

ODBC Sample Application for Tandem NonStop[®] SQL/MX

NonStop Software Developers' Page

The Windows NT[®] Server program discussed in this NonStop Software Application TechNote is the first in a series of samples developed by Tandem's Advanced Technology Group (ATG) expressly for the NonStop Software Developers' Page. As Tandem's ATG continues working with the NonStop Software product line, you'll find other application notes, technical tips, and tutorials to help you with your NonStop Software development efforts. Add a hot-link to this page to your Web browser, and check back here often for updates.

This Application Note contains the following information:

- [Overview and Summary of Steps](#)
- [Running the Sample Application](#)
- [About ODBC](#)
- [Walk-through of the Source Code](#)
- [Summary of Key ODBC Calls Used in "odbcapi.c" Sample Application](#)
- [DDL Listing For the NonStop SQL/MX Database Sample Table](#)
- [Source Code Listing for ODBC Sample Application](#)

1.0 Overview and Summary of Steps

The sample program discussed in this Application Note provides a working example of a Windows NT Server console application that uses ODBC (open database connectivity) function calls to access and update a NonStop SQL/MX database table. (For more information about ODBC in general, see [About ODBC](#).) The program has been written in the C programming language using Microsoft Visual C++ 4.2, but you can compile

and link the program using other compilers as long as you modify the program to account for different filenames in your “include” statements and other such details.

This sample program is designed to help developers understand how to use ODBC API calls to access and perform transactions on a NonStop SQL/MX database, but the program will also work with other relational database management systems. Before running the program, you must create the four-column table in the NonStop SQL/MX database using the statements in the [DDL Listing For the NonStop SQL/MX Database Sample Table](#).

Here’s a summary of the steps you must take before you’ll be able to run the sample program:

1. Install and configure the NonStop SQL/MX database. See the *NonStop SQL/MX Installation Guide*, available on the NonStop SQL/MX CD, for more information. Be sure to consult the Readme file as well for any late breaking changes to the installation and configuration instructions.
2. Install the NonStop SQL/MX ODBC client software (the DLL (dynamic link library) which is included with the NonStop SQL/MX product), and configure the Windows NT Server ODBC Manager to use this driver. The sample program has been used with the Tandem NonStop SQL/MX database product and with the Microsoft SQL Server 6.5 database.
3. In targeting one database or another, developers must remember to configure the appropriate ODBC Driver (sqlodbc.dll or tdm_odbc.dll) in the Control Panel or through the NonStop ODBC/MX MS ODBC Administrator.
4. Create a NonStop SQL/MX table using the DDL listed later in this Note ([DDL Listing For the NonStop SQL/MX Database Sample Table](#)) The sample program interacts with this table. (The User ID and password were configured during NonStop SQL/MX installation, during ODBC installation (ODBC Init phase).
5. Download the **odbcapi.c** file from the NonStop Software Developers’ Page or copy from the NonStop Software SDK CD to your Windows NT Workstation or Windows NT Server system. Make sure all the “included” files are available to the compiler and located in the appropriate subdirectories.

For a step-by-step tutorial about the sample program, refer to the [Walk-through of the Source Code](#) as you read the code listing.

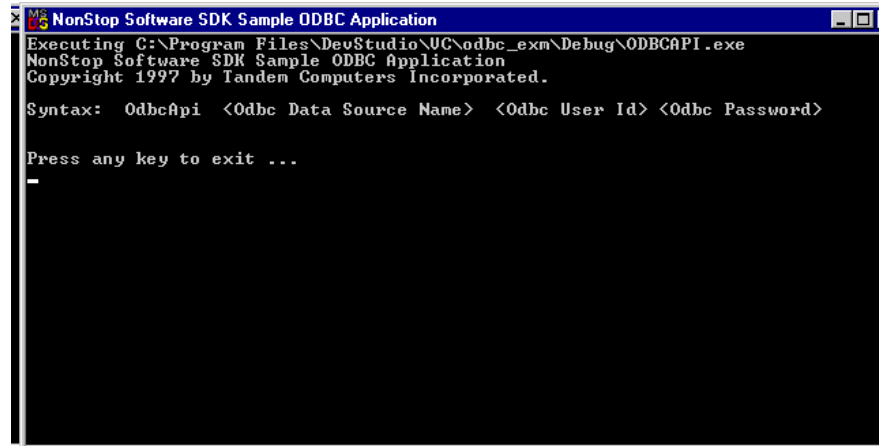
For a quick summary of the ODBC functions in the sample program, refer to the **Summary of the key ODBC calls** used in the sample program.

6. Compile and link the **odbcapi.c** file using Microsoft Visual C++ 4.2 (or higher). You’ll be prompted to create a default workspace; click Ok and proceed. When the build completes, you should have an executable named *odbcapi.exe* in your Visual C++ working directory.
7. You can run the program from within the Microsoft Visual C++ environment; using the Run command from the Windows NT Start menu; or by typing the filename at the Windows NT Server command prompt. See [Running the Sample Application](#) for additional details.

2.0 Running the Sample Application

The sample application was created using the Microsoft Visual C++ console application template as a starting point. The console dialog for the application is shown in Figure 1.

Figure 1. The Sample Application is a Windows NT Server console application



```
NonStop Software SDK Sample ODBC Application
Executing C:\Program Files\DevStudio\UC\odbc_exm\Debug\ODBCAPI.exe
NonStop Software SDK Sample ODBC Application
Copyright 1997 by Tandem Computers Incorporated.

Syntax: OdbcApi <Odbc Data Source Name> <Odbc User Id> <Odbc Password>

Press any key to exit ...
_
```

The NonStop SQL/MX database table and schema need to be created in advance, using the DDL (data definition language) listing, before you can run the program. (The program itself contains DML (data manipulation language)).

To execute the program, enter *odbcapi <database_name> <user name> <password>* as a single command line from the Windows NT Server (or Workstation) Command Prompt. Alternatively, you can enter this same information using the Run command from the Windows NT Server start menu, or by using the Execute command from within the Microsoft Visual C++ development environment (by specifying the command arguments in the command line options in Microsoft Visual development environment). If you don't enter all of this information, you'll be prompted as shown in Figure 1.

When the program executes, it will first connect to the specified database and display database information.

Next, the program deletes any records from the table which may exist from previous runs. After deleting the records, the program inserts records (whose values have been hard-coded into the application code) into the database and then displays these records.

Finally, the sample program updates the records in the database and exits the application.

3.0 About ODBC

Microsoft's ODBC (open database connectivity) is an industry-standard call-level interface which enables access to a wide range of SQL data sources. ODBC aligns with the X/Open and ISO SQL language standards and is supported by virtually all relational database product vendors, thus ensuring interoperability and portability of application source code. Application developers can use ODBC (rather than embedded SQL or the native CLI of a particular database product) in their application code to ensure the widest target market for their application. The ODBC CLI performs the following functions:

- Provides a locally bound call interface
- Instantiates a session with a named data source
- Defines local storage
- Sends SQL requests to that data source
- Retrieves results data produced by that data source

For more information about ODBC, see the Microsoft ODBC programmers reference books, available from Microsoft Press, or on the Microsoft Developers Network Web site (<http://www.microsoft.com/msdn/>). Both high-level and detailed information about using the SQL language in the context of application programming is contained in the Microsoft publications.

4.0 Walk-through of the Source Code

The sample application begins by accepting the data source name (schema name) and logon information (username and password) and providing these to the NonStop SQL/MX for validation. The ODBC API calls then begin with connection to the specified database and displaying the database information (database name and version) to the user. Next, the application deletes any records that may be present in the table, inserts the same record 10 times (which are hard-coded into the program), and then displays these records in the console window.

Note that by default, ODBC transactions are in auto commit mode, and this has not been disabled in this application by use of `SQLSetConnectOption`. In auto commit mode, every database operation is a transaction which is committed when performed. Auto commit mode is suitable for transactions that may consist of a single SQL statement (as in this program, for example).

Here's a more detailed overview of the key ODBC calls that the program uses. Refer to the `odbcapi.c` file as you step through the summary below. For more detailed information about any of the ODBC calls listed, see the Microsoft *ODBC Programmers Reference*, available from the Microsoft Developers Network (MSDN) library.

1. The application first allocates memory for an environment handle and initializes the ODBC call level interface by calling the `SQLAllocEnv` function. An application must call `SQLAllocEnv` prior to calling any other ODBC function. [Show me the code.](#)

2. The application then uses the *SQLAllocConnect* function to allocate memory for a connection handle (within the environment identified above) for connection between the ODBC Driver Manager and the sample application. [Show me the code.](#)
3. Next, the application makes a connection to the data source using the *SQLConnect* call. The *SQLConnect* loads the driver and establishes a connection to the data source, using the connection handle, the data source name, user name, and password. The connection handle references storage of all information about the connection, including status, transaction state, and error information. [Show me the code.](#)
4. Information about the database (ODBC driver, database name, and database version information) is obtained using the *SQLGetInfo* function. (*SQLGetInfo* returns general information about the driver and data source associated with the *hdbc*.)[Show me the code.](#)
5. Once the connection between the client and server have been established, the application now needs to send an SQL command to the server; it uses the *SQLAllocStmt* to allocate a statement handle for this purpose. *SQLAllocStmt* allocates memory for a statement handle and associates the statement handle with the connection specified by *hdbc*. An application must call *SQLAllocStmt* prior to submitting SQL statements. [Show me the code.](#)
6. The *SQLExecDirect* function takes the SQL string and sends the string to the server to perform the 'Insert', 'Update' and 'Select' SQL operations on the database. *SQLExecDirect* is the fastest way to submit an SQL statement for one-time execution. [Show me the code.](#)
7. To retrieve data after performing the 'Select' statement, the application first binds the columns to be retrieved to program variables using the *SQLBindCol* function. *SQLBindCol* assigns a value returned from a DBMS to a program variable in the client application., assigning the storage and data type for a column in a result set, including: 1) a storage buffer that will receive the contents of a column of data; 2) the length of the storage buffer; 3) a storage location that will receive the actual length of the column of data returned by the fetch operation; and 4) data type conversion. [Show me the code.](#)
8. To fetch the data the application then calls the *SQLFetch* function in a loop until all the rows have been fetched. The driver returns data for all columns that were bound to storage locations with *SQLBindCol*. [Show me the code.](#)
9. After the fetch, the statement handle is freed by calling the *SQLFreeStmt* to close the cursor. *SQLFreeStmt* stops processing associated with a specific *hstmt*, closes any open cursors associated with the *hstmt*, discards pending results, and, optionally, frees all resources associated with the statement handle. [Show me the code.](#)
10. All errors are handled through calls to the *SQLError* function after any ODBC function fails or returns a warning. *SQLError* returns error or status information.
11. Just before exiting the application the program frees the environment handle (and releases all memory associated with it) by using *SQLFreeEnv*. [Show me the code.](#)

5.0 Summary of Key ODBC Calls Used in “odbcapi.c” Sample Application

ODBC Call	Functional Description
SQLAllocEnv	Allocates the environment
SQLAllocConnect	Allocates a connection handle (for connection between ODBC Driver Manager and the application)
SQLConnect	Connects to the data source (requires connection handle, a data source name, user name, and password)
SQLGetInfo	Gets the database name and version number
SQLAllocStmt	Allocates a statement handle for sending SQL commands to the server
SQLExecDirect	Sends SQL strings (for INSERT, UPDATE, DELETE) to the server
SQLBindCol	Binds columns to be retrieved to program variables
SQLFetch	Fetches (as in scrollable cursors) the data
SQLFreeStmt	Releases the statement handle
SQLError	Handles errors after any ODBC function failures or warnings
SQLFreeEnv	Releases the environment handle

6.0 DDL Listing For the NonStop SQL/MX Database Sample Table

Use the DDL (data definition language) below to create a target database using NonStop SQL/MX. For more information about creating databases, tables, and using the Non-Stop SQL/MX database product, see the *NonStop SQL/MX Programming Manual for C* and the *NonStop SQL/MX Reference Manual*, available on Tandem’s Developers’ Page. From Tandem’s home page (<http://www.tandem.com>), select “Products,” then “Non-Stop Software,” then “Developers’ Page,” and then “Manuals.”

```

set schema tandem_system_nsk.odbc;
drop table odbcapi;
CREATE TABLE OdbcApi
(
  SYS_NAME  character(16)  not null nondroppable ,
  USE_DATE  character(20)  not null nondroppable ,
  DATA_001 character(08)  not null nondroppable ,
  DATA_002 character(08)  not null nondroppable
)
attributes audit;

```

7.0 Source Code Listing for ODBC Sample Application

```
// ===== //
// OdbcApi.c //
// Copyright 1997 by Tandem Computers Incorporated //
// All Rights Reserved //
// ===== //

// ===== //
// //
// THIS PROGRAM IS PROVIDED TO CUSTOMERS OF TANDEM COMPUTERS //
// INCORPORATED FOR EDUCATIONAL PURPOSES ONLY. //
// //
// THIS SAMPLE PROGRAM IS FOR ILLUSTRATION ONLY AND MAY NOT //
// BE SUITED FOR YOUR PARTICULAR PURPOSE. TANDEM COMPUTERS //
// INCORPORATED DOES NOT WARRANT, GUARANTEE, OR MAKE ANY //
// REPRESENTATION REGARDING THE USE OR THE RESULTS OF THE //
// USE OF THIS SAMPLE PROGRAM. //
// //
// SUPPORT FOR THIS PROGRAM IS NOT AVAILABLE. //
// //
// ===== //

// ===== //
// //
// DROP TABLE OdbcApi ; //
// //
// CREATE TABLE OdbcApi //
// ( //
// SYS_NAME character(16) not null , //
// USE_DATE character(20) not null , //
// DATA_001 character(08) not null , //
// DATA_002 character(08) not null //
// //
// //
// ) //
// ; //
// //
// ===== //

#define APPNAME "NonStop Software SDK Sample ODBC Application"
#define COPYRIGHT "Copyright 1997 by Tandem Computers Incorporated."
#define VERSION "OdbcApi v01.00"
#include <windows.h> // Windows Header Files
#include <conio.h> // Console I/O functions
#include <sql.h> // ODBC Constants and Functions
#include <sqlext.h> // Microsoft SQL Extensions
#include <stdio.h>

//----- //
// OdbcApi is a Windows(R) NT(R) Server Console application designed //
// to be compiled and built using Microsoft(R) Visual C++ 4.2 //
//----- //

// Global Application Variables
HDBC hDbc ;
// ODBC connection handle
```

```
HENV      hEnv ;
    // ODBC environment handle
RETCODE   OdbcRetcode ;
    // ODBC status code
HSTMT     hStmt ;
    // ODBC statement handle

char      szDbmsName[32] ;
SWORD     cbDbmsName ;
char      szDbmsVersion[16] ;
SWORD     cbDbmsVersion ;
char      szOdbcDataSource[32] ;
char      szOdbcPassword[16] ;
char      szOdbcUserId[32] ;
char      szSqlStmt[1024] ;
char      szSystemName[24] ;

/*=====*/
void ConnectAndInteract () ;
void ReportOdbcError (HENV,HDBC,HSTMT,HWND) ;
void TwoStepDropDead (PCHAR) ;
/*=====*/

void ConnectAndInteract ()
{
    char szAttribute_03[9] ;
    ULONG ulAttributeLen_03 ;
    char szAttribute_04[9] ;
    ULONG ulAttributeLen_04 ;
    USHORT i ;
    char szPrimaryKeySysName[17] ;
    ULONG ulPrimaryKeySysNameLen ;
    char szPrimaryKeyDateTime[21] ;
    ULONG ulPrimaryKeyDateTimeLen ;

/* ----- */
```



```
* ALLOCATE AN ODBC CONNECTION *
* ----- */
OdbcRetcode = SQLAllocConnect ( hEnv ,
                               &hDbc );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLAllocConnect() Failed!" );
```

[Return to walk-through.](#)

```
/* ----- *
* CONNECT TO ODBC_API DATA SOURCE *
* ----- */
OdbcRetcode = SQLConnect ( hDbc ,
                           szOdbcDataSource ,
                           SQL_NTS ,
                           szOdbcUserId ,
                           SQL_NTS ,
                           NULL ,
                           0 );

if ( ( SQL_SUCCESS != OdbcRetcode )
    &&( SQL_SUCCESS_WITH_INFO != OdbcRetcode ) )
    TwoStepDropDead ( "OdbcApi SQLConnect() Failed!" );
```

[Return to walk-through.](#)

```
/* ----- *
* GET NAME OF DATABASE MANAGEMENT SYSTEM *
* ----- */
OdbcRetcode = SQLGetInfo ( hDbc ,
                           SQL_DBMS_NAME ,
                           szDbmsName ,
                           (SDWORD)sizeof(szDbmsName) ,
                           &cbDbmsName );
_strupr ( szDbmsName );
OdbcRetcode = SQLGetInfo ( hDbc ,
                           SQL_DBMS_VER ,
                           szDbmsVersion ,
                           (SDWORD)sizeof(szDbmsVersion) ,
                           &cbDbmsVersion );
_strupr ( szDbmsVersion );
cprintf ( "DBMS is %s version %s\n" ,
          szDbmsName ,
          szDbmsVersion );
```

[Return to walk-through.](#)

```
/* ----- *
 * ALLOCATE AN ODBC STATEMENT HANDLE *
 * ----- */
OdbcRetcode = SQLAllocStmt ( hDbc ,
                             &hStmt );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLAllocStmt() Failed!" );
```

[Return to walk-through.](#)

```
/* ----- *
 * EXECUTE A DELETE STATEMENT *
 * ----- */
sprintf ( szSqlStmt
          ,
          "DELETE FROM schema.OdbcApi WHERE SYS_NAME = '%s'",
          szSystemName
          );
OdbcRetcode = SQLExecDirect ( hStmt ,
                             szSqlStmt ,
                             SQL_NTS );
if ( ( SQL_SUCCESS != OdbcRetcode )
     &&( SQL_NO_DATA_FOUND != OdbcRetcode ) )
    TwoStepDropDead ( "SQLExecDirect() DELETE Failed!" );
```

```
/* ----- *
 * EXECUTE AN INSERT STATEMENT *
 * ----- */
for (i= 1; i < 11 ; i++) {
Istrcpy ( szPrimaryKeyDateTime ,
          "12-07-1941 08:01:30" );
sprintf ( szSqlStmt
          ,
          "INSERT INTO schema.OdbcApi VALUES ('%s', '%s', 'AAAAAA00', 'BBBBBB00')",
          szSystemName
          ,
          szPrimaryKeyDateTime
          );
;
OdbcRetcode = SQLExecDirect ( hStmt ,
                             szSqlStmt ,
                             SQL_NTS );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLExecDirect() INSERT Failed!" );
}
```

```
/* ----- *
 * EXECUTE A SELECT STATEMENT *
 * ----- */
sprintf ( szSqlStmt
          ,
          "SELECT * FROM schema.OdbcApi WHERE SYS_NAME = '%s'",
          szSystemName
          );
OdbcRetcode = SQLExecDirect ( hStmt
                              ,
                              szSqlStmt
                              ,
                              SQL_NTS
                              );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLExecDirect() SELECT Failed!" );
```

[Return to walk-through.](#)

```
/* ----- *
 * BIND THE RESULT SET COLUMNS *
 * ----- */
OdbcRetcode = SQLBindCol ( hStmt
                           ,
                           1
                           ,
                           SQL_C_DEFAULT
                           ,
                           szPrimaryKeySysName
                           ,
                           sizeof ( szPrimaryKeySysName )
                           ,
                           &ulPrimaryKeySysNameLen
                           );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLBindCol() #1 Failed!" );
OdbcRetcode = SQLBindCol ( hStmt
                           ,
                           2
                           ,
                           SQL_C_DEFAULT
                           ,
                           szPrimaryKeyDateTime
                           ,
                           sizeof ( szPrimaryKeyDateTime )
                           ,
                           &ulPrimaryKeyDateTimeLen
                           );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLBindCol() #2 Failed!" );
OdbcRetcode = SQLBindCol ( hStmt
                           ,
                           3
                           ,
                           SQL_C_DEFAULT
                           ,
                           szAttribute_03
                           ,
                           sizeof ( szAttribute_03 )
                           ,
                           &ulAttributeLen_03
                           );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLBindCol() #3 Failed!" );
```

```
OdbcRetcode = SQLBindCol ( hStmt          ,
                          4              ,
                          SQL_C_DEFAULT ,
                          szAttribute_04 ,
                          sizeof ( szAttribute_04 ) ,
                          &ulAttributeLen_04 );
if ( SQL_SUCCESS!= OdbcRetcode )
    TwoStepDropDead ( "SQLBindCol() #4 Failed!" );
```

[Return to walk-through.](#)

```
/* ----- *
* FETCH THE RESULT SET ROWS *
* ----- */
while ( SQL_SUCCESS == OdbcRetcode )
{
    OdbcRetcode = SQLFetch ( hStmt );
    /*-----
    if ( SQL_SUCCESS == OdbcRetcode )
        cprintf ( "fetching %s : %s : %s : %s\n" ,
                  szPrimaryKeySysName   ,
                  szPrimaryKeyDateTime ,
                  szAttribute_03       ,
                  szAttribute_04       );
    //-----*/
    if ( ( SQL_SUCCESS != OdbcRetcode )
        &&( SQL_NO_DATA_FOUND != OdbcRetcode ) )
        TwoStepDropDead ( "SQLFetch() failed" );
}
```

[Return to walk-through.](#)

```
/* ----- *
* FREE THE ODBC STATEMENT HANDLE *
* ----- *
* Close its cursor so that the *
* statement handle can be reused *
* ----- */
OdbcRetcode = SQLFreeStmt ( hStmt   ,
                            SQL_CLOSE );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLFreeStmt() CLOSE Failed!" );
```

```
/* ----- *
 * SUBMIT 10 UPDATE STATEMENTS *
 * ----- */
for ( i = 1 ; i < 11 ; i++ )
{
    sprintf ( szSqlStmt
        ,
        "UPDATE schema.OdbcApi SET DATA_001 = 'AAAAAA%02hu', DATA_002 ='BBBBBB%02hu' WHERE SYS_NAME =
'%s'",
        i
        ,
        i
        ,
        szSystemName
    );
    OdbcRetcode = SQLExecDirect ( hStmt
        ,
        szSqlStmt ,
        SQL_NTS );
    if ( SQL_SUCCESS != OdbcRetcode )
        TwoStepDropDead ( "SQLExecDirect() UPDATE Failed!" );
}
/*-----
  end of "for ( i = 0 ; i < 11 ; i++ )"
  -----*/

/* ----- *
 * FREE THE ODBC STATEMENT HANDLE *
 * ----- */
OdbcRetcode = SQLFreeStmt ( hStmt
    ,
    SQL_DROP );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLFreeStmt() DROP Failed!" );

/* ----- *
 * DROP THE ODBC CONNECTION *
 * ----- */
OdbcRetcode = SQLDisconnect ( hDbc );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( "SQLDisconnect() Failed!" );

/* ----- *
```

```
* FREE THE ODBC CONNECTION *
* ----- */
OdbcRetcode = SQLFreeConnect ( hDbc );
if ( SQL_SUCCEEDED ( OdbcRetcode ) )
    TwoStepDropDead ( "SQLFreeConnect() Failed!" );
}

/*-----*/

void ReportOdbcError ( HENV henv ,
                     HDBC hdbc ,
                     HSTMT hStmt ,
                     HWND hWnd )
{
    UCHAR SqlState[6];
    SDWORD pfNativeError;
    UCHAR szRawErrorMsg[SQL_MAX_MESSAGE_LENGTH];
    UCHAR szErrorMsg[SQL_MAX_MESSAGE_LENGTH+22];
    SWORD pcbErrorMsg;
    CHAR *x,*y;
    short z;

    SQLError ( henv ,
              hdbc ,
              hStmt ,
              SqlState ,
              &pfNativeError ,
              szRawErrorMsg ,
              SQL_MAX_MESSAGE_LENGTH ,
              &pcbErrorMsg );

    for(z=0;z<sizeof(szErrorMsg);z++)
        szErrorMsg[z] = '\0';
    x = &szRawErrorMsg[0];
    y = &szErrorMsg[0];
    do {
        *y = *x++;
        if('.')==*y)

```

```
    {
        *y++=']';
        *y ='\n';
    }
    y+=1;
}
while (strlen(szRawErrorMsg)>(x-&szRawErrorMsg[0]));

Istrcat(szErrorMsg, "\nSQL_STATE=");
Istrcat(szErrorMsg, SqlState);
MessageBox ( hWnd
            ,
            szErrorMsg
            ,
            "ODBC Error Detail"
            ,
            MB_ICONINFORMATION );
}

/*=====*/

void TwoStepDropDead ( PCHAR szErrMsg )
{
    MessageBox ( NULL
                ,
                szErrMsg
                ,
                "ODBC Error"
                ,
                MB_ICONEXCLAMATION );
    ReportOdbcError ( hEnv
                    ,
                    hDbc
                    ,
                    hStmt
                    ,
                    NULL );
    ExitProcess ( 2 );
}

/*=====*/

main ( int argc
      ,
      char * argv[] )
{
```

```
USHORT m ;
```

```
SetConsoleTitle ( APPNAME ) ;
```

```
/*--< PRINT EXECUTABLE FILE NAME
```

```
>-----*/
```

```
cprintf ( "Executing %s\n" ,  
         argv[0]      ) ;
```

```
/*--< DISPLAY EXECUTABLE FILE APPLICATION & VERSION INFORMATION
```

```
>-----*/
```

```
cprintf ( APPNAME "\n" ) ;  
cprintf ( COPYRIGHT "\n" ) ;
```

```
/*--< GET NAME OF NT SYSTEM
```

```
>-----*
```

```
lstrcpy (szSystemName , "ECA1");  
cprintf ( "\nRunning on \\\\.%s\n" ,  
         szSystemName      ) ;
```

```
/*--< OBTAIN DATA SOURCE, USER ID, AND PASSWORD
```

```
>-----*/
```

```
if ( 3 == argc )
```

```
{
```

```
lstrcpy ( szOdbcDataSource ,  
         argv[1]      ) ;
```

```
lstrcpy ( szOdbcUserId ,  
         argv[2]      ) ;
```

```
lstrcpy ( szOdbcPassword ,  
         argv[3]      ) ;
```

```
cprintf ( "\n Data Source = %s\n User Id   = %s\n Password = %s\n",  
         szOdbcDataSource,  
         szOdbcUserId,  
         szOdbcPassword ) ;
```

```
}
```

```
else
```

```
{
```

```
cprintf ( "\nSyntax: OdbcApi <Odbc Data Source Name> <Odbc User Id> <Odbc Password>\n\n" ) ;
```



```
    cprintf(“ \nPress any key to exit ... \n” );
    getc ( stdin );
    ExitProcess ( 1 );
}

/*--< ALLOCATE AN ODBC ENVIRONMENT
>-----*/
OdbcRetcode = SQLAllocEnv ( &hEnv );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( “SQLAllocEnv() Failed!” );

/*--< DO IT
>-----*/
cprintf ( “\nExample showing Basic ODBC APIs \n” );

ConnectAndInteract ( m );

/*--< FREE THE ODBC ENVIRONMENT
>-----*/
OdbcRetcode = SQLFreeEnv ( hEnv );
if ( SQL_SUCCESS != OdbcRetcode )
    TwoStepDropDead ( “SQLFreeEnv() Failed!” );

/*--< AND NOW WE'RE DONE
>-----
-*/

cprintf(“ \nPress any key to exit ... \n” );
getc ( stdin );

return FALSE ;
}

// ===== //
//                               //
// THIS PROGRAM IS PROVIDED TO CUSTOMERS OF TANDEM COMPUTERS //
```

```
// INCORPORATED FOR EDUCATIONAL PURPOSES ONLY.      //  
//                                                    //  
// THIS SAMPLE PROGRAM IS FOR ILLUSTRATION ONLY AND MAY NOT //  
// BE SUITED FOR YOUR PARTICULAR PURPOSE. TANDEM COMPUTERS //  
// INCORPORATED DOES NOT WARRANT, GUARANTEE, OR MAKE ANY //  
// REPRESENTATION REGARDING THE USE OR THE RESULTS OF THE //  
// USE OF THIS SAMPLE PROGRAM.                        //  
//                                                    //  
// SUPPORT FOR THIS PROGRAM IS NOT AVAILABLE.        //  
//                                                    //  
// ===== //  
//
```

* * *

Trademark Notice

Tandem, Atalla, Every Second Counts, Himalaya, Integrity, iTP, NonStop, Object Relational Data Mining, Reliability No Limits, ServerNet, Tektonic, The Cluster Is The Computer, the Tandem logo, and other Tandem marks referenced herein are either registered trademarks or trademarks of Tandem Computers Incorporated in the United States and/or other countries.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. TUXEDO is a registered trademark of Novell, Inc., exclusively licensed to BEA Systems, Inc. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd. All other brand and product names are trademarks or registered trademarks of their respective companies.

Copyright (C) 1997, Tandem Computers Incorporated

Restricted to use in connection with Tandem products and systems. All rights reserved.

Warranty Disclaimers

THE INFORMATION CONTAINED HEREIN IS PROVIDED "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND, INCLUDING WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE,

OR NONINFRINGEMENT OF INTELLECTUAL PROPERTY. IN NO EVENT SHALL TANDEM BE LIABLE FOR ANY DAMAGES WHATSOEVER, INCLUDING, WITHOUT LIMITATION, SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA, OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE, OR OTHER TORTIOUS ACTION, ARISING OUT OF THE USE OF THE MATERIALS, EVEN IF TANDEM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT EXCLUSIONS OR LIMITATIONS ON LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

