Novell®

SQL *Connector*

ODBC Programmer's Guide

Printing Date: **October 1, 1999**

About This Manual

Purpose of this Manual

This manual describes the programming and use of SQL *C*–ODBC, which is an interface that enables you to use the ODBC API to create applications that connect with an SQL *Connector* Data Source. The SQL *Connector* Data Source is used to maintain and manage a reference list of physical databases and tables. These database table references are stored in the Data Source, and used by SQL *C*–ODBC to connect client applications to these databases.

This manual provides descriptions and examples that are, as nearly as possible, generic for all databases supported by SQL *Connector*. Unless stated as an exception, SQL statements and procedures developed with the material in this document will be portable across all databases, regardless of the specific SQL requirements of the database.

Intended Audience

This document is intended for programmers who will be creating and maintaining applications which use SQL *C*–ODBC to connect to physical databases. It provides details on creating and configuring ODBC data sources, summarizes ODBC API usage, and provides information on how ODBC functions and SQL grammar map to SQL *C*–ODBC features.

This document assumes that programmers are familiar with the ODBC Application Programming Interface (API). For information about the ODBC API, see the *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide.*

Structure of this Manual

This manual consists of chapters which describes how to configure and test SQL *C*–ODBC.

Associated Documents

The SQL *Connector* document set contains these manuals:

- SQL *Connector Overview*
- SQL *Connector Installation Guide*
- SQL *Connector Administration Guide*
- SQL *Connector SQL Grammar Manual*
- SQL *Connector ODBC Programmer's Guide*
- SQL *Connector JDBC Programmer's Guide*

Operating System Conventions

When there are differences in commands, examples, or syntax between operating systems, the following abbreviations are used:

Abbreviation	Meaning
NetWare	the Novell NetWare operating system
Windows	the Microsoft Windows 95/98/NT operating systems

Table of Contents

1

Overview

1.1 Introduction

SQL *Connector* is a Data Request Broker for database access. It provides the capability of using standard Structured Query Language (SQL) to access data in tables in different databases. The tables are cataloged in an SQL *Connector* Data Source, which can then be accessed by client applications using the Open Database Connectivity (ODBC) interface.

The SQL *Connector* Data Source is created and maintained by the Data Source Administrator and documented in the *Administration Guide*.

1.2 Architecture

ODBC client applications can access the SQL *C*–DRB (Data Request Broker) from any system that supports the ODBC environment. The client application uses the SQL *C*–ODBC API to connect to SQL *C*–DRB running on a network server. SQL *C*–DRB uses the SQL *Connector* Data Source to connect to Oracle on the network server or to remote ODBC databases using the SQL *Connector* ODBC Data Driver.

1.2.1 ODBC Client Configuration

An SQL *C*–ODBC client application uses the following Windows layers to connect to the network server and the database server.

- Programming Layer

```
ODBC Driver Manager     odbc32.dll
```

- System Layer

```
Driver Library          tod32.dll, vtx3.dll
Socket Library          wsock32.dll
```

1.2.2 Data Request Broker Configuration

The system runtime environment uses TCP/IP network sockets to communicate with the server. The server executes a program which listens for incoming requests on a specified TCP/IP socket.

- System Layer

```
Network Listener        vtxnetd.nlm, vtx16.nlm
```

- Connection Layer

```
SQLC Host               sqlcmon.nlm, sqlc.nlm
Data Sources            local and remote databases
```

The SQL *Connector* network listener runs on the Data Broker system (typically from the time the Data Broker system is booted). The listener responds to requests from a specified TCP/IP socket. When it receives a network request from the client, it creates a client thread (if one is not already created), which then changes the network request into a database request and calls the SQL *Connector* Data Broker engine. The Data Broker engine uses the Data Source to access physical databases.

1.2.3 Database Drivers Configuration

The SQL *Connector* Data Drivers are used to connect the Data Source to local and remote databases. Remote connectivity is discussed in the *Administration Guide*.

1.3 ODBC Interface

ODBC (Open Database Connectivity) is an industry-standard API (application programming interface) for connecting client applications and data servers. For more information about the ODBC API, see the *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide*.

The ODBC API documentation specifies the names and arguments of functions which define the interface between the client application and the data source. However, within the documentation, there are defined levels of conformance to the ODBC standard, and there are sections of the standard which allow vendor-defined features (for example, datatypes and error messages).

The purpose of this manual is to discuss the implementation and programming of SQL *C*–ODBC as it relates to the ODBC standard.

For a discussion of the SQL grammar supported by SQL *Connector*, see the *SQL Grammar Manual*.

Installation of SQL *C*–ODBC is discussed in the *Installation Guide*, and creation and maintenance of a Data Source is discussed in the *Administration Guide*. The remainder of this document assumes that SQL *C*–ODBC has been successfully installed and a Data Source is available for use.

2

Data Source Setup and Connection

2.1 Introduction

This chapter discusses creating and configuring an SQL *Connector* ODBC data source name (DSN) on a Windows system. For configuration information on NetWare systems, see Appendix G.

A DSN is a Windows ODBC connection to an SQL *Connector* Data Request Broker and Data Source on a network server. A DSN requires the name of the Data Source and the name of the network server that is running the Data Request Broker. Once the DSN is created, an ODBC application can connect to a SQL *Connector* Data Source by using the DSN.

2.2 Data Source Setup

2.2.1 Introduction

The SQL *C*–ODBC installation procedure installs the SQL *C*–ODBC driver and related software. See the *Installation Guide* for the details. The next step is to create an ODBC data source name.

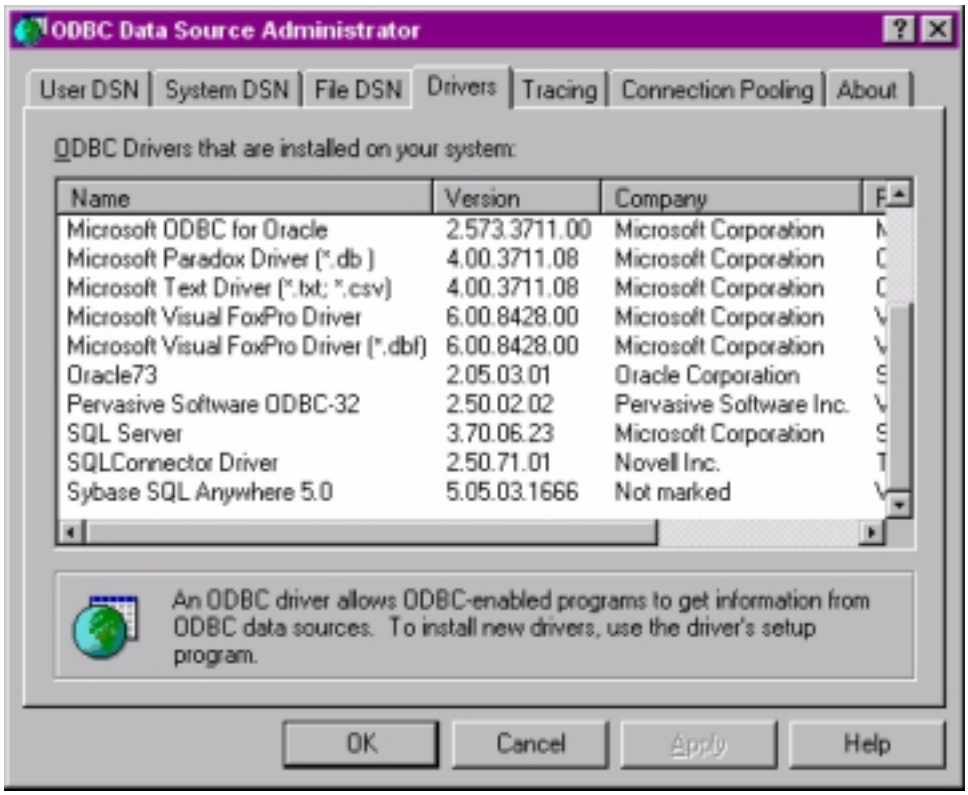
The ODBC data source setup is controlled by the Microsoft ODBC Driver Manager. The Driver Manager displays a series of dialog screens that display the currently installed ODBC drivers and that list, create and remove ODBC data sources that are connected to those drivers. The data source definitions are stored in the system registry.

When a Windows ODBC application executes, the ODBC Driver Manager calls the SQL *Connector* Driver Library (see "ODBC Client Configuration").

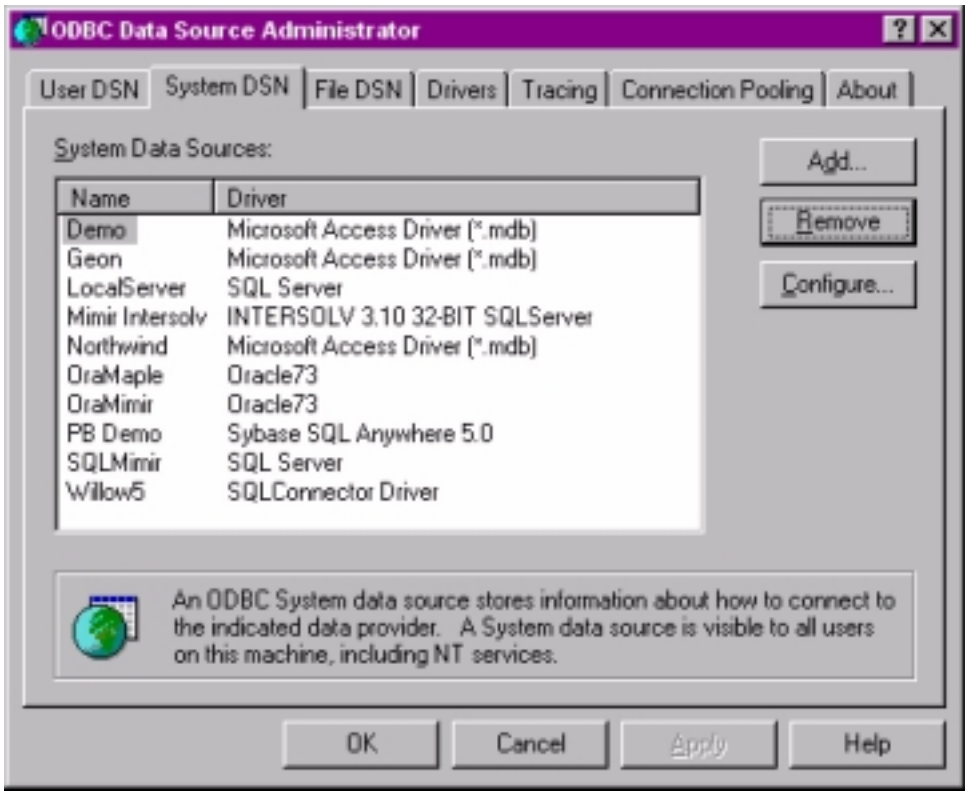
2.2.2 Setup

To access the SQL *C*–ODBC setup dialog box, do the following:

1. From the Control Panel, double-click the ODBC icon.
2. Click on the ODBC Drivers tab.
3. A list of installed drivers will appear. You can confirm installation of the SQL *C*–ODBC driver by the appearance of an entry for the SQL *Connector* Driver in the driver list.

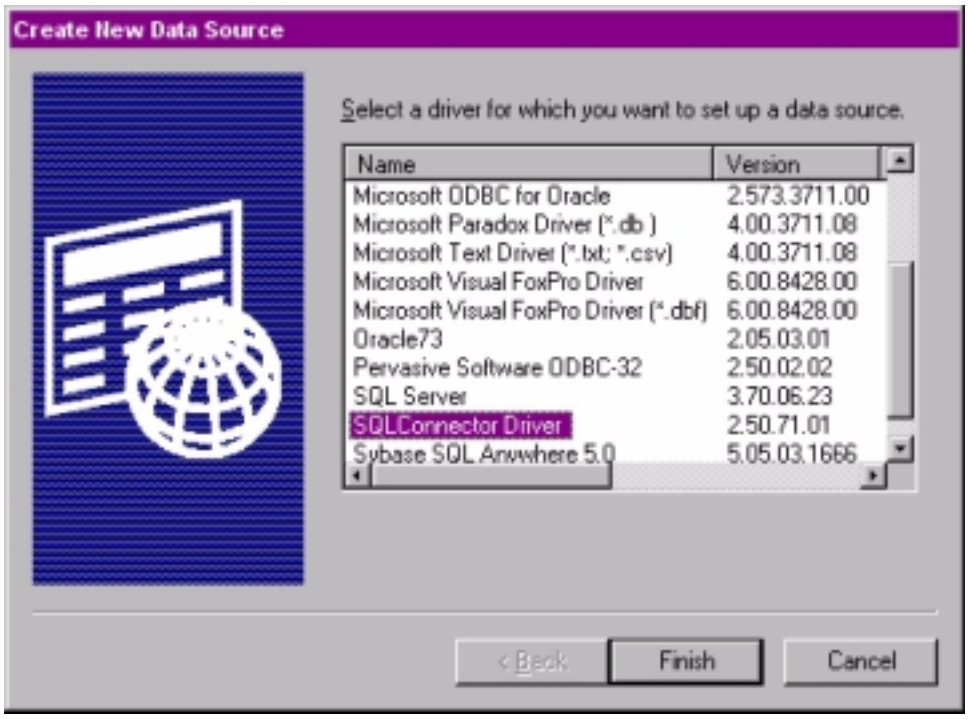


4. Click on the User DSN tab or System DSN tab (System DSN is recommended). A list of currently installed ODBC data sources will be displayed, for example:



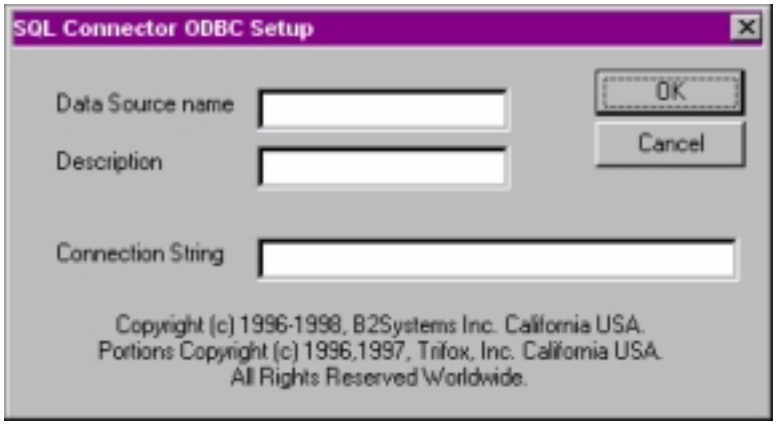
5. Click on the Add button.

The Create New Data Source Dialog box will appear:



6. Click on the row listing the the SQL *Connector* Driver and then click on the Finish button.

The SQL *C*–ODBC Setup dialog box will appear:



2.2.3 Connection Information

The connection information is as follows:

Data Source Name

The Windows name of the ODBC data source, for example, "server1_demo". A recommended convention is to include the Server and Data Source name in the ODBC data source name. In this case, the Server name is "server1" and the Data Source name is "demo".

Description

A description of the data source name. For example "Demo database on Server1".

Connection String

A string for the network connection, which is in the following format:

`/datasource@[port:]host!service[,switchA[,switchB[,switchC...]]]`

datasource – required

The name of the SQL *Connector* Data Source on the network node.

Sample datasource name:

```
demo
```

port – required

A port number used by TCP/IP socket services. This number must match the number used when the SQL *Connector* listener is started on the server.

host – required

The name of the network server or internet address (format nnn.nnn.nnn.nnn) which is running SQL *C*–DRB. The name must be in the system `hosts` file if DNS is not enabled..

Operating System	Hosts File Name and Location
Windows NT	`%SYSTEM_ROOT%\system32\drivers\etc\hosts`
Windows 95/98	`%WINDIR%\hosts`

service – required

Name of the executable host program on the network server:

```
vtx16
```

switchA, switchB, switchC...

SQL *Connector* switch values or environment variables. For a full description, see the *SQL Grammar Manual*. The format is: `name1=value1`

Example:

An example of a complete connection string is shown below:

```
/demo@1958:server1!vtx16,SS_TRACE=yes,SMARTTRACE=sql.trc
```

Example Notes:

1. The Data Source is named demo.
2. TCP/IP port 1958 is used on node server1
3. The host program is named vtx16.

There are two switches used in this example:

a. SS_TRACE = yes

This switch turns on database tracing. All activity with local and remote databases will be written to a trace file. This trace file is useful for debugging and optimizing database applications.

b. SMARTTRACE = sql.trc

This switch names the trace file. If the name is not set, a default name will be used. The default name is s<year-day-number><hour-minute>.trc, for example, s0411634.trc, and the default directory is the location of the loaded programs.

Other switch variables are discussed in the *SQL Grammar Manual.*

2.3 Connecting to the ODBC Data Source

When an application connects to SQL *Connector* using the ODBC function SQLDriverConnect, the application may generate a prompt for data source information. This prompt is usually in the form of a dialog box. Complete the entries as follows:

- Data Source Name: name of the SQL *Connector* data source connection
- Username: <network server username>
- Password: <network server password>

Note that Username and Password in this context provides network server access.

2.4 Changing the ODBC Data Source

To remove or reconfigure an ODBC data source:

1. From the Control Panel, double-click the ODBC icon.
2. Click on the System DSN tab.

   You will see a list of currently installed ODBC data sources. Use the arrow keys to highlight a data source.
3. Click on the Remove button.

   Choose the Yes button to confirm the deletion. The data source will be deleted.
4. Click on the Configure button.

   You will then see same dialog box that was used when the data source was created. You can change the parameters and press OK or Cancel when finished.

2.5 Testing the ODBC Data Source

2.5.1 Sample Programs

The NetWare PUBLIC directory contains a Samples subdirectory that has two programs for testing an SQL *C*–ODBC Data Source connection on two and three tiers. One program (odbcping) verifies the installation by connecting from a client system (first tier) to a Data Source on a network server (second tier) and reporting success or failure. The other program (odbcjoin) connects from a client (first tier) to a Data Source on a network server (second tier) and retrieves joined data using a remote ODBC Data Driver on a database server (third tier). Both programs are supplied in source form (C language) and compiled code (executable file). The source code is listed in Appendices D and E.

The following programs can be copied from the Netware PUBLIC directory to a Windows *client* system (i.e., the system that will run the sample programs). The file locations are:

```
SYS:PUBLIC\SQLC\samples\windows\odbc\odbcjoin.c
SYS:PUBLIC\SQLC\samples\windows\odbc\odbcjoin.exe
SYS:PUBLIC\SQLC\samples\windows\odbc\odbcping.c
SYS:PUBLIC\SQLC\samples\windows\odbc\odbcping.exe
```

2.5.2 Sample Data

A sample database with sample tables is also supplied. The sample data source is a Microsoft Access file that contains tables for departments (dept), jobs (job), employees (emp) and salaries (sal). This sample database must be copied to a Windows remote system that has the ODBC Data Driver installed. The file location is:

```
SYS:PUBLIC\SQLC\samples\windows\access\demo.mdb
```

Once the file is copied, the following steps should be performed:

1. On the database server (third tier), create an ODBC Data Source Name "demo" that uses the Microsoft Access ODBC driver and the database file demo.mdb.
2. On the client system (first tier). run the Data Source Administrator and create a Data Source named "demo" on the network server (second tier) that points to the ODBC Access data source "demo" on the database server (third tier). See the *Administration Guide* for details about this step.

Warning: If the Access database is on a NetWare mounted drive, then the NetWare login and the Windows NT login must be the same username.

The Access database is now available for use with the sample programs.

2.5.3 Using odbcping to Verify a Connection

After creating the SQL *C*–ODBC data source as discussed above, the odbcping program can be executed in an MS–DOS command window or a C/C++ development environment. The SQL *C*–ODBC setup dialog box can have an entry as follows:

```
Data Source name:    Server1_DEMO
Description:         Connection to Node SERVER1 Data Source DEMO
```

```
Connection String:   /demo@1958:server1!vtx16
```

Then odbcping is used as follows:

```
Usage:    odbcping "data source name" username password
Example:  odbcping Server1_DEMO myuser mypass
```

If the ODBC connection is established, this message is displayed:

```
Successful Connection
```

2.5.4 Using odbcjoin to Retrieve Data

After a connection has been verified, the sample data may be retrieved using the odbcjoin program. The odbcjoin program is used as follows:

```
Usage:    odbcjoin "data source name" username password
Example:  odbcjoin DEMO myuser mypass
```

If the data is successfully retrieved, the following text is displayed:

```
select
  dept.deptno, dept.dname, emp.empno, emp.ename, emp.ssno, emp.jobcls
from
  dept, emp
where
  dept.deptno = emp.deptno and dept.deptno < 3
```

Dept.Deptno	Dept.Dname	Emp.Empno	Emp.Ename	Emp.SSNO	Emp.JobCls
1	MARKETING	20265	BASINGER_R	394-90-3583	2
1	MARKETING	20337	GILBREATH_R	553-40-5418	5
1	MARKETING	20365	MILLER_R	248-95-1471	6
1	MARKETING	32024	GOVE_T	742-35-6859	4
1	MARKETING	33297	ELLISON_W	252-60-1283	2
1	MARKETING	33733	BANFIELD_S	688-21-4136	4
1	MARKETING	33771	NASER_K	848-17-0910	8
1	MARKETING	34410	MCCORRY_K	015-84-9007	7
1	MARKETING	35844	MCCARTNEY_H	211-02-7599	7
1	MARKETING	35909	LACY_C	494-12-0675	7
1	MARKETING	38342	KINGSLEY_S	107-42-3232	5
1	MARKETING	39748	BARNETT_N	179-22-5607	4
1	MARKETING	63297	ELLISON_V	152-60-1283	1
1	MARKETING	63647	HAWTHORNE_A	735-64-9259	3
1	MARKETING	63910	BRANNEN_P	355-08-2564	3
1	MARKETING	64689	FULFORD_W	546-61-8895	5
1	MARKETING	67528	BLOCK_F	084-40-0022	3
1	MARKETING	68355	SWIFT_E	358-51-5799	9
2	TRAINING	20384	KNAPP_L	381-26-6148	3
2	TRAINING	31028	ROCHE_D	671-74-2192	9
2	TRAINING	32482	NICHOLL_F	129-79-0929	8
2	TRAINING	34948	ELLINGTON_A	686-40-9928	4
2	TRAINING	35038	ROMANN_H	079-54-8297	9
2	TRAINING	39208	ELDER_N	460-95-5451	2
2	TRAINING	64948	ELLINGTON_Z	586-40-9928	3
2	TRAINING	68551	HOLMES_P	664-23-5820	3
2	TRAINING	69208	ELDER_M	360-95-5451	1

3

ODBC Implementation

3.1 Introduction

The ODBC standard specifies a minimum level of functionality that must be provided by an ODBC driver in order to connect to ODBC applications. This minimum level applies to a set of callable functions, a subset of SQL grammar rules, and a set of datatypes. The ODBC standard also allows vendors to provide additional levels of support, additional functionality, datatypes, etc.

There is a set of ODBC functions that allow an ODBC application to *discover* the functionality available from an ODBC driver. The ODBC application then knows the limits of the driver and does not try to exceed the limits.

This chapter documents the functionality available in the SQL *C*–ODBC driver, as an aid to programmers during application development. The *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide* provides the reference for the information in this chapter.

3.2 Compliance Level

SQL *C*–ODBC supports all of the following ODBC functions:

Core Functions	Level 1 Functions	Level 2 Functions
SQLAllocConnect	SQLBindParameter	SQLBrowseConnect
SQLAllocEnv	SQLColumns	SQLDataSources
SQLAllocStmt	SQLDriverConnect	SQLExtendedFetch
SQLBindCol	SQLDrivers	SQLNumParams
SQLCancel	SQLGetConnectOption	SQLPrimaryKeys
SQLColAttributes	SQLGetData	
SQLConnect	SQLGetFunctions	
SQLDescribeCol	SQLGetInfo	
SQLDisconnect	SQLGetStmtOption	
SQLError	SQLGetTypeInfo	
SQLExecDirect	SQLParamData	
SQLExecute	SQLSetConnectOption	
SQLFetch	SQLSetStmtOption	
SQLFreeConnect	SQLSpecialColumns	
SQLFreeEnv	SQLStatistics	
SQLFreeStmt	SQLTables	
SQLGetCursorName		

Core Functions	Level 1 Functions	Level 2 Functions
SQLNumResultCols		
SQLPrepare		
SQLRowCount		
SQLSetCursorName		
SQLSetParam		
SQLTransact		

3.3 SQL Grammar

SQL *Connector* supports most all of the elements in the minimum SQL grammar, and some elements of the core and extended SQL grammar. For a complete discussion of SQL grammar elements, see Appendix C in the *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide.*

Core elements not supported are Data Definition Language (DDL) statements (`CREATE` and `DROP TABLE`, `CREATE` and `DROP INDEX`). SQL *Connector* uses database table and column information that is imported from existing physical databases. Tables and indexes are first created in the physical database using the appropriate database tools, then imported into SQL *Connector*.

The SQL Grammar elements are categorized as follows:

1. Minimum SQL grammar elements:
   - Data Manipulation Language (DML): simple `SELECT`, `INSERT`, `UPDATE` and `DELETE`
   - Expressions: simple, such as `A > B + C`
   - Data type: `CHAR`
2. Core SQL grammar elements:
   - Minimum SQL grammar and data types
   - DML: full `SELECT`
   - Expressions: subquery and aggregates such as `SUM` and `MIN`
   - Data types: `DECIMAL`, `SMALLINT`, `FLOAT`, `REAL`, `DOUBLE`
3. Extended SQL grammar
   - Minimum and Core SQL grammar and data types
   - Expressions: scalar functions such as `SUBSTRING` and date, time, and timestamp literals
   - Data types: `DATE`, `TIME`, `TIMESTAMP`

3.4 SQL Datatypes

The SQL *Connector* ODBC Driver supports the following SQL datatypes:

SQL *Connector* Datatype	SQL Datatype	Maximum Column Size	Nullable	Case Sensitive	Searchable
char	SQL_CHAR	255	Y	Y	Y
decimal	SQL_NUMERIC	19	Y	N	Y
integer	SQL_INTEGER	10	Y	N	Y
smallint	SQL_SMALLINT	5	Y	N	Y
float	SQL_FLOAT	19	Y	N	Y
real	SQL_REAL	19	Y	N	Y
double	SQL_DOUBLE	19	Y	N	Y
date	SQL_DATETIME	10	Y	N	Y
time	SQL_TIME	8	Y	N	Y
timestamp	SQL_TIMESTAMP	19	Y	N	Y

A

Driver and Connection Attributes

The following table lists the information type and values that are returned by SQLGetInfo when connected to an SQL *Connector* data source. See the discussion of SQLGetInfo in the *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide.*

Information Type	Information Value
Driver Information	
SQL_ACTIVE_CONNECTIONS	8
SQL_ACTIVE_STATEMENTS	256
SQL_DATA_SOURCE_NAME	<ODBC Data Source Name>
SQL_DRIVER_NAME	tod32.dll
SQL_DRIVER_ODBC_VER	02.50
SQL_DRIVER_VER	02.50.1064
SQL_FETCH_DIRECTION	SQL_FD_FETCH_NEXT
SQL_FILE_USAGE	SQL_FILE_NOT_SUPPORTED
SQL_GETDATA_EXTENSIONS	SQL_GD_ANY_COLUMN
	SQL_GD_GET_ANYORDER
	SQL_GD_BOUND
SQL_LOCK_TYPES	***** <not set> *****
SQL_ODBC_API_CONFORMANCE	SQL_OAC_LEVEL1
SQL_ODBC_SAG_CLI_CONFORMANCE	SQL_OSCC_NOT_COMPLIANT
SQL_ODBC_VER	03.00.0000
SQL_POS_OPERATIONS	***** <not set> *****
SQL_ROW_UPDATES	N
SQL_SEARCH_PATTERN_ESCAPE	\
SQL_SERVER_NAME	ODBXHOST
DBMS Product Information	
SQL_DATABASE_NAME	
SQL_DBMS_NAME	SQL Connector
SQL_DBMS_VER	03.01.0000
Data Source Information	
SQL_ACCESSIBLE_PROCEDURES	N
SQL_ACCESSIBLE_TABLES	Y
SQL_BOOKMARK_PERSISTENCE	***** <not set> *****
SQL_CONCAT_NULL_BEHAVIOR	SQL_CB_NON_NULL
SQL_CURSOR_COMMIT_BEHAVIOR	SQL_CB_PRESERVE
SQL_CURSOR_ROLLBACK_BEHAVIOR	SQL_CB_PRESERVE
SQL_DATA_SOURCE_READ_ONLY	N

Information Type	Information Value
SQL_DEFAULT_TXN_ISOLATION	SQL_TXN_REPEATABLE_READ
SQL_MULT_RESULT_SETS	N
SQL_MULTIPLE_ACTIVE_TXN	Y
SQL_NEED_LONG_DATA_LEN	Y
SQL_NULL_COLLATION	SQL_NC_HIGH
SQL_OWNER_TERM	schema
SQL_PROCEDURE_TERM	procedure
SQL_QUALIFIER_TERM	schema
SQL_SCROLL_CONCURRENCY	***** <not set> *****
SQL_SCROLL_OPTIONS	SQL_SO_FORWARD_ONLY
SQL_STATIC_SENSITIVITY	***** <not set> *****
SQL_TABLE_TERM	table
SQL_TXN_CAPABLE	SQL_TC_ALL
SQL_TXN_ISOLATION_OPTION	SQL_TXN_READ_UNCOMMITTED
SQL_USER_NAME	<username>
Supported SQL	
SQL_ALTER_TABLE	***** <not set> *****
SQL_COLUMN_ALIAS	N
SQL_CORRELATION_NAME	SQL_CN_ANY
SQL_EXPRESSIONS_IN_ORDERBY	Y
SQL_GROUP_BY	SQL_GB_GROUP_BY_CONTAINS_SELECT
SQL_IDENTIFIER_CASE	SQL_IC_LOWER
SQL_IDENTIFIER_QUOTE_CHAR	"
SQL_KEYWORDS	<See Appendix B>
SQL_LIKE_ESCAPE_CLAUSE	Y
SQL_NON_NULLABLE_COLUMNS	SQL_NNC_NON_NULL
SQL_ODBC_SQL_CONFORMANCE	SQL_OSC_EXTENDED
SQL_ODBC_SQL_OPT_IEF	N
SQL_ORDER_BY_COLUMNS_IN_SELECT	Y
SQL_OUTER_JOINS	Y
(2.01) SQL_OJ_CAPABILITIES	***** <not set> *****
SQL_OWNER_USAGE	SQL_OU_DML_STATEMENTS
	SQL_OU_TABLE_DEFIITION
	SQL_OU_INDEX_DEFINITION
SQL_POSITIONED_STATEMENTS	***** <not set> *****
SQL_PROCEDURES	N
SQL_QUALIFIER_LOCATION	SQL_QL_START
SQL_QUALIFIER_NAME_SEPARATOR	.
SQL_QUALIFIER_USAGE	SQL_QU_DML_STATEMENTS
	SQL_QU_TABLE_DEFINITION
	SQL_QU_INDEX_DEFINITION

Information Type	**Information Value**
SQL_QUOTED_IDENTIFIER_CASE	SQL_IC_LOWER
SQL_SPECIAL_CHARACTERS	_#$@
SQL_SUBQUERIES	SQL_SQ_EXISTS
	SQL_SQ_IN
	SQL_SQ_CORRELATED_SUBQUERIES
SQL_UNION	***** <not set> *****
SQL Limits	
SQL_MAX_BINARY_LITERAL_LEN	0
SQL_MAX_CHAR_LITERAL_LEN	0
SQL_MAX_COLUMN_NAME_LEN	31
SQL_MAX_COLUMNS_IN_GROUP_BY	0
SQL_MAX_COLUMNS_IN_INDEX	0
SQL_MAX_COLUMNS_IN_ORDER_BY	0
SQL_MAX_COLUMNS_IN_SELECT	0
SQL_MAX_COLUMNS_IN_TABLE	0
SQL_MAX_CURSOR_NAME_LEN	0
SQL_MAX_INDEX_SIZE	0
SQL_MAX_OWNER_NAME_LEN	0
SQL_MAX_PROCEDURE_NAME_LEN	0
SQL_MAX_QUALIFIER_NAME_LEN	0
SQL_MAX_ROW_SIZE	0
SQL_MAX_ROW_SIZE_INCLUDES_LONG	
SQL_MAX_STATEMENT_LEN	0
SQL_MAX_TABLE_NAME_LEN	31
SQL_MAX_TABLES_IN_SELECT	0
SQL_MAX_USER_NAME_LEN	0
Scalar Function Information	
SQL_CONVERT_FUNCTIONS	***** <not set> *****
SQL_NUMERIC_FUNCTIONS	SQL_FN_NUM_ABS
	SQL_FN_NUM_MOD
SQL_STRING_FUNCTIONS	SQL_FN_STR_CONCAT
	SQL_FN_STR_SUBSTRING
	SQL_FN_STR_CHAR
SQL_SYSTEM_FUNCTIONS	SQL_FN_SYS_IFNULL
SQL_TIMEDATE_ADD_INTERVALS	***** <not set> *****
SQL_TIMEDATE_DIFF_INTERVALS	***** <not set> *****
SQL_TIMEDATE_FUNCTIONS	***** <not set> *****
Conversion Information	
SQL_CONVERT_BIGINT	***** <not set> *****
SQL_CONVERT_BINARY	***** <not set> *****
SQL_CONVERT_BIT	***** <not set> *****

Information Type	Information Value
SQL_CONVERT_CHAR	***** <not set> *****
SQL_CONVERT_DATE	***** <not set> *****
SQL_CONVERT_DECIMAL	***** <not set> *****
SQL_CONVERT_DOUBLE	***** <not set> *****
SQL_CONVERT_FLOAT	***** <not set> *****
SQL_CONVERT_INTEGER	***** <not set> *****
SQL_CONVERT_LONGVARBINARY	***** <not set> *****
SQL_CONVERT_LONGVARCHAR	***** <not set> *****
SQL_CONVERT_NUMERIC	***** <not set> *****
SQL_CONVERT_REAL	***** <not set> *****
SQL_CONVERT_SMALLINT	***** <not set> *****
SQL_CONVERT_TIME	***** <not set> *****
SQL_CONVERT_TIMESTAMP	***** <not set> *****
SQL_CONVERT_TINYINT	***** <not set> *****
SQL_CONVERT_VARBINARY	***** <not set> *****
SQL_CONVERT_VARCHAR	***** <not set> *****

B

SQL Keywords

The following keywords are reserved for use by SQL *Connector*. If there is existing SQL metadata (tables and columns) that already use these keywords, then references to the metadata must be surrounded by double quotes. If new metadata (tables and columns) is created, then the metadata should not use words in this list.

abort	att_name	audit	average	bcd	bcdfixed
bcdflt	binary	bottom	breakable	call	carrycolumns
center	checkpoint	columnno	committed	concat	copyin
copyout	cumulative	currentdate	currentmoment	currenttime	database
databasename	datatype	days	daytime	dba	detail
detailno	editstring	elseif	enddo	endif	endpage
endsection	endstore	endstream	errorlimit	exclusive	exit
fetchnumber	fieldcomment	fieldinfo	fieldlabel	filestatus	fixed
float4	float8	footsize	format	forms	function
getdate	gettime	giving	groupcount	groupno	guide
head	header	headformat	heading	highlighting	host
hostfield	hostparameter	hours	ifnull	index	initpage
initsection	inout	inquire	int1	int2	int4
int8	isnull	killdbin	leave	leftmargin	lineno
lowercase	maximum	message	midnight	minimum	minlocks
minutes	months	need	none	nvl	oddpage
outerjoin	overwriting	page	pagebottom	pagelength	pageno
pagetop	paging	panelwidth	parameter comment	presorted	print
prompt	prompting	protected	reconfigure	recordno	recordtype
recover	rel_id	rel_name	repeatable	report	reportbottom
reporttop	reserving	return	returns	rightmargin	savepoint
seconds	sectionno	segmentsize	separation	serializable	shared
show	skip	skipping	space	spacing	spreading
start	startpage	startsection	status	stddev	store
stream	sync	synonym	to_char	to_date	to_number
today	top	total	truncate	unbreakable	uncommitted
unsafe	unset	uppercase	userid	variance	vaxday
vaxhour	vaxmidnight	vaxminute	vaxsecond	vaxtime	verify
week	weeks	while	width	years	

C

ODBC Errors

When an error occurs, the SQL *C*–ODBC driver returns the native error number, the SQLSTATE (an ODBC error code), and an error message. The SQL *C*–ODBC driver gets this information from errors that are detected by the driver itself and from errors that are returned by SQL *C*–DRB.

For errors that occur within the Data Source or from data sources connected to the Data Source, the SQL *C*–ODBC driver returns the native error number that is returned by SQL *C*–DRB, and the message identification and the message text. For a list of native error numbers, see the table `sqlmsg` in the Data Source named `msgdb`. This table has four columns:

```
msgnum       integer not null
facility     char (8) not null
msgid        char (26) not null
msgtxt       char (160) not null
```

The message number, message identification and message text are the first, third and fourth columns.

Error Message Syntax

Error messages have the following format:

```
SQLSTATE native_error [vendor][ODBC_component][data_source] error_message
```

The error message returned by SQL *C*–DRB is the concatenation of the message identification and the message text. There may be more than one error line returned.

Example Error Messages

If the ODBC data source is not found:

```
IM002    0 [Microsoft][ODBC Driver Manager] Data source name not found
                                        and no default driver specified
```

If the Data Source is not found:

```
S1000   -1 [TOD][ODBC Driver][Net]%RQP-E-DICPARSE,
   Can't Parse Dictionary File Name '../examples/xxx'
   %RMS-E-FNF, file not found
```

If a table in the Data Source is not found:

```
S1000  -23457819 [TOD][ODBC][SQLI]%RQP-E-TABUND,
    Table emp Undefined in Dictionary File ../examples/demo
```

D

ODBCPING Source Code

```
// ODBC Ping Test

#include <windows.h>
#include <stdio.h>
#include <sqlext.h>

void* checkError (SQLHENV envHandle, SQLHDBC dbcHandle,
                  SQLHSTM stmtHandle, SQLRETURN retCode, char* retmsg);

//--------------------------------------------------------------------
main (int argc, char *argv[])
{
  SQLRETURN rc;
  SQLHENV   henv;
  SQLHDBC   hdbc;
  SQLCHAR   dsn[63];
  SQLCHAR   uid[63];
  SQLCHAR   pwd[63];
  char      msg[255];
  // arg[0]= executable name, arg[1] = dsn, arg[2] = uid, arg[3] = pwd
  if (argc < 3) {
    printf ("Usage: odbcping \"data source name\" username password\n");
    goto finish;
    }

  strcpy ((char*)dsn, argv[1]);
  strcpy ((char*)uid, argv[2]);
  strcpy ((char*)pwd, argv[3]);
  printf ("\nConnecting to ODBC Data Source, dsn=%s, uid=%s, pwd=%s\n",
         dsn, uid, pwd);
  rc = SQLAllocEnv(&henv);
  checkError (SQL_NULL_HENV, SQL_NULL_HDBC, SQL_NULL_HSTMT, rc, msg);
  if (rc == SQL_SUCCESS) {
    rc = SQLAllocConnect (henv, &hdbc);
    checkError (henv, SQL_NULL_HDBC, SQL_NULL_HSTMT, rc, msg);
    if (rc == SQL_SUCCESS) {
      rc = SQLConnect (hdbc, dsn, SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
      checkError (henv, hdbc, SQL_NULL_HSTMT, rc, msg);
      if (rc == SQL_SUCCESS)
  printf ("\nSuccessful Connection\n");
    }
  }
  rc = SQLDisconnect (hdbc);
  rc = SQLFreeConnect (hdbc);
  rc = SQLFreeEnv (henv);
  finish:
  return 0;
}
```

```
//--------------------------------------------------------------------
void* checkError (SQLHENV envHandle, SQLHDBC dbcHandle,
                  SQLHSTMT stmtHandle, SQLRETURN retCode, char* retmsg)
{
  SQLRETURN    krc;
  SQLCHAR      sqlState[5];
  SQLINTEGER   sqlNativeError;
  SQLCHAR      msg[255];
  SQLSMALLINT  lmsg;
  strcpy ((char*)retmsg, "[SQL_SUCCESS]");
  if (retCode != SQL_SUCCESS) {
    krc = SQLError (envHandle, dbcHandle, stmtHandle, sqlState,
          &sqlNativeError, msg, sizeof(msg), &lmsg);
    while (krc != SQL_NO_DATA_FOUND) {
      sprintf (retmsg, "%s %5i %s", sqlState, sqlNativeError, msg);
      printf ("%s\n", retmsg);
      krc = SQLError (envHandle, dbcHandle, stmtHandle, sqlState,
          &sqlNativeError, msg, sizeof(msg), &lmsg);
      }
    }
  return 0;
}
```

E

ODBCJOIN Source Code

```
// ODBC Join Test

#include <windows.h>
#include <stdio.h>
#include <sqlext.h>

void* checkError (SQLHENV envHandle, SQLHDBC dbcHandle,
      SQLHSTMT stmtHandle, SQLRETURN returnCode, char* retmsg);
char* getData (SQLHENV envHandle, SQLHDBC dbcHandle,
      SQLHSTMT stmtHandle, SQLUSMALLINT colNo, int outLen);
//---------------------------------------------------------------------
main (int argc, char *argv[])
{
  SQLRETURN  rc;
  SQLHENV    henv;
  SQLHDBC    hdbc;
  SQLHSTMT   hstmt;
  char       dsn[31];
  char       uid[31];
  char       pwd[31];
  char       msg[255];

  // arg[0]= executable name, arg[1] = dsn, arg[2] = uid, arg[3] = pwd
  if (argc < 4) {
    printf ("\nUsage: odbcjoin "data source name" username password\n");
    goto error;
    }
  strcpy ((char*)dsn, argv[1]);
  strcpy ((char*)uid, argv[2]);
  strcpy ((char*)pwd, argv[3]);

  // Allocate environment, connection and statement handles
  // connect to database
  rc = SQLAllocEnv (&henv);
  rc = SQLAllocConnect (henv, &hdbc);
  rc = SQLConnect (hdbc, (SQLCHAR*)dsn, SQL_NTS, (SQLCHAR*)uid,
           SQL_NTS, (SQLCHAR*)pwd, SQL_NTS);
    checkError (henv, hdbc, SQL_NULL_HSTMT, rc, msg);
    if (rc != SQL_SUCCESS) goto error;
  rc = SQLAllocStmt (hdbc, &hstmt);

  printf ("\nselect\n");
  printf ("  dept.deptno, dept.dname, emp.empno, emp.ename,
                emp.ssno, emp.jobcls\n");
  printf ("from\n");
  printf ("  dept, emp\n");
  printf ("where\n");
  printf ("  dept.deptno = emp.deptno and dept.deptno < 3\n\n");
```

```
  printf ("Dept.Deptno  Dept.Dname  Emp.Empno  Emp.Ename      Emp.SSNO
             Emp.JobCls\n");
  printf ("-------------------------------------------------------\n");
  rc = SQLExecDirect (hstmt, (SQLCHAR*)"select d.deptno, d.dname,
          e.empno, e.ename, e.ssno, e.jobcls from dept d, emp e where
          dept.deptno = emp.deptno and dept.deptno < 3", SQL_NTS);
    checkError (henv, hdbc, hstmt, rc, msg);
    if (rc != SQL_SUCCESS) goto finish;
  rc = SQLFetch (hstmt);
    checkError (henv, hdbc, hstmt, rc, msg);
    if (rc != SQL_SUCCESS) goto finish;

  while (rc != SQL_NO_DATA_FOUND) {
    strcpy (msg, "     ");
    strcat (msg, getData (henv, hdbc, hstmt, 1,  8));
    strcat (msg, getData (henv, hdbc, hstmt, 2, 13));
    strcat (msg, getData (henv, hdbc, hstmt, 3, 10));
    strcat (msg, getData (henv, hdbc, hstmt, 4, 15));
    strcat (msg, getData (henv, hdbc, hstmt, 5, 16));
    strcat (msg, getData (henv, hdbc, hstmt, 6,  7));
    printf ("%s\n",msg);
    rc = SQLFetch (hstmt);
    checkError (henv, hdbc, hstmt, rc, msg);
    }
  printf("-------------------------------------------------------\n");
finish:
  rc = SQLFreeStmt (hstmt, SQL_DROP);
  rc = SQLDisconnect (hdbc);
  rc = SQLFreeConnect (hdbc);
  rc = SQLFreeEnv (henv);
error:
  return 0;
}

//---------------------------------------------------------------------
void* checkError (SQLHENV envHandle, SQLHDBC dbcHandle,
                  SQLHSTMT stmtHandle, SQLRETURN returnCode, char* retmsg)
{
  SQLRETURN    lrc;
  SQLCHAR      sqlState[5];
  SQLINTEGER   sqlNativeError;
  SQLCHAR      msg[255];
  SQLSMALLINT  lmsg;

  strcpy ((char*)retmsg, "[SQL_SUCCESS]");
  if (returnCode != SQL_SUCCESS) {
    lrc = SQLError (envHandle, dbcHandle, stmtHandle, sqlState,
                    &sqlNativeError, msg, sizeof(msg), &lmsg);
    while (lrc != SQL_NO_DATA_FOUND) {
      sprintf (retmsg, "%s %5i %s", sqlState, sqlNativeError, msg);
      printf ("%s\n", retmsg);
      lrc = SQLError (envHandle, dbcHandle, stmtHandle, sqlState,
                      &sqlNativeError, msg, sizeof(msg), &lmsg);
      }
    }
  return NULL;
}
```

```
//---------------------------------------------------------------------
char* getData (SQLHENV envHandle, SQLHDBC dbcHandle,
               SQLHSTMT stmtHandle, SQLUSMALLINT colNo, int outLen)
{
  SQLRETURN    lrc;
  SQLINTEGER   ind;
  static char  val[255], msg[255];
  int          i;

  lrc = SQLGetData (stmtHandle, colNo, SQL_CHAR, (SQLCHAR*)val, sizeof
                    val, &ind);
  checkError (envHandle, dbcHandle, stmtHandle, lrc, msg);

  if (ind == SQL_NULL_DATA) {
    strcpy (val,"~");
    ind = 1;
    }
  for (i=ind; i<outLen; i++) {
    strcat (val, " ");
    }
  return val;
 }
```

F

Programming Notes

F.1 SQL Data Types

F.1.1 SQL Double Datatype Restriction

The maximum value of an SQL Double datatype is `1.0e+126`. The SQL-92 documented maximum value is `1.797693134862316e+308`.

F.2 SQL Functions

F.2.1 SQLBindParameter Restriction

SQLBindParameter is used to bind a program variable and its null value indicator to a parameter in an SQL statement. The null value indicator must be set before the SQLBindParameter is called.

G

ODBC NetWare Client Driver

G.1 Introduction

The SQL *Connector* ODBC client driver is also available as a NetWare NLM. The characteristics and behavior of the NetWare version of the driver are identical to the Windows version of the driver. The NetWare ODBC Client Driver can be accessed by NetWare applications which call ODBC API functions, such as Web Servers. The NetWare ODBC Client Driver uses the NetWare Data Request Broker and Data Source to gain access to local and remote databases. The Data Source is created and maintained by the Data Source Administrator and documented in the *Administration Guide*. There are also two test programs (`odbcping` and `odbcjoin`) that are distributed as NLMs.

G.2 Architecture

NetWare ODBC client applications can access the SQL *C*–DRB (Data Request Broker) from the same system that supports the NetWare ODBC Client. The client application uses the ODBC API (see Section 1.3) to connect to SQL *C*–DRB running on a network server. SQL *C*–DRB uses the Data Source to connect to Oracle on the network server or to remote databases using the SQL *Connector* ODBC Data Driver.

G.2.1 NetWare ODBC Client Configuration

An SQL *C*–ODBC client application uses the following NetWare layers to connect to the network server and the database server.

- Programming Layer

```
ODBC Driver:                sqlcodbc.nlm
Connection Driver:          vtxodbc.nlm
Environment Variables:      vtxenv.nlm
```

- The NetWare ODBC Client Driver is loaded using:

```
load sqlcodbc
```

G.2.2 Data Request Broker Server Configuration

The ODBC driver directly connects to the SQL *Connector* Data Request Broker.

- Connection Layer

```
SQLC Host:              module sqlcmon.nlm, sqlc.nlm
Data Source:            local and remote databases
```

G.3 Installation

The NetWare ODBC client driver is automatically installed when NetWare is installed. No additional steps are necessary.

G.4 Data Source Setup

G.4.1 Data Source Names

On Windows systems, the ODBC data sources are controlled by the Microsoft OBDC Driver Manager. The Driver Manager displays a series of dialog screens that display the currently installed OBDC drivers and that list, create and remove OBDC data sources that are connected to those drivers. The data source definitions are stored in the system registry.

On NetWare systems, the ODBC data sources are controlled by the Data Source Administrator. The Data Source Administrator has a series of web pages the display the current Data Sources and that create and remove Data Sources and list tables in Data Sources. For further information, see the *Administration Guide*.

Warning: Data Sources names are unique on each Netware server. If a Data Sources name already exists when *any* user creates a new Data Source with the same name using the Data Source Administrator, an error will occur, and the Data Source will not be created.

G.4.2 ODBC Trace Files

ODBC tracing can be started by setting the environment variable `odbc_trace = y`. When tracing is on, files are created in `SYS:\` with names like `ODBC9999.LOG`, where `9999` is a portion of the SQL *Connector* thread id.

The environment variable is set by loading the ODBC environment NLM as follows:

```
load vtxenv odbc_trace=y
```

G.5 Testing the ODBC Data Source

G.5.1 Sample Programs

The NetWare PUBLIC directory contains a Samples subdirectory that has two NLM programs for testing an SQL *C*–ODBC Data Source using one and two tiers. One program (odbcping) verifies the installation by connecting to a Data Source on the NetWare server (first tier) and reporting success or failure. The other program (odbcjoin) connects to a Data Source on the NetWare server (first tier) and retrieves joined data using a remote ODBC Data Driver on a database server (second tier). Both programs are supplied in source form (C language) and compiled code (Netware loadable module). The source code is listed in Appendices D and E.

The following programs can be copied from the Netware PUBLIC directory to any NetWare directory that is in the Search Path. The file locations are:

```
SYS:PUBLIC\SQLC\samples\netware\odbc\odbcjoin.c
SYS:PUBLIC\SQLC\samples\netware\odbc\odbcjoin.nlm
SYS:PUBLIC\SQLC\samples\netware\odbc\odbcping.c
SYS:PUBLIC\SQLC\samples\netware\odbc\odbcping.nlm
```

G.5.2 Sample Data

A sample database with sample tables is also supplied. The sample data source is a Microsoft Access file that contains tables for departments (dept), jobs (job), employees (emp) and salaries (sal). This sample database must be copied to a Windows remote system that has the ODBC Data Driver installed. The file location is:

```
SYS:PUBLIC\SQLC\samples\windows\access\demo.mdb
```

Once the file is copied, the following steps should be performed:

1. On the database server, create an ODBC Data Source Name "demo" that uses the Microsoft Access ODBC driver and the database file demo.mdb.
2. Run the Data Source Administrator and create a Data Source named "demo" on the network server that points to the ODBC Access data source "demo" on the database server. See the *Administration Guide* for details about this step.

Warning: If the Access database is on a NetWare mounted drive, then the NetWare login and the Windows NT login must be the same username.

The Access database is now available for use with the sample programs.

G.5.3 Using odbcping to Verify a Connection

After creating the Data Source as discussed above, the odbcping program can be executed from the NetWare console (since it has been copied to a directory in the search path). The odbcping program is used as follows:

```
Usage:    odbcping "Data Source" username password
Example:  odbcping demo myuser mypass
```

If the ODBC connection is established, this message is displayed:

```
Successful Connection
```

G.5.4 Using odbcjoin to Retrieve Data

After a connection has been verified, the sample data may be retrieved using the odbcjoin program (since it has been copied to a directory in the search path). The odbcjoin program is used as follows:

```
Usage:    odbcjoin "Data Source" username password
Example:  odbcjoin demo myuser mypass
```

If the data is successfully retrieved, the following text is displayed:

```
select
  dept.deptno, dept.dname, emp.empno, emp.ename, emp.ssno, emp.jobcls
from
  dept, emp
where
  dept.deptno = emp.deptno and dept.deptno < 3
```

Dept.Deptno	Dept.Dname	Emp.Empno	Emp.Ename	Emp.SSNO	Emp.JobCls
1	MARKETING	20265	BASINGER_R	394-90-3583	2
1	MARKETING	20337	GILBREATH_R	553-40-5418	5
1	MARKETING	20365	MILLER_R	248-95-1471	6
1	MARKETING	32024	GOVE_T	742-35-6859	4
1	MARKETING	33297	ELLISON_W	252-60-1283	2
1	MARKETING	33733	BANFIELD_S	688-21-4136	4
1	MARKETING	33771	NASER_K	848-17-0910	8
1	MARKETING	34410	MCCORRY_K	015-84-9007	7
1	MARKETING	35844	MCCARTNEY_H	211-02-7599	7
1	MARKETING	35909	LACY_C	494-12-0675	7
1	MARKETING	38342	KINGSLEY_S	107-42-3232	5
1	MARKETING	39748	BARNETT_N	179-22-5607	4
1	MARKETING	63297	ELLISON_V	152-60-1283	1
1	MARKETING	63647	HAWTHORNE_A	735-64-9259	3
1	MARKETING	63910	BRANNEN_P	355-08-2564	3
1	MARKETING	64689	FULFORD_W	546-61-8895	5
1	MARKETING	67528	BLOCK_F	084-40-0022	3
1	MARKETING	68355	SWIFT_E	358-51-5799	9
2	TRAINING	20384	KNAPP_L	381-26-6148	3
2	TRAINING	31028	ROCHE_D	671-74-2192	9
2	TRAINING	32482	NICHOLL_F	129-79-0929	8
2	TRAINING	34948	ELLINGTON_A	686-40-9928	4
2	TRAINING	35038	ROMANN_H	079-54-8297	9
2	TRAINING	39208	ELDER_N	460-95-5451	2
2	TRAINING	64948	ELLINGTON_Z	586-40-9928	3
2	TRAINING	68551	HOLMES_P	664-23-5820	3
2	TRAINING	69208	ELDER_M	360-95-5451	1