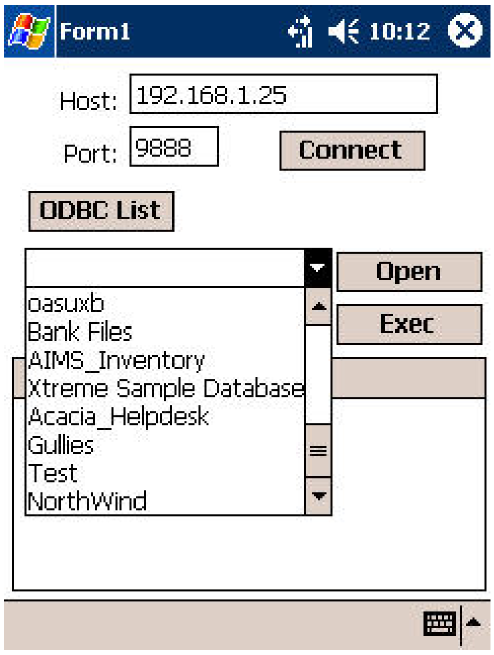
GUI Innovations Limited

SqlLinkCE



Host: 192.168.1.25

Port: 9888

oasuxb
Bank Files
AIMS_Inventory
Xtreme Sample Database
Acacia_Helpdesk
Gullies
Test
NorthWind

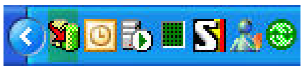
Table of Contents

Contents

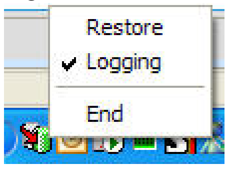
Introduction

SqlLinkCEServer allows you to access ANY ODBC databases from your Windows CE device. The system is implemented as a server process on the PC, and as a .NET dll on the CE Device.

The server process can run as a service.

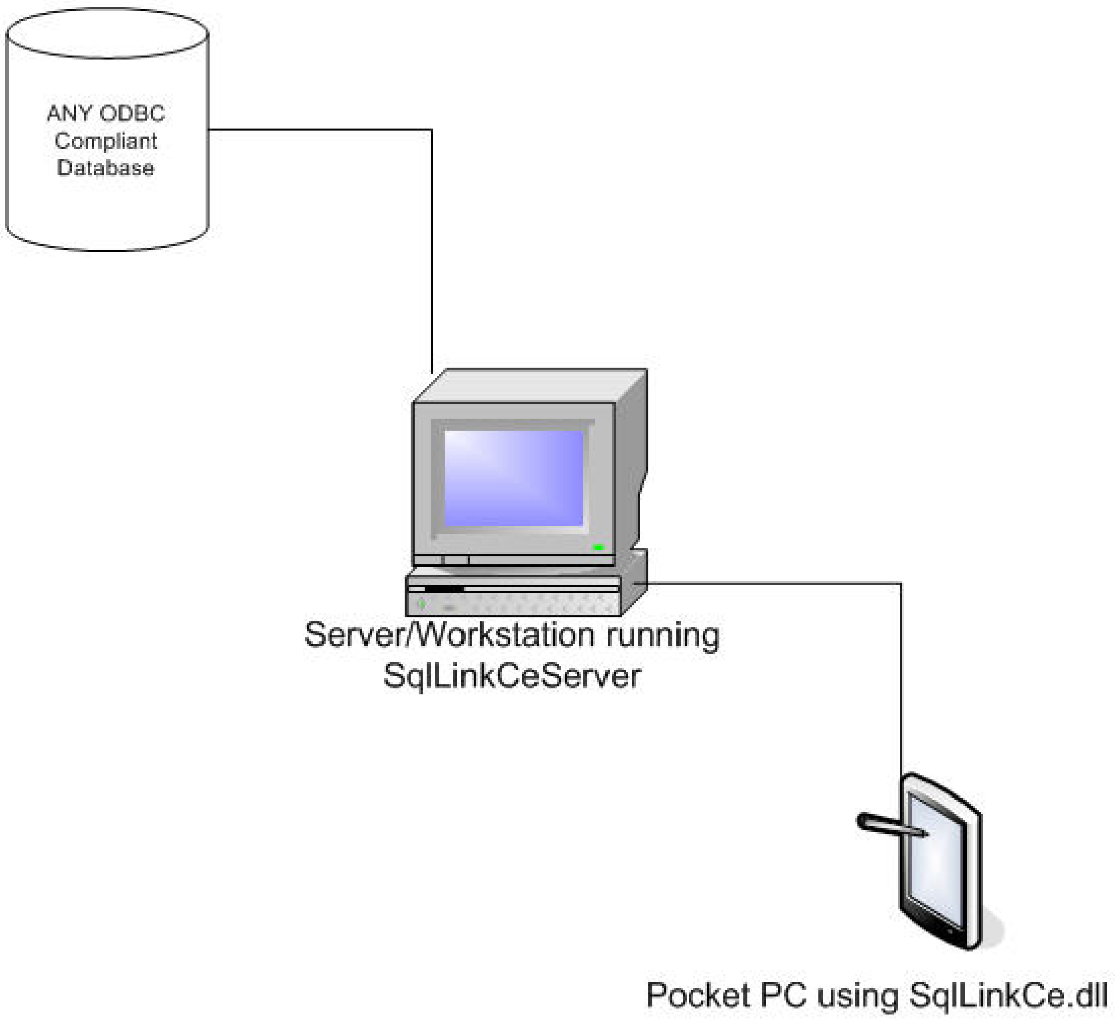
The SqlLinkCEServer process runs mimimised in the system tray 

Right click on it to access the menu



Where you can restore the screen, turn logging on and off, and end the process.

The SqlLinkCEServer must be running before your CE programs can manage databases.


ANY ODBC Compliant Database
Server/Workstation running SqlLinkCeServer
Pocket PC using SqlLinkCe.dll

-o-

Command Line Parameters

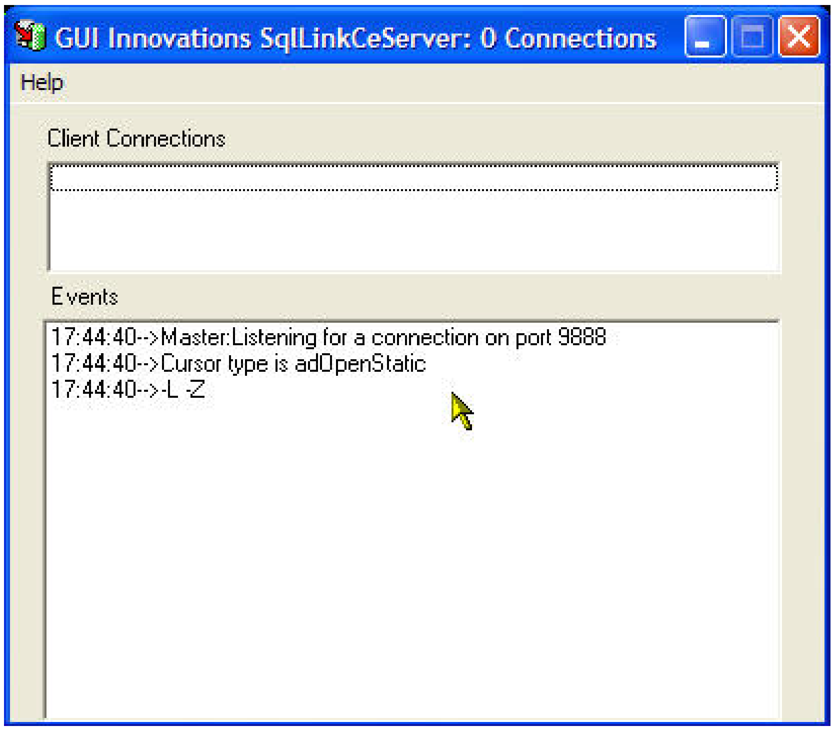
The following command line parameters can be specified when starting SqlLinkCEServer

-L Log all output to the screen
-P The port to listen on. If this is not specified then the default port of 9888 is used
-C and the cursor type to specify a cursor. The server uses ADO, and defaults its cursort type to adOpenStatic.
To change this simply specify the cursor type you want.

-C0 adopenforwardonly
-C1 adopenkeyset
-C2 adopendynamic
-C3 adopenstatic

-D The delimiter of the data (Default pipe (|)). This can be changed to any character you want, e.g.
-D~ Change the delimiter to ~
-D11 Change the delimiter to a tab character. To specify a non printing character, use its ASCII value

System being run with logging.



```
17:44:40-->Master:Listening for a connection on port 9888
17:44:40-->Cursor type is adOpenStatic
17:44:40-->-L -Z
```

-o-

Running SqlLinkCEServer as a service

GUIInnovationsService allows you to run our 'servers' as services

To install the service, run the following at the command prompt: GUIInnovationsService -i
To un-install the service, run the following at the command prompt:
GUIInnovationsService -u

By default, the installed service will be started automatically when you reboot the computer. You can also start and shutdown the service from the Control Panel using the Services icon. When the service is started, it will create all the processes you defined in the GUIInnovationsService.ini file one by one. When the service is shutdown, it will terminate each of the processes it created (in reverse order). The GUIInnovationsService.ini file should be placed in the same directory as the executable.

The ProcCount property specifies how many processes you want this service to create. The sections [Process0], [Process1], ..., etc., define properties related to each of these processes. As you can see, there is 1 processes to create in this example, SqlLink3000.exe is the name of the programs, and you can specify parameters for each of these processes in the CommandLine property. You must specify the full path of the executable file for the corresponding process in the CommandLine property unless the executable is already in the system path.

The CheckProcess property specifies whether and how often you want to check processes started by GUIInnovationsService. If the property has value 0, then no checking is done. If the property value is 30, for example, then every 30 minutes GUIInnovationsService will query the operating system to see if the processes it started are still running and the dead ones will be restarted if the Restart property value (explained later) is defined to be Yes for that process. The default value of this property (if you don't specify it) is 60.

GUIInnovationsService can check the processes it started periodically. A dead process will be restarted by GUIInnovationsService if you specify the Restart property for this process in the GUIInnovationsService.ini file.

The WorkingDir property is the working directory of the current process. If you don't specify this property, then the working directory of the current process will be c:\winnt\system32. The PauseStart property is the number of milliseconds the service will wait after starting the current process (and before starting the next process). This is useful in the case where the next process depends on the previous process. For example, the second process has to "connect" to the first process so that it should not be run until the first process is finished with initialization. If you don't specify the PauseStart property, the default value will be 100 milliseconds.

When GUIInnovationsService is shutdown, it will post WM_QUIT messages to the processes it created first and then call the WIN32 function TerminateProcess. The PauseEnd property is the number of milliseconds the service will wait before TerminateProcess is called. This property can be used to give a process (started by GUIInnovationsService) a chance to clean up and shutdown itself. If you don't specify the PauseEnd property, the default value will be 100 milliseconds.

The UserInterface property controls whether a logged on user can see the processes created by GUIInnovationsService. However, this only works when GUIInnovationsService is running under the local system account, which is the default. In this case, processes created by GUIInnovationsService will not be able to access a specific user's settings (e-mail profiles, etc.). You can configure GUIInnovationsService to run under a user account, which is done easily from the Control Panel (double click the Services icon and then double click GUIInnovationsService in the installed services list to bring up a dialog box).

The Restart property is used to decided whether you want GUIInnovationsService to restart a dead process. If this property is No (which is the default if you don't specify it), then the corresponding process will not be restarted. If this property is Yes, then the dead process will be restarted by GUIInnovationsService. See the CheckProcess property above on how often dead processes are restarted.

You can bounce (stop and restart) any process defined in the .ini file from the command line. For example, the following command:

```
GUIInnovationsService -b 0
```
will stop and restart the process defined in the [Process2] section of the .ini file.

GUIInnovationsService can also be used to start and stop other services from the command line. Here are the commands to start (run) and stop (kill) other services.

```
GUIInnovationsService -r NameOfServiceToRun
GUIInnovationsService -k NameOfServiceToKill
```

In particular, you can use the above commands to start and stop GUIInnovationsService itself from command line! Please note that you cannot start GUIInnovationsService by running it from the command prompt without any argument.

All errors while running GUIInnovationsService are written into a log file in the same directory as the executable. The error code in the log file is a decimal number returned by the GetLastError API, you can look it up in MSDN.

Obviously, you are not restricted to running our servers as services, this function will allow you to run any process as a service.

-o-

Sample GUIInnovationsService.ini

```
[Settings]
ServiceName = GUIInnovationsService
ProcCount = 1
CheckProcess = 1
[Process0]
CommandLine = C:\Program Files\SqlLinkCE\SqlLinkCEServer.exe
WorkingDir = C:\Program Files\SqlLinkCE
PauseStart = 1000
PauseEnd = 1000
UserInterface = Yes
Restart = Yes
```

-o-

SqlLink.dll

SqlLink.dll

SqlLink.dll allows you to read and update databases from your CE Device.

For more details, see Methods and Properties

A sample VB.Net program is included to show how to use the dll

-o-

Properties

Connected

Data Type: Boolean

Returns a value to say whether the PC is connected to the Pocket PC.

Syntax: *sqlLink*.Connected

See also: Connect_To_Server

-o-

Connection_String

Data Type: String

Sets or returns the connection string to open the database

Syntax: *sqllink*.Connection_String

See also: Open_Database

-o-

Database_User

Data Type: String

Sets or returns the database user name

Syntax: *sqllink*.Database_User

See also: Open_Database

-o-

Database_Password

Data Type: String

Sets or returns the database password

Syntax: *sqllink*.Database_Password

See also: Open_Database

-o-

EOF

Data Type: Boolean

Returns a value to say whether end of file has been reached following a select statment.

Syntax: *sqllink*.EOF

See also: Sql, Execute_Query

-o-

Error_Message

Data Type: String

Returns a value showing any error message returned from an operation.

Syntax: *sqllink*.Error_Message

See also Error_Number

-o-

Error_Number

Data Type: Integer

Returns a value to say whether an error was encountered during the last operation. Non-zero is an error condition.

Syntax: *sqllink*.Error_Number

See also Error_Message

-o-

Field_Count
Data Type: Integer

Returns a value to say how many fields were retrieved by a Select statement.

Syntax: *sqllink*.Field_Count

See also: Sql, Execute_Query

```
Dim ict as integer
for ict = 0 to rs.Field_Count -1
  list1.additem rs.Field_Name(ict), rs.Field_Value(ict),rs.Field_Len(ict), rs.Field_Type(ict)
Next
```

-o-

Field_Name

Data Type: String Array

Returns a value showing the name of each field returned.

Syntax: *sqllink*.Field_Name(0)

See also: Field_Count

-o-

Field_Length

Data Type: Integer Array

Returns a value showing the length of each field returned.

Syntax: *sqllink*.Field_Len(0)

See also: Field_Count

-o-

Field_Value

Data Type: Array

Returns a value showing the value of each field returned. You can use the field number or the field name to get the value.

Syntax: *sqllink*.Field_Value(0) or *sqllink*.Field_Value("Inspector_Name")

See also: Field_Count

-o-

Field_Type

Data Type: Integer Array

Returns a value showing the type of each field returned.

Syntax: *sqllink*.Field_Type(0)

Values are the same as ADO constants,

ADOX	SQL Server CE data types
adSmallInt	smallint
adInteger	integer
adSingle	real
adDouble	float
adCurrency	money
adBoolean	bit
adUnsignedTinyInt	tinyint
adBigInt	bigint
adGUID	uniqueidentifier
adVarBinary	varbinary
adBinary	binary
adVarWChar	nvarchar
adWChar	nchar
adNumeric	numeric
adDBTimestamp	datetime
adLongVarBinary	image
adLongVarWChar	ntext

See also: Field_Count

-o-

Host

Data Type: String

Sets or returns the host name/ip to connect to

Syntax: *sqllink*.Host

See also: Connect_To_Server

-o-

Licensed_User

Data Type: String

Returns a value containing the name of the user the software is licensed to

Syntax: *sqllink*.Licensed_User

See also:

-o-

Records

Data Type: Long

Returns a value showing the number of records affected by the last operation.

Syntax: *sqllink*.Records

See also: Sql, Execute_Query

-o-

Returned_Data

Data Type: String

Returns a value of all the fields returned separated by the pipe (|) character or whatever delimiter you have chosen. This can be used as an alternative to using Field_Value

Syntax: *sqllink*.Returned_Data

See also Field_Value

-o-

Remote_Port

Data Type: Integer

Sets or returns the port on the PC to connect to. Default port on the PC is 9888

Syntax: *sqllink*.Remote_Port

See also: Connect_To_Server

-o-

SQL

Data Type: String

Returns/Sets a value of the sql string you wish to excute.

Syntax: *sqllink*.Sql = "Select * from tDepots"

See also: Execute_Query, Execute_SQL

-o-

Table_Names

Data Type: String Array

Returns a value showing the names of all tables in the database. These values are returned automatically when the database is opened.

Syntax: *sqllink*.Tables(0)

-o-

Table_Type

Data Type: String Array

Returns a value showing the type of each table returned. Values are "Table" or "View". These values are returned automatically when the database is opened.

Syntax: *sqllink*.Table_Type(0)

-o-

Version_Number

Data Type: String

Returns a value showing the version number of the DLL. You may be asked for this if you have a problem.

Syntax: *sqllink*.Version_Number

See also

-o-

Methods

Connect_To_Server

Connect to the Pocket PC when the Pocket PC is functioning as a server. Needs the Host of the Pocket PC and the Remote_Port

Syntax: *sqlLink*.Connect_To_Server

Returns an integer value. 0 is no error. Error values are

-1 No host name or IP
-2 No remote port number set
-3 No host process listening on remote port
-4 The evaluation period for SqlLinkCe has expired
-5 The maximum number of connections has been reached

These are returned in Error_Number and Error_Message

-o-

Close_Database

Close the currently open database

Syntax: *sqlLink*.Close_Database

-o-

Disconnect

Disconnect from the remote server

Syntax: *sqlLink*.Disconnect

-o-

Execute_Query

Execute the query specified in the SQL property. This is used to select and return records. For statements that do not return values, use the Execute_Sql method.

Syntax: *sqlLink*.Execute_Query

```
    sqlLink.Sql = "Select * from Inspectors"
    sqlLink.Execute_Query()

     While Not sqlLink.EOF
         For Ict = 1 To sqlLink.Field_Name.Count
            lvRecords.Columns.Add(sqlLink.Field_Name(Ict).ToString, -2,
HorizontalAlignment.Left)
        Next

        For Ict = 1 To sqlLink.Field_Value.Count
            If Ict = 1 Then
                lvi = New ListViewItem
                 lvi.Text = (sqlLink.Field_Value("Inspector_ID").ToString)
           Else
                lvi.SubItems.Add(sqlLink.Field_Value(Ict).ToString)
           End If
        Next
        lvRecords.Items.Add(lvi)
        lvi = Nothing
        sqlLink.MoveNext()
     End While
```

-o-

Execute_SQL

Execute a SQL statement such as 'Delete from mytable where status = 1'

To use a 'Select' statement that returns records, use Execute_Query

Syntax: *sqlLink*.Execute_SQL

-o-

Return_Dataset

Execute the query specified in the Sql statement, and return a dataset. This is used to select and return records.
Dataset returned is called dsSqlLink

Syntax: *sqlLink*.Return_Dataset(sql)

```
        Dim ds As New System.data.DataSet
        ds = sqlLink.Return_DataSet("Select * from products order by productid")

        With ds.Tables("dsSqlLink")
            If .Rows.Count > 0 Then
                For iRows = 0 To .Rows.Count - 1
                         listbox1.items.add (.Rows(iRows).Item("ProductID").ToString & ":" &
.Rows(iRows).Item("ProductName").ToString
                Next
            End If
        End With
```

or....

```
         With ds.Tables("dsSqlLink")
            If .Rows.Count > 0 Then
                If Not bLVInit Then
                    lvRecords.Items.Clear()
                    For iCols = lvRecords.Columns.Count To 1 Step -1
                         lvRecords.Columns.Remove(lvRecords.Columns(iCols - 1))
                    Next
                    For iCols = 0 To .Columns.Count - 1
                         lvRecords.Columns.Add(.Columns(iCols).ToString, -2,
HorizontalAlignment.Left)
                    Next
                    bLVInit = True
                End If
                For iRows = 0 To .Rows.Count - 1
                     For iCols = 0 To .Rows(iRows).ItemArray.GetUpperBound(0)
                         If iCols = 0 Then
                             lvi = New ListViewItem
                             lvi.Text = (.Rows(iRows).Item(iCols).ToString)
                        Else
                             lvi.SubItems.Add(.Rows(iRows).Item(iCols).ToString)
                        End If
                    Next
                    lvRecords.Items.Add(lvi)
                    lvi = Nothing
                Next
            End If
        End With
```

-o-

Stored_Procedure_Returning_Records

Execute the stored procedure specified in the SQL property. This is used to select and return records. For stored procedures that do not return records, simply use the Execute_SQL method or the Stored_Procedure_Return_Code for stored procedures that return a value using the Return statement in SQL

The example below uses the Ten Most Expensive Products store procedure from the Northwind database

Syntax: *sqlLink*.Stored_Procedure_Returning_Records

```
    sqlLink.Sql = "exec " & Chr(34) & "Ten Most Expensive Products" & Chr(34)
    If sqlLink.Stored_Procedure_Returning_Records <> 0 Then
      MsgBox(sqlLink.Error_Message, MsgBoxStyle.Information, "Error " & sqlLink.Error_Number)
       Exit Sub
    End If

      While Not sqlLink.EOF
          For Ict = 1 To sqlLink.Field_Name.Count
              lvRecords.Columns.Add(sqlLink.Field_Name(Ict).ToString, -2,
HorizontalAlignment.Left)
          Next

          For Ict = 1 To sqlLink.Field_Value.Count
              If Ict = 1 Then
                  lvi = New ListViewItem
                  lvi.Text = (sqlLink.Field_Value("TenMostExpensiveProducts").ToString)
              Else
                  lvi.SubItems.Add(sqlLink.Field_Value(Ict).ToString)
              End If
          Next
          lvRecords.Items.Add(lvi)
          lvi = Nothing
          sqlLink.MoveNext()
      End While
```

-o-

Store_Procedure_Return_Code

Execute the stored procedure specified in the SQL property. This is used to run stored procedures that use the 'Return' statement in SQL. The name of the parameter you are returning, is specified after the store procedure name, and the value (which MUST be an integer) is returned from the procedure.

Syntax: *sqlLink*.Stored_Procedure_Return_Code

```
 'Return the value of ReturnValue from the stored procedure sp_test_proc
sqlLink.Sql = "sp_test_proc,Return"
 Dim iRet as integer
 iRet = sqlLink.Stored_Procedure_Return_Code
```

iRet will contain the value of Return. If the stored procedure fails, the iRet will return -1, and sqlLink. Error_Message and sqlLink.Error_Number will contain details of the error.

-o-

Get_ODBC_List

This will return a list of ODBC connections available to you.

Syntax: *sqlLink*.Get_ODBC_List

```
If sqlLink.Get_ODBC_List() = 0 Then
    Dim ict As Integer
    For ict = 1 To sqlLink.Connections.Count
        cmbBases.Items.Add(sqlLink.Connections(ict).ToString)
    Next
End If
```

-o-

Movenext

Moves to the next record in the recordset.

Syntax: *sqlLink*.Movenext

See also: Execute_Query, Sql

-o-

MovePrevious

Moves to the previous record in the recordset.

Syntax: *sqlLink*.MovePrevious

See also: Execute_Query, Sql

-o-

MoveFirst

Moves to the first record in the recordset.

Syntax: *sqlLink*.MoveFirst

See also: Execute_Query, Sql

-o-

MoveLast
Moves to the last record in the recordset.

Syntax: *sqlLink*.MoveLast
See also: Execute_Query, Sql

-o-

Open_Database

Opens the the database on the server.

Syntax: *sqllink*.Open_Database

```
sqlLink.Connection_String = "Provider=MSDASQL;DSN=" & cmbBases.Text
sqlLink.Database_Password = ""
sqlLink.Database_USer = "sa"
If sqlLink.Open_Database <> 0 Then
    MsgBox(sqlLink.Error_Message)
Else
    MsgBox("Opened OK!")
End If
```

See also: Close_Database, Disconnect

-o-

Sample Program

Adding the DLL

Click on Project>Add Reference and browse to \Program Files\SqlLinkCE and add the SqlLink.dll file.

In your program do

```
Imports SqlLinkCE
```

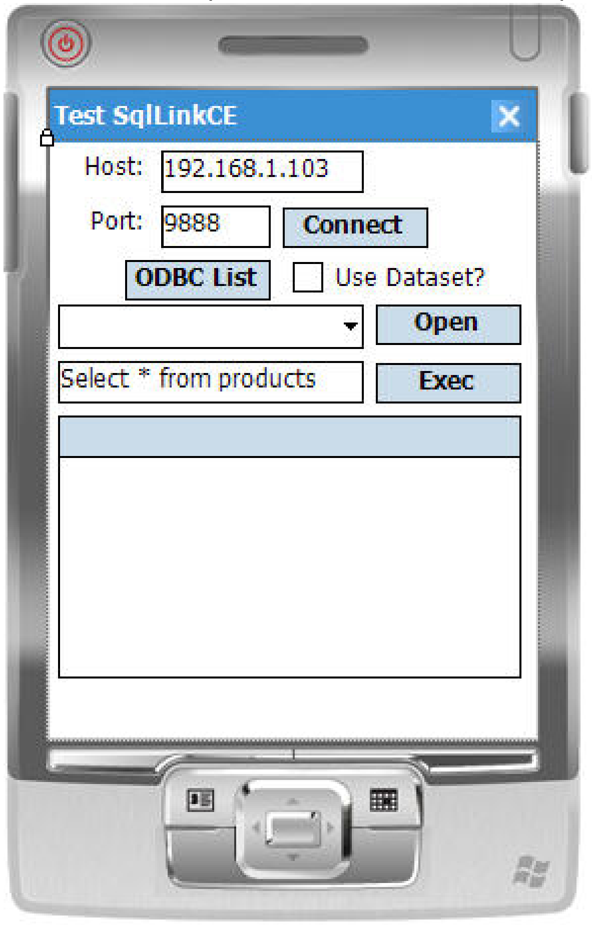
And then define it as...

```
Private sqlLink As New GUI_SqlLinkCe
```

-o-

How it works

Click on a hot spot below, to see how simple it is to access a remote database.



-o-

Index