

# Dr. DeeBee<sup>®</sup> ODBC Driver Kit (Bronze Edition)

## Overview

The Dr. DeeBee ODBC Driver Kit (Bronze Edition) contains the C source code for a fully functional ODBC driver (Level 1 API, minimum+ SQL grammar). Programmers can modify this source code to provide ODBC access their own proprietary databases.

The source code provides a complete ODBC driver implementation. The driver has the following limitations:

- Only minimum level SQL is supported, with some additional core level SQL functionality. See Appendix A for the SQL grammar supported.
- Only the level 1 ODBC API calls are supported.
- Column and table names are not case-sensitive (this can easily be changed). String data comparisons are case sensitive.
- Character and binary values supplied for parameterized queries (SELECT \* FROM EMP WHERE NAME = ?) are limited to 255 bytes.
- Transactions are not supported.
- Qualifiers or owners are not allowed on databases, tables, etc.
- There is limited query optimization. No indexes are used, although there are hooks in place to allow you to use indexes.

In the source code provided, simple dBASE files (dBASE files without indexes or memos) are used as the underlying database. All dBASE specific code is isolated to two modules: ISAM.C and DBASE.C.

Our initial testing shows that this driver will work with off-the-shelf applications such as Access, Visual Basic, PowerBuilder, MFC, Visual C/C++, etc. If you encounter problems, please let us know.

## New Features in the Bronze Edition

This release adds the following new features to the Standard Edition:

- SQL statements can be now be passed through to be processed by the backend.
- DATE, TIME, TIMESTAMP, BINARY, and all the numerical data types have been implemented.
- A true BCD (Binary Coded Decimal) representation is used for SQL\_NUMERIC, SQL\_DECIMAL, and SQL\_BIGINT.
- SELECT DISTINCT is now implemented.

- Multiple types from the underlying database can be mapped to the same SQL\_x type.
- The parser returns more meaningful error messages, rather than "Syntax Error".
- The semanticizer returns more meaningful error messages, rather than "Semantic Error".
- The back slash (\) can be used as the search pattern escape character in the catalog functions.
- Errors from the ISAM layer are preserved throughout the system, rather than being changed to "ISAM Error". It is also possible to specify the error messages returned by SQLError() for these errors. See ISAMGetErrorMessage() in ISAM.H for instructions on how to do this.
- The escape clause for dates, times, and timestamps has been implemented.
- The optimizer has been improved to use ISAMRestrict() more often.
- Tables are iterated over in the order specified in the FROM clause of a SELECT. They used to be iterated over in the reverse order.
- Multiple columns can be specified in an ORDER BY clause of a SELECT.
- GROUP BY and HAVING can be specified in a SELECT.
- The AVG(), COUNT(), MAX(), MIN(), and SUM() functions have been implemented.
- The IN and NOT IN operator have been implemented.
- The anonymous union has been removed from PARSE.H.
- Comments in the code are enclosed in /\* \*/ rather than be preceded by //.

See Appendix B for notes on how to migrate from the Standard Edition to the Bronze Edition.

## What You Need

In addition to the Dr. DeeBee ODBC Driver Kit (Bronze Edition), you will need the following to build and distribute your ODBC driver:

- A C/C++ compiler such as Microsoft C 7.0, Visual C/C++, etc.
- The ODBC SDK. The ODBC SDK is available directly from Microsoft as part of MSDN Level 2. Call Microsoft at 1-800-759-5474 or 1-206-882-8080 for more information.

It is assumed that the user has working knowledge of what ODBC is, the ODBC API, how to use ODBC, etc. Recommended reading in these areas is:

- *Windows Multi-DBMS Programming*; Ken North; John Wiley & Sons; 1995
- *Inside ODBC*; Kyle Geiger; MS Press; expected June 1995
- *The ODBC Solution*; Robert Signore, John Creamer, Micheal O. Stegman; McGraw-Hill; 1995
- *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide*; Microsoft Press; 1994

## SYWARE Support

Included with your purchase is one hour of telephone support at 617-497-1300. Additional telephone support is available on a fee-per-event basis. Questions can also be E-mailed to support@syware.com or CompuServe 75310,2100.

If you find bugs or have other feedback, we welcome your comments. Please E-mail them to support@syware.com or CompuServe 75310,2100.

## Installing the Dr. DeeBee ODBC Driver Kit (Bronze Edition)

### **WARNING:**

**If you are installing over a previous version of the kit, backup any old installation of the kit before installing. The installation overwrites everything in any previous installation without making any backup files.**

To install the Dr. DeeBee ODBC Driver Kit (Bronze Edition), do the following:

- Install your C compiler.
- Install the ODBC SDK.
- Put the installation disk in drive A: and run SETUP.EXE.

Once installed, you will see the following:

C:\DRDBDR	- Your installation directory (the default)
C:\DRDBDR\SOURCE	- ODBC driver source code
C:\DRDBDR\SETUP	- 16-bit ODBC driver DLL and installer
C:\DRDBDR\SETUP32	- 32-bit ODBC driver DLL and installer

C:\DRDBDR\SETUP\DRDBDR.DLL is a 16-bit compiled version of the source code found in C:\DRDBDR\SOURCE. C:\DRDBDR\SETUP32\DRDBDR32.DLL is a 32-bit compiled version of the source code found in C:\DRDBDR\SOURCE.

## Getting Started

You will probably want to start by rewriting ISAM.C to read and write data from your database (rather than dBASE files). Once this is done, you will have an ODBC driver for your DBMS or file format with the same limitations specified in the Overview section above. See Appendix C for a suggested strategy on how to implement ISAM.C.

After this is done, you can add functionality to your driver by modifying the rest of the code.

## Distributing Your Driver

Except for DRDBDR.DLL and DRDBDR32.DLL; none of the components in the Dr. DeeBee ODBC Driver Kit (Bronze Edition) are redistributable. The driver you create using the kit (in the form of a DLL) is redistributable royalty-free in most cases. See your license (Appendix D) for more details.

If you copy C:\DRDBDR\SETUP\\*. \* to a diskette, you will have a setup disk that will load the 16-bit Dr. DeeBee ODBC Sample Driver. If you copy C:\DRDBDR\SETUP32\\*. \* to a diskette, you will have a setup disk that will load the 32-bit Dr. DeeBee ODBC Sample Driver. However, your license agreements with SYWARE only allows you to redistribute DRDBDR.DLL and DRDBDR32.DLL. Your right to redistribute the other components in C:\DRDBDR\SETUP and C:\DRDBDR\SETUP32 can be obtained from your MSDN Level 2 license.

To create a setup disk for your 16-bit driver, do the following:

- Delete C:\DRDBDR\SETUP\DRDBDR.DLL.
- Copy your DLL into C:\DRDBDR\SETUP.
- Edit C:\DRDBDR\SETUP\ODBC.INF as follows:
  - In the section [Dr. DeeBee Sample Driver 16], change DRDBDR.DLL to the name of your DLL, change 1997-02-11 to the date on your DLL, and change 179344 to the size of your DLL.
  - Search for "Dr. DeeBee" elsewhere in the ODBC.INF and make the appropriate changes.

To create a setup disk for your 32-bit driver, do the same in the C:\DRDBDR\SETUP32 directory.

## Passthrough SQL

Before an SQL query is processed by the upper levels of the system, it is first passed to ISAMPrepare(). This gives the ISAM layer the option of passing it to a SQL backend or having the upper levels of the system process it.

ISAMPrepare() responds in one of three ways:

1. It indicates that the upper levels of the system should process the query.
2. It indicates that the query will be processed by the backend and no result set will be returned (for example, an INSERT statement).
3. It indicates that the query will be processed by the backend and a result set will be returned (for example, a SELECT statement).

If a result set is returned, it is returned like any other table. The ISAMPrepare() call returns the name of this virtual table. The upper level of the system will process a "SELECT \* FROM <virtual-table>" to retrieve the values.

After ISAMPrepare() is called, ISAMExecute() is called to actually run the query. ISAMExecute() may be called several times after a single call to ISAMPrepare(). When the upper levels of the system no longer needs the query, ISAMFreeStatement() is called.

The SQL prepared by ISAMPrepare() can contain parameters, and these parameters are specified by ISAMParameter() calls that are made after the call to ISAMPrepare() but before the ISAMExecute().

The sample code shows two simple examples of passthrough SQL. The first processes the query "SQL" by returning a table with one column and two rows. The second processes the query "MessageBox(?,?)" by putting up a message box whose content and title are specified by parameters one and two (respectively).

## Sorting

Sorting is used to implement the ORDER BY and GROUP BY clauses of a SELECT statement. The Dr. DeeBee ODBC Driver Kit (Bronze Edition) sorts in two ways: pushdown sorts and upper-level sorts.

The pushdown sort is implemented by pushing the sorting functionality down to the ISAM layer. If the upper layers of the system determine that a pushdown sort can be used, it calls

ISAMSort(). If the ISAM layer can implement the sort, the ISAM layer returns a status of NO\_ISAM\_ERR and a pushdown sort is done.

If the upper layers of the system determine that a pushdown sort cannot be used, or if an ISAMSort() call of a pushdown sort returns a status other than NO\_ISAM\_ERR, an upper-level sort is done.

## Use of Indexes

To solve a query, the upper levels of the system opens a table and retrieves the records in the table. For each record retrieved, it tests the selection criteria specified and filters out the records that do not meet the criteria. This filtering can be pushed down to the ISAM layer. When this is done, the query would take less time to execute since fewer records would have to be processed by the upper levels.

This section describes this pushdown mechanism in general terms. For details, see the documentation of ISAM.H (in particular, ISAMOpenTable(), ISAMRestrict(), ISAMNextRecord(), and the declaration of COLUMNDEF).

When the upper levels of the system calls the ISAM layer to open a table, the ISAM layer returns the name and type of each of the columns. In addition to this information, the ISAM layer also returns the selectivity of each column. The selectivity is an indication of how selective the column is (for example, RECORD\_ID would be very selective, NAME would be moderately selective, ZIPCODE would not be very selective).

When the upper levels of the system receives a query, it walks the predicate to find the selective clauses (based on the selectivity specified when the table was opened). After the upper levels of the system open the table, but before any records are retrieved from the table, it passes these clauses down to the ISAM layer to tell the ISAM layer "you only need to return records that satisfy this criteria". It then starts to retrieve the records from the ISAM layer.

A typical ISAM implementation will use its indexes to decide which records to return.



- SQLConnect
- SQLDriverConnect
- SQLBrowseConnect
- SQLDisconnect
- SQLFreeConnect
- SQLFreeEnv

EXECUTE.C - ODBC entry points for SQL execution functions:

- SQLExecute
- SQLExecDirect
- SQLNativeSql
- SQLParamData
- SQLPutData
- SQLCancel

INFO.C - ODBC entry points for informational functions:

- SQLGetInfo
- SQLGetTypeInfo
- SQLGetFunctions

OPTIONS.C - ODBC entry points for connection and statement option functions:

- SQLSetConnectOption
- SQLSetStmtOption
- SQLGetConnectOption
- SQLGetStmtOption

PREPARE.C - ODBC entry points for query preparation setup functions:

- SQLAllocStmt
- SQLFreeStmt
- SQLPrepare
- SQLBindParameter
- SQLDescribeParam
- SQLParamOptions
- SQLNumParams
- SQLSetScrollOptions
- SQLSetCursorName
- SQLGetCursorName

RESULTS.C - ODBC entry points for result retrieval functions:

- SQLNumResultCols
- SQLDescribeCol
- SQLColAttributes
- SQLBindCol

SQLFetch  
SQLGetData  
SQLMoreResults  
SQLRowCount  
SQLSetPos  
SQLExtendedFetch  
SQLError

**SORT.C** - Sorting operations.

**TRANSACT.C** - ODBC entry points for transaction functions:  
SQLTransact

**DLL.C** - LibMain, etc.

**PARSE.C** - Parses SQL statement (as a text string) and creates a parse tree. This module implements a recursive descent parser for SQL.

**SEMANTIC.C** - Check a parse tree for semantic correctness. This module also contains a routine to display parse trees on the debug monitor.

**OPTIMIZE.C** - Query optimizer. Currently the only optimization done is to find restrictions on tables to cut down the search space.

**EVALUATE.C** - Evaluates SQL expressions and executes SQL statements.

**COLDEFS.C** - This module contains the definition of the columns returned by the virtual tables returned by the catalog functions and SQLGetTypeInfo(). It is unlikely you will ever need to modify these.

**SQLTYPE.C** - Definition of the SQL types supported by the driver. As installed, the Dr. DeeBee ODBC Driver Kit (Bronze Edition) supports SQL\_BIT, SQL\_CHAR, SQL\_DATE, and SQL\_DECIMAL. If you want your driver to support other types, modify the entries in the tables in this module. See the comments in SQLTYPE.C.

**BCD-** Operations to compare, negate, add, subtract, multiply, and divide Binary Coded Decimal values.

**ISAM.C** - Low level record access interface. The majority of the work needed to access data in a format other than dBASE is done in this module. The driver writer is expected to provide the implementation of the following routines (as documented in ISAM.H):

ISAMOpen

ISAMGetTableList  
ISAMGetNextTableName  
ISAMFreeTableList  
ISAMCreateTable  
ISAMAddColumn  
ISAMOpenTable  
ISAMRewind  
ISAMSort  
ISAMRestrict  
ISAMNextRecord  
ISAMGetData  
ISAMPutData  
ISAMInsertRecord  
ISAMUpdateRecord  
ISAMDeleteRecord  
ISAMGetBookmark  
ISAMPosition  
ISAMCloseTable  
ISAMDeleteTable  
ISAMPrepare  
ISAMParameter  
ISAMExecute  
ISAMFreeStatement  
ISAMClose  
ISAMGetErrorMessage  
ISAMGetColumnType  
ISAMCaseSensitive  
ISAMName  
ISAMVersion  
ISAMDriver  
ISAMMaxTableNameLength  
ISAMMaxColumnNameLength  
ISAMUser  
ISAMDatabase

UTIL.C - Utility functions:

CharToDouble - Converts strings to doubles  
CharToDate - Converts strings to dates  
CharToTime - Converts strings to times  
CharToTimestamp - Converts strings to timestamps  
DoubleToChar - Converts doubles to strings  
DateToChar - Converts dates to strings

TimeToChar - Converts times to strings  
TimestampToChar - Converts timestamps to strings  
ReturnString - Copies strings and their length  
ReturnStringD - Copies strings and their length  
ConvertSqlToC - Converts data from one type to another  
PatternMatch - Wildcard pattern matching  
TrueSize - Determines size of a string

DBASE.C - Routines to read and write records in dBASE files. This module is only used by ISAM.C.

TRACE.C - Tracing facilities. To enable tracing, set ISAM\_TRACE to TRUE in UTIL.H (not TRACE.H), delete all your .OBJ and .PCH files, and recompile. Setting ISAM\_TRACE to TRUE will trace every ISAM.C call. The output is sent to the debug window (DBWIN.EXE).

## Tips and Tricks

The following tips and tricks will make your development easier:

1. Before modifying ISAM.C or any other modules, recompile the system. This will allow you find problems in your development environment independently of bugs you might enter into the code.
2. Run SETUP.EXE in C:\DRDBDR\SETUP or C:\DRDBDR\SETUP32 to install the driver and create a datasource (MYSOURCE) using the driver. When creating the new datasource, be sure to use the "Database" edit control to specify the name of the directory that holds the .DBF files.

Once you do this, redirect the MYSOURCE datasource to use the DLL created by your compiler rather than the one in the Windows system. To do this...

...in Windows 3.x - Using a text editor, such as NOTEPAD, edit C:\WINDOWS\ODBC.INI. Look for the [MYSOURCE] section and change the <file> designated by the "Driver=<file>" keyword/value pair to specify the complete pathname of the .DLL file created by your compilation.

...in Windows NT/Win95 - Using the registry editor change the value of /HKEY\_CURRENT\_USER/Software/ODBC/ODBC.INI/MYSOURCE/Driver to specify the complete pathname of the .DLL file created by your compilation.

Once you do this, you will not have to reinstall after each compilation of your driver, since the MYSOURCE datasource will always be pointing at the DLL just created.

3. If you are using Microsoft Visual C/C++ 1.5, set "Calling Program" under Options | Debug to be C:\ODBCSDK\BIN\ODBCTEST.EXE. You can then set breakpoints in your driver, and run (DEBUG | GO).
4. If you are using Microsoft Visual C/C++ 2.0, set "Executable for Debug Session" under Project | Settings | Debug to be C:\ODBCSDK\BIN32\ODBCTE32.EXE. You can then set breakpoints in your driver, and run (DEBUG | GO).

## Appendix A: SQL Grammar supported

statement ::= CREATE create | DROP drop | SELECT select | INSERT insert |  
DELETE delete | UPDATE update | passthroughSQL

passthroughSQL ::= any statement supported by the backend

create ::= TABLE tablename ( createcols )

tablename ::= identifier

createcols ::= createcol , createcols | createcol

createcol ::= columnname datatype | columnname datatype ( integer ) |  
columnname datatype ( integer , integer )

datatype ::= identifier

columnname ::= identifier

drop ::= TABLE table

select ::= selectcols FROM tablelist where groupby having orderby

delete ::= FROM table where

insert ::= INTO table insertvals

update ::= table SET setlist where

setlist ::= set | setlist , set

set ::= column = NULL | column = expression

insertvals ::= ( columnlist ) VALUES ( valuelist ) | VALUES ( valuelist )

columnlist ::= column , columnlist | column

column ::= columnname

valuelist ::= NULL , valuelist | expression , valuelist | expression | NULL

selectcols ::= selectallcols \* | selectallcols selectlist

selectallcols ::= | ALL | DISTINCT

selectlist ::= expression , selectlist | expression

where ::= | WHERE boolean

having ::= | HAVING boolean

boolean ::= and | and OR boolean

and ::= not | not AND and

not ::= comparison | NOT comparison

comparison ::= ( boolean ) | colref IS NULL | colref IS NOT NULL |  
expression LIKE pattern | expression NOT LIKE pattern |  
expression IN ( valuelist ) | expression NOT IN ( valuelist ) |  
expression op expression

op ::= > | >= | < | <= | = | <>

pattern ::= string | ? | USER

expression ::= expression + times | expression - times | times

times ::= times \* neg | times / neg | neg

neg ::= term | + term | - term

term ::= ( expression ) | colref | simpleterm | aggterm

aggterm ::= COUNT ( \* ) | AVG ( expression ) | MAX ( expression ) |  
MIN ( expression ) | SUM ( expression )

simpleterm ::= string | realnumber | ? | USER | date | time | timestamp

groupby ::= | GROUP BY groupbyterms

groupbyterms ::= colref | colref , groupbyterms

orderby ::= | ORDER BY orderbyterms

orderbyterms ::= orderbyterm | orderbyterm , orderbyterms

orderbyterm ::= colref asc | integer asc

asc ::= | ASC | DESC

colref ::= aliasname . columnname | columnname

aliasname ::= identifier

tablelist ::= tableref, tablelist | tableref

tableref ::= table | table aliasname

table ::= tablename

identifier ::= an identifier (identifiers containing spaces must be enclosed in double quotes)

string ::= a string (enclosed in single quotes)

realnumber ::= a non-negative real number

integer ::= a non-negative integer

date ::= a date in ODBC escape clause format (for example, {d '1996-02-05'} or  
--(\*vendor(Microsoft),product(ODBC) d '1996-02-05' \*)--

time ::= a time in ODBC escape clause format (for example, {t '10:19:48'} or  
--(\*vendor(Microsoft),product(ODBC) t '10:19:48' \*)--

timestamp ::= a timestamp in ODBC escape clause format (for example,  
{ts '1996-02-05 10:19:48.529'} or  
--(\*vendor(Microsoft),product(ODBC) ts '1996-02-05 10:19:48.529" \*)--



## Appendix B: Migrating from the Standard Edition to the Bronze Edition

If you implemented your old driver by only modifying ISAM.C and DBASE.C, for the most part, the ISAM.C and DBASE.C you implemented for the Standard Edition can be used with only minor changes in the Bronze Edition. If you modified any other code you will have to re-implement those changes.

In ISAM.C there are eight changes you must make:

1. The new upper-level sorts require a bookmarking capability from ISAM.C. Bookmarking allows the upper levels of the system to request a 32-bit identifier for the current record of an open table and, at a later time, allows the upper levels of the system to instruct ISAM to position back to the record with a given identifier. Two functions have been added to ISAM.C for this purpose: ISAMBookmark() and ISAMPosition(). These are described in ISAM.H. The dBASE implementation of ISAM.C uses the record number as the bookmark.
2. The definition of the following ISAM functions have been changed:
  - ISAMOpen() - Now returns a status code. The handle is returned as an output parameter.
  - ISAMGetTableList() - Now returns a status code. The handle is returned as an output parameter.
  - ISAMGetNextTableName() - Now returns a status code instead of a boolean. If a table name was found, NO\_ISAM\_ERR is returned. If there are no more tables to return, ISAM\_EOF is returned.
  - ISAMFreeTableList() - now returns a error code.
  - ISAMOpenTable() - Now returns a status code. The handle is returned as an output parameter.
  - ISAMCreateTable() - Now returns a status code. The handle is returned as an output parameter.
3. ISAMSort() should return ISAM\_NOTSUPPORTED if the ISAM layer does not implement sorting. If any other status code (other than NO\_ISAM\_ERR) is returned, it will be returned to the ODBC application.
4. ISAMRestrict() should return ISAM\_NOTSUPPORTED if the ISAM layer does not implement the restriction. If any other status code (other than NO\_ISAM\_ERR) is returned, it will be returned to the ODBC application.
5. A new function, ISAMGetErrorMessage(), has been added. This routine returns a textual message describing the most recent error returned by the ISAM layer.

6. A new function, ISAMGetColumnType(), has been added. See the implementation of this routine in the sample code for more details.
7. The interface to ISAMAddColumn() has changed. Instead of a SQL\_\* type identifier being specified by fSqlType, an index into the SQLTypes[] array is specified by iSqlType. The SQL\_\* type identifier for this entry is SQLTypes[iSqlType].type.
8. The interface to ISAMSort() has also changed. It now allows for multiple columns to be specified as the sort criteria. The easiest way to migrate this function is to rename your old ISAMSort() to be OldSort() and implement ISAMSort() as follows:

```

        SWORD INTFUNC ISAMSort(LPISAMTABLEDEF lpISAMTableDef,
                               UWORD          count,
                               UWORD FAR *    icol,
                               BOOL FAR *     fDescending)
    {
        if (count > 1)
            return ISAM_NOTSUPPORTED;

        if (count == 0)
            return NO_ISAM_ERR;

        return OldSort(lpISAMTableDef, *icol, *fDescending);
    }

```

## Appendix C: Implementation Strategy for ISAM.C

It is suggested that you use the following code...test...code...test...code...test strategy when rewriting ISAM.C to connect to your database. The testing can be done using ODBC Test (from the ODBC SDK). The testing assumes that there is a table called EMP with a column called NAME and a column called SALARY:

### *Step 1: Connecting:*

Code: ISAMOpen()  
ISAMClose()  
ISAMGetErrorMessage()

Test: Connect | Full Connect  
Connect | Full Disconnect

### *Step 2: Get driver description:*

Code: ISAMCaseSensitive()  
ISAMMaxColumnNameLength()  
ISAMMaxTableNameLength()  
ISAMName()  
ISAMVersion()  
ISAMDriver()  
ISAMUser()  
ISAMDatabase()

Test: Connect | Full Connect  
Connect | SQLGetInfo(SQL\_IDENTIFIER\_CASE)  
Connect | SQLGetInfo(SQL\_MAX\_COLUMN\_NAME\_LEN)  
Connect | SQLGetInfo(SQL\_MAX\_TABLE\_NAME\_LEN)  
Connect | SQLGetInfo(SQL\_DBMS\_NAME)  
Connect | SQLGetInfo(SQL\_DBMS\_VER)  
Connect | SQLGetInfo(SQL\_DRIVER\_NAME)  
Connect | SQLGetInfo(SQL\_USER\_NAME)  
Connect | SQLGetInfo(SQL\_DATABASE\_NAME)  
Connect | Full Disconnect

### *Step 3: List of tables:*

Code: ISAMGetTableList()  
ISAMGetNextTableName()  
ISAMFreeTableList()

Test: Connect | Full Connect  
Catalog | SQLTables  
Results | GetDataAll  
Connect | Full Disconnect

*Step 4: List of columns for a table:*

Code: ISAMOpenTable()  
ISAMCloseTable()  
  
Test: Connect | Full Connect  
Catalog | SQLColumns(EMP)  
Results | GetDataAll  
Connect | Full Disconnect

*Step 5: Simple select:*

Code: ISAMRewind()  
ISAMNextRecord()  
ISAMGetData()  
  
Test: Connect | Full Connect  
Statement | SQLExecDirect("select \* from EMP")  
Results | GetDataAll  
Connect | Full Disconnect

*Step 6: Select with restriction:*

Code: ISAMRestrict()  
  
Test: Connect | Full Connect  
Statement | SQLExecDirect("select \* from EMP where NAME = 'FRED'")  
Results | GetDataAll  
Connect | Full Disconnect

Note: ISAMRestrict() will only be called if ISAMTableOpen() returns a non-zero value for fSelectivity for the NAME column.

*Step 7: Select with ORDER BY clause using an upper-level sort:*

Code: ISAMGetBookmark()  
ISAMPosition()

Test: Connect | Full Connect  
Statement | SQLExecDirect("select \* from EMP order by NAME")  
Results | GetDataAll  
Connect | Full Disconnect

Note: Suppress pushdown sorts by implementing ISAMSort() as:

```
if (count != 0)
    return ISAM_NOTSUPPORTED;
else
    return NO_ISAM_ERR;
```

*Step 8: Select with ORDER BY clause using a pushdown sort:*

Code: ISAMSort()

Test: Connect | Full Connect  
Statement | SQLExecDirect("select \* from EMP order by NAME")  
Results | GetDataAll  
Connect | Full Disconnect

*Step 9: Modifying data:*

Code: ISAMPutData()  
ISAMUpdateRecord()

Test: Connect | Full Connect  
Statement | SQLExecDirect("update EMP set SALARY = 20000")  
Connect | Full Disconnect

*Step 10: Inserting records:*

Code: ISAMInsertRecord()

Test: Connect | Full Connect  
Statement | SQLExecDirect(  
    "insert into EMP(NAME, SALARY) values ('CRAIG', 20000)")  
Connect | Full Disconnect

*Step 11: Deleting records:*

Code: ISAMDeleteRecord()

Test: Connect | Full Connect  
Statement | SQLExecDirect("delete from EMP where NAME ='CRAIG'")  
Connect | Full Disconnect

*Step 12: Creating a table:*

Code: ISAMCreateTable()  
ISAMAddColumn()

Test: Connect | Full Connect  
Statement | SQLExecDirect(  
"create table EMP2(NAME CHAR(10), SALARY DECIMAL(14,2))")  
Connect | Full Disconnect

*Step 13: Deleting a table:*

Code: ISAMDeleteTable()

Test: Connect | Full Connect  
Statement | SQLExecDirect("drop table EMP2")  
Connect | Full Disconnect

*Step 14: Passthrough SQL:*

Code: ISAMPrepare()  
ISAMParameter()  
ISAMExecute()  
ISAMFreeStatement()

Test: Connect | Full Connect  
Statement | SQLExecDirect(<<SQL to be passed to the back end >>)  
If the query returns a result set: Results | GetDataAll  
Connect | Full Disconnect

# Appendix D: SYWARE License Agreement

## Dr. DeeBee ODBC Driver Kit (Bronze Edition)

---

### SYWARE SOFTWARE LICENSE

1. GRANT OF LICENSE. SYWARE grants you the right to use the enclosed SYWARE software product in the manner provided below:

a. You may use one copy of the SYWARE software product identified above, which includes "online" or electronic documents (the "SOFTWARE") on a single computer. The SOFTWARE is in "use" on a computer when it is loaded into temporary memory (i.e., RAM) or installed into permanent memory (e.g., hard disk, CD-ROM, or other storage device) of that computer. However, installation on a network server for the sole purpose of internal distribution to one or more other computer(s) shall not constitute "use" for which a separate license is required, provided you have a separate license for each computer to which the SOFTWARE is distributed.

b. Solely with respect to the electronic documents, you may make unlimited number of copies (either in hardcopy or electronic form), provided that such copies shall be used only for internal purposes and are not republished or distributed beyond the user's premises.

2. If the SOFTWARE is an upgrade from a SYWARE product, you may now use that upgraded product only in accordance with this License.

3. COPYRIGHT. The SOFTWARE (including any images, "applets", photographs, animations, video, audio, music, and text incorporated in the SOFTWARE) is owned by SYWARE or its suppliers and is protected by United States copyright laws and international treaty provisions. Therefore you must treat the SOFTWARE like any other copyrighted material (e.g. a book or recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the software to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the printed materials accompanying the SOFTWARE.

4. OTHER RESTRICTIONS. You may not rent or lease the SOFTWARE. **You may not transfer the SOFTWARE.** You may not reverse engineer, decompile, or disassemble the SOFTWARE except to the extent such foregoing restriction is expressly prohibited by applicable law.

5. REDISTRIBUTABLE COMPONENTS.

a. **Sample Code.** Notwithstanding Section 1, SYWARE grants you the right to use and modify the source code version of the SOFTWARE located in \DRDBDR\SOURCE ("SAMPLE CODE") provided you comply with Section 5.c. You may not distribute SAMPLE CODE, or any modified version of SAMPLE CODE, in source code form.

b. **Redistributable Files.** Notwithstanding Section 1, SYWARE grants you a non-exclusive right to reproduce and distribute the object code version of those portions of the SAMPLE CODE, \DRDBDR\SETUP16\DRDBDR.DLL, and \DRDBDR\SETUP32\DRDBDR32.DLL (collectively called the REDISTRIBUTABLES), provided you comply with Section 5.c. and that the object code version of the SAMPLE CODE, or any modified version of the SAMPLE CODE, is an ODBC driver and is only capable of accessing a single DBMS or file format.

c. **Redistribution Requirements.** If you redistribute the REDISTRIBUTABLES, you must (i) not use SYWARE's name, logo, trademarks, to market your software application product; (ii) include a valid copyright notice on your software product; (iii) agree to indemnify, hold harmless, and defend SYWARE from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your software application product; and (iv) do not permit further distribution of the REDISTRIBUTABLES by your end-user.

The license in this section to distribute the REDISTRIBUTABLES is royalty-free.

6. EXPORT RESTRICTIONS. You agree that neither you nor your customers intends to or will, directly or indirectly, export or transmit (i) the SOFTWARE or related documentation and technical data or (ii) your software product as described in section 5 of this License (or any part thereof), or process, or service that is the direct product of the SOFTWARE, to any country to which such export or transmission is restricted by any applicable U.S. regulation or statute, without prior written consent, if required, of the Bureau of Export Administration of the U.S. Department of Commerce, or such other governmental entity as may have jurisdiction over such export or transmission.

---

### LIMITED WARRANTY

LIMITED WARRANTY. **Except with respect to the REDISTRIBUTABLES, which are provided "as-is", without warranty of any kind,** SYWARE warrants that (a) the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt, and (b) any hardware accompanying the SOFTWARE will be free from defects in materials and workmanship under normal use and service for a period of one (1) year from date of receipt. Any implied warranties on the SOFTWARE and hardware are limited to ninety (90) days and one (1) year, respectively. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

CUSTOMER REMEDIES. SYWARE's and its suppliers' entire liability and your exclusive remedy shall be, at SYWARE's option, either (a) return of the price paid, or (b) repair or replacement of the SOFTWARE or hardware that does not meet SYWARE's Limited Warranty and which is returned to SYWARE with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE or hardware has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE or hardware will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. **Outside the United States, neither these remedies nor any product support services offered by SYWARE are not available without proof of purchase from an authorized U.S. source.**

**NO OTHER WARRANTIES.** To the maximum extent permitted by applicable law, SYWARE and its suppliers disclaim all other warranties, either express or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, with regard to the SOFTWARE, the accompanying written materials, and any accompanying hardware.

This limited warranty gives you specific legal rights. You may have others, which vary from state/jurisdiction to state/jurisdiction.

**NO LIABILITY FOR CONSEQUENTIAL DAMAGES.** To the maximum extent permitted by applicable law, in no event shall SYWARE or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this SYWARE product, even if SYWARE has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

This Agreement is governed by the laws of the Commonwealth of Massachusetts.

For Licensee: \_\_\_\_\_  
X \_\_\_\_\_  
Name/Title: \_\_\_\_\_  
Date: \_\_\_\_\_

For SYWARE:  
X \_\_\_\_\_  
Seymour A. Danberg, President  
Date: \_\_\_\_\_