SQL Server Driver

For All Users

The following topics discuss the SQL Server driver and how to install it for use by an application.

Overview

Hardware and Software Requirements

Setting Up the SQL Server Driver

Installing the Catalog Stored Procedures

Adding, Modifying, and Deleting SQL Server Data Sources

Connecting to a SQL Server Data Source

Troubleshooting

For Advanced Users

The following topics discuss how to use the SQL Server driver directly.

Connection Strings (Advanced)

SQL Statements (Advanced)

Data Types (Advanced)

Error Messages (Advanced)

For Programmers

The following topics discuss how to use the SQL Server driver programmatically. They are intended for application programmers and require knowledge of the Open Database Connectivity (ODBC) application programming interface (API).

SQLGetInfo Return Values (Programming)

ODBC API Functions (Programming)

Implementation Issues (Programming)

SQL Server Driver-Specific Connection Options (Programming)

What's New

For a listing of the new features and issues documented in this help file, and jumps to topics documenting the features and issues, click here.

What's New

The following features or characteristics of the SQL Server driver are new or have changed for version 2.0. The topics listed under each feature document the changes. Click on the topic name to jump to the topic text.

Version Updates

Hardware and Software Requirements

Addition of 32-Bit Driver

Hardware and Software Requirements
Overview

Addition of Cursor Library

Overview
Cursor Library (Programming)
Active *hstmt* Definition (Programming)
Error Messages (Advanced)

Setup Option Changes

SQL Server Driver-Specific Connection Options
    Fast Connect Option
    Generate Stored Procedures for Prepared Statements
ODBC SQL Server Setup Dialog Box
SQLConfigDataSource Implementation (Programming)
Procedure Invocation Limitations
SQLPrepare Implementation (Programming)

Connection String Changes

Connection String (Advanced)
SQLConnect Implementation
SQLDriverConnect Implementation
SQLBrowseConnect Implementation

SQL Grammar Support Changes

SQL Statements (Advanced)
Limitations to ODBC SQL Grammar (Advanced)
    New Scalar Functions
    Create-Index Statement
    Date Translation
    NULLCHAR and NULLBINARY Columns
Limitations to ODBC SQL Grammar (Programming)
    Data-at-Execution Parameters
    Outer Join
    Procedure Invocation Limitations
    Remote Procedure Calls
Unsupported ODBC SQL Grammar (Advanced)
    DAYOFWEEK Scalar Function
    USER Keyword

ODBC API Function Support Changes

ODBC API Functions (Programming)
Implementation of ODBC API Functions (Programming)
Limitations to ODBC API Functions (Programming)

SQLBindParameter
SQLCancel
SQLGetConnectOption/SQLSetConnectOption
SQLGetStmtOption/SQLSetStmtOption
SQLTables

SQLGetInfo Return Values (Programming)

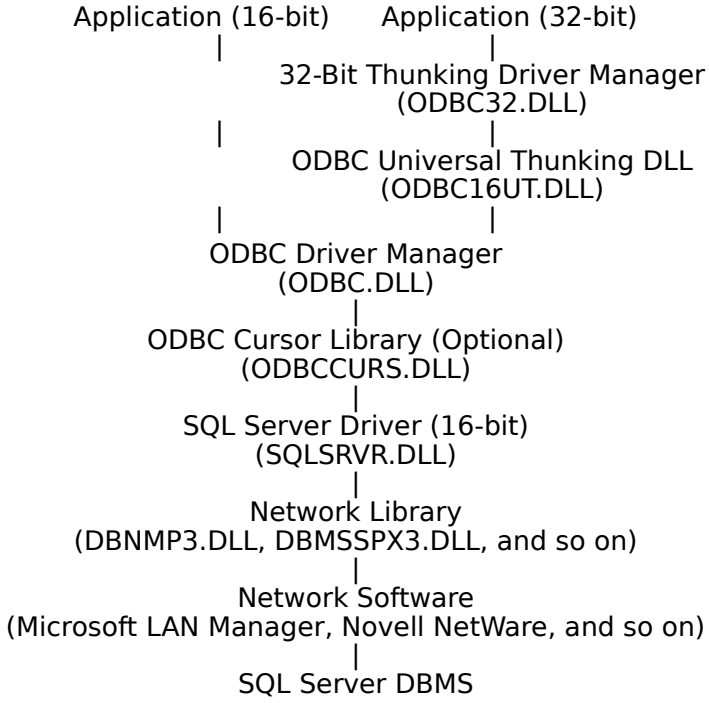
Overview

See Also

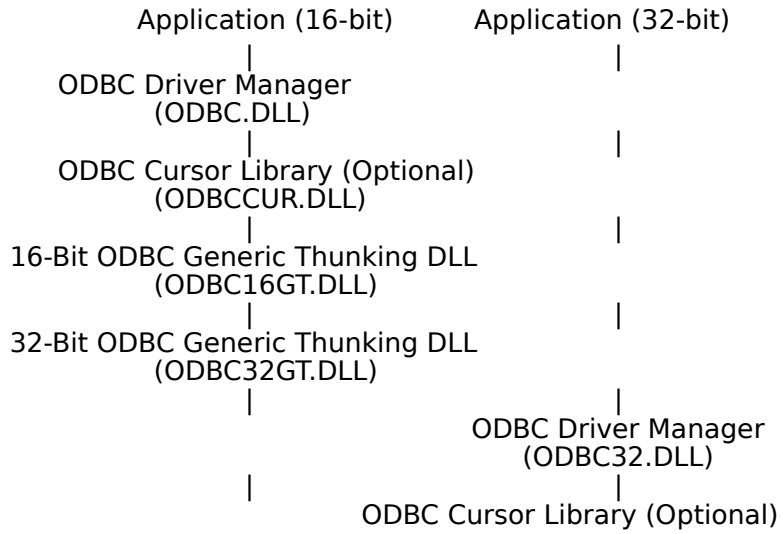
SQL Server is a multi-user relational database management system (DBMS) that runs on local area networks. Microsoft SQL Server runs on the OS/2 and Windows NT operating systems. Both 16- and 32-bit versions of the driver are available. Structured Query Language (SQL) is used to access data in a SQL Server database. Client workstations communicate with SQL Server across a network such as Microsoft LAN Manager, Novell NetWare, Banyan VINES, or a TCP/IP network.

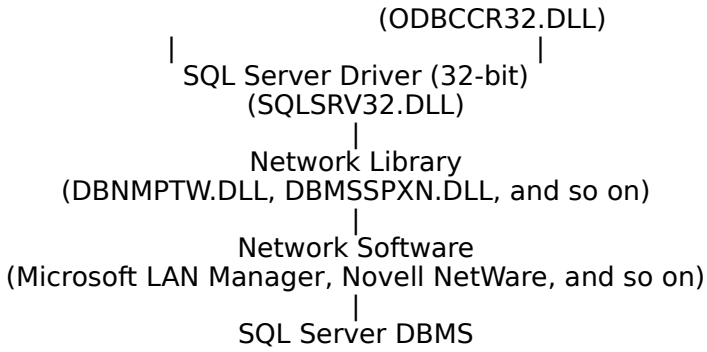
The SQL Server driver enables applications to access data in Microsoft SQL Server databases through the Open Database Connectivity (ODBC) interface.

The application/driver architecture for 16-bit environments is:



The application/driver architecture for 32-bit environments is:




(ODBCCR32.DLL)
SQL Server Driver (32-bit)
(SQLSRV32.DLL)
Network Library
(DBNMPTW.DLL, DBMSSPXN.DLL, and so on)
Network Software
(Microsoft LAN Manager, Novell NetWare, and so on)
SQL Server DBMS

See Also

For All Users

- Adding, Modifying, and Deleting SQL Server Data Sources
- Connecting to a SQL Server Data Source
- Hardware and Software Requirements
- Setting Up the SQL Server Driver

Hardware and Software Requirements

See Also

To access SQL Server data, you must have:

- The SQL Server driver.
- A SQL Server DBMS.
- A network connecting the computers on which these reside.

The following paragraphs describe the hardware and software required by each of these components.

SQL Server Driver

The SQL Server driver requires the following hardware:

- An Industry Standard Architecture (ISA) computer, such as the IBM PC/AT or compatible, or
- A Micro Channel Architecture (MCA) computer, such as an IBM PS/2 or compatible, or
- An Extended Industry Standard Architecture (EISA) computer with an 80286, 80386, or 80486 microprocessor.
- At least 2 megabytes of random-access memory (RAM); 4 MB of RAM are recommended.
- A hard disk drive and approximately 200 kilobytes of hard disk space for the SQL Server driver and ODBC Driver Manager (approximately 300 kilobytes with the cursor library installed).

The SQL Server driver requires the following software:

- MS-DOS version 3.3 or later
- Microsoft Windows version 3.1 or later (16-bit driver), or Microsoft Windows NT 3.1 or later (32-bit driver)
- ODBC Driver Manager version 2.0 or later (ODBC.DLL for 16-bit or ODBC32.DLL for 32-bit)

SQL Server

To access data in SQL Server with the SQL Server driver, you must have Microsoft SQL Server version 1.2 or later. The catalog stored procedures must be installed on your copy of SQL Server. You may need to install the catalog stored procedures shipped with this driver on versions 4.2 and earlier of Microsoft SQL Server for OS/2. For information about the hardware and software required by SQL Server, see the *Microsoft SQL Server Configuration Guide* (for Windows NT), or the *Microsoft SQL Server Installation Guide* (for OS/2).

Network Software

A network is required to connect the platforms on which SQL Server and the SQL Server driver reside. To connect to Microsoft SQL Server, you can use Microsoft Windows for Workgroups; Microsoft LAN Manager or a compatible network, such as IBM LAN Server or DEC Pathworks; Novell NetWare; TCP/IP (Windows NT only); or Banyan VINES (OS/2 only). For information about the hardware and software required by each network, see that network's documentation.

The SQL Server driver communicates with the network software through the SQL Server Net-Library interface and requires a Net-Library dynamic-link library (DLL). The following table lists the network library DLLs that can be used with each network for Microsoft SQL Server.

Note that the 32-bit SQL Server driver is thread-safe.

With this network	Use one of these DLLs	Shipped with this package
Microsoft Windows NT; Microsoft Windows for	DBNMP3.DLL (16-bit) DBNMPTW.DLL (32-bit)	SQL Server driver; Microsoft SQL Server

Workgroups; Microsoft LAN Manager and compatibles, such as IBM LAN Server or DEC Pathworks		
Novell NetWare	DBMSSPX3.DLL (16-bit) DBMSSPXN.DLL (32-bit)	Microsoft SQL Server for Windows NT; Network Integration Kit for Novell NetWare
Banyan VINES	DBMSVIN3.DLL (16-bit) DBMSVINN.DLL (32-bit)	Microsoft SQL Server for Windows NT; Network Integration Kit for Banyan VINES (for OS/2)
TCP/IP networks*	DBMSSOC3.DLL (16-bit) DBMSSOCN.DLL (32-bit)	Microsoft SQL Server 4.21 (for Windows NT)

*DBMSSOC3.DLL and DBMSSOCN.DLL use the Windows Sockets interface. Contact your TCP/IP vendor to find out if this interface is supported on your network.

See Also

For All Users

Setting Up the SQL Server Driver

Setting Up the SQL Server Driver

See Also

To set up the SQL Server driver

1. If you have Microsoft SQL Server (for OS/2) version 4.2 or earlier, follow the instructions for installing the catalog stored procedures.
2. Add a data source for each copy of SQL Server in which you want to access data.

To set up a new version of the SQL Server driver

1. In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

   Note For Microsoft Windows NT when using 16-bit drivers, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

   The Data Sources dialog box is displayed.
2. In the Data Sources dialog box, choose the Drivers button.

   The Drivers dialog box is displayed.
3. In the Drivers dialog box, choose the Add button.

   The Add Driver dialog box is displayed.
4. In the text box, type the name of the drive and directory containing the SQL Server driver in the text box. Or choose the Browse button to select a drive and directory name.
5. In the Add Driver dialog box, choose the OK button.

   The Install Drivers dialog box is displayed
6. In the Available ODBC Drivers list, select SQL Server.
7. Choose the OK button.

   The SQL Server driver is installed.

To delete the SQL Server driver

1. In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

   Note For Microsoft Windows NT when using 16-bit drivers, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

   The Data Sources dialog box is displayed.
2. In the Data Sources dialog box, choose the Drivers button.

   The Drivers dialog box is displayed.
3. In the Installed ODBC Drivers list, select SQL Server.
4. Choose the Delete button.

   A message asks you to confirm that you want to remove the driver and all of the data sources that use the driver.
5. Choose the Yes button.

See Also

For All Users

Adding, Modifying, and Deleting SQL Server Data Sources

Hardware and Software Requirements

data source (SQL Server)

A data source includes the data a user wants to access and the information needed to get to that data. For the SQL Server driver, a data source is a SQL Server database, the server on which it resides, and the network used to access that server.

Installing the Catalog Stored Procedures

The SQL Server driver uses a set of system stored procedures, known as the catalog stored procedures, to obtain information from the SQL Server system catalog. With Microsoft SQL Server for Windows NT and Microsoft SQL Server (for OS/2) version 4.2a and later, the catalog stored procedures are installed automatically when you install or upgrade SQL Server, and select the procedures for installation. For earlier versions of Microsoft SQL Server, your system administrator must install these stored procedures unless they have already been installed.

To install the catalog stored procedures

1. Insert the disk on which the SQL Server driver was shipped into drive A.
2. Run the INSTCAT.SQL batch file in the **isql** utility.

   C:> ISQL /U sa /P *sa-password* /S *server-name* /i A:\INSTCAT.SQL

   The arguments in this format are:

Argument	Meaning
sa-password	The system administrator's password.
server-name	The name of the server on which SQL Server resides.

Note To run **isql**, your computer must be installed as a client workstation for SQL Server.

Note The INSTCAT.SQL file will generate warning messages. These can be ignored.

Note The INSTCAT.SQL batch file will fail if there is not enough room in the master database to store the catalog stored procedures or to log the changes to existing procedures. To create more room, your system administrator can dump the transaction log or remove unused non-system stored procedures and tables from the master database. Your system administrator can also back up the master database and expand its size. For more information, see the SQL Server documentation.

The SQL Server driver uses some or all of the following catalog stored procedures.

This procedure	Returns
sp_column_privileges	Information about column privileges for the specified table or tables.
sp_columns	Information about columns for the specified table or tables.
sp_databases	A list of databases.
sp_datatype_info	Information about the data types.
sp_fkeys	Information about logical foreign keys.
sp_pkeys	Information about primary keys.
sp_server_info	A list of attribute names and matching values for the server.

sp_special_columns	Information for a single table about columns in the table that have special attributes.
sp_sproc_columns	Column information for a stored procedure.
sp_statistics	A list of indexes for a single table.
sp_stored_procedures	A list of stored procedures.
sp_table_privileges	Information about table privileges for the specified table or tables.
sp_tables	A list of objects that can be queried.

You can find additional information about the catalog stored procedures in the *Microsoft SQL Server Transact-SQL Reference*.

Adding, Modifying, and Deleting SQL Server Data Sources

See Also

Before you can access data with the SQL Server driver, you must add a data source for each of your copies of SQL Server. The SQL Server driver uses the information you enter when you add the data source to access the data. You can change or delete a data source at any time.

To add a SQL Server data source

1. In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

   Note For Microsoft Windows NT when using 16-bit drivers, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

2. In the Data Sources dialog box, choose the Add button.

   The Add Data Source dialog box is displayed.

3. In the Installed ODBC Drivers list, select SQL Server and choose the OK button.

   The ODBC SQL Server Setup dialog box is displayed.

4. In the ODBC SQL Server Setup dialog box, set the option values as necessary and choose the OK button.

To modify a SQL Server data source

1. In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

   Note For Microsoft Windows NT when using 16-bit drivers, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

2. In the Data sources dialog box, select the data source from the Data Sources list and choose the Setup button.

   The ODBC SQL Server Setup dialog box is displayed.

3. In the ODBC SQL Server Setup dialog box, set the option values as necessary and choose the OK button.

To delete a SQL Server data source

1. In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

   Note For Microsoft Windows NT when using 16-bit drivers, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

2. In the Data Sources dialog box, select the data source you want to delete in the Data Sources list.
3. Choose the Delete button, and then choose the Yes button to confirm the deletion.

   You may be asked if you want to remove the data source name and its associated information from the WIN.INI file. Other applications that call SQL Server programmatically may use this information to connect to SQL Server data sources.

4. Choose the Yes button if you are certain that no other applications use the information about the data source; otherwise, choose the No button.

To add, modify, or delete a SQL Server data source dynamically

You can call the SQLConfigDataSource API function to add, modify, or delete a data source dynamically. This function uses keywords to set connect options that in the above procedures are set through the ODBC SQL Server Setup Dialog Box. This function

should be used when you want to add, modify, or delete a data source without displaying the setup dialog box.

See Also

For All Users

- Connecting to a SQL Server Data Source
- Setting Up the SQL Server Driver
- SQL Server Driver-Specific Connection Options (Programming)
- SQLConfigDataSource Implementation (Programming)

Connecting to a SQL Server Data Source

See Also

As part of the connection process, an application can prompt you for information. If an application prompts you for information about a SQL Server data source, do the following:

To connect to a SQL Server data source

1. If there is a Server combo box (rather than a Data Source line), enter or select the server name.
2. In the Login ID box, type your login ID for SQL Server.
3. In the Password box, type your password for SQL Server.
4. Choose OK.

Although it is not required, you can enter additional connection information, such as the database to access and the language for SQL Server to use.

To connect to a SQL Server data source with additional information

1. Follow steps 1 through 3 above, and then choose the Options button.
2. Select the name of the database you want to access in the Database box. When the list box is first opened, the selection highlights the user's default database.
3. Select the name of language for SQL Server to use in the Language box. This option is unavailable if you're using a version of SQL Server earlier than version 4.2. When the list box is first opened, the selection highlights the user's default language.
4. Enter an application name if the displayed application name is incorrect. The application name is the name of the application that is calling the SQL Server driver.
5. Enter a workstation ID if the displayed workstation ID is incorrect. Typically, this is the network name of the computer on which the application resides.
6. Choose OK.

See Also

For All Users

Adding, Modifying, and Deleting SQL Server Data Sources

For Advanced Users

Connection Strings (Advanced)

For Programmers

SQLBrowseConnect Implementation (Programming)

SQLDriverConnect Implementation (Programming)

Troubleshooting

The following sections discuss how to solve problems you might encounter while using the SQL Server driver.

Trouble establishing or maintaining connections to servers

See the *Troubleshooting Guide* included with Microsoft SQL Server for tips on resolving connection problems.

Procedures named "odbc#..." left in sysobjects table

The SQL Server driver creates a procedure when it prepares a statement for execution. Normally, the SQL Server driver deletes any procedures it created when it disconnects from a data source. However, if the connection between the driver and SQL Server terminates abnormally, as in the case of a power failure, these procedures are left on SQL Server.

These procedures are named "odbc#<*user*><*identifier*>", where <*user*> is up to 15 characters of the user name and <*identifier*> is up to 10 digits that identify the prepared SQL statement. They are created in the sysobjects table in the current database. Any procedures abnormally left in this table by the SQL Server driver can be safely dropped by your SQL Server administrator.

ODBC SQL Server Setup Dialog Box

See Also

The ODBC SQL Server Setup dialog box has the following options.

Data Source Name

A name by which you will identify the data source. For example, "Personnel Data."

Description

A description of the data in the data source. For example, "Hire date, salary history, and current review of all employees."

Server

The name that identifies SQL Server on your network. You can select a server from the list or enter the server name. For more information, click your network name.

Banyan VINES

Microsoft Windows for Workgroups, Microsoft LAN Manager, and Compatibles (for example, IBM LAN Server or DEC Pathworks)

Novell NetWare

TCP/IP Networks (Sockets)

"(local)" can be entered as the server on a Microsoft Windows NT network. The user can then use a local copy of SQL Server, even when a non-networked version of SQL Server is used.

Note If you are configuring using Microsoft LAN Manager, you are using the SQL Server-supplied network library. You will only see servers that are in your workgroup/domain.

Network Address

An address that specifies the location of the SQL Server database management system (DBMS) from which the driver will retrieve data. For Microsoft SQL Server, you can usually leave this value set to (Default). For information on network address, click your network name.

Banyan VINES

Microsoft Windows for Workgroups, Microsoft LAN Manager, and Compatibles (for example, IBM LAN Server or DEC Pathworks)

Novell NetWare

TCP/IP Networks (Sockets)

Network Library

The name of the SQL Server Net-Library DLL that the SQL Server driver uses to communicate with the network software. If the value of this option is (Default), the SQL Server driver uses the library for the Default Network specified in the SQL Server Client Configuration Utility. For information on the network library to use, click your network name.

Banyan VINES

Microsoft Windows for Workgroups, Microsoft LAN Manager, and Compatibles (for example, IBM LAN Server or DEC Pathworks)

Novell NetWare

TCP/IP Networks (Sockets)

Options

To access the following fields, click the Options button.

Database Name

The name of the SQL Server database.

Language Name

The national language to be used by SQL Server. This is used only by SQL Server versions 4.2 and later. The selected language must have been installed on the server by the System Administrator.

Generate Stored Procedures for Prepared Statements

When this option is selected (the default), stored procedures are created for prepared statements The SQL Server driver prepares a statement by placing it in a procedure and compiling that procedure. When this option checkbox is cleared, the creation of stored procedures for prepared statements is disabled. In this case, a prepared statement will be stored and re-executed at execution time.

Fast Connect Option

Select this option to disable the execution of ODBC informational queries at connect time. These queries retrieve information required to implement meta data support, including retrieving UDT (user-defined datatypes) information. The Fast-Connect option reduces the connect time required. The option is designed to increase performance for applications that are transaction-oriented, and therefore connect and disconnect repeatedly.

Translation

The description of the current translator is displayed. To select a new translator, choose the Select button and select a new translator from the list in the Select Translator dialog box.

Convert OEM to ANSI Characters

If the SQL Server client machine and SQL Server are using the same non-ANSI character set, select the Convert OEM to ANSI Characters check box.

If the SQL Server client machine and SQL Server are using different character sets, you must specify a character set translator.

See Also

SQL Server Driver-Specific Connection Options (Programming)

Server for Microsoft Windows for Workgroups, Microsoft LAN Manager, and Compatibles

Enter the network name of the computer on which SQL Server is running.

Server for Novell NetWare

Enter the name that SQL Server registers with the NetWare Bindery when it starts. On Windows NT, this name is configured by SQL Server Setup. On OS/2, it is specified as a command line parameter for the Network Manager.

Server for Banyan VINES

Enter the PC-based service name that SQL Server registers with the StreetTalk directory service when it starts (for example, Server1@engineering@xyzcorp). On OS/2, this name is specified as a command line parameter for the Network Manager.

Server for TCP/IP Networks (Sockets)

For TCP/IP sockets, enter an alias and also enter values in the Network Library and Network Address text boxes.

For Microsoft Windows for Workgroups, Microsoft LAN Manager, and compatibles, you can connect to SQL Server using named pipes on top of the TCP/IP transport protocol. In this case, enter the network name of the computer on which SQL Server is running.

Network Address for Microsoft Windows for Workgroups, Microsoft LAN Manager, and Compatibles

To use the default named pipe for SQL Server (*server*\pipe\sql\query), where *server* is specified in the Server text box, enter (Default) in the Network Address text box. To use a different named pipe, specify the complete pipe name in the Network Address text box. The complete pipe name uses the format *server*\pipe*pipename*, where *server* is specified in the Server text box and *pipename* is the pipe name.

Network Address for Novell NetWare

The SQL Server driver retrieves the network address associated with a name in the NetWare Bindery. To use the name specified in the Server text box, enter (Default) in the Network Address text box. To use a different name, enter that name in the Network Address text box.

Network Address for Banyan VINES

The SQL Server driver retrieves the network address associated with a name in the StreetTalk directory service. To use the name specified in the Server text box, enter (Default) in the Network Address text box. To use a different name, enter that name in the Network Address text box.

Network Address for TCP/IP Networks (Sockets)

The network address uses the format

IP-address,*socket-address*

where *IP-address* is the IP address of the server and *socket-address* is the socket address of the server. For example, 11.1.8.166,2025.

Network Library for Microsoft Windows for Workgroups, Microsoft LAN Manager, and Compatibles

You must use the Named Pipes network library DBNMP3.DLL (16-bit) or DBNMPTW.DLL (32-bit) to connect with these networks. If the Default Network setting in the SQL Server Client Configuration Utility is Named Pipes, enter (Default) in the Network Library text box. If the Default Network setting is anything else, enter DBNMP3 in the Network Library text box (do not include the .DLL extension).

DBNMP3.DLL is shipped with the SQL Server driver; it is also shipped with a number of other Microsoft products, including Microsoft SQL Server, Microsoft Access, and Visual Basic. Make sure that the SQL Server driver uses the most current version of this DLL.

Network Library for Novell NetWare

For Microsoft SQL Server for Windows NT, you must use the Novell IPX/SPX network library DBMSSPX3.DLL (16-bit) or DBMSSPXN.DLL (32-bit) to connect with Novell NetWare. For Microsoft SQL Server running on OS/2, use the Novell IPX/SPX network library and the Network Integration Kit for Novell NetWare.

If the Default Network setting in the SQL Server Client Configuration Utility is Novell IPX/SPX, enter (Default) in the Network Library text box and the SQL Server driver will use the specified library. If the Default Network setting is anything else, enter DBMSSPX3 (16-bit) or DBMSSPXN (32-bit) in the Network Library text box (do not include the .DLL extension).

DBMSSPX.DLL (16-bit) or DBMSSPXN.DLL (32-bit) is shipped on the Windows Client Setup disks included with Microsoft SQL Server. Make sure that the SQL Server driver uses the most current version of this DLL.

Network Library for Banyan VINES

For Microsoft SQL Server running on OS/2, use the Banyan VINES network library DBMSVIN3.DLL (16-bit) or DBMSVINN.DLL (32-bit) and the Network Integration Kit for Banyan VINES.

If the Default Network setting in the SQL Server Client Configuration Utility is Banyan VINES, enter (Default) in the Network Library text box and the SQL Server driver will use the specified library. If the Default Network setting is anything else, enter DBMSVIN3 (16-bit) or DBMSVINN.DLL (32-bit) in the Network Library text box (do not include the .DLL extension).

DBMSVIN3.DLL (16-bit) or DBMSVINN.DLL (32-bit) is shipped with the Network Integration Kit for Banyan VINES. Make sure that the SQL Server driver uses the most current version of this DLL.

Network Library for TCP/IP Networks (Sockets)

For Microsoft SQL Server for Windows NT, you must use the network library DBMSSOC3.DLL (16-bit) or DBMSSOCN.DLL (32-bit) to connect with TCP/IP networks.

For Microsoft SQL Server for Windows NT, if the Default Network setting in the SQL Server Client Configuration Utility is TCP/IP, enter (Default) in the Network Library text box and the SQL Server driver will use the specified library. If the Default Network setting is anything else, enter DBMSSOC3 (16-bit) or DBMSSOCN (32-bit) in the Network Library text box (do not include the .DLL extension).

DBMSSOC3.DLL (16-bit) or DBMSSOCN.DLL (32-bit) is shipped on the Windows Client Setup disks included with Microsoft SQL Server. Make sure that the SQL Server driver uses the most current version of this DLL.

SQL Server Client Configuration Utility

A Windows-based utility shipped on the Windows Client Setup disks included with Microsoft SQL Server. It is also shipped with the Network Integration Kit for Novell NetWare and the Network Integration Kit for Banyan VINES.

Connection Strings (Advanced)

See Also

The connection string for the SQL Server driver uses the following keywords. Some keywords are optional.

Keyword	Description
DSN	The name of the data source. If the DSN keyword is used, the DRIVER keyword must not be used.
DRIVER	The name of the driver. If the DRIVER keyword is used, the DSN keyword must not be used.
SERVER	The name of the computer on the network on which the data source resides. Required if the DRIVER keyword is used.
UID	The user login ID.
PWD	The user-specified password.
APP	The name of the application calling the SQL Server driver (optional).
WSID	The workstation ID. Typically, this is the network name of the computer on which the application resides (optional).
DATABASE	The name of the SQL Server database (optional).
LANGUAGE	The national language to be used by SQL Server. This is used only by SQL Server versions 4.2 and later (optional).

For example, to connect to the Human Resources data source on the server HRSRVR using the login ID Smith and the password Sesame, you would use the following connection string:

```
DSN=Human Resources;UID=Smith;PWD=Sesame
```

To specify the Payroll database on the same server, you would use the following connection string:

```
DSN=Human Resources;UID=Smith;PWD=Sesame;DATABASE=Payroll
```

See Also

For All Users

Connecting to a SQL Server Data Source

For Programmers

SQLBrowseConnect Implementation

SQLDriverConnect Implementation

SQL Statements (Advanced)

See Also

The SQL Server driver fully supports the core SQL grammar, with the exception that the USER keyword is not supported. In addition, it supports almost all SQL statements in the extended ODBC grammar. In accordance with the design of ODBC, the SQL Server driver will pass native SQL grammar to SQL Server.

The following Help topics describe the SQL grammar implemented by the SQL Server driver.

For Advanced Users

Implementation of the ODBC SQL Grammar (Advanced)

Limitations to the ODBC SQL Grammar (Advanced)

Unsupported ODBC SQL Grammar (Advanced)

For Programmers

Limitations to the ODBC SQL Grammar (Programming)

See Also

For Advanced Users

Data Types (Advanced)

For Programmers

SQLGetInfo Return Values (Programming)

Implementation of the ODBC SQL Grammar (Advanced)

The only noteworthy part of the implementation of the ODBC SQL grammar is the implementation of the CREATE TABLE and ALTER TABLE statements.

The SQL Server driver adds a NULL specification to each column definition in a CREATE TABLE statement that does not specify whether the column is nullable (except for BIT columns, which are not nullable). It adds a NULL specification to each column definition in an ALTER TABLE statement (except for BIT columns, which are not nullable).

It does this to resolve a difference in the SQL grammars defined by ODBC and SQL Server:

- In the ODBC grammar, columns for which nullability is not specified are assumed to be nullable.
- In the SQL Server grammar, columns for which nullability is not specified are assumed to be not nullable.

Limitations to the ODBC SQL Grammar (Advanced)

The SQL Server driver and SQL Server impose the following limitations on the ODBC SQL grammar:

Limited SQL	Description
ABS, CEILING, DEGREES, FLOOR, POWER, RADIANS, ROUND, and SIGN scalar functions	These scalar functions return a value of the same type as that supplied for the parameter.
Batched SQL statements	Batched SQL statements can't include CREATE VIEW statements.
	The statements in a batch that start with a restricted DDL statement cannot be rolled back.
Create-Index Statements	The Create-Index statement does not allow the ASC and DESC options for each column identifier. ASC and DESC in the Create-Index statement are deleted.
CURDATE scalar function	Because the SQL Server driver doesn't support the SQL_DATE data type, the CURDATE scalar function returns a value of type SQL_VARCHAR instead of type SQL_DATE.
CURTIME scalar function	Because the SQL Server driver doesn't support the SQL_TIME data type, the CURTIME scalar function returns a value of type SQL_VARCHAR instead of type SQL_TIME.
Date Translation	The SQL Server driver translates dates and timestamps from the canonical form (e.g.,1994-10-23) into an order-insensitive literal (e.g., 19941023) in order to avoid misinterpretation of the month/day ordering.
NULL CHAR and NULL BINARY columns	Data input to a NULL CHAR or NULL BINARY column is not padded with spaces. The data is stored as VARCHAR, so that trailing spaces are removed.
WHERE clause	Multiple updates of columns with SQL_LONGVARBINARY or SQL_LONGVARCHAR data

	types (using a WHERE clause) are not fully supported. Only the first row of a set of rows matching a value is updated.

Unsupported ODBC SQL Grammar (Advanced)

The SQL Server driver completely supports all SQL statements and clauses in both the core and extended ODBC grammars except those listed below.

Statement not supported	Description
DAYOFWEEK	The scalar function DAYOFWEEK is not supported.
DELETE	The WHERE CURRENT OF *cursor-name* clause isn't supported (positioned delete statement).
DROP INDEX	Instead of *index-name*, *table-name.index-name* must be used.
IEF	None of the clauses in the Integrity Enhancement Facility (IEF) is supported.
MAX, MIN	The DISTINCT keyword isn't supported for these set functions.
SELECT	The FOR UPDATE OF clause isn't supported (select-for-update statement).
UPDATE	The WHERE CURRENT OF *cursor-name* clause isn't supported (positioned update statement).
USER	The USER keyword is not supported.

Limitations to the ODBC SQL Grammar (Programming)

The SQL Server driver and SQL Server impose the following limitations on the ODBC SQL grammar:

Limited SQL	Description
Data-at-Execution Parameter Limitations	Data-at-execution parameters that are used to send more than 65,536 bytes of data for an SQL_LONGVARCHAR or SQL_LONGVARBINARY column are subject to a number of restrictions.
OUTER JOIN	The search condition for the outer join must be an equals condition.

	Conditions using ">=" or "<=" return an incorrect-syntax error message.
	SQL Server does not allow both nested outer joins and inner joins nested within an outer join. A table cannot be the inner table in an outer join and still participate in an inner join. A table cannot be the inner table in an outer join and the outer table in another outer join. A table can, however, participate in an inner join and be the outer table in an outer join.
Procedures	With the SQL Server native grammar, if a procedure is not invoked as the first statement in a prepared batch of statements, it must be invoked with the EXECUTE keyword.
	If literals are used as inputs to stored procedure calls, no output parameter is returned.

Procedure Invocation Limitations (Programming)

In the SQL grammar used by SQL Server, the EXECUTE keyword isn't needed if a statement that executes a procedure is the first statement in a batch. If such a statement is prepared, however, the EXECUTE keyword must be present. This is because the SQL Server driver surrounds a statement it is preparing with other SQL statements. Thus, the statement being prepared is no longer the first in the batch.

This problem should be avoided because, for maximum interoperability, procedures should be invoked using the ODBC extension to SQL designed for this purpose. With the SQL Server driver, there is no advantage to preparing a statement that invokes a procedure (instead of executing it directly). When the Generate Stored Procedures for Prepared Statements option is selected, the SQL Server driver prepares a statement simply by placing it in a procedure and compiling that procedure.

Preparing a statement by placing it in a procedure and compiling can be disabled, however. To ensure that stored procedures are never used to implement **SQLPrepare**, clear the Generate Stored Procedures option checkbox in the SQL Server Driver Setup dialog box, or set the SQL_USE_PROCEDURE_FOR_PREPARE option in **SQLSetConnectOption** to SQL_UP_OFF. This will ensure that a prepared statement will be stored and re-executed at execution time. Stored procedures will then never be used to implement **SQLPrepare**. In addition, all syntax error checking will be delayed until execution time.

If a SET NOCOUNT ON statement has been executed, multiple statements embedded in a stored procedure don't create multiple results as they should. Row counts generated by SQL statements inside of a stored procedure are ignored by the driver.

Literals as Parameters to Stored Procedure Calls

Note that if a literal is passed to a stored procedure call, no output parameter can be

returned. The only way to get an output parameter is to execute the procedure as a remote procedure call. In order to execute a remote procedure call, the data type of each parameter must be known. Since a literal can map into several data types, no output parameter is returned.

Data-at-Execution Parameter Limitations (Programming)

If a data-at-execution parameter is used to send more than 65,536 bytes of data for an SQL_LONGVARCHAR or SQL_LONGVARBINARY column, it is subject to the following restrictions:

- It can be used only as an *insert-value* in an INSERT statement and as an *expression* in the SET clause of an UPDATE statement.
- It cannot be used in a statement with data-at-execution parameters that have a data type other than SQL_LONGVARCHAR or SQL_LONGVARBINARY. (It can be used in a statement with non-data-at-execution parameters of any data type.)
- An INSERT statement containing such a parameter must contain a list of columns. (In all other cases, the list of columns is optional.)
- If its value is NULL, the *cbColDef* argument in **SQLBindParameter** and the *cbValueMax* argument in **SQLPutData** must be SQL_NULL_DATA.

Data Types (Advanced)

See Also

The SQL Server driver maps SQL Server data types to ODBC SQL data types. The following table lists all SQL Server data types and shows the ODBC SQL data types they are mapped to.

SQL Server SQL data type	ODBC SQL data type
binary	SQL_BINARY
bit	SQL_BIT
char	SQL_CHAR
datetime	SQL_TIMESTAMP
float	SQL_FLOAT
image	SQL_LONGVARBINARY
int	SQL_INTEGER
money	SQL_DECIMAL
real	SQL_REAL
smalldatetime	SQL_TIMESTAMP
smallint	SQL_SMALLINT
smallmoney	SQL_DECIMAL
sysname	SQL_VARCHAR
text	SQL_LONGVARCHAR
timestamp*	SQL_VARBINARY
tinyint	SQL_TINYINT
varbinary	SQL_VARBINARY
varchar	SQL_VARCHAR

* The timestamp data type is converted to the SQL_VARBINARY data type because values in timestamp columns are not datetime data, but varbinary(8) data, indicating the sequence of SQL Server activity on the row.

Note The SQL Server driver cannot convert SQL data of types SQL_CHAR, SQL_VARCHAR, or SQL_LONGVARCHAR to C data of types SQL_C_DATE or SQL_C_TIME. It supports all other conversions in Appendix D of the *Microsoft ODBC SDK Programmer's Reference* for the ODBC SQL data types listed earlier in this topic.

The following Help topics describe the data types implemented by the SQL Server driver.

For Advanced Users

Limitations to Data Types (Advanced)

For Programmers

Implementation of Data Types (Programming)

Limitations to Data Types (Programming)

See Also

For Advanced Users

SQL Statements (Advanced)

Limitations to Data Types (Advanced)

The SQL Server driver and SQL Server impose the following limitations on the data types.

Limited data type	Description
Date literals	Date literals, when stored in an SQL_TIMESTAMP column (SQL Server types of datetime or smalldatetime) have a time value of 12:00:00.000am (midnight).
money and smallmoney	Only the integer parts of the money and smallmoney data types are significant. If the decimal part of SQL money data is truncated during data type conversion, the SQL Server driver returns a warning, not an error.
SQL_BINARY	If an SQL_BINARY column is nullable, the data stored in the data source isn't padded with zeroes. When data from such a column is retrieved, the SQL Server driver pads it with zeroes on the right. However, data created in operations performed by SQL Server, such as concatenation, don't have such padding.
SQL_CHAR (truncation)	When data is inserted into an SQL_CHAR column, SQL Server truncates it on the right without warning if it is too long to fit into the column. The SQL Server driver is thus unable to provide a warning to the application when character data is truncated on the right.
SQL_CHAR (nullable)	If an SQL_CHAR column is nullable, the data stored in the data source isn't padded with blanks. When data from such a column is retrieved, the SQL Server driver pads it with blanks on the right. However, data created in operations performed by SQL Server, such as concatenation, don't have such padding.

SQL_INTEGER	The most negative integer value (-2147483648) cannot be input into a table as a string.
SQL_LONGVARBINARY, SQL_LONGVARCHAR	Multiple updates of columns with SQL_LONGVARBINARY or SQL_LONGVARCHAR data types (using a WHERE clause) are not fully supported. Only the first row of a set of rows matching a value is updated.
String Function Parameters	*string_exp* parameters to the string functions must be of type SQL_CHAR or SQL_VARCHAR. SQL_LONG_VARCHAR types are not supported in the string functions. The *count* parameter must be less than or equal to 255, since the SQL_CHAR and SQL_VARCHAR data types are limited to a maximum length of 255 characters.
Time literals	Time literals, when stored in an SQL_TIMESTAMP column (SQL Server types of datetime or smalldatetime) have a date value of January 1, 1900.
timestamp	A timestamp column cannot be updated, and only a NULL value can be inserted into a timestamp column. However, because timestamp columns are automatically updated by SQL Server, a NULL value is overwritten.
tinyint	The SQL Server tinyint data type is unsigned. A tinyint column is bound to a variable of type SQL_C_UTINYINT by default.
User-defined data types	Because the SQL Server driver adds NULL to a column definition that doesn't explicitly declare a column's nullability, the nullability stored in the definition of a user-defined data type is ignored.

Implementation of Data Types (Programming)

The SQL Server driver implements the ODBC SQL data types as follows.

Data type	Description
SQL_CHAR	The SQL Server driver adds NULL to a column definition that doesn't explicitly declare the column's nullability. SQL Server doesn't support nullable SQL_CHAR columns. Therefore, SQL_CHAR columns declared as NULL (by the user or the SQL Server driver) are created as type SQL_VARCHAR. The SQL Server driver recognizes these columns and returns the correct data type and data type name for them in **SQLColAttributes**, **SQLColumns**, and **SQLDescribeCol**. It also pads data retrieved from these columns.
tinyint	The SQL Server tinyint data type is unsigned. A tinyint column is bound to a variable of type SQL_C_UTINYINT by default.

Limitations to Data Types (Programming)

The SQL Server driver and SQL Server impose the following limitations on the data types.

Limited data type	Description
LONG data types	SQL_LONGVARBINARY data must be passed to **SQLPutData** as raw binary data, not as binary data converted to character data. Also, data-at-execution parameters are restricted for both the SQL_LONGVARBINARY and SQL_LONGVARCHAR data types.
User-defined data types	Columns with a user-defined data type that has a base type of char and for which no nullability or NULL was declared are created as type varchar. **SQLColAttributes**, **SQLColumns**, and **SQLDescribeCol** return SQL_VARCHAR as the data type for these columns. Data retrieved from these columns isn't padded.

User-Defined Data Types Limitations (Programming)

The SQL Server driver adds NULL to a column definition that doesn't explicitly declare the column's nullability. SQL Server doesn't support nullable char columns. Therefore, columns declared as NULL (by the user or by the SQL Server driver) that have a user-defined data type with a base data type of char are created as type varchar.

Because of this, **SQLColAttributes**, **SQLColumns**, and **SQLDescribeCol** return the declared user-defined data type name and an ODBC SQL data type of SQL_VARCHAR. Also, data returned from these columns isn't padded, as char data is.

Error Messages (Advanced)

When an error occurs, the SQL Server driver returns the native error number, the SQLSTATE (an ODBC error code), and an error message. The driver derives this information both from errors detected by the driver and errors returned by SQL Server.

Native Error

For errors that occur in the data source, the SQL Server driver returns the native error returned to it by SQL Server. For errors detected by the driver, the SQL Server driver returns a native error of zero. For a list of native errors, see the error column of the sysmessages system table in the master database in SQL Server.

SQLSTATE

For errors that occur in the data source, the SQL Server driver maps the returned native error to the appropriate SQLSTATE. If the error occurs in the data source and can't be mapped, the SQL Server driver returns SQLSTATE 37000 (Syntax error or access violation). For errors that are detected by the driver, the SQL Server driver generates the appropriate SQLSTATE.

Error Message

For errors that occur in the data source, the SQL Server driver returns an error message based on the message returned by SQL Server. For errors that occur in the SQL Server driver or the Driver Manager, the SQL Server driver returns an error message based on the text associated with the SQLSTATE. For a list of error messages that can be returned by SQL Server, see the description column of the sysmessages system table in the master database in SQL Server.

Error messages have the following format:

[vendor][ODBC-component][data-source]error-message

where the prefixes in brackets ([ ]) identify the source of the error. The following table shows the values of these prefixes returned by the SQL Server driver.

Note When the error occurs in the data source, the [*vendor*] and [*ODBC-component*] prefixes identify the vendor and name of the ODBC component that received the error from the data source.

Error source	Prefix	Value
Driver Manager	[*vendor*] [*ODBC-component*] [*data-source*]	[Microsoft] [ODBC DLL] N/A
Cursor Library	[*vendor*] [*ODBC-component*] [*data-source*]	[Microsoft] [ODBC Cursor Library] N/A
SQL Server driver	[*vendor*] [*ODBC-component*] [*data-source*]	[Microsoft] [ODBC SQL Server Driver] N/A
SQL Server	[*vendor*] [*ODBC-component*] [*data-source*]	[Microsoft] [ODBC SQL Server Driver] [SQL Server]

SQLGetInfo Return Values (Programming)

The following table lists the C language #defines for the *fInfoType* argument and the corresponding values returned by **SQLGetInfo**. An application retrieves this information by passing the listed C language #defines to **SQLGetInfo** in the *fInfoType* argument.

***fInfoType* value (#define)**	**Returned value**

SQL_ACCESSIBLE_PROCEDURES	"Y"
SQL_ACCESSIBLE_TABLES	"Y"
SQL_ACTIVE_CONNECTIONS	0. (The actual number of active connections is determined by the number of network connections available on the client computer and the number of connections allowed by the server DBMS.)
SQL_ACTIVE_STATEMENTS	1
SQL_ALTER_TABLE	SQL_AT_ADD_COLUMN
SQL_BOOKMARK_PERSISTENCE	0
SQL_COLUMN_ALIAS	"Y"
SQL_CONCAT_NULL_BEHAVIOR	SQL_CB_NON_NULL
SQL_CONVERT_FUNCTIONS	SQL_FN_CVT_CONVERT
SQL_CONVERT_*type*, where *type* is the SQL data type (such as CHAR)	See table below.
SQL_CORRELATION_NAME	SQL_CN_ANY
SQL_CURSOR_COMMIT_BEHAVIOR	SQL_CB_CLOSE
SQL_CURSOR_ROLLBACK_BEHAVIOR	SQL_CB_CLOSE
SQL_DBMS_NAME	"Microsoft SQL Server"
SQL_DEFAULT_TXN_ISOLATION	SQL_TXN_READ_COMMITTED.
SQL_DRIVER_NAME	"SQLSRVR.DLL" or "SQLSRV32.DLL"
SQL_DRIVER_ODBC_VER	"02.00"
SQL_DRIVER_VER	"02.00.*nnnn*", where *nnnn* specifies the build date.
SQL_EXPRESSIONS_IN_ORDERBY	"Y"
SQL_FETCH_DIRECTION	SQL_FD_FETCH_NEXT
SQL_FILE_USAGE	SQL_FILE_NOT_SUPPORTED
SQL_GETDATA_EXTENSIONS	0
SQL_GROUP_BY	SQL_GB_NO_RELATION
SQL_IDENTIFIER_CASE	Depends on whether SQL Server was installed as case-sensitive or not case-sensitive.
SQL_IDENTIFIER_QUOTE_CHAR	" "
SQL_KEYWORDS	"BREAK", "BROWSE", "BULK", "CHECKPOINT", "CLUSTERED", "COMPUTE", "CONFIRM", "CONTROLROW", "DATABASE", "DBCC", "DEBUG", "DISK". "DUMMY", "DUMP", "ERRLVL", "ERROREXIT", "EXIT", "FILE", "FILLFACTOR", "GETDEFAULT", "HOLDLOCK", "IF", "INIT", "KILL", "LINENO", "LOAD", "MIRROREXIT", "NONCLUSTERED", "NOT NULL", "NOUNLOAD", "OFF", "OFFSETS", "ONCE", "OVER", "PERM", "PERMANENT", "PLAN", "PRINT", "PROC", "PROCESSEXIT", "RAISERROR",

	"READTEXT", "RECONFIGURE", "RETURN", "ROWCOUNT", "RULE", "SAVE", "SETUSER", "STATISTICS", "TAPE", "TEMP", "TEXTSIZE", "TRAN", "TRIGGER", "TRUNCATE", "TSEQUEL", "UNLOAD", "USE", "WAITFOR", "WHILE", "WRITETEXT"
SQL_LIKE_ESCAPE_CLAUSE	"N"
SQL_LOCK_TYPES	0
SQL_MAX_BINARY_LITERAL_LEN	131072
SQL_MAX_CHAR_LITERAL_LEN	131072
SQL_MAX_COLUMN_NAME_LEN	30
SQL_MAX_COLUMNS_IN_GROUP_BY	16
SQL_MAX_COLUMNS_IN_INDEX	16
SQL_MAX_COLUMNS_IN_ORDER_BY	16
SQL_MAX_COLUMNS_IN_SELECT	1310
SQL_MAX_COLUMNS_IN_TABLE	250
SQL_MAX_CURSOR_NAME_LEN	32
SQL_MAX_INDEX_SIZE	256
SQL_MAX_OWNER_NAME_LEN	30
SQL_MAX_PROCEDURE_NAME_LEN	36 (1 to 30 characters followed by a semicolon [;] and one to five digits)
SQL_MAX_QUALIFIER_NAME_LEN	30
SQL_MAX_ROW_SIZE	1962
SQL_MAX_ROW_SIZE_INCLUDES_LONG	"N"
SQL_MAX_STATEMENT_LEN	131072
SQL_MAX_TABLE_NAME_LEN	30
SQL_MAX_TABLES_IN_SELECT	16
SQL_MAX_USER_NAME_LEN	30
SQL_MULT_RESULT_SETS	"Y"
SQL_MULTIPLE_ACTIVE_TXN	"Y"
SQL_NEED_LONG_DATA_LEN	"Y"
SQL_NON_NULLABLE_COLUMNS	SQL_NNC_NON_NULL
SQL_NULL_COLLATION	SQL_NC_LOW
SQL_NUMERIC_FUNCTIONS	SQL_FN_NUM_ABS SQL_FN_NUM_ACOS SQL_FN_NUM_ASIN SQL_FN_NUM_ATAN SQL_FN_NUM_ATAN2 SQL_FN_NUM_CEILING SQL_FN_NUM_COS SQL_FN_NUM_COT SQL_FN_NUM_DEGREES SQL_FN_NUM_EXP SQL_FN_NUM_FLOOR SQL_FN_NUM_LOG SQL_FN_NUM_LOG10 SQL_FN_NUM_MOD SQL_FN_NUM_PI

	SQL_FN_NUM_POWER SQL_FN_NUM_RADIANS SQL_FN_NUM_RAND SQL_FN_NUM_ROUND SQL_FN_NUM_SIGN SQL_FN_NUM_SIN SQL_FN_NUM_SQRT SQL_FN_NUM_TAN
SQL_ODBC_API_CONFORMANCE	SQL_OAC_LEVEL_1
SQL_ODBC_SAG_CLI_CONFORMANCE	SQL_OSCC_NOT_COMPLIANT
SQL_ODBC_SQL_CONFORMANCE	SQL_OSC_CORE
SQL_ODBC_SQL_OPT_IEF	"N"
SQL_OJ_CAPABILITIES	SQL_OJ_LEFT, SQL_OJ_RIGHT, SQL_OJ_NESTED
SQL_ORDER_BY_COLUMNS_IN_SELECT	"N"
SQL_OUTER_JOINS	"Y"
SQL_OWNER_TERM	"owner"
SQL_OWNER_USAGE	SQL_OU_DML_STATEMENTS, SQL_OU_PROCEDURES_ INVOCATION, SQL_OU_TABLE_DEFINITION, SQL_OU_INDEX_DEFINITION, SQL_OU_PRIVILEGE_ DEFINITION
SQL_POS_OPERATIONS	0
SQL_POSITIONED_STATEMENTS	0
SQL_PROCEDURE_TERM	"stored procedure"
SQL_PROCEDURES	"Y"
SQL_QUALIFIER_LOCATION	SQL_QL_START
SQL_QUALIFIER_NAME_SEPARATOR	"." (period)
SQL_QUALIFIER_TERM	"database"
SQL_QUALIFIER_USAGE	SQL_QU_DML_STATEMENTS, SQL_QU_PROCEDURE_ INVOCATION, SQL_QU_TABLE_DEFINITION
SQL_QUOTED_IDENTIFIER_CASE	0
SQL_ROW_UPDATES	"N"
SQL_SCROLL_CONCURRENCY	SQL_SCCO_READ_ONLY
SQL_SCROLL_OPTIONS	SQL_SO_FORWARD_ONLY
SQL_SEARCH_PATTERN_ESCAPE	"\" (backslash)
SQL_SPECIAL_CHARACTERS	"@#$"
SQL_STATIC_SENSITIVITY	0
SQL_STRING_FUNCTIONS	SQL_FN_STR_ASCII, SQL_FN_STR_CHAR, SQL_FN_STR_CONCAT, SQL_FN_STR_DIFFERENCE, SQL_FN_STR_INSERT, SQL_FN_STR_LCASE, SQL_FN_STR_LEFT, SQL_FN_STR_LENGTH, SQL_FN_STR_LOCATE_2,

	SQL_FN_STR_LTRIM, SQL_FN_STR_REPEAT, SQL_FN_STR_RIGHT, SQL_FN_STR_RTRIM, SQL_FN_STR_SOUNDEX, SQL_FN_STR_SPACE, SQL_FN_STR_SUBSTRING, SQL_FN_STR_UCASE
SQL_SUBQUERIES	SQL_SQ_COMPARISON, SQL_SQ_EXISTS, SQL_SQ_IN, SQL_SQ_QUANTIFIED, SQL_SQ_CORRELATED_ SUBQUERIES
SQL_SYSTEM_FUNCTIONS	SQL_FN_SYS_DBNAME, SQL_FN_SYS_IFNULL, SQL_FN_SYS_USERNAME,
SQL_TABLE_TERM	"table"
SQL_TIMEDATE_ADD_INTERVALS	SQL_FN_TSI_FRAC_SECOND, SQL_FN_TSI_SECOND, SQL_FN_TSI_MINUTE, SQL_FN_TSI_HOUR, SQL_FN_TSI_DAY, SQL_FN_TSI_WEEK, SQL_FN_TSI_MONTH, SQL_FN_TSI_QUARTER, SQL_FN_TSI_YEAR
SQL_TIMEDATE_DIFF_INTERVALS	SQL_FN_TSI_FRAC_SECOND, SQL_FN_TSI_SECOND, SQL_FN_TSI_MINUTE, SQL_FN_TSI_HOUR, SQL_FN_TSI_DAY, SQL_FN_TSI_WEEK, SQL_FN_TSI_MONTH, SQL_FN_TSI_QUARTER, SQL_FN_TSI_YEAR,
SQL_TIMEDATE_FUNCTIONS	SQL_FN_TD_NOW, SQL_FN_TD_CURDATE, SQL_FN_TD_DAYOFMONTH, SQL_FN_TD_DAYOFYEAR, SQL_FN_TD_DAYNAME, SQL_FN_TD_MONTH, SQL_FN_TD_MONTHNAME, SQL_FN_TD_QUARTER, SQL_FN_TD_WEEK, SQL_FN_TD_YEAR, SQL_FN_TD_CURTIME, SQL_FN_TD_HOUR, SQL_FN_TD_MINUTE, SQL_FN_TD_SECOND, SQL_FN_TD_TIMESTAMPADD, SQL_FN_TD_TIMESTAMPDIFF
SQL_TXN_CAPABLE	SQL_TC_DML
SQL_TXN_ISOLATION_OPTION	SQL_TXN_READ_COMMITTED SQL_TXN_SERIALIZABLE
SQL_UNION	SQL_U_UNION, SQL_U_UNION_ALL

The following table shows the conversions supported by SQL Server from one SQL data type to another using the CONVERT scalar function.

Convert From \ Convert To	SQL_CHAR	SQL_VARCHAR	SQL_LONGVARCHAR	SQL_DECIMAL	SQL_NUMERIC	SQL_BIT	SQL_TINYINT	SQL_SMALLINT	SQL_INTEGER	SQL_BIGINT	SQL_REAL	SQL_FLOAT	SQL_DOUBLE	SQL_BINARY	SQL_VARBINARY	SQL_LONGVARBINARY	SQL_DATE	SQL_TIME	SQL_TIMESTAMP
SQL_CHAR	•	•	•	•		•	•	•	•		•	•		•	•	•			•
SQL_VARCHAR	•	•	•	•		•	•	•	•		•	•		•	•	•			•
SQL_LONGVARCHAR	•	•	•			•	•	•	•		•	•		•	•				
SQL_DECIMAL	•	•		•		•	•	•	•		•	•		•	•				
SQL_NUMERIC																			
SQL_BIT	•	•				•	•	•	•		•	•		•	•				
SQL_TINYINT	•	•		•		•	•	•	•		•	•		•	•				
SQL_SMALLINT	•	•		•		•	•	•	•		•	•		•	•				
SQL_INTEGER	•	•		•		•	•	•	•		•	•		•	•				
SQL_BIGINT																			
SQL_REAL	•	•		•		•	•	•	•		•	•							
SQL_FLOAT	•	•		•		•	•	•	•		•	•							
SQL_DOUBLE																			
SQL_BINARY	•	•					•	•	•					•	•	•			
SQL_VARBINARY	•	•					•	•	•					•	•	•			
SQL_LONGVARBINARY														•	•	•			
SQL_DATE																			
SQL_TIME																			
SQL_TIMESTAMP	•	•												•	•				•

ODBC API Functions (Programming)

See Also

The SQL Server driver supports all core and Level 1 functions and the following Level 2 functions:

SQLBrowseConnect	SQLNativeSql
SQLColumnPrivileges	SQLNumParams
SQLDataSources	SQLPrimaryKeys
SQLDrivers	SQLProcedureColumns
SQLForeignKeys	SQLProcedures
SQLMoreResults	SQLTablePrivileges

In addition, the SQL Server driver supports translation DLLs.

The following Help topics describe the ODBC API functions implemented by the SQL Server driver.

For Programmers

Implementation of ODBC API Functions (Programming)

Limitations to ODBC API Functions (Programming)

See Also

For Advanced Users

Error Messages (Advanced)

For Programmers

SQLGetInfo Return Values (Programming)

Implementation of ODBC API Functions (Programming)

The following table describes how the SQL Server driver implements specific functions.

Function	Description
SQLBrowseConnect	**SQLBrowseConnect** uses three levels of keywords: 1 DSN, DRIVER 2 SERVER, UID, PWD, APP, and WSID 3 LANGUAGE and DATABASE
SQLColAttributes, **SQLDescribeCol**, **SQLNumResultCols**	If any of these functions are called after a SELECT statement has been prepared and before it has been executed, the SQL Server driver forces SQL Server to generate an empty result set to obtain the necessary information about the result set.
SQLConfigDataSource	**SQLConfigDataSource** adds, modifies, or deletes a data source dynamically by using keywords to set connect options.
SQLConnect	**SQLConnect** uses the DSN, UID, and AuthStr (usually password) arguments.
SQLDriverConnect	**SQLDriverConnect** uses the DSN, DRIVER, SERVER, UID, PWD, APP, WSID, DATABASE, and LANGUAGE keywords.
SQLPrepare	SQL Server doesn't directly support the Prepare/Execute model of ODBC. To prepare an SQL statement, the SQL Server driver stores it as a procedure and compiles it for later execution. Note that generation of stored procedures for **SQLPrepare** can be disabled in either the ODBC SQL Server Driver Setup dialog box or the **SQLSetConnectOption** function. If disabled, the statement will be stored and re-executed at execution time.
SQLTables	The *szTableQualifier* argument accepts a string search pattern.

SQLBrowseConnect Implementation (Programming)

SQLBrowseConnect uses three levels of connection information. For each keyword in a level, the following tables indicate whether a list of valid values is returned for the keyword in the *szConnStrOut* argument and whether the keyword is optional; they also provide a description of the keyword.

Level 1:

	User-	List

Keyword	Friendly Name	returned?	Optional?	Description
DSN	N/A	N/A	No	The name of the data source as listed in the ODBC.INI file. The DSN keyword is not used if DRIVER is used.
DRIVER	N/A	N/A	No	The name of the driver as listed in the ODBC.INI file or the Windows NT registry. The DRIVER keyword is not used if DSN is used.

Level 2:

Keyword	User-Friendly Name	List returned?	Optional?	Description
SERVER	Server	Yes	No	The name of the server on the network on which the data source resides. On a Microsoft Windows NT network, "(local)" can be entered as the server, in which case a local copy of SQL Server can be used, even when this is a non-networked version.
UID	Login ID	No	No	The user login ID.
PWD	Password	No	Depends on the user.	The user-specified password.
APP	AppName	No	Yes	The name of the application (AppName) calling **SQLBrowseConnect**.
WSID	WorkStation ID	No	Yes	The workstation ID. Typically, this is the network name of the computer on which the application resides.

Level 3:

Keyword	User-Friendly Name	List returned?	Optional?	Description
DATABAS	Database	Yes	Yes	The name of the

E				SQL Server database.
LANGUAGE	Language	Yes	Yes	The national language to be used by SQL Server. This is used only when connecting to SQL Server versions 4.2 and later.

SQLBrowseConnect ignores the values of the **Language** and **Database** keywords in the ODBC.INI file. If the language or database specified in the connection string passed to **SQLBrowseConnect** is invalid, **SQLBrowseConnect** returns SQL_NEED_DATA and the level 3 connection attributes.

SQLBrowseConnect doesn't check whether a user has access to all the databases it lists with the **DATABASE** keyword. If the user doesn't have access to the chosen database, **SQLBrowseConnect** returns SQL_NEED_DATA and the level 3 connection attributes.

SQLColAttributes, SQLDescribeCol, and SQLNumResultCols Implementation (Programming)

SQL Server returns information about a result set before it returns the data in the result set. The SQL Server driver returns this information to an application through the **SQLColAttributes**, **SQLDescribeCol**, and **SQLNumResultCols** functions.

If an application calls any of these functions after a SELECT statement has been prepared and before it has been executed, the SQL Server driver submits the SELECT statement with the clause WHERE 1=2. This forces SQL Server to generate a result set without any rows, but with the information about the result set.

To add the clause WHERE 1=2 to the SELECT statement, the SQL Server driver:

1. Checks if the statement is a batch of statements separated by semi-colons. If it is, the driver deletes all statements except the first.
2. Searches for a WHERE or ORDER BY clause. If one is found, the driver replaces the clause and all of the statement following the clause with WHERE 1=2.
3. Adds WHERE 1=2 to the end of the statement if no WHERE or ORDER BY clause is found.

Note **SQLColAttributes**, **SQLDescribeCol**, and **SQLNumResultCols** cannot return information about a result set generated by a procedure if that procedure has been prepared but not executed. If the SELECT statement is the first statement in a batched statement and the SQL Server native grammar is used (no semi-colons between statements), the results of these functions are unpredictable. Note also that the word "SELECT" must be the first token in the buffer. If anything precedes the word "SELECT" in the statement to be prepared, "WHERE 1=2" will not be added to the SELECT statement.

SQLConnect Implementation (Programming)

See Also

The **SQLConnect** function uses the following arguments:

Keyword	Description
DSN	The name of the data source as listed in the ODBC.INI file.
UID	The user login ID.
AuthStr	The user-specified authentication string (typically the password).

SQLConnect retrieves the value of the LANGUAGE keyword from the ODBC.INI file. If SQL Server is unable to use the specified language, it uses the default language for the specified user ID, and **SQLConnect** returns SQL_SUCCESS_WITH_INFO. If SQL Server is unable to use the default language for the specified user ID, **SQLConnect** returns SQL_ERROR.

SQLConnect ignores the value of the DATABASE keyword.

See Also

For Programmers

SQLBrowseConnect Implementation (Programming)

SQLDriverConnect Implementation (Programming)

SQLDriverConnect Implementation (Programming)

See Also

The **SQLDriverConnect** connection string uses the following keywords:

Keyword	Description
DSN	The name of the data source as listed in the ODBC.INI file. The DSN keyword is not used if DRIVER is used.
DRIVER	The name of the driver as listed in the ODBC.INI file or the Windows NT registry. The DRIVER keyword is not used if DSN is used.
SERVER	The name of the server on the network on which the data source resides. On a Microsoft Windows NT network, "(local)" can be entered as the server, in which case a local copy of SQL Server can be used, even when this is a non-networked version.
UID	The user login ID.
PWD	The user-specified password.
APP	The name of the application calling **SQLDriverConnect** (optional).
WSID	The workstation ID. Typically, this is the network name of the computer on which the application resides (optional).
DATABASE	The name of the SQL Server database (optional).
LANGUAGE	The national language to be used by SQL Server. This is used only by SQL Server versions

4.2 and later
(optional).

SQLDriverConnect uses keyword values from the dialog box (if one is displayed). If a keyword value isn't set in the dialog box, **SQLDriverConnect** uses the value from the connection string. If the value isn't set in the connection string, it uses the value from the ODBC.INI file.

If the *fDriverCompletion* argument is SQL_DRIVER_NOPROMPT or SQL_DRIVER_COMPLETE_REQUIRED, the language or database comes from the connection string, and the language or database is invalid, **SQLDriverConnect** returns SQL_ERROR.

If the *fDriverCompletion* argument is SQL_DRIVER_NOPROMPT or SQL_DRIVER_COMPLETE_REQUIRED, the language or database comes from the ODBC.INI file, and the language or database is invalid, **SQLDriverConnect** uses the default language or database for the specified user ID and returns SQL_SUCCESS_WITH_INFO.

If the *fDriverCompletion* argument is SQL_DRIVER_COMPLETE or SQL_DRIVER_PROMPT and the language or database is invalid, **SQLDriverConnect** redisplays the dialog box.

See Also

For Advanced Users

Connection Strings (Advanced)

For Programmers

SQLBrowseConnect Implementation (Programming)

SQLConnect Implementation (Programming)

SQLConfigDataSource Implementation (Programming)

The **SQLConfigDataSource** function that is used to add, modify, or delete a data source dynamically uses the following arguments. Note that only the SERVER keyword is required for this function; all other keywords are optional.

Keyword	Description
ADDRESS	The location of the SQL Server database management system from which the driver will retrieve data.
DATABASE	The name of the SQL Server database.
DESCRIPTION	A description of the data in the data source.
FASTCONNECT OPTION	Disables execution of ODBC informational queries at connect time. Valid values are YES for ON (execution is disabled) and NO for OFF. The default value in the setup dialog box is OFF.
LANGUAGE	The national language to be used by SQL Server. (This is only used by SQL Server versions 4.2 and later.)
NETWORK	The network connecting the platforms on which SQL Server and the SQL Server driver reside.
OEMTOANSI	Enables conversion of the OEM character set to the ANSI character set if the SQL Server client machine and SQL Server are using the same non-ANSI character set. Valid values are YES for ON (conversion is enabled) and NO for OFF. The default value in the setup dialog box is OFF.
SERVER	The name of the computer on the network on which the data source resides.
TRANSLATION DLL	Name of the DLL that translates data passing between an application and a data source.
TRANSLATION NAME	Name of the translator that translates data passing between an application and a data source.
TRANSLATION OPTION	Enables translation of data passing between an application and a data source.
USEPROCFOR PREPARE	Turns off generation of stored procedures for SQLPrepare. Valid values are NO for OFF (generation is disabled) and

YES for ON. The default value in the setup dialog box is ON.

SQLPrepare Implementation (Programming)

SQL Server doesn't directly support the Prepare/Execute model of ODBC. To implement this model, the SQL Server driver performs two separate operations related to statement preparation.

In the first operation, **SQLPrepare** submits the statement to SQL Server with the SET NOEXEC or SET PARSEONLY option (depending on the statement type). SQL Server checks the syntax of the statement and returns any errors.

In the second operation, a stored procedure is created from the statement, since stored procedures are an efficient way to execute a statement more than once. The procedure is named "odbc#<*user*><*identifier*>", where <*user*> is up to 15 characters of the user name and <*identifier*> is up to 10 digits that identify the statement. The procedure is created at prepare time if all parameters have been set, or at execute time if all parameters were not set at prepare time or if any parameter has been reset since the procedure was created. Because of this, **SQLExecute** can return any errors that **SQLPrepare** can return.

If a user can't create a stored procedure for any reason (such as lack of permission), the SQL Server driver doesn't use a stored procedure but submits the SQL statement each time **SQLExecute** is called.

Generation of stored procedures for **SQLPrepare** can be disabled. If disabled, the statement will be stored and re-executed at execution time. Stored procedure generation can be disabled in either of two ways: by clearing the Generate Stored Procedures for Prepared Statements option checkbox in the ODBC SQL Server Driver Setup dialog box, or by setting the SQL_USE_PROCEDURE_FOR_PREPARE option in the **SQLSetConnectOption** function to SQL_UP_OFF. If stored procedure generation is disabled, all syntax error checking will be delayed until execution time.

Note that if SET NOCOUNT ON has been executed, multiple statements embedded in a stored procedure don't create multiple results as they should. Row counts generated by SQL statements inside of a stored procedure are ignored by the driver.

Limitations to ODBC API Functions (Programming)

The following functions in the SQL Server driver don't meet the specifications in the *Microsoft ODBC SDK Programmer's Reference*.

Function	Description
SQLBindParameter	For all output parameters, *fParamType* must be set to SQL_INPUT_OUTPUT or SQL_OUTPUT. Return value parameters must be SQL_INPUT_OUTPUT or SQL_OUTPUT. Parameters are always sent to the server, even if they will not be used on the server, so they must contain data of the proper type to be converted.
SQLCancel	Some network libraries do not support the out-of-band cancel function, and will abort the cancel (returning a 70100 error code) if an out-of-band cancel is attempted.
Catalog functions	If the qualifier is a null pointer, only entries in the current database are returned.
SQLColumnPrivileges, **SQLForeignKeys**, **SQLPrimaryKeys**, **SQLStatistics**, and **SQLTablePrivileges**	These functions cannot be used in manual-commit mode because they create temporary tables and CREATE TABLE statements aren't allowed in transactions.
SQLProcedureColumns	**SQLProcedureColumns** doesn't return columns in any result sets created by a procedure.
SQLPutData	**SQLPutData** can only accept data for an SQL_LONGVARBINARY column as raw binary data, not as binary data converted to character data.
SQLSetConnectOption/ SQLGetConnectOption	The following connection options are supported: SQL_ACCESS_MODE, SQL_AUTOCOMMIT, SQL_CURRENT_QUALIFIER, SQL_LOGIN_TIMEOUT, SQL_ODBC_CURSORS, SQL_OPT_TRACE, SQL_OPT_TRACEFILE, SQL_PACKET_SIZE, SQL_QUIET_MODE, SQL_TRANSLATE_DLL,

	SQL_TRANSLATE_OPTION, SQL_TXN_ISOLATION. Driver-specific connection options are SQL_FAST_CONNECT, SQL_USE_PROCEDURE_FOR_PREPARE, and SQL_REMOTE_PWD. All other options return SQLSTATE S1C00 (Driver not capable).
	The SQL_PACKET_SIZE connection option has a range of 512 to 41710. Any number less than 512 will be adjusted to 512, and a SQLSTATE 01S02 (Option Value Changed) warning will be returned. This option must be set prior to the SQLConnect, SQLDriverConnect, or SQLBrowseConnect function. If the server does not support changing the packet size, a SQLSTATE IM006 (Driver's **SQLSetConnectOption** failed) warning will be returned. Only Microsoft SQL Server 4.21 (for Windows NT) and later releases support this feature.
SQLSetStmtOption/ SQLGetStmtOption	SQL_ASYNC_ENABLE, SQL_MAX_LENGTH, SQL_NOSCAN, and SQL_MAX_ROWS are supported. All other values return SQLSTATE S1C00 (Driver Not Capable).
	Because SQL Server uses a signed 32-bit integer, the SQL_MAX_LENGTH and SQL_MAX_ROWS options in **SQLSetStmtOption** cannot be set higher than 2 to the 31st power minus 1 (2,147,483,647). If a larger value is specified, this value is used.

SQL Server Driver-Specific Connection Options (Programming)

The following driver-specific connection options in SQLSetConnectOption and SQLGetConnectOption are supported by the SQL Server driver:

Option	Description
SQL_FAST_CONNECT	Possible values are:

	SQL_FC_OFF (the default). All informational queries are run at connect time to retrieve information required to implement meta data support, including retrieving UDT (user defined datatypes) information. SQL_FC_ON. No informational queries are run at connect time or after connect time. This will reduce the required connect time. Note that the driver will not always operate according to the ODBC 2.0 specification when fast connections have been selected. Anomalies are listed below this table.
SQL_USE_PROCEDURE_FOR_PREPARE	Possible values are: SQL_UP_ON (the default). Stored procedures will be generated for **SQLPrepare**. SQL_UP_OFF. Stored procedures will not be generated for **SQLPrepare**. The statement will be stored, compiled, and executed at execution time. All syntax error checking will be delayed until execution time.
SQL_REMOTE_PWD	The value for this option is a string that allows setting the remote password section of the login record passed to SQL Server. The *vParam* argument is a pointer to a double-zero byte-terminated string formatted as follows: *server1*\0*password1*\0*server2*\0...*password2*\0*servern*\0*passwordn*\0\0.

Driver-Specific Option #Defines

The following are the SQL Server-specific defines.

SQLSetConnectOption/SQLSetStmtOption driver specific defines:
Microsoft has 1200-1204 reserved for Microsoft driver usage.

```
#define SQL_FAST_CONNECT                1200
#define SQL_REMOTE_PWD                  1201
#define SQL_USE_PROCEDURE_FOR_PREPARE   1202
#define RESERVED                        1203
#define RESERVED                        1204
```

```
        Defines for use with SQL_FAST_CONNECT
#define SQL_FC_OFF                      0L
#define SQL_FC_ON                       1L
#define SQL_FC_DEFAULT                  SQL_FC_OFF

        Defines for use with SQL_USER_PROCEDURE_FOR_PREPARE
#define SQL_UP_OFF                      0L
#define SQL_UP_ON                       1L
#define SQL_UP_DEFAULT                  SQL_UP_ON
```

Fast-Connect Anomalies

The following anomalies occur when fast connections have been selected:

- For the metadata information returned by **SQLColAttributes**, the native type name for UDTs will not be returned, only the base type name on which the UDT is based.

Implementation Issues (Programming)

The following implementation-specific issues might affect the use of the SQL Server driver.

Issue	Description
Active hstmt definition	An *hstmt* is defined as active if it has results pending.
Arithmetic errors	SQL Server returns a data value of 0 for arithmetic errors and doesn't report the error until after all data has been retrieved.
Cursor Library	The cursor library allows an application to work with more than one active statement on a single connection.
DB-Library	The SQL Server driver doesn't use DB-Library and therefore doesn't behave like DB-Library.
Manual-commit mode transactions	In manual-commit mode, the SQL Server driver initiates a transaction when there is no current transaction and a statement other than a restricted DDL statement is pending. (See the "BEGIN TRANSACTION" section in the *Microsoft SQL Server Language Reference* for the list of statements that cannot be in a transaction.)
MAX_LENGTH and MAX_ROW Options	If an application uses multiple *hstmts* on the same connection, and uses the **SQLSetStmtOption** function with either the SQL_MAX_LENGTH or SQL_MAX_ROW *fOption*, the application should have the same option value set for each *hstmt*. If there are different values for each one of the multiple statements, the driver must change the value at the server whenever a different *hstmt* is used, which is inefficient. The SQL_MAX_LENGTH or SQL_MAX_ROW *fOption* can be set for all *hstmts* on an *hdbc* by calling the **SQLSetConnectOption** function. This sets the statement option for any *hstmt*s associated with the specified *hdbc*, and establishes the statement option as a default for any *hstmts* later allocated for that *hdbc*.
Remote procedure calls	The SQL Server driver uses the remote procedure call (RPC) facility in SQL Server to invoke prepared statements (which are stored as procedures), procedures called with the ODBC procedure extension, and the stored procedures used to implement the catalog functions.
SET TEXTSIZE	So that the driver will be able to retrieve text and image data in parts of any size, it issues a SET TEXTSIZE statement when it connects to SQL Server. This

	SET TEXTSIZE statement specifies that up to 2 gigabytes of data can be returned with a SELECT statement.
Setup DLL	The ODBC Administrator calls the function **ConfigDSN** when users configure data sources. For the SQL Server driver, this function is in the driver DLL (SQLSRVR.DLL or SQLSRV32.DLL).
Serializable transactions	The SQL Server driver implements the SQL_TXN_SERIALIZABLE transaction isolation level by adding the HOLDLOCK keyword after each table name in a SELECT statement. For more information, see SELECT in the *Microsoft SQL Server for Windows NT Transact-SQL Reference*.

Active *hstmt* Definition (Programming)

The SQL Server driver can have only one active *hstmt*; it returns this information through **SQLGetInfo** with the SQL_ACTIVE_STATEMENTS option. An *hstmt* is defined as active if it has results pending. In this context, *results* are any information returned by SQL Server, such as a result set or a count of the rows affected by an UPDATE statement.

Note An *hstmt*'s activity isn't related to its state. For example, if a SELECT statement is executed and it doesn't return any rows, the statement isn't active, since no results are pending. However, before the statement can be re-executed, the cursor associated with it must be closed with **SQLFreeStmt**.

The SQL Server driver supports only one active statement on a connection. The cursor library shipped with the SQL Server driver, on the other hand, allows applications to use multiple active statements on a connection.

Cursor Library (Programming)

The SQL Server driver supports only one active statement on a connection. The cursor library shipped with the SQL Server driver, on the other hand, allows applications to use multiple active statements on a connection, and scrollable, updateable cursors.

The cursor library (ODBCCURS.DLL (16-bit) or ODBCCR32.DLL (32-bit)) must be loaded in order to support this functionality. The user calls **SQLSetConnectOption** to specify how the cursor library should be used, and **SQLSetStmtOption** to specify the cursor type, concurrency, and rowset size.

Arithmetic Errors (Programming)

If an arithmetic error such as divide-by-zero or a numeric overflow occurs while data is being retrieved, SQL Server returns a data value of 0 for the column and doesn't report the error until after all data has been retrieved. Consequently, **SQLFetch** (for bound data) or **SQLGetData** (for unbound data) return the error only when the last row of data is retrieved. It is not possible for the SQL Server driver or an application to determine how many errors occurred or in which rows or columns they occurred.

For example, suppose that MyTable has a single column (IntCol), which is of type SQL_INTEGER and is bound to a SQL_C_SHORT storage location and that there are four rows of data: 0, 1, 2, and 3. The statement

```
SELECT 1/IntCol FROM MyTable
```

causes a divide-by-zero error when SQL Server attempts to resolve the expression 1/IntCol for the row containing the value 0. SQL Server sets the value for the row in the result set to 0 (because an arithmetic error occurred). The driver doesn't detect the error when it fetches that row; it detects the error after it has fetched the last (fourth) row of data in the result set, since that is when the error is returned by SQL Server. Consequently, the driver returns SQL_SUCCESS the first three times **SQLFetch** is called and SQL_ERROR the fourth time **SQLFetch** is called.

For more information, see SET ARITHABORT and SET ARITHIGNORE in the *Microsoft SQL Server Language Reference*.

Manual-Commit Mode Transactions (Programming)

When the SQL Server driver is in manual-commit mode, it initiates a transaction with a BEGIN TRANSACTION statement when:

- An SQL statement is pending.
- There is no current transaction.
- The pending SQL statement isn't a restricted Data Definition Language (DDL) statement.

For more information about restricted DDL statements, see the section on the BEGIN TRAN statement in the SQL Server Transact-SQL Reference Manual for the list of statements that cannot be in a TRANS.

To commit or roll back a transaction in manual-commit mode, the application must call **SQLTransact**. The SQL Server driver sends a COMMIT TRANSACTION statement to commit a transaction; it sends a ROLLBACK TRANSACTION statement to roll back a transaction.

A restricted DDL statement can be executed only in manual-commit mode under one of the following circumstances:

- After manual commit mode has been set and before a non-restricted DDL or a Data Manipulation Language (DML) statement has been executed, or
- After a transaction has been committed or rolled back and before a non-restricted DDL or a DML statement has been executed.

For more information about manual-commit mode, see **SQLSetConnectOption** in the *Microsoft ODBC SDK Programmer's Reference*.

Note See the "BEGIN TRANSACTION" section in the *Microsoft SQL Server Language Reference* for the list of statements that cannot be in a TRANS.

Remote Procedure Calls (Programming)

With tabular data stream (TDS) version 4.0 or later, the SQL Server driver uses the remote procedure call (RPC) facility in SQL Server to invoke procedures rather than pass procedures to SQL Server in an SQL statement. A procedure can be a prepared statement (which is stored as a procedure), a procedure called with the ODBC procedure extension, or a stored procedure that the SQL Server driver uses to implement a catalog function. RPCs have the following advantages over procedures passed in an SQL statement:

- RPCs are faster than procedures passed in an SQL statement.
- RPCs can have output parameters; procedures passed in an SQL statement cannot.

(A procedure can return a return value in either case.)

To invoke a statement as an RPC, an application

1. Constructs an SQL statement.
2. Calls **SQLBindParameter** for each parameter in the statement.
3. Prepares the statement with **SQLPrepare**. (Note that the SQL_USE_PROCEDURE_FOR_PREPARE connection option must be set to SQL_UP_ON.)
4. Executes the statement with **SQLExecute**.

To invoke a procedure as an RPC, an application

1. Constructs an SQL statement that uses the ODBC procedure syntax. The statement uses parameter markers for each input, input/output, and output parameter and for the procedure return value (if any).
2. Calls **SQLBindParameter** for each input, input/output, and output parameter and for the procedure return value (if any).
3. Executes the statement with **SQLExecDirect**.

Note If an application submits a procedure using the SQL Server syntax (as opposed to the ODBC procedure extension), the SQL Server driver passes the procedure call to SQL Server as an SQL statement.

Restrictions:

The procedure will be executed as a language event (EXEC *Procname Params*) if:

1. Any literal parameters, or
2. Any parameter is bound to SQL_LONGVARCHAR or SQL_LONGVARBINARY, or
3. Any parameter is bound using SQL_DATA_AT_EXEC.

When the procedure is executed as a language event, no return values or output parameter values are returned.

API

Application programming interface. A set of routines that an application, such as Microsoft Access, uses to request and carry out lower-level services.

character set

A character set is a set of 256 letters, numbers, and symbols specific to a country or language. Each character set is defined by a table called a code page. An OEM (Original Equipment Manufacturer) character set is any character set except the ANSI character set. The ANSI character set (code page 1007) is the character set used by Microsoft Windows.

conformance level

Some applications can use only drivers that support certain levels of functionality, or conformance levels. For example, an application might require that drivers be able to prompt the user for the password for a data source. This ability is part of the Level 1 conformance level for the application programming interface (API).

Every ODBC driver conforms to one of three API levels (Core, Level 1, or Level 2) and one of three SQL grammar levels (Minimum, Core, or Extended). Drivers may support some of the functionality in levels above their stated level.

For detailed information about conformance levels, programmers should see the *Microsoft ODBC SDK Programmer's Reference*.

data source

A data source includes the data a user wants to access and the information needed to get to that data. Examples of data sources are:

- A SQL Server database, the server on which it resides, and the network used to access that server.
- A directory containing a set of dBASE files you want to access.

DBMS

Database management system. The software used to organize, analyze, search for, update, and retrieve data.

DDL

Data definition language. Any SQL statement that can be used to define data objects and their attributes. Examples include CREATE TABLE, DROP VIEW, and GRANT statements.

DLL

Dynamic-link library. A set of routines that one or more applications can use to perform common tasks. The ODBC drivers are DLLs.

DML

Data manipulation language. Any SQL statement that can be used to manipulate data. Examples include UPDATE, INSERT, and DELETE statements.

ODBC

Open Database Connectivity. A Driver Manager and a set of ODBC drivers that enable applications to access data using SQL as a standard language.

ODBC Driver Manager

A dynamic-link library (DLL) that provides access to ODBC drivers.

ODBC driver

A dynamic-link library (DLL) that an ODBC-enabled application, such as Microsoft Excel, can use to gain access to a particular data source. Each database management system (DBMS), such as Microsoft SQL Server, requires a different driver.

SQL

Structured Query Language. A language used for retrieving, updating, and managing data.

SQL statement

A command written in Structured Query Language (SQL); also known as a query. An SQL statement specifies an operation to perform, such as SELECT, DELETE, or CREATE TABLE; the tables and columns on which to perform that operation; and any constraints to that operation.

translation option

An option that specifies how a translator translates data. For example, a translation option might specify the character sets between which a translator translates character data. It might also provide a key for encryption and decryption.

translator

A dynamic-link library (DLL) that translates all data passing between an application, such as Microsoft Access, and a data source. The most common use of a translator is to translate character data between different character sets. A translator can also perform tasks such as encryption and decryption or compression and expansion.