog box.

The port number that you use should be an unused port number in the TCP/IP services file. (For larger networks, contact your network administrator to obtain an available port number.) The port number must be greater than 1024, because any port number equal to or less than 1024 is reserved. The protocol must always be TCP. The server name can be up to eight characters. The first character must be a letter or an underscore. Subsequent characters can be letters, numeric digits, underscores, the dollar ($) sign, or the at (@) sign.

For example, in the ODBC Data Source Administrator dialog box, if you configured a local data source named **local**, and a remote SAS/SHARE server named

machine.shr2, then you should add entries to the TCP/IP services file similar to the following entries (substituting the appropriate port numbers):

```
local    6000/tcp    # service name for local access to SAS data
shr2     5010/tcp    # service name for SAS/SHARE server
```

Note: In the case of **shr2**, the network administrator of the remote system named **machine** should have already edited the services file on the remote system to include the same **shr2** entry, and he should have started the SAS/SHARE server.

SQL Options on the General Tab

SQL options affect the interactions among the driver, SAS, and applications that are ODBC compliant. The default settings for the SQL options are those that most ODBC-compliant applications expect and work best with. However, you can override the default settings by selecting any of the SQL options that are listed. Select the check box to enable the SQL option.

Preserve trailing blanks
preserves trailing blanks at the end of character fields. The default is that trailing blanks are removed so that each field ends in a null value.

Support VARCHAR
causes character fields that are longer than 80 characters to be reported as variable-length fields, and causes the trailing blanks to be removed. For more information, see "Support VARCHAR Option" on page 45.

Infer INTEGER from FORMAT
causes SAS numeric data types (typically reported as SQL_DOUBLE) to be reported as SQL_INTEGER. For more information, see "Infer INTEGER from FORMAT Option" on page 45.

Disable _0 override parsing
prevents the driver from removing the _0 string from SQL queries. For more information, see "Disable _0 override parsing Option" on page 45.

Use client code page
tells the driver to use the client's code page information when transcoding multi-byte characters to and from WideChar. If this option is not selected, the server's code page information is used.

DQUOTE=ANSI
strings enclosed in double quotation marks are treated as variables. This feature enables you to use reserved words such as AS, JOIN, and GROUP as table names, column names, or aliases. It also enables using DBMS names and other names that are not normally permissible in SAS when the values are enclosed in double quotations marks. The VALIDVARNAME=ANY system option is also set by the driver to enable non-ASCII characters to be used in column names when the value is enclosed in double quotation marks. If non-ASCII characters are needed in a table name and a SAS 9.3 or later SAS/SHARE server is used, then enclose the table name in double quotation marks and enable the VALIDMEMNAME=EXTEND system option in a SAS/SHARE server configuration file. When this option is not selected, values within double quotation marks are treated as strings.

Return SQLTables REMARKS
causes the driver to read and return the SAS data set label for each data set in the library you are accessing. (SQLTables is the name of an ODBC function that can be

used for this purpose.) For SAS data sets, this task can have a negative impact on performance, because each data set must be opened to read and return the label. Therefore, you should not select this option unless there is information in the label that you need.

UNDO_POLICY=REQUIRED
implements the UNDO_POLICY option of the SAS SQL procedure with a setting of REQUIRED. With this setting enabled, INSERT or UPDATE statements that fail are undone. However, this option is only for statements that affect multiple records. This option has no effect for an SQL statement that affects a single record. When UNDO_POLICY=REQUIRED, the associated statement handle (**`hstmt`**) of an UPDATE or INSERT statement must be the only active **`hstmt`** against the table. If another user or an **`hstmt`** within the same user's application has an active SELECT statement, then the UPDATE or INSERT statement fails.

Fuzz Numbers at *N* Places
specifies the degree of precision to use when comparing numbers. For more information, see "Fuzz Numbers at N Places Option" on page 45. By default, this option is selected. You can change the default value of 12 by typing over it.

Infer type in min/max functions
tells the driver to attempt to determine the format of the column used as input to a MIN or MAX function, and to use that format to infer the SQL type of the column.

Quote Char is NULL
tells the driver to use an empty string for the SQLGetInfo(SQL_IDENTIFIER_QUOTE_CHAR) method when this option is enabled. When this option is not enabled, the driver returns a blank space. This option should be enabled only when directed by SAS Technical Support or the application vendor.

Using the General Tab

When you first access the SAS ODBC Driver Configuration dialog box, the **General** page is in the foreground. Use the **General** page for the following reasons:

- provide the Data Source Name that you use in your ODBC application
- provide a description of the data in the data source
- select a server from the **Server** field to associate the Data Source Name with a server
- configure the number of records to buffer
- set SQL options

Data Source Name
provide a name for the data source that you want to access. The name must begin with a letter, and it cannot contain commas, semicolons, or any of the following special characters: **`[ ] { } ( ) ? * = ! @`**. For example, if you are defining SAS data that is stored on a machine named Cicero, you might call your data source SAS_Cicero. If you or other users are concerned only with the type of data (or with the type of application that uses that data), and not with where the data is stored, then you might have data sources with names like Finance or Payroll.

Description
provide a description of the data source. Providing a description is optional.

Server
a menu from which you can select a defined server. The first time you set up a data source, the menu is empty. Define one or more servers, and then come back to the **General** page to make a selection. You must specify a server for every data source.

Records to Buffer
provide the number of rows to request from the SAS/SHARE server in a single transmission. The default value is 100 and the maximum value is 32,000.

The size of the server's transmission buffer limits the actual number of rows returned. Usually you do not change the default value. However, specifying a larger value might improve performance when you retrieve a very large result set.

Using the Servers Tab

Purpose of the Servers Tab

The **Servers** tab of the SAS ODBC Driver Configuration dialog box is used for adding, deleting, and modifying server definitions. The driver can communicate with a SAS ODBC server for local data access, or with a SAS/SHARE server and an SPD Server for remote data access. This tab has the following fields:

Name
identifies the SAS server name of the data source. This value can be a one-part name, such as `__5001`, or a two-part name, such as `machine.__5010`. Use a one-part name when you want to access local data or a two-part name when you want to access remote data.

Password
identifies the password that the driver sends to a password-protected SAS/SHARE server.

Configure
opens the Local Options dialog box or the SAS/SHARE Options dialog box.

SAS Server Type
identifies whether the server is local or remote. For a server with a one-part name, this field says `Local (Single User)`. For a server with a two-part name, this field says `SAS/SHARE (Multi-User)`.

Clear
click this button to clear all of the fields in the **Server Settings** area.

Deleting a Server Definition

To delete a defined server, complete these steps:

1. Select the server name from the `Servers` list on the **Servers** page.
2. Click **Remove**.

CAUTION:
If you delete a server, then any data sources that use the server are no longer accessible until you redefine the server.

Modifying a Server Definition

To change the information for a defined server, complete these steps:

1. Select the server name from the **`Servers`** list on the **Servers** page.
2. To make changes, click **Configure** in the Local Options or SAS/SHARE Options dialog box.

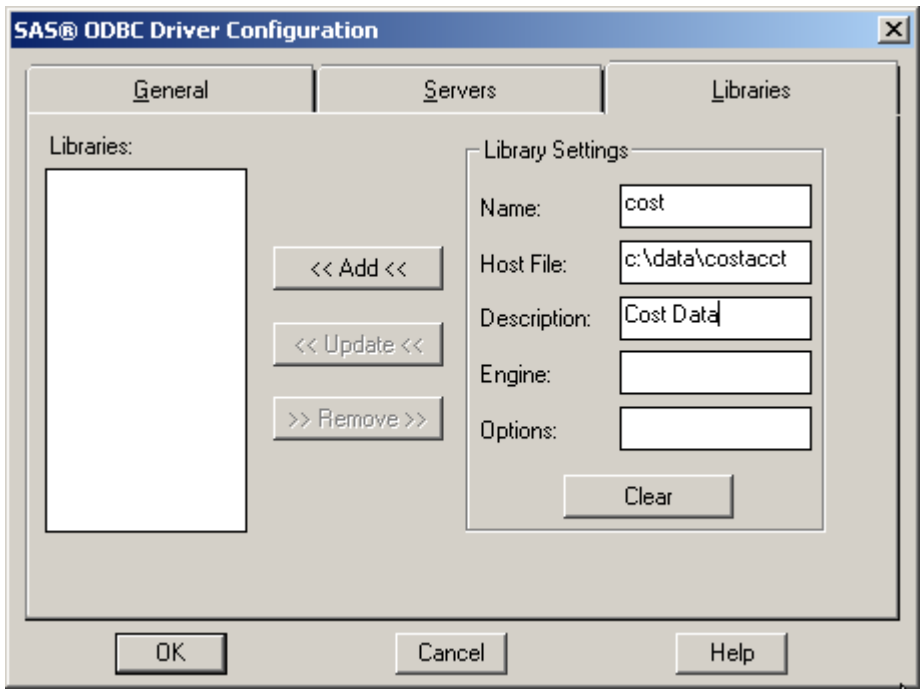
   The **Name** field is disabled to indicate that you cannot change the name of the server. The reason is that you might have defined data sources that use that server. If you change the server name, then you can no longer access those data sources.
3. Click **Update** to save your changes.

For instructions on how to specify a different server for a data source that you have already defined, see "Specifying a Different Server for a Defined Data Source" on page 29.

Using the Libraries Tab

Purpose of the Libraries Tab

Each DSN is associated with a server, and a server can access multiple data libraries. Therefore, you provide information about each data library that you want to access from a server.



This tab has the following fields:

Libraries
provides a list of libraries that are associated with a DSN.

Name
identifies the name for an existing physical SAS library that you want to access. (If you are familiar with SAS, this field corresponds to the libref in the SAS LIBNAME statement.) The name can be up to eight characters. The first character must be a letter or an underscore. Subsequent characters can be letters, numeric digits, or underscores. Blank spaces and special characters are not allowed. For example, you might use the name **cost** to designate a library of cost accounting data.

Host file
identifies the pathname of the library. For example, **c:\data\costacct**, **d:\data**, or **\\acctsrv\customers**. If you connect to a SAS/SHARE server, then identify the pathname of the library on the machine that is hosting the SAS/SHARE server.

Description
identifies a description of the library. Providing this value is optional.

Engine
identifies the name of the SAS engine that is required for writing to and reading from this library. This setting is necessary only if you do not want to use the **V9** engine, which is the default for SAS 9.3. For information about other engines that might be available, see the description of the LIBNAME statement in the *SAS Companion for Windows*. Providing this value is optional.

Options
identifies options for the library that you are defining, such as **ACCESS=READONLY**.

Defining Libraries at Server Start-Up Time

Server administrators might prefer to define SAS libraries at server start-up time, rather than defining them through the SAS ODBC Driver Configuration dialog boxes. Defining libraries at server start-up time can make opening the data source faster. It enables you to avoid hardcoding the physical names of your libraries in your SAS ODBC data source definitions.

The driver communicates with a SAS/SHARE server (invoked by PROC SERVER) to access remote data or a SAS ODBC server (invoked by PROC ODBCSERV) to access local data. To define a data library at server start-up time, you precede the PROC SERVER or PROC ODBCSERV statement with a SAS LIBNAME statement. For example, you could define a library of cost accounting data to a SAS/SHARE server as follows:

```
options VALIDVARNAME=any;
libname cost 'c:\data\costacct';
proc server id=acctserv authenticate=optional;
run;
```

Note: Depending on whether the server is running in secured mode, the **authenticate=optional** option might not be needed.

To define this library to a SAS ODBC server, you would add only the previous LIBNAME statement to the **!SASROOT\core\sasmacro\sasodbc.sas** file.

When a user requests access to a SAS ODBC data source from an ODBC client application, the server automatically makes the library available, along with any libraries that were defined on the **Libraries** tab.

For more information about the LIBNAME statement, see the SAS Companion for the operating system under which your data library is stored.

Deleting a Data Library Definition

To delete a defined data library, complete these steps:

1. Select the library name from the **`Libraries`** on the **Libraries** page.
2. Click **Remove**.

Modifying a Data Library Definition

To change the information for a defined library, complete these steps:

1. Select the library name from the **`Libraries`** list on the **Libraries** page.
2. Make changes to the **Host File**, **Description**, **Engine**, and **Options** fields.
3. Click **Update** to save your changes.

Modifying a Data Source Definition

To change the information for a defined data source, complete these steps:

1. Select the data source name from the **User DSN** or **System DSN** tab in the ODBC Data Source Administrator dialog box.
2. Click **Configure**. The SAS ODBC Driver Configuration dialog box appears.
3. Use the **General**, **Servers**, and **Libraries** tabs in this dialog box to access the information. Use these tabs to modify server definitions, data library definitions, or SQL options.
4. Click **OK** to save your changes.

Specifying a Different Server for a Defined Data Source

Suppose a data source named **`Payroll`** has been moved from a server named **`CICERO`** to a server named **`DAVINCI`**. In this case, you need to change the server that you specified for the **`Payroll`** data source by completing these steps:

1. Select the **`Payroll`** data source from the ODBC Data Source Administrator dialog box.
2. Click **Configure**. The SAS ODBC Driver Configuration dialog box appears, with **`CICERO`** listed in the **Server** field.
3. Use the **Servers** tab to define the **`DAVINCI`** server if it is not already defined.
4. From the **General** page, select **`DAVINCI`** from the **Server** menu.
5. Click **OK** to save your changes.

Note: The server name specified must be defined in the TCP/IP services file. For more information, see "TCP/IP Services File" on page 23.

Deleting a Data Source Definition

To delete a defined data source, complete these steps:

1. Select the name of the data source from the ODBC Data Source Administrator dialog box.
2. Click **Remove**.

Chapter 3

Using the SAS Drivers for ODBC

Introduction to Using the SAS Drivers for ODBC

This chapter provides an overview of how to use the drivers to access your SAS data sources. It provides information about the SAS servers and the communications access methods that are used by the driver.

Accessing Your Data Sources

The details of how you access your SAS data sources depend on which ODBC-compliant application you are using (for example, Microsoft Access or Excel), and on whether you are accessing local or remote data. To access your data source, complete these steps:

1. Install the SAS Drivers for ODBC. (See the installation instructions for the driver.) The installation program provides the drivers for the ODBC Data Source Administrator to use.

2. Double-click the ODBC Data Sources or ODBC Administrator icon in the Control Panel window to access the ODBC dialog box and the SAS ODBC Driver Configuration dialog box.
3. Use both of these dialog boxes to provide the driver with the necessary information about your SAS data sources. (For more information, see Chapter 2, "Defining Your Data Sources," on page 11.) SAS data sources can include SAS data sets, DATA step views, PROC SQL views, or SAS/ACCESS views. All of these contain definitions of data that is stored elsewhere from the physical data itself. (For more information, see "Types of Data Accessed with the SAS Drivers for ODBC" on page 4 and "SAS Data Sets" on page 5.)
4. For information about how to access or import data from other sources, see the documentation for your Windows application. From the list of available data sources, select the name of your SAS data source.
5. Select or enter the name of the SAS library.
6. Select or enter the name of the SAS data file or view.

For information about configuring file data source names (DSN) for applications that use file DSNs only (such as Microsoft Excel 97), see the online Help that is available when you configure data sources in the ODBC Data Source Administrator.

Understanding Access to Local Data Sources

Starting a SAS ODBC Server

If there isn't a SAS ODBC server running on your PC when you access a SAS data source from your ODBC application, then the driver starts a SAS ODBC server. The driver uses the information that you provided in the Local Options dialog box to start one for you automatically.

To access local data sources, the driver uses TCP/IP to communicate with a SAS ODBC server. You must edit your TCP/IP services file to define your servers before starting the SAS ODBC server. It is not necessary for the server to be running when you define your data sources. However, the server must be running on your PC for you to access your SAS data sources. For information about editing the TCP/IP services file, see "TCP/IP Services File" on page 23. For more information about SAS servers, see "SAS Servers" on page 6.

If you already have a SAS session running on your PC, then you can start the SAS ODBC server in that session by submitting the following statements:

```
options comamid=tcp;
proc odbcserv id=servername authenticate=optional;
run;
```

The value for server name must be the same server name you specified on the **Servers** page when you defined your local data source, as explained in "Setting Up a Connection to Local Data" on page 14.

Alternatively, you can terminate your SAS session so that the driver can start a SAS ODBC server for you in a new SAS session.

Note: When the SAS ODBC server is running in a SAS session, the SAS session does not accept user input from the keyboard.

If the SAS session cannot be started before the timeout value that you specified in the Local Options dialog box is reached, a timeout error is returned to your ODBC client application. An error message is returned to the client if the SAS session was started, but PROC ODBCSERV could not execute.

Stopping a SAS ODBC Server

When you are finished using your ODBC client application to access your local data sources, the SAS ODBC server continues to execute in case you want to access additional SAS data sources. To terminate the SAS ODBC server, do either of the following:

- Open the Windows Task Manager. Under the **Applications** tab, select **SAS** from the list of tasks, and then click **End Task**. Alternatively, under the **Processes** tab, select **SAS.EXE** in the Image Name column, and then click **End Process**.
- Bring the SAS session into focus (make it the active window), and then press CTRL+BREAK. The Task Manager appears. Select **ODBCSERV** from the list of tasks, and then click **OK**. Click **OK** again to halt the procedure.

Understanding Access to SAS/SHARE Data Sources

The drivers can communicate with SAS/SHARE servers for access to remote data. For complete information about SAS/SHARE, see *SAS/SHARE User's Guide*. It is not necessary for the server to be running when you define your data sources. However, the server must be running on your remote host in order for you to access your data sources.

Because a SAS/SHARE server is used by multiple users, it is usually invoked on the remote host at system start-up time. Therefore, users usually do not need to locally invoke the server.

The SAS/SHARE administrator must complete these steps:

1. Define a service name and network port in the TCP/IP services file for the SAS/SHARE server. (For more information, see "TCP/IP Services File" on page 23.)
2. Create a SAS program that uses the SERVER Procedure. See the following example:

   ```
   options comamid=tcp;
   proc server id=servername authenticate=optional;
   run;
   ```

   Invoke this SAS program to run in the background using host-specific command syntax. For more information, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE*.

Userid/Password Override

The **Userid/Password Override** option is in the SAS/SHARE Options dialog box on the **Servers** page. This option enables you to use the UID keyword and PWD keyword in the ODBC client application. The driver passes the value of the UID keyword as the user

ID, and passes the value of the PWD keyword as the user login password to the SAS/SHARE server or SPD Server. The values for UID and PWD in the ODBC client application override the values that are stored in the DSN definition.

If the driver fails to authenticate with the SAS/SHARE server or the SPD Server, the driver provides a prompt for the user ID and password credentials (up to three times). If the user ID and password credentials are successfully authenticated, then the **Userid/ Password Override** check box is selected. After a successful login, the driver returns the credentials if the application calls the SQLDriverConnect() method.

Note: For sites that use a SAS/SHARE server with a user password (the UAPW= option), the password must be stored in the DSN definition. The SAS/SHARE server password has no interaction with the **Userid/Password Override** setting, and the driver does not prompt for the credentials if authentication fails.

Using Data Sets That Have One-Level Names

If you use an ODBC application such as Microsoft Access that exports databases using one-level names, you should use the ODBC Data Source Administrator to define a library that is named User. SAS usually places any data set that has a one-level name in the Work library, which is deleted at the end of the SAS session. But, if a User library has been defined, SAS places all one-level named data sets in that library. That library is saved at the end of the SAS session. In a multi-user environment, each client connection to a SAS server can have its own User library defined.

Updating Attached Tables

Some Microsoft products that are based on the Jet database engine (such as Microsoft Access) have certain requirements to be able to update database tables. This might be true of other ODBC applications as well. These requirements might make it necessary for you to specify two SQL options when you define your data sources.

- The attached table must have a unique primary key that is not a floating-point value. You can use the data source SQL option **`Infer INTEGER from FORMAT`** to indicate that SAS numeric data types without fractional parts (for example, FORMAT(*n*,0), where *n* is less than 12) are actually integer values that can be used to index the table.
- All of the values in a row might be used to uniquely select the row for updating. This can be a problem in rows that contain floating-point fields (SAS numerics). Insignificant differences in values can be caused by differences in floating-point representation on different machines or by conversion between character and binary formats. By specifying the data source SQL option **Fuzz Numbers at 12 places**, you can cause WHERE clauses to select values that are acceptably close, rather than requiring exact comparisons.

These options are specified on the **General** tab. For more information about these SQL options, see "User-Specified SQL Options" on page 45.

Using SQL Statements to Access SAS Data Sources

All applications that are ODBC compliant use SQL to access and manipulate data. However, most of these applications transform your actions into SQL statements so that you do not need to know anything about SQL.

For more information, see "The SQL Procedure" in *Base SAS Procedures Guide*. The elements of SQL grammar that are supported by the drivers are the same elements described in that document.

Accessing the SAS Libraries MAPS, SASUSER, and SASHELP

By default, every SAS session (including SAS server sessions) provides access to the SAS libraries Maps, Sasuser, and Sashelp. These libraries contain sample data sets and other files that are generally not of interest to ODBC users. Therefore, the driver does not report the contents of these libraries when it invokes a SAS ODBC server. (From a programming standpoint, when SQLTables, SQLStatistics, or SQLColumns is called, the result set that is returned does not include rows for the SAS libraries Maps, Sasuser, or Sashelp.) If you want information from these libraries, you can do either of the following tasks:

- Use a LIBNAME statement to define the SAS library to the SAS ODBC server.
- Use the SAS ODBC Driver Configuration dialog box to define the SAS library to your ODBC applications.

In both cases, you must use a different name (that is, not Maps, Sasuser, or Sashelp) as your libref or library name.

Passwords

The drivers provide a text field to store a user password on the SAS/SHARE Options dialog box when you define a connection to a SAS/SHARE server or an SPD Server. If you provide a password, then the driver encrypts the password and stores the encrypted value in the Windows registry.

If you do not provide a password, when you use the DSN in an application, the Enter User Name and Password dialog box prompts for credentials. If you do not want to store encrypted passwords, the driver can accept credentials from the application. For more information, see "Userid/Password Override" on page 33.

Chapter 4

Programmer's Reference

Introduction to the Programmer's Reference

This chapter is intended for application programmers and others who need information about how the drivers are implemented. It provides information about support for ODBC functions, SQL grammar, and SQL data types.

The drivers are ODBC 3.5 version drivers and conform to the Core interface level. The following table identifies exceptions to the Core interface conformance. Overwhelmingly, when the drivers do not implement a function, it is because the type of SAS server that the driver connects to does not provide the functionality. (In this case, the type of SAS server is either SAS/SHARE, SPD Server, or Base SAS running PROC ODBCSERV.)

For applications that use ODBC version 2.0, Level 1 interface functionality is included in the ODBC version 3.5 Core interface level.

For complete information about the ODBC standard, see *ODBC Programmer's Reference* at **http://msdn2.microsoft.com/en-us/library/ms714177.aspx**.

Support for and Implementation of ODBC Functions

Core Functions

Table 4.1 *Core Functions*

Function Name	Purpose	SAS Drivers for ODBC Implementation
SQLBindParameter	Assigns storage for a parameter in an SQL statement.	SQL_DATA_AT_EXEC is not supported because SAS does not support large data fields.
SQLCancel	Cancels an SQL statement.	This function is used only in asynchronous mode, which SAS does not support. The SAS interprocess communication library does not provide a way to interrupt a transaction in process. SQLCancel does not cause an active statement to terminate immediately, and it does not halt an operation that is already in process. This function always returns SQL_SUCCESS.
SQLColAttributes	Describes the attributes of a column in the result set.	A common reason for applications to call this function is to determine whether a column of data is a dollar amount. With SAS data sets or views, this information is inferred from the format. For more information about SAS formats, see "Supported Data Types" on page 41.
SQLColumns	Returns the list of column names in specified tables.	SAS uses a specially formatted query to query the virtual table DICTIONARY.COLUMNS.
SQLDriverConnect	Connects to a specific driver by connection string or requests that the ODBC driver manager and the driver display connection dialog boxes for the user.	SAS uses the same dialog boxes that are used in configuration. If input was adequate, SAS continues with the connection, rather than saving the parameters. However, in many cases, input is not required. The connection to the host is made at this time.
SQLGetCursorName	Returns the cursor name that is associated with a statement.	SAS does not support cursors, so this function returns SQL_ERROR with SQLSTATE set to IM001 ("Driver does not support this function").
SQLGetData	Returns data for a single unbound column in the current row. The application must call SQLFetch or SQLExtendedFetch before calling SQLGetData.	This function enables you to use multiple calls to retrieve data from character variables. Some applications need this functionality to access long character variables. The extension SQL_GD_BLOCK (see SQL_GETDATA_EXTENSIONS under SQLGetInfo) is not supported. This means that you must call SQLExtendedFetch with a rowset size of 1.

Function Name	Purpose	SAS Drivers for ODBC Implementation
SQLMoreResults		
SQLParamData	Returns the storage value that is assigned to a parameter at execution time.	This function is used for large data fields, which SAS does not support. The function returns SQL_ERROR with SQLSTATE set to IM001 (“Driver does not support this function”).
SQLPrepare	Prepares an SQL statement for execution.	This function does not check syntax; syntax is checked later in the SQLExecute function by the server.
SQLPutData	Sends part or all of a data value for a parameter.	This function is used for large data fields, which SAS does not support. The function returns SQL_ERROR with SQLSTATE set to IM001 (“Driver does not support this function”).
SQLSetCursorName	Specifies a cursor name.	SAS does not support cursors, so this function returns SQL_ERROR with SQLSTATE set to IM001 (“Driver does not support this function”).
SQLSpecialColumns	Retrieves information about the optimal set of columns that uniquely identifies a row in a specified table, or about the columns that are automatically updated when any value in the row is updated by a transaction.	The drivers issue a query to the DICTIONARY.INDEXES view to obtain this information.
SQLStatistics	Retrieves statistics about a single table and the list of indexes that are associated with the table.	The drivers issue a query to the DICTIONARY.INDEXES view to obtain this information.
SQLTables	Returns the list of table names stored in a specific data source.	The drivers issue a query to the DICTIONARY.TABLES and DICTIONARY.MEMBERS views to obtain this information.
SQLTransact	Commits or rolls back a transaction.	Always returns SQL_SUCCESS for SQL_COMMIT. Returns an error for SQL_ROLLBACK because SAS does not support transactions.

ODBC Cursors and the SQLExtendedFetch Function

The drivers support the SQLExtendedFetch function to retrieve multiple rows of result data in a single operation. However, SAS does not support real cursor functionality, so the drivers cannot truly support cursors.

The implementation of SQLExtendedFetch uses a forward-only cursor to support the syntax of the function. Rows in the result data cannot be skipped, so the parameter IROW (which is the number of the row to begin fetching) is ignored. You cannot retrieve an arbitrary subset of the result data. However, because a single operation can fetch multiple rows, there is less overhead compared to when you use multiple calls to the SQLFetch function.

Note: To support the SQLExtendedFetch function, the SQLSetStmtOption function enables you to set the rowset size and to select rowwise or columnwise binding.

For more information about the SQLExtendedFetch function, see *ODBC Programmer's Reference* at **http://msdn2.microsoft.com/en-us/library/ms714177.aspx**.

Catalog Functions

You might want to remove certain SAS libraries from the results of the functions such as SQLColumns, SQLStatistics, SQLSpecialColumns, and SQLTables. Many client applications (for example, Microsoft Access) use these functions to retrieve a list of available tables. Currently, the results from the functions filter out the Maps, Sashelp, and Sasuser libraries. The filtering is done primarily so unwanted tables do not appear in lists that are based on results from these functions. The filtering can improve the performance of these functions.

You can replace this filter by adding the LibWhere key to a data source entry in the system registry. The LibWhere key includes a logical expression to conditionally specify which results to retrieve from the SAS server. The current filter uses the following logical expression:

```
tbl.libname NE 'MAPS' AND tbl.libname NE 'SASUSER' AND
tbl.libname NE 'SASADMIN' AND tbl.libname NE 'SASHELP'
```

The tbl name is important if you need to construct your own filter in these functions.

CAUTION:

When you use the registry editor, you can damage the Windows environment if your changes are invalid or if you accidentally delete information. Do not create a filter unless it is absolutely necessary. Editing the registry can damage registered programs and make them unusable.

If many libraries are defined on a server, but only specific libraries are needed, you can edit the registry to specify which libraries to access. For information about how to edit the registry, see the discussion of catalog functions in the online Help of the SAS ODBC Driver Configuration dialog box.

ODBC Scalar Functions

The drivers support the following ODBC scalar functions:

String Functions:
ASCII, CHAR, CONCAT, LCASE, LEFT, LTRIM, REPEAT, REPLACE, RTRIM, SOUNDEX, SPACE, SUBSTRING, UCASE

Numeric Functions:
ABS, ACOS, ASIN, ATAN, CEILING, COS, EXP, FLOOR, LOG, LOG10, MOD, POWER, RAND, ROUND, SIGN, SQRT, TAN

Time, Date, and Interval Functions:
CURDATE, CURTIME, DAYNAME, DAYOFMONTH, DAYOFWEEK, DAYOFYEAR, HOUR, MINUTE, MONTH, MONTHNAME, NOW, QUARTER, SECOND, YEAR

The ODBC RAND scalar function is translated to a SAS function. For more information about the usage and parameters of ODBC scalar functions, see *ODBC Programmer's Reference* at **http://msdn2.microsoft.com/en-us/library/ms714177.aspx**.

Support for SQL Grammar

The Microsoft ODBC specification defines three levels of support for SQL grammar: Minimum, Core, and Extended. The drivers support all Minimum-level and some Core-level SQL statements and statement elements. For more information about supported grammar, see *Base SAS Procedures Guide*.

Currently, the drivers do not support the following SQL syntax:

```
SELECT FOR UPDATE OF
```

Records are locked during data manipulation operations, but cannot be explicitly locked before a data manipulation operation.

Supported Data Types

SAS Data Types

Internally, SAS supports two data types for storing data:

CHAR
: fixed-length character data, 32,767-character maximum

NUM
: double-precision floating-point number

Note: If the data field is longer than 254 characters, the drivers process it as the ODBC data type SQL_VARCHAR.

By using SAS format information, the drivers are able to represent other ODBC data types, both when responding to queries, and in CREATE TABLE statements. (A SAS format is a string that describes how data should be printed. SAS associates format information with each column in a table.) For more information about SAS formats, see *SAS Formats and Informats: Reference*.

The following sections explain conventions for data type representation that the drivers follow.

ODBC SQL Data Types

The drivers support the ODBC SQL data types listed in the following table. Internally, SAS supports a numeric and character data type using informats and formats to modify the data representation. The drivers use the informat and format to determine the corresponding ODBC data type.

Table 4.2 *SAS Drivers for ODBC and Corresponding Data Types*

SQL Data Type	SAS Drivers for ODBC Implementation
SQL_CHAR	Supports text fields up to 200 characters long

SQL Data Type	SAS Drivers for ODBC Implementation
SQL_VARCHAR	Interpreted for text columns of length 80 or greater when the **Support VARCHAR** option is selected in the SQL Options section of the SAS ODBC Driver Configuration dialog box; the maximum length of an SQL_VARCHAR field is 32,767 characters, and is available when the version of the SAS server is Version 7 or later; for Version 6 SAS servers, the SQL_VARCHAR field is limited to 200 characters
SQL_DOUBLE	1.E-308 to 1.E308
SQL_FLOAT	1.E-308 to 1.E308
SQL_INTEGER	-2,147,483,648 to 2,147,483,647
SQL_DATE	Valid dates range from 1582 A.D. to 9999 A.D.
SQL_TIME	ODBC supports time values within the range of a 24-hour day (00:00:00 to 23:59:59)
SQL_TIMESTAMP	Valid dates range from 1582 A.D. to 9999 A.D. with a 24-hour day (00:00:00 to 23:59:59) time portion

Data Types Reported on Queries

When the SQLDescribeCol and SQLColAttributes functions are called against active queries, the drivers report data types as follows:

- When the SQLDescribeCol function is called, the drivers report CHAR data types as SQL_CHAR. NUM data types are generally reported as SQL_DOUBLE.

  However, SAS stores dates and times as numbers, and the drivers use SAS format information to infer the following SQL data types from NUM data types:[1]

  Table 4.3 *SAS and SQL Data Types*

SAS Data Type	SQL Data Type
NUM FORMAT=DATE*n*.	SQL_DATE
NUM FORMAT=TIME*n*.	SQL_TIME
NUM FORMAT=DATETIME*n*.	SQL_TIMESTAMP

  In each of the previous FORMAT= strings, *n* is a number that selects the printable representation by specifying a width for printing. The value of *n* is not relevant to the driver.

1 For a complete list of date and time formats that the drivers support, see the table of formats listed by categories in *SAS Formats and Informats: Reference*.

- When the SQLColAttributes function is called, if a NUM column has a format of DOLLAR*n*., the drivers identify the column as financial data (having a column attribute of SQL_COLUMN_MONEY).

Retrieving Native SAS Format Information

You can find the SAS format for a column by using the SQLColumns function. The drivers return the additional column, FORMAT. The nineteenth column in the SQLColumns function result set contains a string with the format information for a SAS column of data.

Using Data Types in CREATE TABLE Statements

In CREATE TABLE requests, the drivers interpret certain data-type specifications by creating NUM variables and associating SAS formats with them, as shown in the following table:

Table 4.4 *CREATE TABLE Data Types and SAS Data Types*

CREATE TABLE Data Type Name	**ODBC Data Type**	**SAS Data Type**
char(*w*)	SQL_CHAR	CHAR(*w*)
num(*w*, *d*)	SQL_DOUBLE	NUM
num(*w*, *d*)	SQL_FLOAT	NUM
integer	SQL_INTEGER	NUM FORMAT=11.0
date9x	SQL_DATE	NUM FORMAT=DATE9X.
datetime19x	SQL_TIMESTAMP	NUM FORMAT=DATETIME19X.
time8x	SQL_TIME	NUM FORMAT=TIME8X.
timestamp	SQL_TIMESTAMP	NUM FORMAT=DATETIME26.6

The data type names listed in the first column are the values that are returned by SQLColAttributes (with the parameter SQL_COLUMN_TYPE_NAME) and by SQLGetTypeInfo. For all CREATE TABLE statements, the drivers translate these data type names into the SAS data types shown in the third column. Do not try to use the ODBC data types directly in SAS.

As an alternative to using **`num(w, d)`** for creating SQL_FLOAT and SQL_DOUBLE columns, you can specify FORMAT= in a CREATE TABLE statement. In this case, the drivers pass the information to SAS unmodified. A column within a table (or data set) can be created based on any exact specification that is required for its use within SAS. For example, in the following CREATE TABLE statement, variable B's data type and format are passed directly to SAS:

```
CREATE TABLE SASUSER.TABLE1 (
  A INTEGER,
```

```
  B NUM FORMAT=9.5,
  C CHAR(40)
);
```

The following code shows how to create a table with the time-related data types:

```
CREATE TABLE library.TIME_EXAMPLES (
  TIMESTAMP_COL TIMESTAMP,
  DATETIME_COL DATETIME19X,
  DATE_COL     DATE9X,
  TIME_COL     TIME8X
);
```

To insert date-related values, use SQL syntax similar to the following code:

```
INSERT INTO TABLE library.TIME_EXAMPLES (
  {ts'1960-01-01 00:00:00.000001'},
  {ts'1960-01-01 00:00:00},
  {d'1960-01-01'},
  {t'00:00:00'}
);
```

If you are more familiar with SAS date formats, then you can insert date-related values as shown in the following code:

```
INSERT INTO TABLE library.TIME_EXAMPLES (
  '01jan1960:00:00:00.000000'dt,
  '01jan1960:00:00:00'dt,
  '01jan1960'd,
  '00:00:00't
);
```

Comparing Date, Time, and Datetime Values

When you compare date, time, and datetime values in SAS data sets from an ODBC application, you must consider the following:

- A SAS time value is the number of seconds since the current day began. That is, 0 is 00:00:00 or 12:00:00 a.m., and 86399 is 11:59:59 p.m.

  Note: ODBC does not support negative time values or values greater than one day's worth of seconds. The drivers return an error for time values that are less than 0 or greater than 86399 (the last second of the day).

- A SAS date value is the number of days since January 1, 1960. That is, 0 is 01jan1960, and -1 is 31dec1959.
- A SAS datetime value is the number of seconds since midnight on January 1, 1960. That is, 0 is 01jan1960:00:00:00, and -1 is 31dec1959:11:59:59.

Both ODBC and SAS date, time, and datetime literals are supported by the drivers.

CAUTION:
You can compare equivalent literals against SAS date, time, or datetime values only because they each have a different unit of measure.

For example, you cannot compare a SAS data set value that has been defined with a datetime format against a date literal using either of the following:

```
select * where hiredate = {d'1995-01-02'}

select * where hiredate = '02jan1995'd
```

Instead, use a datetime literal, such as either of the following:

```
select * where hiredate = {ts'1995-01-02 00:00:00'}

select * where hiredate = '02jan1995:00:00:00'dt
```

User-Specified SQL Options

Infer INTEGER from FORMAT Option

This option affects how other default conversions of data types or data values can be made. Even when no format string is specified for SAS data, SAS assigns a default width and number of decimal places to the data. If the SQL option **`Infer INTEGER from FORMAT`** is selected, then the drivers report SAS columns of NUM(*n*,0) data type as SQL_INTEGER, where *n* is less than 12. This can be important because some software products do not use indexes on floating-point columns. If those columns actually contain only integer values, then using this SQL option enables these products to read the index and allow updates. For more information, see "Updating Attached Tables" on page 34.

Support VARCHAR Option

This option affects how other default conversions of data types or data values can be made. Enabling this option causes the drivers to report the data type CHAR(*n*) as SQL_VARCHAR, where *n* is greater than 80. Because SAS is fixed width, CHAR fields are often specified at the maximum value. For example, for a list of messages, the text width might be specified as 200 characters, even though the average width of a message is much less. Reporting the data type as SQL_VARCHAR enables some software products to use less memory.

Disable _0 override parsing Option

Sometimes, the _0 string occurs at the end of names in SQL queries constructed by certain applications. Enabling this SQL option prevents a SAS error by specifying that the driver keeps the _0 string at the end of a table name. For example, the string _0 is added to table aliases in SQL queries that Microsoft Query constructs so that the table name **`mytable`** becomes **`mytable_0`**. Disabling this option might be helpful if you need to access data on a Version 6 SAS/SHARE server because table names are limited to eight characters in Version 6.

Fuzz Numbers at N Places Option

This SQL option addresses a problem that occurs during the conversion of floating-point numbers. Floating-point numbers are stored in different binary representations on different computer hardware. Even when data is transferred between different applications on the same type of hardware, the precision of floating-point numbers might be affected slightly because of the conversion between ASCII and binary representations.

This effect is usually so slight that it is insignificant when a number is used in calculations. For example, the numbers 65.8 and 65.799999999999 are almost identical

for mathematical purposes. The difference between them might be the result of conversion between representations, rather than any purposeful change in value.

However, such a slight difference in value can keep a number from comparing correctly. For example, many ODBC applications include a WHERE clause that lists every column in a record at its current value whenever the application issues an UPDATE statement. This is done to ensure that the record has not changed since the last time it was read. Sometimes, a comparison might fail because of the problem with floating-point conversion.

To solve this problem, SAS fuzzes numbers (standardizes the degree of precision to use, overriding the hardware-specific representations). Instead of using exact comparisons, SAS checks to make sure that the numbers are acceptably close.

By default, the degree of precision is 12 decimal places. Given a number **N**, if **N1** were to be checked for equality with **N**, then the driver would use the SQL BETWEEN function to determine the following:

```
N1 > (N - (ABS(N * 10**-12))) AND N1  (N + (ABS(N * 10**-12)))
```

If **N=0**, the driver checks for the following:

```
BETWEEN -(10**-12) AND (10**-12)
```

Security Notes

To specify a secure SAS/SHARE server, submit code similar to the following code on your Windows system:

```
/* Start a remote secured SAS/SHARE server that requires user login  */
/* name and password specified by the client.                        */
/* ADOMAIN is the domain that authenticates username and password    */
%let tcpsec=_secure_;             /* Require user logon id/password  */
options authserver=adomain;
proc server id=shr1 authenticate=req;
run;
```

For this code to work on Windows 2000 or Windows NT you must assign **Act as part of the operating system** to the user account that is running the SAS/SHARE server. This privilege is not needed for later versions of Windows such as Windows XP and Windows Server 2003. You must assign **Log as a batch job rights** to the user account that wants to connect to the SAS/SHARE server.

For example, suppose your Windows system is named TRISTAN, and the user account adomain\stephmc wants to connect to the SAS/SHARE server. Also, suppose the user logged in to TRISTAN (and running the SAS/SHARE server) is the user account adomain\joshua. On the system named TRISTAN, you must assign user rights. To do this, use the group policy.

Complete the following steps:

1. Select **Start** ⇨ **Run**, type **gpedit.msc**, and then click **OK**.
2. Select **Windows Settings** under **Computer Configuration**. Select **Security Settings** ⇨ **Local Policies** ⇨ **User Rights Assignment**.
3. Right-click **Act as part of the operating system**, and then select **Properties**. Click **Add User or Group**, type **adomain\joshua**, and then click **OK**.

4. Right-click **Log on as a batch job**, and then select **Properties**. Select **Add User or Group**, type **`adomain\stephmc`**, and then click **OK**.

After the system TRISTAN is shut down and restarted (rebooting is usually required), you submit the preceding code on TRISTAN.

To log on to the remote Windows system, the user account adomain\stephmc specifies the user name adomain\stephmc in the SAS ODBC Driver Configuration dialog box and the password in the SAS/SHARE Options dialog box. The user specifies an address such as tristan.mynet.com. This action enables the user account adomain\stephmc to connect to the secure SAS/SHARE server.

For more information, see *SAS/SHARE User's Guide*.

SAS Drivers for ODBC Error Codes

For information about the SQLSTATE values (return codes) and associated messages that can be returned from the SQLError function, see *ODBC Programmer's Reference* at **`http://msdn2.microsoft.com/en-us/library/ms714177.aspx`**.

For information about messages that might be returned by your communications software, see Chapter 5, “Return Codes and Associated Messages,” on page 49.

Chapter 5

Return Codes and Associated Messages

SAS Drivers for ODBC Return Codes

For information about SQLSTATE values (return codes) and associated messages that can be returned for the SQLError function, see *ODBC Programmer's Reference* at **`http://msdn2.microsoft.com/en-us/library/ms714177.aspx`**. The associated messages might be generated by the driver, the SAS server, or by your communications access method. The ODBC driver manager passes return codes and messages to client applications.

Trace Files

In addition to the error messages in the following sections, information can sometimes be found in a trace file. A trace file is created in the working directory of the ODBC client application if it fails to connect to a SAS server. The trace file has a name in the form WQExxxxx.TRC, where xxxxx is the process ID of the ODBC client application at the time of failure.

The following trace file is a sample response for a DSN that does not have the service name identified in the TCP/IP services file on the client machine. Add the service name to the TCP/IP services file on the client machine. Review the task for setting up your connection.

```
[2786321593]:WQEOpen: Winsock getservbyname failed rc = 11004.
[2786321593]:WQEOpen failed: rc -1005 trc 0x2afc.
```

The following trace file is a sample response for a DSN that has the service name identified in the TCP/IP services file on the client machine, but the same service name is not found in the TCP/IP services file on the server. Add the same service name to the TCP/IP services file on the server.

```
[2788365437]:WQEOpen: Winsock sendrecv failed rc = 0.
[2788365437]:WQEOpen failed: rc -1007 trc 0x0.
```

For more information about service names, see "TCP/IP Services File" on page 23.

The information in the previous paragraphs provides information about how to use an automatically generated trace file to assist with diagnosing communications access errors. A trace file can also be created intentionally to assist with diagnosing undesirable behavior. Make sure that the ODBC client application is closed, and then follow the instructions provided at **`http://support.microsoft.com/kb/274551`**.

S1000 Communications Access Method Errors

The S1000 (SAS API error) return code is often accompanied by error messages that are returned by your communications access method. The following tables list some of the messages and explain them.

Table 5.1 *S1000 Communications Access Method Errors*

Message Text	Explanation
Memory failure	Not enough memory is available.
Network failure	An unspecified network failure occurred.
No server found	The remote server was not found.
Remote closed connection	The SAS server disconnected.
Remote refused connection	The remote system disallowed a connection. Check the remote services file.
Start SAS failure—please check your SAS server parameters	The ShellExecute statement failed when starting SAS. Check to see whether the SAS paths are specified properly.
TCP method Winsock API <*function-name*> failed with WSAGetLastError <**`rc`**>	A TCP/IP Winsock return code (**`rc`**) was returned. The return codes are listed in Table 5.2 on page 51..
Timeout waiting for the SAS server—check the startup options	A SAS server did not register itself as a server within the specified time period.
Unable to locate remote host	TCP/IP could not find the remote host name.
Unable to locate service	TCP/IP could not find the server name in the services file.
Userid.password security failure	User ID and password verification failed on the remote machine.
You must connect to SAS/SHARE on a remote machine	You must select SAS/SHARE in the SAS ODBC Driver Configuration dialog box **Servers** page to connect to a remote machine.

TCP/IP Winsock Return Codes

Table 5.2 *TCP/IP Winsock Return Codes*

Return Code	Return-Code Mnemonic	Description
10004	WSAEINTR	The (blocking) call was canceled via WSACancelBlockingCall.
10013	WSAEACCES	The requested address is a broadcast address, but the appropriate flag was not set.
10014	WSAEFAULT	The function argument is incorrect.
10022	WSAEINVAL	Invalid argument or function sequence, or the socket has not been bound with bind.
10024	WSAEMFILE	No more file descriptors are available.
10035	WSAEWOULDBLOCK	The socket is marked as non-blocking and the operation would block.
10036	WSAEINPROGRESS	A blocking Windows sockets call is in progress.
10037	WSAEALREADY	The asynchronous routine being canceled has already completed.
10038	WSAENOTSOCK	The description is not a socket.
10039	WSAEDESTADDREQ	A destination address is required.
10040	WSAEMSGSIZE	The datagram was too large to fit into the specified buffer and was truncated.
10041	WSAEPROTOTYPE	The specified protocol is the wrong type for this socket.
10042	WSAENOPROTOOPT	The option is unknown or unsupported.
10043	WSAEPROTONOSUPPORT	The specified protocol is not supported.
10044	WSASOCKTNOSUPPORT	The specified socket type is not supported in this address family.
10045	WSAEOPNOTSUPP	The referenced socket is not the proper type.
10046	WSAEPFNOSUPPORT	The protocol family is not supported.
10047	WSAEAFNOSUPPORT	The specified address family is not supported.
10048	WSAEADDRINUSE	The specified address is already in use.

Return Code	Return-Code Mnemonic	Description
10049	WSAEADDRNOTAVAIL	The specified address is not available from the local machine.
10050	WSAENETDOWN	The Windows sockets implementation has detected that the network subsystem has failed.
10051	WSAENETUNREACH	The network cannot be reached from this host at this time.
10052	WSAENETRESET	The connection must be reset because the Windows sockets implementation dropped it.
10053	WSAECONNABORTED	The virtual circuit was terminated due to timeout or another failure.
10054	WSAECONNRESET	The virtual circuit was reset by the remote side.
10055	WSAENOBUFS	No buffer space is available.
10056	WSAEISCONN	The socket is already connected.
10057	WSAENOTCONN	The socket is not connected.
10058	WSAESHUTDOWN	The socket has been shut down.
10059	WSAETOOMANYREFS	Too many references: cannot splice.
10060	WSAETIMEDOUT	Attempt to connect timed out without establishing a connection.
10061	WSAECONNREFUSED	The attempt to connect was forcefully rejected.
10062	WSAELOOP	Too many levels of symbolic links.
10063	WSAENAMETOOLONG	The filename is too long.
10064	WSAEHOSTDOWN	The host is down.
10065	WSAEHOSTUNREACH	No route to host.
10066	WSAENOTEMPTY	The directory is not empty.
10067	WSAEPROCLIM	Too many processes.
10068	WSAEUSERS	Too many users.
10069	WSAEQUOT	The disk quota was exceeded.
10070	WSAESTALE	Stale NFS file handle.
10071	WSAEREMOTE	Too many levels of remote in path.
10091	WSAESYSNOTREADY	The underlying network subsystem is not ready for network communication.

Return Code	Return-Code Mnemonic	Description
10092	WSASVERNOTSUPPORTED	The version of Windows sockets API support requested is not provided by this particular Windows sockets implementation.
10093	WSANOTINITIALISED	A successful WSAStartup must occur before using this API.
11001	WSAHOST_NOT_FOUND	Authoritative Answer Host not found.
11002	WSATRY_AGAIN	Non-Authoritative Answer Host not found, or SERVERFAIL.
11003	WSANO_RECOVERY	Non-recoverable errors, FORMERR, REFUSED, NOTIMP.
11004	WSANO_DATA	Valid name, no data record of requested type.

Glossary

access descriptor
a SAS/ACCESS file that describes data that is managed by SAS, by a database management system, or by a PC-based software application such as Microsoft Excel, Lotus 1-2-3, or dBASE. After creating an access descriptor, you can use it as the basis for creating one or more view descriptors.

access method
the communications protocol that the SAS(c) Drivers for ODBC use to exchange data with a SAS server. The driver currently supports the use of TCP/IP and Network DDE for remote data exchange, and DDE for local data exchange.

API
See application programming interface

application programming interface
a set of software functions that facilitate communication between applications and other kinds of programs or services. Short form: API.

data source name
a persistent identifier that is associated with a data source definition. The data source definition specifies how to locate and access a data source, including any authentication (such as a user name and password) that a user must supply in order to access the data. Short form: DSN.

data view
See SAS data view

database management system
a software application that enables you to create and manipulate data that is stored in the form of databases. Short form: DBMS.

DBMS
See database management system

DDE
See Dynamic Data Exchange

DSN
See data source name

Dynamic Data Exchange
a standard mechanism in the PC environment for sharing data among applications. Short form: DDE.

file DSN
a data source name that is stored completely within a file (unlike a user DSN or a system DSN, which are stored in the Windows registry).

ODBC
See Open Database Connectivity

ODBC driver
a loadable library module that provides a standardized interface for accessing, manipulating, and updating data that is created and maintained by a particular vendor's data management software. For example, the SAS(c) Drivers for ODBC enable you to access, manipulate, and update SAS data sources from any application that conforms to the ODBC standard.

Open Database Connectivity
an interface standard that provides a common application programming interface (API) for accessing data. Many software products that run in the Windows operating environment adhere to this standard so that you can access data that was created using other software products. Short form: ODBC.

SAS data view
a type of SAS data set that retrieves data values from other files. A SAS data view contains only descriptor information such as the data types and lengths of the variables (columns) plus other information that is required for retrieving data values from other SAS data sets or from files that are stored in other software vendors' file formats. Short form: data view.

SAS SPD Server
a SAS Scalable Performance Data Server. An SPD Server restructures data in order to enable multiple threads, running in parallel, to read and write massive amounts of data efficiently.

SAS/ACCESS view
a type of file that retrieves data values from files that are stored in other software vendors' file formats. You use the ACCESS procedure of SAS/ACCESS software to create SAS/ACCESS views.

SPD Server
See SAS SPD Server

system DSN
a data source name that can be accessed by any user of the system on which the data source is stored. System DSNs are stored in the Windows registry.

user DSN
a data source name that can be accessed only by the user who created it. User DSNs are stored in the Windows registry.

view descriptor
a SAS/ACCESS file that defines part or all of the DBMS data that is described by an access descriptor.

Index

T

U

V

W