

ODBC: Windows to the Outside World

*S. David Riba, JADE Tech, Inc., Clearwater, FL
Elisabeth A. Riba, JADE Tech, Inc., Clearwater, FL*

ABSTRACT

Until now SAS® datasets have remained largely inaccessible to non-SAS software packages. This has led to requests such as "How do I get my SAS data into Lotus 1-2-3, Excel, xBase, etc.?" While the SAS® System allows many different ways of retrieving non-SAS data, users have had only a limited ability to directly share their SAS data with other, non-SAS, applications and databases.

New to the SAS System for the Windows environment is support for ODBC. ODBC, which stands for Open Data Base Connectivity, is a Microsoft standard to allow software packages that are compliant with the ODBC specification to exchange information. Through the use of an ODBC engine, compliant software packages such as Lotus 1-2-3, Excel, and Paradox can now extract information directly from SAS datasets. In addition, SAS software can now extract information directly from ODBC compliant data sources.

INTRODUCTION

At SUGI 19 in 1994, SAS Institute announced support for Microsoft Corporation's Open Data Base Connectivity (ODBC) standard. The inclusion of ODBC support is part of the SAS Institute plan to provide data warehousing solutions for client/server environments.

The ODBC standard was established by Microsoft to create a common method of accessing information from a variety of data sources. ODBC allows applications to access different database file types whose drivers are installed the same way that Windows uses a common print driver routine to allow different applications to access any printer whose driver is installed.

WHAT IS ODBC?

ODBC is a software architecture developed by Microsoft that provides a common Application Programming Interface (API) to send requests to different database management systems (DBMS).

The standard way to communicate with relational databases is Structured Query Language (SQL). SQL is designed to allow users to request information (query) from a database management system on an ad-hoc basis. SQL commands are supported by many DBMS's.

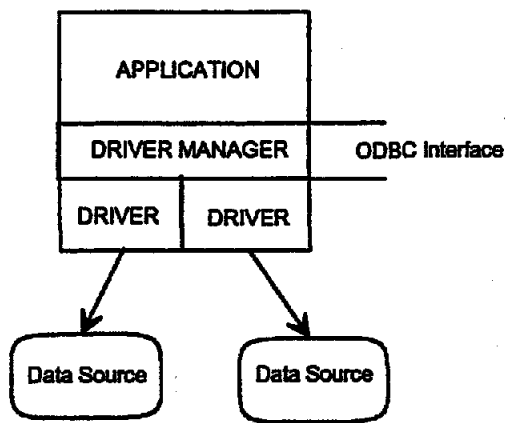
Since different database management systems can differ in the way they implement the same SQL functionality, there needed to be a common method of access across these differences. ODBC was designed to provide this common interface to different DBMS. It is based on the SQL Access Group's Client Library Interface (SAG-CLI) specification. The ODBC API allows programs to write common code for database access. This API is "translated" by software libraries called "drivers" into a format that is intelligible to the backend DBMS that is being accessed.

With ODBC, the client application is not required to know where the database is located (local or on a remote server), what communications method is used to get to the database (TCP/IP, DDE, etc.), or what vendor's DBMS is being accessed (DB2, Oracle, SQL Server, etc.). ODBC does require that all accesses to a DBMS use Structured Query Language (SQL). SQL queries form the common basis for communications between all of the different DBMS's that ODBC can access.

ODBC ARCHITECTURE

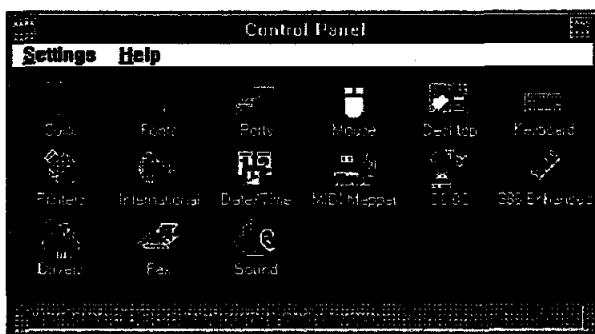
There are four components to ODBC:

- ♦ An **APPLICATION** such as SAS that accesses data through the ODBC drivers
- ♦ A **DRIVER MANAGER**, which is the intermediary that controls the drivers on behalf of an application
- ♦ **DRIVERS** which process ODBC requests and returns data to the application
- ♦ **DATA SOURCES**, which are the files or databases accessed by a driver (similar to an access descriptor in SAS/ACCESS)



Applications are Windows 3.x, Windows NT, and Windows for Workgroups software that includes support for ODBC. The application calls ODBC functions to connect to a data source, send and receive data, and disconnect from the datasource. The list is constantly changing, but currently includes such applications as SAS, Lotus Notes, Lotus Approach, and Microsoft Excel.

The **Driver Manager** was originally a separate component called the ODBC Administrator. It is now part of the Windows Control Panel as an icon just like the Printers and Fonts icons. It is supplied by Microsoft and is redistributed with ODBC drivers from other vendors. The purpose of the Driver Manager is to load vendor-supplied drivers and make them available to the ODBC compliant applications. The Driver Manager is a dynamic link library (DLL) which provides information to the application. *It is important to note that all ODBC applications communicate only with the Driver Manager and not directly with any individual drivers.*



Drivers operate between the Driver Manager and the network. They process ODBC function calls, manage all exchanges between an application and a specific DBMS, and may translate the standard SQL syntax into the native SQL of the target data source.

Only those Drivers that have been installed are displayed in the Control Panel. Additional drivers can be installed as needed. While most of the drivers are supplied by the database vendors themselves, there are third party companies such as Pioneer Software that are developing ODBC packs with drivers for many of the more popular database types such as Oracle, Sybase, DB2, etc.

The ODBC drivers provide either Single Tier or Multiple Tier functionality.

- ◆ A single tier driver allows access to data on a single (local) machine and serves as a SQL Engine. It parses the SQL statements, validates them, optimizes them, and retrieves the data from the underlying data source directly. It is typically used when the data source is an ISAM (Indexed Sequential Access Method) file, such as xBase, Btrieve, mdb, XLS, or text files. These types of data typically do not have an SQL interface, so the driver must provide the SQL functionality. Therefore, these drivers only provide a limited subset of SQL and do not support advanced features like transactions.
- ◆ A multiple tier driver allows access to data on both local (two tier driver) and remote (three tier driver) machines. The application that controls access to this data typically provide a SQL engine to parse, validate, optimize, and execute the SQL statements. The role of a multiple tier ODBC driver is to translate the function requests into a format that the DBMS can understand and to make the results available to the requesting application.

Since different data sources may provide different levels of support for SQL queries, a "lowest common denominator" approach would restrict the functionality of all ODBC drivers to the bare minimum. Instead, the ODBC API has defined three levels of compliance: CORE, Level 1, and Level 2 (or low, medium, and high functionality, respectively). These levels correspond to the functionality of the SQL operations that a driver can perform, such as read-only or read / write. When it receives a SQL query, the ODBC driver determines whether an application can support the functions requested (such as outer joins). Only those functions supported by the DBMS are passed through the query from the application. If the ODBC driver can not adequately perform a function, an error condition occurs.

The definitions for all installed drivers are maintained in the Windows directory in a file named ODBCINST.INI. There is one additional file created when drivers are installed: ODBCISAM.INI. This file contains the definitions of Single Tier (ISAM) data sources.

The SAS ODBC driver is a Multiple Tier, Level 1 driver. It is supplied as part of the Base product in Release 6.10 for Windows. It allows ODBC compliant applications to access, manipulate, and update SAS data sources. The SAS ODBC driver uses a SAS server (PROC ODBC SERVER) to access data from local SAS data sources. For remote data access, communications software (either TCP/IP or Network DDE) is required along with SAS/SHARE and SAS/SHARE*NET on the remote host.

The ODBC Control Panel displays a list of **Data Sources**. A data source is the ODBC term for a collection of data tables that a user can connect to. The data source defines the data to access, the associated DBMS, the platform on which the DBMS resides, and the network (if any) used to access that platform. An application can be connected to more than one data source at a time.

Data sources are listed in the ODBC.INI file in the Windows directory, and contain information about the file type and the directory of a database. To print output from Windows, the printer control panel can not connect simply to a "Laserjet 4", but it must connect to the "Laserjet 4 on port LPT1". Similarly, ODBC can not just connect to a database, but must connect to a specific file type in a specific directory.

The ODBC.INI file contains the definitions for all Data Sources that are currently defined.

There are two main sections to the ODBC.INI file:

- ♦ First is the section [ODBC Data Sources] which is the list that displays in the data source screen of the Control Panel
- ♦ Second is a separate section of definitions for each data source

SAS SYSTEM INTERFACES TO ODBC

In addition to the SAS ODBC driver which is part of Base SAS, the SAS/ACCESS® Interface to ODBC is a separately licensed product from SAS Institute to access other ODBC data sources. It was introduced in Release 6.08 of the SAS System. SAS/ACCESS to ODBC uses a data access method called SQL

Pass-Through to send PROC SQL commands directly to a database from within the SAS System.

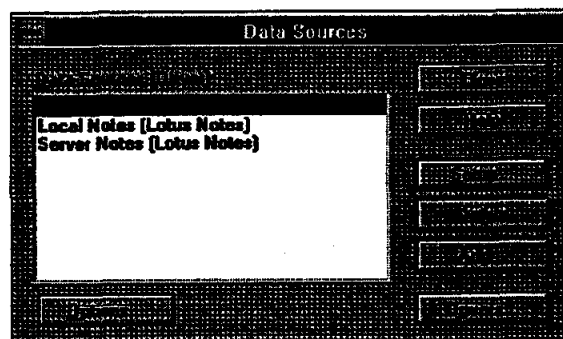
SAS/ACCESS® to ODBC is not a complete SAS/ACCESS engine. For example, it can not create view descriptors that can be accessed with a dataset. The primary purpose is to query ODBC data sources and subset the data into a SAS dataset. It has the capability to connect to an ODBC data source, execute SQL queries or other SQL statements, and then disconnect from the database.

For those sites that license SAS/ASSIST®, the Query window now has an access type for ODBC. This allows users to generate ODBC queries without the knowledge of SQL. SAS/ASSIST allows the user to define the tables to use, the data to select for a query, the additional logic needed, and then generates the SQL necessary for the query.

DEFINING THE SAS® ODBC DRIVER

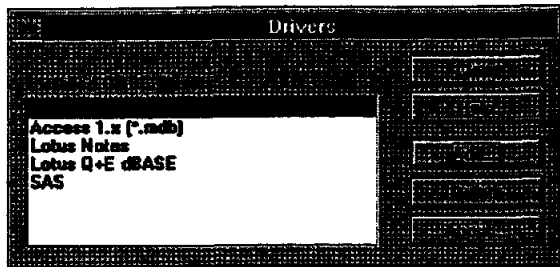
There are several steps necessary to setup and define the SAS ODBC driver for Windows. Once a Windows application has been installed which supports ODBC, such as Release 6.10 of the SAS System, an ODBC icon will be available in the Windows Control Panel. This ODBC icon is used to install ODBC drivers and to configure ODBC data sources.

When the ODBC icon is selected, a Data Sources window appears which contains a list of all defined data sources and their associated drivers. This list is based on the definitions in the ODBC.INI file. As new data sources are defined to Windows, they will be added to the Data Sources window. Before the SAS ODBC driver or data sources are defined, the Data Sources window may look like this:

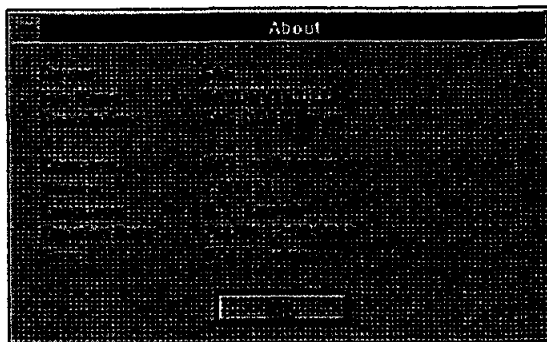


The **Drivers...** button will present a window with a list of all ODBC drivers that are currently installed

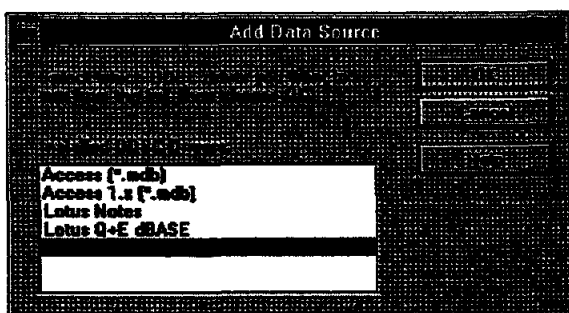
and available to ODBC. This window is used to manage the ODBC drivers and to install or delete drivers. Changes to the ODBC drivers will also change the definitions in the ODBCINST.INI file.



Information about any ODBC driver can be displayed by selecting the **About...** button. The following information was displayed For the SAS ODBC driver shipped with Release 6.10:

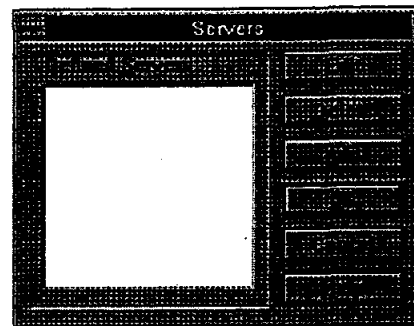


To define a new ODBC data source, select **Add...** from the Data Sources window. The list of currently installed ODBC drivers will be presented so the ODBC driver to use can be selected.

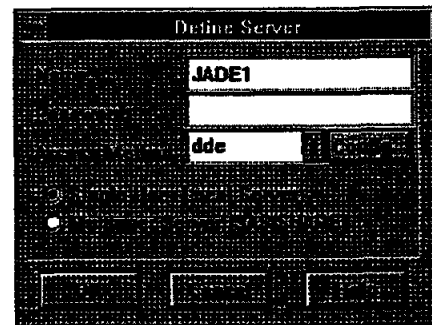


Since the SAS System uses a server (an ODBC server, SAS/SHARE, or SAS/SHARE*NET) to control the access to ODBC, one of two things will happen when the SAS ODBC driver is selected. If a server has already been defined, the SAS ODBC Setup window will be presented. If not, a server must be defined first.

If no servers are currently defined, the Servers window will look like the following:



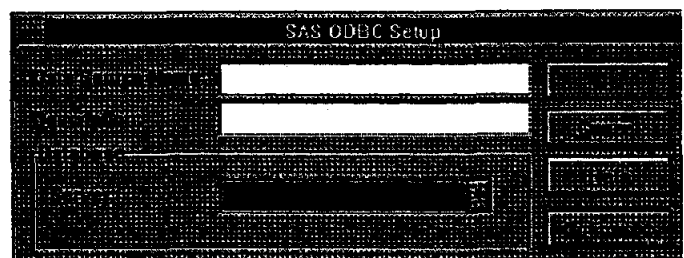
The **Edit** button allows existing server definitions to be modified. Select **Add** to create a new server definition.



The server name is required and a password is optional. There are two choices under **Access Method**: dde and tcp. Select DDE for a single user, non-network version of Windows. Select TCP to access remote data sources using TCP/IP or Network DDE. The **Config...** button allows for customization of the access method.

Additional servers can be added, or existing servers can be modified at any time through this procedure. They can also be selected from the **Servers...** button from the SAS ODBC Setup window.

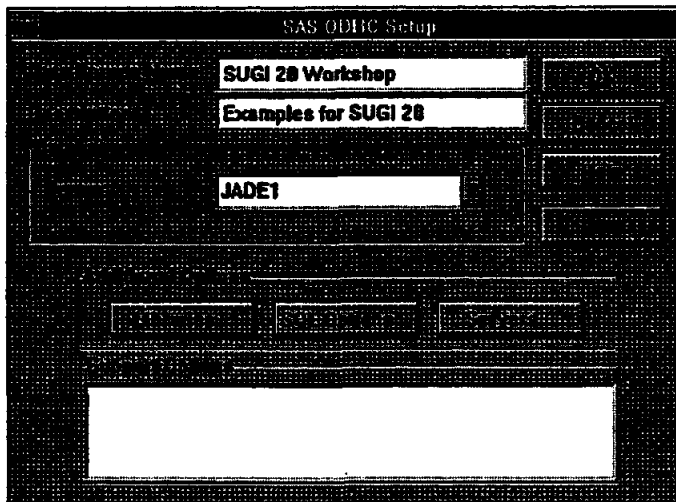
Once at least one server has been defined, individual ODBC data sources can be defined. They are defined from the SAS ODBC Setup window. The default window will look like the following:



- ♦ The **Data Source Name** field is required. It is used to define a descriptive name for the data source that will display in the ODBC Data Sources selection window.
- ♦ The **Description** field is optional and can be used to give a more informative description of the data source.
- ♦ The **Server** field is a drop-down list of all currently defined servers.

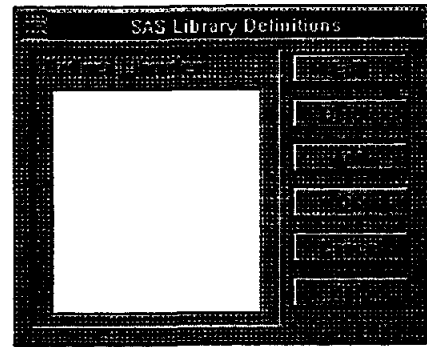
As soon as alphanumeric text is entered in the Data Source Name field, the **Options>>** button becomes available. It presents several additional options to further define this data source to ODBC.

- ♦ **Libraries...** is used to define SAS LIBNAME statements that will be valid with this data source. Multiple libraries can be defined for each data source.
- ♦ **SQL Options...** is used to change some default SQL options
- ♦ **Servers...** is used to add new servers or modify existing servers. When selected, this option presents the server definition windows discussed on the preceding page.

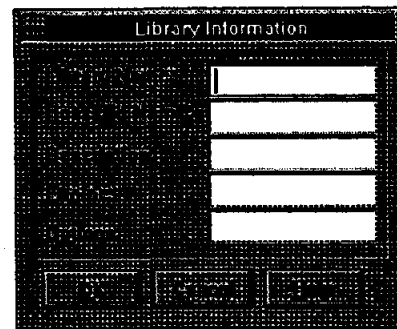


When the **Libraries** option is selected, a window of the currently defined LIBNAMES for this data source is presented. LIBNAMES can be added or deleted as required. These are the same as would normally be defined with a SAS LIBNAME statement, and are valid during the ODBC SAS session.

The default SAS Library Definitions window looks like the following:

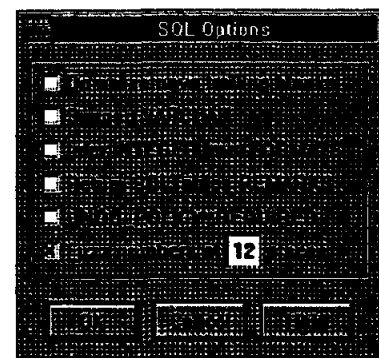


The following window is presented to **Add** a new Library definition, or to **Edit** an existing definition:



- ♦ **Library Name** defines the SAS LIBREF name for the SAS data library to be used.
- ♦ **Host File Name** defines the path and the name of the directory on the machine that the data file is stored.
- ♦ **Description** is an optional field to supply a description of the library.
- ♦ **Engine** is an optional field to define the SAS engine required to read or write to this library.
- ♦ **Options** is an optional field that allows several additional options to be defined.

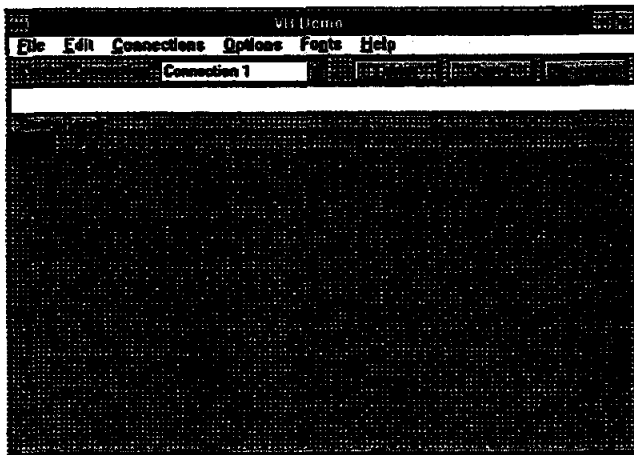
The **SQL Options** button from the SAS ODBC Setup window allows several SQL options to be modified for this data source. Each option may be selected or deselected as desired.



TESTING THE SAS ODBC INSTALLATION

Once the SAS ODBC driver has been installed and SAS data sources have been defined, it is prudent to test the setup and validate that everything is operating correctly. An easy way to do this is a Visual Basic application that can be found on the SAS Institute BBS (SIBBS) or by anonymous ftp from ftp.sas.com. It is a self extracting file named ODBCDEMO.EXE.

By invoking the program VBDEMO.EXE from within Windows, the following will be displayed:



VBDEMO will allow simple SQL queries to SAS and other ODBC defined data sources.

USING THE SAS ODBC DRIVER

Once SAS data sources have been defined with the SAS ODBC driver, any ODBC compliant application can query those data sources. The specific procedures to set up an ODBC connection will differ depending on the application involved. Some applications, like Lotus Approach, allow the user to directly open an ODBC data source the same as any other file. Other applications, like Lotus 1-2-3, allow connections to ODBC data sources through less direct means. Consult the documentation for each application for specific instructions.

Regardless of the method needed to access SAS ODBC data, the SAS ODBC driver provides a common interface to the SAS System. There are some differences in operation depending on whether SAS is being accessed locally (using DDE) or remotely (using TCP/IP or Network DDE). The

access method to be used by the SAS ODBC driver (DDE or TCP/IP) is defined and configured from the Define Server window which is accessible from the ODBC Setup window.

To connect to SAS data sources from other applications, the SAS ODBC driver communicates with the SAS System through a server. For a local DDE session, PROC ODBC SERVER is invoked. For a TCP/IP or Network DDE session, PROC SERVER is invoked.

When the ODBC Driver Manager calls the SAS ODBC driver to establish a connection to SAS data sources, the ODBC driver first initiates a SAS session and starts the server. The parameters specified in the server configuration window are used by the ODBC driver to define and start a SAS session. If the SAS System is not currently running, then a SAS session is started and the server is invoked. If a SAS session is already running but the server is not currently running, then it is necessary to manually start the server for the ODBC connection to complete.

The SAS code necessary to start the server is similar for DDE or TCP/IP.

For DDE submit the following:

```
OPTIONS COMAMID = DDE ;
PROC ODBC SERVER ID = server-name ;
RUN ;
```

For TCP/IP submit the following:

```
OPTIONS COMAMID = TCP ;
PROC SERVER ID = server-name ;
RUN ;
```

There are additional requirements to configure the remote server. Consult the *SAS® ODBC Driver Technical Report* for specific instructions for configuring TCP/IP and Network DDE connections.

While PROC ODBC SERVER is running in a local DDE SAS session, SAS System resources are totally committed to the server. No other SAS operations can be performed until the server is terminated. Since multiple SAS sessions can be invoked on networked SAS, using TCP/IP or Network DDE, another SAS session can be started while the ODBC-connected session is tied up.

Currently there are only two methods to terminate the SAS ODBC server from a local DDE session.

Even after the ODBC data source has been disconnected from the client application, the SAS server does not complete its task and return the user to the SAS session. The only options are:

- ♦ Terminate the SAS session from the Windows Task Manager
- ♦ Terminate the server by entering Ctrl and Break from the SAS session

USING SAS/ACCESS TO ODBC

SAS/ACCESS to ODBC is a separately licensed part of the SAS System. It installs the same as other SAS/ACCESS modules. However, unlike other SAS/ACCESS modules, the purpose of the ODBC interface is solely to communicate with the ODBC Driver Manager and pass SQL requests to it. The ODBC interface is not a full SAS/ACCESS engine. It does not use view descriptors, so a SAS view into an ODBC data source is not possible.

Once installed, SAS/ACCESS to ODBC submits PROC SQL queries to the ODBC Driver Manager for processing and receives the results of the query back from the ODBC Driver Manager. This communications with the ODBC Driver Manager is via the SQL Pass-Through facility. It is possible to store SQL queries as SQL views, and SQL queries can be invoked as part of a SAS program, for example within a SAS/AF application.

While it is not necessary to learn SQL to use ODBC (there are alternatives like SAS/ASSIST), it is useful to understand some of the basic SQL commands that are used by SAS/ACCESS to ODBC. For a more complete understanding of the SQL language and syntax, there are a number of excellent tutorials and other training on the subject.

SQL SYNTAX FOR SAS ODBC REQUESTS

Since a SQL query must be interpreted by drivers on both sides of a query, the ODBC specifications mandate a lowest-common denominator approach to queries. Each ODBC driver interprets as much of a SQL query as the underlying software understands. Requests for functionality that the underlying software is unable to perform (for example, certain types of JOINS) are ignored. As required, the ODBC driver also performs certain conversions, such as from one data type to another.

The SQL Procedure Pass-Through facility consists of three PROC SQL statements and one component: The three SQL statements are:

- ♦ CONNECT to establish a connection with the DBMS
- ♦ EXECUTE to send dynamic, non-query DBMS-specific SQL statements to the DBMS
- ♦ DISCONNECT to terminate the connection with the DBMS

The SQL component is:

- ♦ CONNECTION TO in the FROM clause of the PROC SQL SELECT statement to retrieve data directly from the DBMS

To begin an ODBC connection from a SAS session, it is necessary to establish a connection to the ODBC Driver Manager. This is done from PROC SQL with the CONNECT statement:

```
PROC SQL ;  
CONNECT TO ODBC .....
```

There are several options that can be specified on the CONNECT statement. They are:

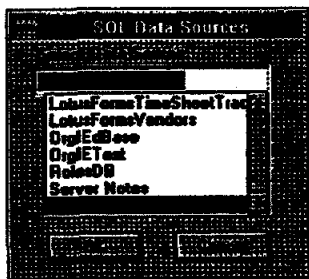
- ♦ AS *alias* (optional) SAS alias
- ♦ DSN = "*data source*" the ODBC data source name defined in the ODBC setup
- ♦ UID = "*user id*" (optional)
- ♦ PWD = "*password*" (optional)
- ♦ PROMPT dialog box prompt
- ♦ NOPROMPT no dialog box
- ♦ COMPLETE allows additional information or a change to the request if the SQL request is incomplete
- ♦ REQUIRED allows additional information but no change to the request if the SQL request is incomplete
- ♦ LOG any warnings are written to the SAS log

Either a DSN = option or the PROMPT, NOPROMPT, COMPLETE, LOG, or REQUIRED arguments must be specified on the CONNECT statement. All SQL arguments are enclosed in parenthesis and can be combined as necessary. The SAS System implementation is not case sensitive. However, since some DBMS's are case sensitive the SQL Pass-Through Facility treats all DBMS names and text as if they were.

For example, to request an ODBC connection where the user can select the data source:

```
PROC SQL ;
  CONNECT to ODBC (PROMPT) ;
```

The PROMPT option will present the user with a selection list of all currently defined SQL Data Sources. The user can scroll through the list and select a data source to use for this connection. Note in the example below that 'SUGI 20 Workshop' was the Data Source Name which had been defined in the SAS ODBC Setup window.



To define an alias, *SUGI*, and to specify an ODBC data source to connect to (*SUGI 20 Workshop*):

```
PROC SQL ;
  CONNECT to ODBC as SUGI
    (dsn = "SUGI 20 Workshop") ;
```

Any text within the quotation marks is not parsed by the SQL Pass-Through facility, but is passed intact to the ODBC Driver Manager

Once a connection has been made to an ODBC data source, other SQL statements can be used to define the query. The **CONNECTION TO** component of the **SELECT** statement is used to point the **SELECT** statement to ODBC. For example, to make a query against a data source:

```
SELECT * from CONNECTION TO SUGI
  (SELECT * from data-source) ;
```

This SQL statement points the query to the ODBC connection defined with the alias *SUGI*. The statement within the parenthesis is passed intact to the ODBC driver to perform a query against the data source named in the statement.

To create a SAS dataset from an ODBC data source, the **CREATE** statement can be used to define either a **VIEW** or a **TABLE**. To generate a SQL view from the ODBC data source, the above example might be modified as follows:

```
CREATE view sqlview as
SELECT * from CONNECTION TO SUGI
  (SELECT * from data-source
    where some-where-clause) ;
```

It is important to note that even though the SQL query might be a valid SAS SQL query, it might request actions that the SQL engine on the other side of ODBC is incapable of processing. The SAS LOG will contain information about the results of each query made to the ODBC driver.

The return codes that are generated by SQL error conditions are available through two SAS macro variables:

- ◆ SQLXRC SQL Return Code
- ◆ SQLXMSG Descriptive RC Message

The SQL return codes and messages can be printed to the SAS LOG with the %put macro statement:

```
%put &sqlxrc ;
%put &sqlxmsg ;
```

The **EXECUTE** statement is used to send dynamic DBMS-specific SQL statements to the DBMS and to process those statements. These statements can perform actions on a non-SAS DBMS data source, such as creating a DBMS table, deleting a DBMS table, adding or deleting rows, or modifying data in the table. For example, to **DROP** a table from the ODBC data source with the alias *SUGI*:

```
EXECUTE (drop table tablename)
  BY SUGI ;
```

The **DISCONNECT** statement terminates the connection with the DBMS. If an alias has been defined for the connection, use the alias. Otherwise, use the name you originally defined on the **CONNECT** statement (ODBC).

```
DISCONNECT from SUGI ;
```

There is an implied **DISCONNECT** when **PROC SQL** terminates.

In addition to the above, there are several special ODBC queries that are supported. These queries communicate directly with the ODBC API to extract information from both SQL and non-SQL data sources. SAS/ACCESS to ODBC supports the following special queries:

- ♦ ODBC::SQLTables
- ♦ ODBC::SQLColumns
- ♦ ODBC::SQL ColumnPrivileges
- ♦ ODBC::SQLForeignKeys
- ♦ ODBC::SQLPrimaryKeys
- ♦ ODBC::SQLProcedureColumns
- ♦ ODBC::SQLProcedures
- ♦ ODBC::SQLStatistics
- ♦ ODBC::SQLTablePrivileges

The above has been a brief summary of the SQL options that can be used with ODBC. For more complete details refer to *SAS/ACCESS® Software Changes and Enhancements: SQL Procedure Pass-Through Facility*. For more information about SQL refer to the *SAS® Guide to the SQL Procedure*.

CONCLUSION

Now that the SAS System provides support for the Microsoft ODBC standard, it is easier to share information between the SAS System and non-SAS applications. Through the use of the SAS ODBC driver, SAS/ACCESS to ODBC, and SQL, different applications can exchange data regardless of data format or location. An understanding of how the SAS System can work with ODBC compliant data is becoming an essential cornerstone to the successful implementation of a client / server environment.

GLOSSARY

There were enough acronyms used in this paper to make a Glossary useful for clarity. Below is an ad-hoc **GLOSSARY OF ACRONYMS** used and their meanings. Note that the descriptions are my own, and should not be considered definitive.

API	Applications Programming Interface a library of functions that one computer program can use to communicate with another program
DBMS	Data Base Management System a computer application that stores data in an organized way. Examples of a DBMS include Oracle, Sybase, SAS, and DB2.
DDE	Dynamic Data Exchange a method for Windows programs to exchange information
DLL	Dynamic Link Library a program module called by a Windows application
ftp	File Transfer Protocol protocol used to transfer files over the Internet
ODBC	Open DataBase Connectivity a Microsoft standard for exchanging SQL requests to access data from a variety of DBMS's
SAG-CLI	SQL Access Group Client Library Interface (also SQL Access Group Call Level Interface) a standard under development to define common SQL functions
SIBBS	SAS Institute Bulletin Board System
SQL	Structured Query Language language used to query and update DBMS data
TCP/IP	Terminal Communications Protocol / Internet Protocol a method for networked computers to exchange information

REFERENCES

SAS Institute, Inc., *SAS® ODBC Driver Technical Report: User's Guide and Programmers Reference*, Release 6.10, Cary, NC. SAS Institute Inc., 1994

SAS Institute, Inc., *SAS/ACCESS® Software Changes and Enhancements: SQL Procedure Pass-Through Facility*, Version 6, Cary, NC. SAS Institute Inc., 1994

SAS Institute, Inc., *SAS® Guide to the SQL Procedure: Usage and Reference*, Version 6, First Edition, Cary, NC. SAS Institute Inc., 1989

Barnhart, Andy (1994). Sharing Enterprise Data with ODBC. *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. SAS Institute, Inc., Cary, NC. pp. 284-288.

Dickson, Alan and Pass, Ray (1994). Select Items from PROC SQL Where Items > Basics. *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. SAS Institute, Inc., Cary, NC. pp. 1440-1449.

Mehler, Gary (1994). Integrating the SAS® System for Personal Computers into Your Enterprise. *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. SAS Institute, Inc., Cary, NC. pp. 428-437.

Wallace, Jack (1994). New Features for Remote and Open Access to Enterprise Data: Enhanced Distributed Data Access Through Multiple Engine Architecture. *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. SAS Institute, Inc., Cary, NC. pp. 258-267

Venkatrao, Murali. Microsoft ODBC Interface: A Tutorial. Microsoft Corporation, 1993.

ACKNOWLEDGEMENTS

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. * indicates USA registration. Lotus 1-2-3, Notes, and Approach are trademarks of Lotus Development Corporation. ODBC and Windows are trademarks of Microsoft Corporation. Other brand and product names are registered trademarks or trademarks of their respective companies.

The author would like to thank Andy Kuligowski and Lindi Ingold for their time, patience, and invaluable assistance.

AUTHORS

The authors may be contacted at:

S. David Riba
JADE Tech, Inc.
P O Box 4517
Clearwater, FL 34618
(813) 726-6099
INTERNET: 0006495817@mcimail.com

Elisabeth A. Riba
JADE Tech, Inc.
P O Box 4517
Clearwater, FL 34618
(813) 726-6099
INTERNET: lis@netcom.com