Remote Data Access

Access your SQL databases from your HPe3000
Using

SqlLink3000

Oracle
PostgreSQL
SqlCommand
Access
MySQL
SqlLink3000 from GUI Innovations Limited builds on our base of TCP/IP and migration tools. SqlLink3000 has 2 constituents, a multi-threaded data server that runs on your PC server, and a library of callable routines that run on your HPe3000.

The calls and their functions are:-

- **SqlConnect** – Connect to the remote data server, and open your database.
- **SqlClose** – close your database and disconnect from your remote data server.
- **SqlExec** – execute a Sql command, i.e. Insert, Update, Delete
- **SqlRead** – initiate a read operation on a table, a view, or linked tables.
- **SqlGetNext** – retrieves the next record from your database
- **SqlGetStringField** – retrieve a string field from your record
- **SqlGetIntegerField** – retrieve a numeric field from your record
- **SqlGetDecimalField** – retrieve a decimal numeric field from your record.

When you issue a SqlGetNext call, the data is returned in a buffer, separated by the '|' character.

To best illustrate how to use the routines, we will examine the COBOL sample program that accompanies the product. The remote database is called GUI30ALES, and is a Microsoft Access database. This will just demonstrate the principle. SqlLink3000 will run against any databases for which you have drivers on your PC server.

GUI30ALES is our test HPe3000 database, which was migrated to Access using GUI3000/iX. It is a rough 'clone' of the NorthWind database supplied as a sample by Microsoft.

The COBOL sample program, simply reads down the 'products' table, and displays information. In addition, it inserts a record, updates the values, and then deletes the record. There are options to run a formatted report, a 'query' like report, and a 'complex' query. Standard SQL statements are used in SqlLink3000. They are sent to the remote data server, which processes them, and returns the results.
A Working Example

On the next several pages, we ‘dissect’ the test program to illustrate what is happening.

1) The usual COBOL program header information.

$CONTROL USLINIT
IDENTIFICATION DIVISION.
PROGRAM-ID. SQLTEST.
AUTHOR. Pete Vickers.
DATE-WRITTEN. MON, MAR 18, 2002.
DATE-COMPILED.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING- STORAGE SECTION.

2) Our area for displaying our returned values.

01 DISP-HEADINGS.
  03 FILLER PIC X(30) VALUE "Product".
  03 FILLER PIC X(20) VALUE "Price".
  03 FILLER PIC X(20) VALUE "In Stock".
01 DISP-UNDERLINE.
  03 FILLER PIC X(30) VALUE "========".
  03 FILLER PIC X(20) VALUE "=====".
  03 FILLER PIC X(20) VALUE "========".

3) Our sample Insert statement

01 INSERT-STATEMENT.
  03 FILLER PIC X(45) VALUE "INSERT INTO PRODUCTS VALUES (1234,'TEST',26,5".
  03 FILLER PIC X(27) VALUE ",'1 Box',12.34,1,2,3)".

4) Our sample Update statement

01 UPDATE-STATEMENT.
  03 FILLER PIC X(35) VALUE
"UPDATE PRODUCTS SET Qty_Per_Unit = ".
03 FILLER PIC X(40) VALUE
"2 Boxes' WHERE Product_ID = 1234".

5) Our sample Delete statement

01 DELETE-STATEMENT.
  03 FILLER PIC X(40) VALUE
  "DELETE FROM PRODUCTS WHERE Product_ID =".
  03 FILLER PIC X(10) VALUE
  "1234".

6) Display area for the full record

01 SERVER-RECORD.
  03 SV-PRODUCT PIC X(4).
  03 SV-DESCRIPTION PIC X(60).
  03 SV-SUPPLIER PIC X(8).
  03 SV-CATEGORY PIC X(8).
  03 SV-QTY-PER-UNIT PIC X(30).
  03 SV-UNIT-PRICE PIC X(8).
  03 SV-STOCK PIC X(8).
  03 SV-ON-ORDER PIC X(8).
  03 SV-REORDER PIC X(8).

7) The first thing you need to change. Supply the IP address of your PC server where SqlLink3000 is running.

01 SERVER-IP PIC X(20) VALUE "192.192.192.46".

8) The port number of SqlLink3000. The routine connects to this port, and then transparently re-connects to a system assigned port, to enable multi-threading (Can be overridden - see later).

01 SERVER-PORT PIC S9(9) COMP VALUE 9875.

9) The connect string. Commented out is a SQL server connect string. SqlLink3000 uses Microsoft ActiveX Data Objects (ADO) to access the remote database. If this is not on your system, it can be downloaded from http://www.microsoft.com/data/download.htm or we include it in the setup program.
We have set it up to connect to the database supplied with the samples.

* 01 SERVER-CONNECT.
* 03 FILLER PIC X(40) VALUE
* "driver={SQL Server};server=pentium450;ui".
* 03 FILLER PIC X(60) VALUE
* "d=sa;pwd=;database=northwind".

01 SERVER-CONNECT.
  03 FILLER PIC X(45) VALUE
  "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=".
  03 FILLER PIC X(40) VALUE
  "C:\program files\SqlLink3000\db\GUISALES.MDB".
10) The status area. The 'server-status' will return 0 for a successful call, 11 for 'end of file' (as in Image), and a negative number for an error. Server-error will contain a descriptive error message. If it is an ADO error, it will contain the standard ADO error messages.

01 SERVER-STATUS-AREA.
   03 SERVER-STATUS    PIC S9(18) COMP.
   03 SERVER-_SOCKET   PIC S9(9) COMP.
   03 SERVER-ERROR     PIC X(256).

11) The buffer for returning the data. The data is returned separated by the '|' character (Can be overridden - see later)

01 SERVER-BUFFER        PIC X(4096).

12) The number of records (if known) found by the SqlRead statement. This is dependent on the database returning the number of records.

01 SERVER-RECORDS       PIC S9(9) COMP.

13) The number of fields returned.

01 SERVER-NFIELDS       PIC S9(4) COMP.

14) The area for converting a string used by SqlGetStringField

01 SERVER-STRING        PIC X(512).

15) The field name for converting. This is CASE SENSITIVE!!

01 SERVER-FNAME         PIC X(48).

16) The area for converting a decimal field, used by SqlGetDecimalField

01 SERVER-DECIMAL       PIC S9(9)V9(9) COMP.

17) The area for converting a numeric field, used by SqlGetIntegerField

01 SERVER-INTEGER       PIC S9(18) COMP.

18) Where it begins to happen

PROCEDURE DIVISION.
AA-DRIVER SECTION.
A1AA.

   PERFORM BA-CONNECT.

   IF NOT COMPLEX-QUERY
      PERFORM BB-READ-RECORDS
   ELSE
      PERFORM BC-COMPLEX-QUERY
PERFORM BD-TERMINATE.
GOBACK.
Z9AA.
EXIT.
BA-CONNECT SECTION.
  A1BA.
  *
  * Show debug information if required - Answer Y or y
  *
  DISPLAY "Do you wish to display debug information? "
     WITH NO ADVANCING.
  ACCEPT IN-DEBUG.
  DISPLAY "Option 1) Query Like Report".
  DISPLAY "  2) Formatted Report".
  DISPLAY "  3) Grouped query".
  DISPLAY "Enter your option... " WITH NO ADVANCING.
  ACCEPT QUERY-OPTION.
  IF NOT VALID-QUERY-OPTION
     MOVE 1
     TO QUERY-OPTION.
  *
  * Connect to SQLINK3000 using the server IP address, the port number
  * and the connection string.
  *
  CALL "SQLConnect" USING SERVER-IP,
     SERVER-PORT,
     SERVER-CONNECT,
     SERVER-STATUS-AREA.
  IF DISPLAY-DEBUG
     DISPLAY "Server status is ", SERVER-STATUS
  END-IF.
  IF SERVER-STATUS <> 0 THEN
     DISPLAY "Error in SqlConnect ", SERVER-ERROR
     DISPLAY "Please ensure that SQLINK3000 is running "
     "on the PC Server"
     GOBACK.
  *
  * Insert our test record
  *
  PERFORM CA-INSERT-A-RECORD.
  *
  * Update it to prove the update works
  *
  PERFORM CB-UPDATE-A-RECORD.
Z9BA.
  EXIT.
BB-READ-RECORDS SECTION.
  A1BB.
  *
* Set up our standard SQL select statement
*
  MOVE "SELECT * FROM products" to SERVER-BUFFER.

  IF DISPLAY-DEBUG
    DISPLAY "Calling SqlRead"
  END-IF.

  MOVE 0 TO RECORD-COUNT.
*
  * Call SqlRead sending the Sql statement
  * Returns the number of records found, and the number of fields
  * Most database providers return -1 in the number of records.
*
  CALL "SqlRead" USING SERVER-STATUS-AREA
    SERVER-BUFFER,
    SERVER-RECORDS,
    SERVER-NFIELDS.

  IF SERVER-STATUS <> 0 THEN
    DISPLAY "Error in Read ", SERVER-ERROR
    GOBACK.
  END-IF.
*
  * Read all our records
  *
  PERFORM CC-READ-RECORDS
    UNTIL END-OF-FILE.

Z9BB.
  EXIT.
BC-COMPLEX-QUERY SECTION.
A1BC.
*
* Set up our standard SQL select statement
*
  MOVE COMPLEX-SELECT
  TO   SERVER-BUFFER.

  IF DISPLAY-DEBUG
    DISPLAY "Calling SqlRead"
  END-IF.

  MOVE 0 TO RECORD-COUNT.
*
  * Call SqlRead sending the Sql statement
  * Returns the number of records found, and the number of fields
  * Most database providers return -1 in the number of records.
*
CALL "SqlRead" USING SERVER-STATUS-AREA
SERVER-BUFFER,
SERVER-RECORDS,
SERVER-NFIELDS.

IF SERVER-STATUS <> 0 THEN
  DISPLAY "Error in Read ", SERVER-ERROR
  GOBACK.

IF DISPLAY-DEBUG
  DISPLAY "No of Records ", SERVER-RECORDS
  DISPLAY "No of Fields ", SERVER-NFIELDS
END-IF.

* Read all our records
* PERFORM CE-READ-RECORDS
  UNTIL END-OF-FILE.

Z9BB.
EXIT.
BD-TERMINATE SECTION.
A1BD.

  MOVE RECORD-COUNT
  TO RECORD-DISP.

  DISPLAY ": ".
  DISPLAY RECORD-DISP, " records reported".

* Delete the test record we created
*

  PERFORM CD-DELETE-A-RECORD.

  IF DISPLAY-DEBUG
    DISPLAY "Calling SQLClose"
  END-IF.

* Close the database, and our connection
*

  CALL "SqlClose" USING SERVER-STATUS-AREA.

Z9BD.
EXIT.

CA-INSERT-A-RECORD SECTION.
A1CA.

  IF DISPLAY-DEBUG
    DISPLAY "Inserting a record"
  END-IF.

  MOVE INSERT-STATEMENT TO SERVER-BUFFER.

* Using our pre-prepared insert statement, insert a record
* Server-records will tell us how many records were inserted
* In this case, if more than 1, we are in trouble!
* CALL "SqlExec" USING SERVER-STATUS-AREA,
  SERVER-BUFFER,
  SERVER-RECORDS.

IF SERVER-STATUS <> 0 THEN
  DISPLAY "Error in Insert ", SERVER-ERROR
  GOBACK.
IF DISPLAY-DEBUG
  DISPLAY "Records inserted = ", SERVER-RECORDS
END-IF.

Z9CA.
EXIT.

CB-UPDATE-A-RECORD SECTION.
A1CB.

IF DISPLAY-DEBUG
  DISPLAY "Updating a record"
END-IF.

MOVE UPDATE-STATEMENT TO SERVER-BUFFER.
* Using our pre-prepared update statement, update the
* field in the record we just created.
* This will prove when we read, that we have indeed
* updated the record
* CALL "SqlExec" USING SERVER-STATUS-AREA,
  SERVER-BUFFER,
  SERVER-RECORDS.

IF SERVER-STATUS <> 0 THEN
  DISPLAY "Error in UPDATE ", SERVER-ERROR
  GOBACK.
IF DISPLAY-DEBUG
  DISPLAY "Records updated = ", SERVER-RECORDS
END-IF.

Z9CB.
EXIT.

CC-READ-RECORDS SECTION.
A1CC.
* Call SqlGetNext to retrieve the next record.
* Data is returned in Server-buffer, separated by "|
* CALL "SqlGetNext" USING SERVER-STATUS-AREA
  SERVER-BUFFER.

* Are we at end of file?
IF SERVER-STATUS = 11 THEN
   MOVE 1 TO END-OF-FILE-FLAG
   GO TO Z9CC.

* If displaying debug messages, unload the buffer into
* our Server-record area using unstring.
* We are treating all the data as text!
*

IF DISPLAY-DEBUG
   DISPLAY "Status is ", SERVER-STATUS
   DISPLAY "Buffer ", SERVER-BUFFER(1:80)
END-IF.

ADD 1 TO RECORD-COUNT.

IF SIMPLE-REPORT
   UNSTRING SERVER-BUFFER
      DELIMITED BY "|
      INTO
         SV-PRODUCT,
         SV-DESCRIPTION,
         SV-SUPPLIER,
         SV-CATEGORY,
         SV-QTY-PER-UNIT,
         SV-UNIT-PRICE,
         SV-STOCK,
         SV-ON-ORDER,
         SV-REORDER
   DISPLAY "   
   DISPLAY "Product: ", SV-PRODUCT
   DISPLAY "Name: ", SV-DESCRIPTION
   DISPLAY "Supplier: ", SV-SUPPLIER
   DISPLAY "Category: ", SV-CATEGORY
   DISPLAY "Unit Qty: ", SV-QTY-PER-UNIT
   DISPLAY "Price: ", SV-UNIT-PRICE
   DISPLAY "Stock: ", SV-STOCK
   DISPLAY "On Order: ", SV-ON-ORDER
   DISPLAY "ReOrder: ", SV-REORDER
   GO TO Z9CC
END-IF.

IF DISPLAY-DEBUG
   DISPLAY "Get Product Dese"
END-IF.

IF RECORD-COUNT = 1
   DISPLAY DISP-HEADINGS
   DISPLAY DISP-UNDERLINE
END-IF.

* Get the value of Product_Name.
* This does no physical I-O, as the record has been retrieved
* and is sitting in memory.
*
   MOVE "Product_Name" TO SERVER-FNAME.
CALL "SqlGetStringField" USING SERVER-STATUS-AREA
   SERVER-FNAME
   SERVER-STRING.

*  
* If we get a negative status, the field we are retrieving
* is not a character field
*  
* IF SERVER-STATUS <> 0 THEN
   MOVE "Not a string field"
   TO   DISP-PRODUCT
ELSE
   MOVE SERVER-STRING(1:30)
   TO   DISP-PRODUCT
END-IF.

IF DISPLAY-DEBUG
   DISPLAY "Get Product price"
END-IF.

*  
* Get the value of Unit_Price
* This is held on the database as a 'money' type field
* SQLLINK3000 will take care of the decimal alignment
*  

   MOVE "Unit_Price" TO SERVER-FNAME.

   CALL "SqlGetDecimalField" USING SERVER-STATUS-AREA
      SERVER-FNAME
      SERVER-DECIMAL.

*  
* If we get a negative status, the field we are retrieving
* is not a 'decimal' type field
*  
* IF SERVER-STATUS <> 0 THEN
   MOVE 0
   TO   DISP-PRICE
ELSE
   MOVE SERVER-DECIMAL
   TO   DISP-PRICE
END-IF.

IF DISPLAY-DEBUG
   DISPLAY "Get Product Qty"
END-IF.

*  
* Get the value of Units_In_Stock
*  
   MOVE "Units_In_Stock" TO SERVER-FNAME.

   CALL "SqlGetIntegerField" USING SERVER-STATUS-AREA
      SERVER-FNAME
      SERVER-INTEGER.

*  
* If we get a negative status, the field we are retrieving
* is not an 'integer' type field
*  
* IF SERVER-STATUS <> 0 THEN
MOVE 0 TO DISP-QTY
ELSE
  MOVE SERVER-INTEGER TO DISP-QTY
END-IF.

DISPLAY DISP-LINE.

Z9CC.
EXIT.

CD-DELETE-A-RECORD SECTION.
A1CD.

  IF DISPLAY-DEBUG
    DISPLAY "Deleting a record"
  END-IF:

  * Using our pre-prepared delete statement
  * delete the record we created earlier
  * If server-records is not 1, then we are in trouble!
  *
  MOVE DELETE-STATEMENT TO SERVER-BUFFER.

  CALL "SqlExec" USING SERVER-STATUS-AREA,
     SERVER-BUFFER,
     SERVER-RECORDS.

  IF SERVER-STATUS <> 0 THEN
    DISPLAY "Error in Delete ", SERVER-ERROR
    GOBACK.
  END-IF.

  IF DISPLAY-DEBUG
    DISPLAY "Records deleted = ", SERVER-RECORDS
  END-IF.

Z9CD.
EXIT.

CE-READ-RECORDS SECTION.
A1CE.

  * Call SqlGetNext to retrieve the next record.
  * Data is returned in Server-buffer, separated by "|
  *
  CALL "SqlGetNext" USING SERVER-STATUS-AREA
     SERVER-BUFFER.

  * Are we at end of file?
  *
  IF SERVER-STATUS = 11 THEN
    MOVE 1 TO END-OF-FILE-FLAG
    GO TO Z9CE.

  * If displaying debug messages, unload the buffer into
  * our Server-record area using unstring.
* We are treating all the data as text!

*  

IF DISPLAY-DEBUG
  DISPLAY "Status is ", SERVER-STATUS
  DISPLAY "Buffer ", SERVER-BUFFER(1:80)
END-IF.

ADD 1 TO RECORD-COUNT.

UNSTRING SERVER-BUFFER
  DELIMITED BY "|
  INTO
  CP-COMPANY,
  CP-DATE,
  CP-VALUE,
  CP-ORDERS.

DISPLAY " ".
DISPLAY "Company: ", CP-COMPANY.
DISPLAY "Date: ", CP-DATE.
DISPLAY "Total Value: ", CP-VALUE.
DISPLAY "No Of Orders: ", CP-ORDERS.
DISPLAY " ".
Z9CE.
EXIT.

That is all there is to accessing a remote SQL Database. In less than 600 lines, we have connected to a remote database, inserted updated and deleted records, and run 2 queries, displaying the results.

Multiple Connections.

SqlLink3000 supports connections to multiple databases at the same time, and multiple connections to the same database at the same time. To achieve this, simply declare another status area, and issue a SqlConnect. Think of the status area in the same terms as the database name in a program, where the first 2 characters are the unique id of the database.

Alternately, store away the SERVER-_SOCKET, after the connect, and treat this as your unique id. Please ask us if you need a further sample of this.
The results from running the above test program are:-

GUI Innovations SqlLink3000
===========================
Please enter the IP address of the 'Server' where SqlLink3000 is running..213.122.63.64
Option 1) Query Like Report
      2) Formatted Report
      3) Grouped query
Enter your option... 2

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>In Stock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice Mutton</td>
<td>39.00</td>
<td>0</td>
</tr>
<tr>
<td>Aniseed Syrup</td>
<td>10.00</td>
<td>13</td>
</tr>
<tr>
<td>Boston Crab Meat</td>
<td>18.40</td>
<td>123</td>
</tr>
<tr>
<td>Camembert Pierrot</td>
<td>34.00</td>
<td>19</td>
</tr>
<tr>
<td>Carnarvon Tigers</td>
<td>62.50</td>
<td>42</td>
</tr>
<tr>
<td>Vegie-spread</td>
<td>43.90</td>
<td>24</td>
</tr>
<tr>
<td>Wimmers gute Semmelknodel</td>
<td>33.25</td>
<td>22</td>
</tr>
<tr>
<td>Zaanse koeken</td>
<td>9.50</td>
<td>36</td>
</tr>
</tbody>
</table>

Start time: 12:01:42:40
End time  : 12:01:43:20

78 records reported
Elapsed time of .8 seconds to retrieve and display the records.
The 'grouped query' returns:-

GUI Innovations SqlLink3000
===========================
Please enter the IP address of the 'Server' where SqlLink3000 is running..213.122.63.64
Option 1) Query Like Report
      2) Formatted Report
      3) Grouped query
Enter your option... 3

... Company: Vins et alcools Chevalier
   Date: 01/09/1994
   Total Value: 440
   No Of Orders: 3

Company: Hanari Carnes
   Date: 24/08/1994
   Total Value: 1444.8
   No Of Orders: 3

Start time: 12:03:50:30
End time  : 12:04:00:50

823 records reported
SqlQuery allows you to access your databases from the HPe3000 without writing code. Simply connect to the database and issue SQL statements.

```
start time: 19:03:59.70
end time  : 19:04:01.80
```

```
Connection Problems
```

You may encounter problems where 2 or more clients attempt to connect to the server at the same time. If SqlLink detects this, it will pause and try again. By default it will try 10 times, pausing for 3 seconds each time.

You can override these using system variables, e.g.
SETVAR SQL_TRIES 5
SETVAR SQL_PAUSE 1

Will make SqlLink try to connect a maximum of 5 times, pausing for 1 second between each try.

Delimiting Data

The standard data-delimiting character for SqlLink3000 is the "|" character. This can be changed by running SqlLink3000 with a -D option, e.g.
C:\SqlLink\SqlLink3000.exe -D #
C:\SqlLink\SqlLink3000.exe -D 255

These will delimit the data by "#" and the ascii value of 255 respectively. SqlQuery will also know if the delimiter varies from the standard.

Translating Data

Providing your system has Adager Level1, SqlLink will provide CCDC to BIG5 translation. This is achieved by doing 2 things:-
 SETVAR SQL_BIG5
 Will instruct SqlLink to call the translator. In addition, you need to specify an Adager XL on your run statement. To make SqlQuery do BIG5 translation, you would do...

 SETVAR SQL_BIG5 1
 RUN SQLQUERY;XL="ADAGER67.PUB.REGO"

The system will then internally call the translator - this should be transparent to your own program

SqlLink3000 – The remote PC server component.

SqlLink3000 runs on the PC. It can be started with a -L option (C:\SqlLink\SqlLink3000.exe -L), which will log what it is doing. This will inevitably slow the program down. The program will 'sit' in the system tray. Double-click the icon to restore the screen. Right-clicking the icon will allow the options of Restore and End.
SqlLink3000 shows the number of connections it is currently serving, and clients connecting and disconnecting. With logging turned on, you get details of all transactions as they are happening. If you cannot read the full line of the log, simply double-click on the line, and it will display in full.

SqlLink3000 is multi-threaded. Each requestor talks to its own socket. Hovering over the icon in the system tray, will show you the number of connections.

The default listener port for SqlLink3000 is 9875. This can be overridden by a run time parameter, e.g.

C:\SqlLink\SqlLink3000.exe -P1234
will make SqlLink3000 listen on port 1234. If you change the port on SqlLink3000, SqlQuery needs to know, so simply RUN SQLQUERY;PARM=1234 in the above case.

According to the ADO specification, there are 4 types of SQL 'cursors'.

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adOpenDynamic</td>
<td>2</td>
<td>Uses a dynamic cursor. Additions, changes, and deletions by other users are visible, and all types of movement through the Recordset are allowed, except for bookmarks, if the provider doesn't support them.</td>
</tr>
<tr>
<td>adOpenForwardOnly</td>
<td>0</td>
<td>Default. Uses a forward-only cursor. Identical to a static cursor, except that you can only scroll forward through records. This improves performance when you need to make only one pass through a Recordset.</td>
</tr>
<tr>
<td>adOpenKeyset</td>
<td>1</td>
<td>Uses a keyset cursor. Like a dynamic cursor, except that you can't see records that other users add, although records that other users delete are inaccessible from your Recordset. Data changes by other users are still visible.</td>
</tr>
<tr>
<td>adOpenStatic</td>
<td>3</td>
<td>Uses a static cursor. A static copy of a set of records that you can use to find data or generate reports. Additions, changes, or deletions by other users are not visible.</td>
</tr>
</tbody>
</table>

SqlLink3000 by default will use adOpenStatic, which should return a valid record count. If you wish to override this, simply start of SqlLink3000 with a -C option, e.g. C:\SqlLink\SqlLink3000.exe -C2 Will use adOpenDynamic. This gives you complete flexibility.

Server Installation

There are 5 files to upload to your HPe3000. They are in the \program files\SqlLink3000\hp3000\ folder on the PC where you installed SqlLink3000.

- SQLTESCB.wrq - The source for the COBOL test program
- SQLTEST.wrq - The compiled version of the above
- SQLTRNOB.wrq - The NMOBJ file of the transaction server to allow you to link into your own programs
- GUISQLXL.wrq - The XL file built from the above containing the calls.
- BUILDSQL.wrq - A command file for compiling and linking SQLTESCB
- SqlQuery.wrq - A 'query' look-alike to access your SQL databases without coding!
When compiling and linking programs using SQLTRNOB, you need to add the statement 'XL=SOCKETXL.NET.SYS' to link in the Sockets library. You can either do this at 'link' time, e.g.

```
link from=sqltesob,sqltrnob;to=sqltest;XL="SOCKETXL.NET.SYS"
```

Or at run time, e.g.

```
RUN TESTPROG;XL="GUI\$QLXL.PUB\$GUI3KIX,SOCKETXL.NET.SYS"
```

Transfer the files to your HPe3000 using Reflection, and the 'labels' format.

Alternately, use FTP and DOS. This is a transcript of an FTP session with your input underlined:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\>ftp invent3k.external.hp.com
Connected to INVENT3K.EXTERNAL.HP.COM.
220 HP ARPA FTP Server [A0011B03] (C) Hewlett-Packard Co. 2000 [PASV SUPPORT]
User (INVENT3K.EXTERNAL.HP.COM:(none)): mgr.gui3kix
331 Password required for MGR.GUI3KIX. Syntax: [,]acctpass
Password:________
230 User logged on
ftp> binary
200 Type set to I.
ftp> put c:\hp_rda\sqlstore
sqlstore;code=store;rec=128,1,f,binary;disc=4000
200 PORT command ok.
150 File: sqlstore;code=store;rec=128,1,f,binary;disc=4000 opened;
data connecti
on will be opened
226 Transfer complete.
ftp: 592128 bytes sent in 66.84Seconds 8.86Kbytes/sec.
ftp> quit
221 Server is closing command connection
C:\>exit
```

Then log on to the account where you uploaded the file on the hpe3000 and:

```
:FILE SQL=SQLSTORE;DEV=DISC
:RESTORE *SQL;@.@.@;SHOW;LOCAL
```

>> TURBO-STORE/RESTORE VERSION C.70.10 B5152AA <<
SqlLink3000 has been tested against SqlServer, Oracle, Access, PostgreSQL and MySql, but will run against any database for which there are ADO drivers.

Connecting to your Database

**MS Access.**

SETVAR SQL_CONNECT "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\program files\SqlLink3000\db\GUISALES.mdb"

**MS Sql Server**

SETVAR SQL_CONNECT "driver={SQL Server};server=pentium450;uid=sa;pwd=;database=northwind"

**Oracle**

SETVAR SQL_CONNECT "provider=MSDAORA;user id=system;data source=guiordb;password=manager;"

**ODBC**

SETVAR SQL_CONNECT "PROVIDER=MSDASQL;DSN=SE_System"
What follows is an example of how to set up an ODBC connection to PostgreSQL, although ODBC setups generally follow the same format. Start the ODBC admin program, and select System DSN.

Click on Add, and select your preferred ODBC driver, and click on Add...
The screen you are presented with depends on your driver. This is the PostgreSQL screen. Fill in your details and click on OK.

Your newly added connection will now appear in your list of ODBC connections.
This is a sample setup for MySQL and the database DJI01

The Connection strings from the above 2 samples would be:

**PostgreSQL**

```sql
SETVAR SQL_CONNECT "PROVIDER=MSDASQL;DSN=Postgres_Test"
```

**MySql**

```sql
SETVAR SQL_CONNECT "PROVIDER=MSDASQL;DSN=MySql_DJI01"
```

More details and evaluation copies are available from:

[www.gui-innovations.com](http://www.gui-innovations.com),
[www.gainsborough.com](http://www.gainsborough.com)

Please report any suggestions for improvements or problems to

guisqlink@gui-innovations.com

Thank You