

**DATA MIRROR FOR PT**

**USER'S GUIDE**

**Multiware, Inc.**

**December 31, 2009**

# 1. Introduction

The *Data Mirror for PT* is an application that maintains a copy, or mirror, of a Peachtree Accounting database in an external Sql Server database, allowing users to develop applications that will work with the Peachtree data using the development environment Microsoft has built around Sql Server and ensures that there is no interference between Peachtree and external applications that work with the mirrored data.

The heart of the *Data Mirror for PT* is a .NET class library that performs the mirroring operations and exposes a user interface that allows applications to control the way mirroring is performed. Initially all records in the selected files are read into the Sql Server database. Then when mirroring is enabled the Peachtree database is monitored for changes (additions, modifications, and deletions) and the Sql Server database is updated with the changed information. This is a significant improvement over the way that previous Multiware products *PawCom* and *MWToolkitForPT* got data, where all the records in the external tables were refreshed because it was not known what had changed. The *Data Mirror for PT* also reads data much faster. *PawCom* is approximately seven times faster than *MWToolkitForPT* and the *Data Mirror for PT* is 3-4 times faster than *PawCom* at filling the external tables.

The mirroring of the Peachtree data is performed as follows: The files that make up the Peachtree databases are monitored for changes to their Last Modified date. The application keeps track of files that have changed and waits a while until there are no further changes to the files for a short period of time, by default 10 seconds. The reason it is necessary to wait for modifications to the files to stop is because Peachtree typically makes several changes to a given file over a period of seconds. For example, when a sales invoice is entered in Peachtree it can take up to 15-20 seconds for the resulting changes of data files to complete even on a standalone system with a local database. When the database is stable the application reads new Audit Trail records to determine what operation was performed in Peachtree, then the application optionally copies the appropriate files to a local directory, then reads the modified records and updates the *Sql Server* tables accordingly.

Copying the modified files to a local directory insures that the application can access the data quickly without any interference from Peachtree. Previous experience has shown when Peachtree is being used on several workstations that attempts to read the direct data files from an external application can be extremely slow because Peachtree is tying up essential resources for its own use. Also reading Peachtree data across a network can be much slower than reading the data from a local directory.

The *Data Mirror for PT* includes an application *DisplayMirroredData* that illustrates all the features of the library and allows you to control mirroring and to view the contents of the Sql Server database. VB.NET source code for this application is provided.

## 2. Prerequisites

*Data Mirror for PT* supports Peachtree 2005 and later.

*Data Mirror for PT* requires that a Sql Server 2005 or 2008 version be installed. Fortunately Microsoft provides a free Express version of Sql Server that you can download and install.

To download Sql Server 2008 Express go to

<http://www.microsoft.com/exPress/download/>

We recommend you download the Runtime with Management Tools or the Runtime with Advanced Tools rather than just the Runtime version as the Management Studio is very useful for managing and examining Sql Server databases. A word of warning: installation of these kits takes quite a while to complete, so be sure to set aside some time to perform the installation.

Sql Server 2008 Express is a completely functional version of Sql Server. However, it does have some limitations compared to the higher version Sql Server. The most significant has to do with the maximum size of the database. Although we have tested the *Data Mirror for PT* with some large Peachtree databases, it is possible that some extremely large Peachtree databases will contain more data than the Express version can handle, in which case you would need to purchase and install a higher version.

The application *DisplayMirroredData* was built with VB.NET 2008 Express, which is a free development tool that you can also download from the link above. If you want to modify the application you will need either the Express or full version of VB.NET 2008. Microsoft also provides other free Express tools that you may want to consider installing.

The first time the MirrorControl object is invoked it will look for the specified Sql Server database. If it can't find it then it will create the database and build all the necessary tables, views, and stored procedures. This insures that the database will be compatible with the version of Sql Server that you are using. You can also specify an existing Sql Server database that does not have the necessary database components and they will be added to that database.

### 3. MirrorControl Class

The *MirrorControl* is a .NET native class that provides the interface to the *Data Mirror for PT* allowing you to completely control the mirroring process. The class should be referenced in .NET applications that you develop.

Sample code in this section will be for VB.NET. However, the code is very straightforward, so translation to other languages such as C#.NET should be fairly obvious.

Since the MirrorControl class contains events, you can refer to the MirrorControl as follows:

```
Dim WithEvents oControl As New EDataMirrorForPT.MirrorControl
```

#### 3.1 Methods

##### ClearALLPTData

Deletes all Peachtree data from the Sql Server tables. Example:

```
oControl.ClearAllPTData()
```

##### Connect(DataSource, InitialCatalog, Username, Password)

Instruct the MirrorControl to make a connection to the specified Sql Server database. The parameters are optional:

- DataSource:** The name or network address of a Sql Server to which you want to connect. If no value is specified, “.SQLEXPRESS” is used.
- InitialCatalog:** The name of the database to which you want to connect. If no value is specified “DataMirrorForPT” is used.
- Username:** The User ID you want to connect to the database with.
- Password:** The password you want to connect to the database with.

A username and password are required if the database has been secured. By default the generic DataMirror database does not require a username or password. If you need to specify a username and password you can alternatively specify their value with the Username and Password properties before invoking the Connect method. Examples:

```
oControl.Connect()
```

Connect to database DataMirrorForPT using local Sql Server Express instance.

```
oControl.Connect(DataSource, InitialCatalog, , ,)
```

Connect to specified database using specified Sql Server instance.

```
oControl.Connect(DataSource, InitialCatalog, Username, Password)
```

Connect to the specified secured database with the username/password using the specified Sql Server instance.

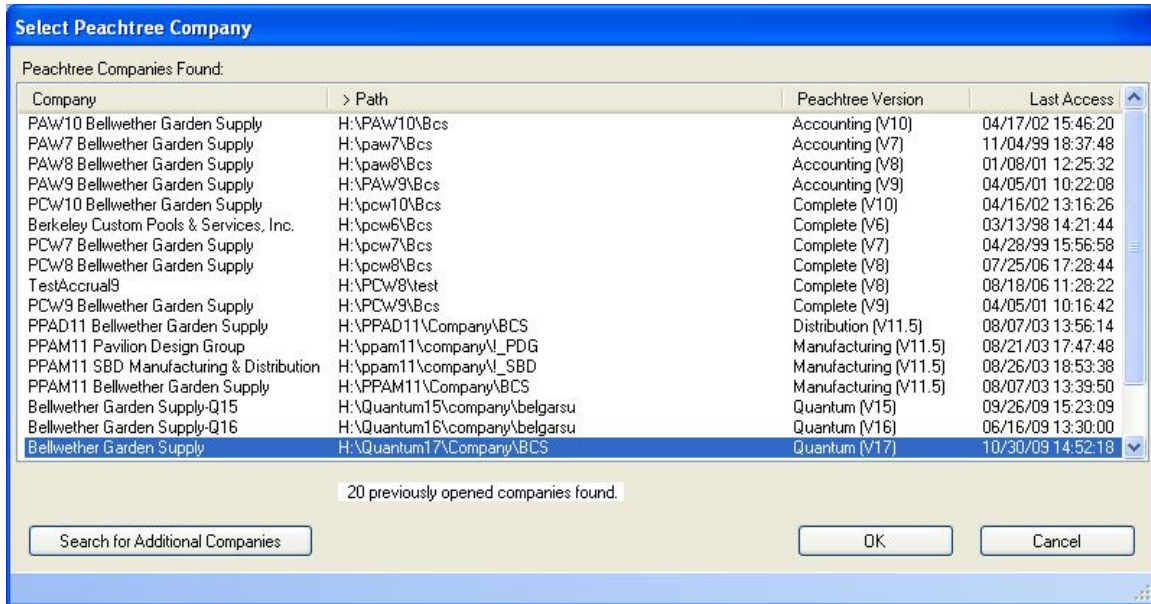
## ReadAllPTData

For each component specified on the DataToMirror form read all records. Example:

```
Result = oControl.ReadAllPTData()
```

## SelectCompany

Display a list of Peachtree company databases found on the system.



The initial list of companies is found by searching the registry and Peachtree ini files for any databases that any version of Peachtree has previously opened. Click on the company you want to use and click OK. You can also click on any column heading to sort by that column. Click on the column a second time and the sort will be in reverse order.

If you click the *Search for Additional Companies* button a more exhaustive search will be performed by scanning all your disk partitions. This can take some time to perform. After the search any additional companies found will be highlighted in light yellow and any databases located in Peachtree backup files will be highlighted in light green.

If you do not want to use the form for selecting a company and you know the company's path, you can enter the path directly in the Sql Server table named *Options*.

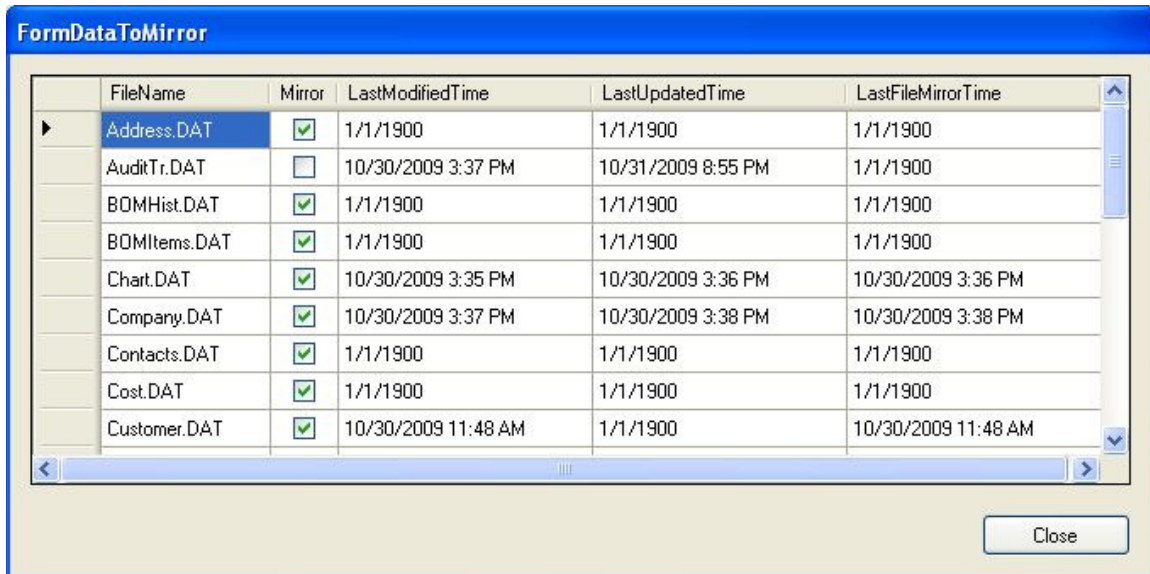
If you do not select a new company or click Cancel the return value will be false. If a problem is encountered that doesn't allow the form to display, an error will be raised.

Example:

```
Try
    Result = oControl.SelectCompany
Catch ex As Exception
    MsgBox ("Error selecting company:" & vbCrLf & ex.Message)
End Try
```

## ShowDataToMirror

Display a form that lets you select the files you want to mirror by checking the item in the Mirror column.



The screenshot shows a window titled "FormDataToMirror" containing a table with the following data:

FileName	Mirror	LastModifiedTime	LastUpdatedTime	LastFileMirrorTime
Address.DAT	<input checked="" type="checkbox"/>	1/1/1900	1/1/1900	1/1/1900
AuditTr.DAT	<input type="checkbox"/>	10/30/2009 3:37 PM	10/31/2009 8:55 PM	1/1/1900
BOMHist.DAT	<input checked="" type="checkbox"/>	1/1/1900	1/1/1900	1/1/1900
BOMItems.DAT	<input checked="" type="checkbox"/>	1/1/1900	1/1/1900	1/1/1900
Chart.DAT	<input checked="" type="checkbox"/>	10/30/2009 3:35 PM	10/30/2009 3:36 PM	10/30/2009 3:36 PM
Company.DAT	<input checked="" type="checkbox"/>	10/30/2009 3:37 PM	10/30/2009 3:38 PM	10/30/2009 3:38 PM
Contacts.DAT	<input checked="" type="checkbox"/>	1/1/1900	1/1/1900	1/1/1900
Cost.DAT	<input checked="" type="checkbox"/>	1/1/1900	1/1/1900	1/1/1900
Customer.DAT	<input checked="" type="checkbox"/>	10/30/2009 11:48 AM	1/1/1900	10/30/2009 11:48 AM

The LastModifiedTime is the date and time that the file was last changed by Peachtree. The LastUpdatedTime is the date and time the records in the file were written to the Sql Server database. The LastFileMirrorTime is the date and time that the file was last copied to the local mirror folder. For all three fields a value of 1/1/1900 indicates the date and time have not changed since the last time the entire database was read, either because ReadAllPTData was called or because the Peachtree company was changed.

Notice that AuditTr.DAT is unchecked. The application will read new audit trail records anyway but this will prevent the application from reading old audit trail records since the number of old records in the Audit Trail file can be very large and are not of any interest unless you specifically want to examine the audit trail.

Example:

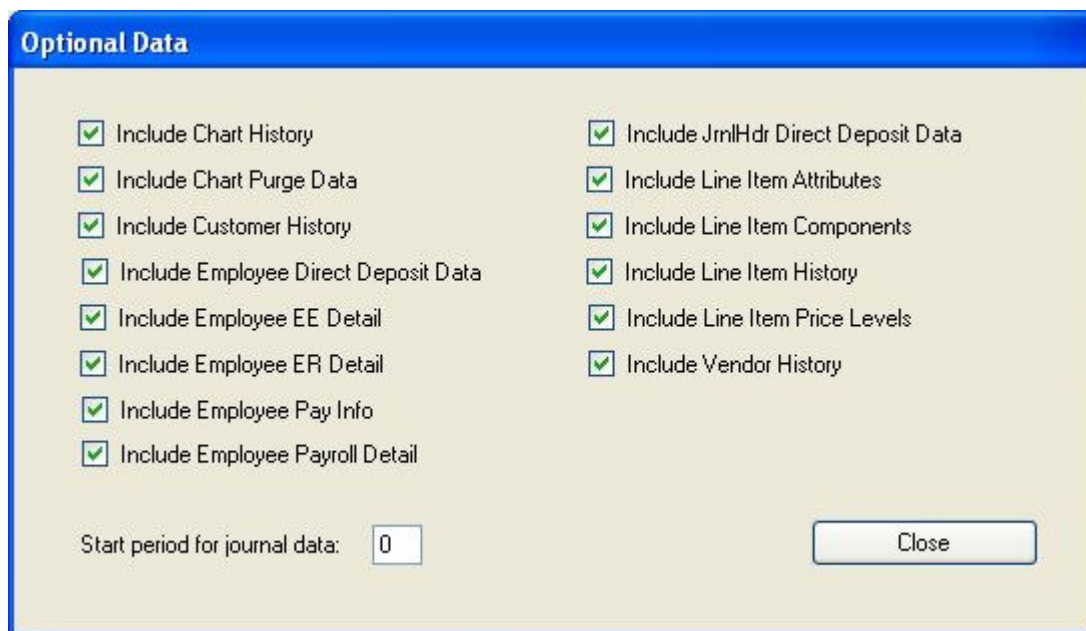
```
oControl.ShowDataToMirror()
```

The data on this form is stored in the Sql Server table r. *DataToMirror* If you do not want to display the form above in your application you can edit the Sql Server table directly as desired.

## ShowOptionalData

Display a form that lets you select some optional data to copy from the Peachtree database to the Sql Server database. The items on this form all have a fair amount of additional overhead associated with reading the information, so performance can be improved if you only select optional data that you need. By default all optional data is read.

The *Start period for journal data* value lets you specify the internal period at which the application should start reading journal records when ReadAllPTData is invoked. A value of 0 means read the complete set of journal transactions going back to the start of the company. If you want to only read data in open periods the value should be set to 15.



**Optional Data**

<input checked="" type="checkbox"/> Include Chart History	<input checked="" type="checkbox"/> Include JnlHdr Direct Deposit Data
<input checked="" type="checkbox"/> Include Chart Purge Data	<input checked="" type="checkbox"/> Include Line Item Attributes
<input checked="" type="checkbox"/> Include Customer History	<input checked="" type="checkbox"/> Include Line Item Components
<input checked="" type="checkbox"/> Include Employee Direct Deposit Data	<input checked="" type="checkbox"/> Include Line Item History
<input checked="" type="checkbox"/> Include Employee EE Detail	<input checked="" type="checkbox"/> Include Line Item Price Levels
<input checked="" type="checkbox"/> Include Employee ER Detail	<input checked="" type="checkbox"/> Include Vendor History
<input checked="" type="checkbox"/> Include Employee Pay Info	
<input checked="" type="checkbox"/> Include Employee Payroll Detail	

Start period for journal data:

Close

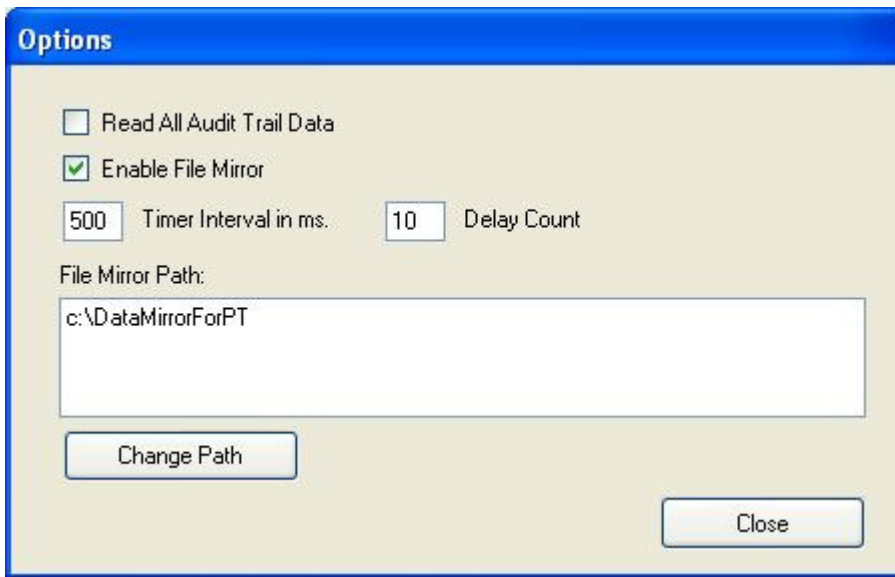
Example:

```
oControl.ShowOptionalData()
```

The data on this form is stored in the Sql Server table *OptionalData*. If you do not want to display the form above in your application you can edit the Sql Server table directly as desired.

## ShowOptions

Display a form that lets you select some controlling options.



When the *Read All Audit Trail Data* box is checked the entire audit trail history will be read when ReadAllPTData is invoked.

When the *Enable File Mirror* box is checked, Peachtree data files will be copied to a local mirror folder before they are read. If the box is not checked then the Peachtree data files will be read from their original location where Peachtree is using the data. Copying the files to a local folder eliminates interference from Peachtree and network access issues when reading the files. However, there may be situations where you want to access the Peachtree files in their original location.

The *Timer Interval in ms.* Value represents the period of time the application pauses in the main control loop that performs operations. The default value is 500 ms. You can probably change this to 100 ms if you want to see faster response to events in applications that call the library, but you may see some degradation in system performance if you make this value too small.

The *Delay Count* is the number of seconds that the application waits after the last file has been modified before reading data from the Peachtree files. The default value is 10. The reason this value is so large is because Peachtree typically modifies files for a period of 15-20 seconds after a change has been made on one of the Peachtree forms.

The *File Mirror Path* is the local location where Peachtree data files are copied when the *Enable File Mirror* box is checked.

The data on this form is stored in the Sql Server table DataToMirror. If you do not want to display the form above in your application you can edit the Sql Server table directly as desired.

## **StartMirroring**

Inform the library to start the mirroring process. Example:

```
oControl.StartMirroring()
```

## **Stop Mirroring**

Inform the library to stop the mirroring process. Example:

```
oControl.StopMirroring()
```

## **3.2 Properties**

### **CompanyName**

Return the currently selected Peachtree company name. *ReadOnly*.

### **CompanyPath**

Return the currently selected Peachtree company's directory path. *ReadOnly*.

### **ConnectionString**

Return the Sql Server connection string that is currently in effect. *ReadOnly*.

### **Initialized**

Return a Boolean value indicating whether a successful connection to a Sql Server database has been performed yet. *ReadOnly*.

### **Password**

Specify a password to use for the Sql Server connection string. If the database is secured the password can either be specified with this property before calling the Connect method or the password can be specified as a parameter for the Connect method. *Writeonly*.

### **TimerEnabled**

Return or set a Boolean value specifying whether the library internal timer for mirroring is running.

### **TimerInterval**

Return or set the timer interval in ms of the library internal timer for mirroring.

### **Username**

Specify a User ID to use for the Sql Server connection string. If the database is secured the username can either be specified with this property before calling the Connect method or the username can be specified as a parameter for the Connect method. *Writeonly*.

### 3.3 Events

If the calling application has included the `WithEvents` parameter when defining the `MirrorControl` object then .NET will include the following event handlers in your application.

#### **`oControl.MirrorStateChanged(ByVal State As Short)`**

Triggered each time the mirroring state is changed while the library is mirroring data. The values of the `State` parameter are

```
Off = 0
Stable = 1
Waiting = 2
MirroringData = 3
```

If mirroring is turned off then the state will be set to `Off`.

If mirroring has finished processing any changes to the Peachtree data files then the state will be set to `Stable`.

If mirroring has detected that a file has been modified then the state will be set to `Waiting`.

If mirroring has detected that the Peachtree database has changed and no further changes have occurred for the number of seconds set in the Options `DelayCount` parameter, then processing of the changes will begin and the state will be set to `Mirroring Data`.

#### **`oControl.LogTransactionAdded(ByVal myTime As DateTime, ByVal Milliseconds As Integer, ByVal Message As String)`**

The library maintains a transaction log in the Sql Server table `Transaction Log`. Each time a new entry is made to this log this event is triggered so that the calling application can monitor the log entries in real time. The log entries are defined by the date and time they are entered. The milliseconds gives finer division of the entry time so that entries that occur within the same second can be sorted properly. The `Message` is the text of the log entry.

#### **`oControl.StatusText(ByVal Text As String)`**

The library also generates informational text that is not associated with a `Transaction Log` entry. This information typically indicates what the library is doing with finer detail than the `Transaction Log` data, so is useful to display as a status line in the calling application.

#### **`oControl.StatusProgressInit(ByVal MaxValue as Integer)`**

The library provides progress information for operations that will take a while. This information is useful for displaying a progress bar in the calling application. This event is triggered before progress begins and indicates the maximum value the progress bar should have.

### **oControl.StatusProgressUpdate(ByVal Value as Integer)**

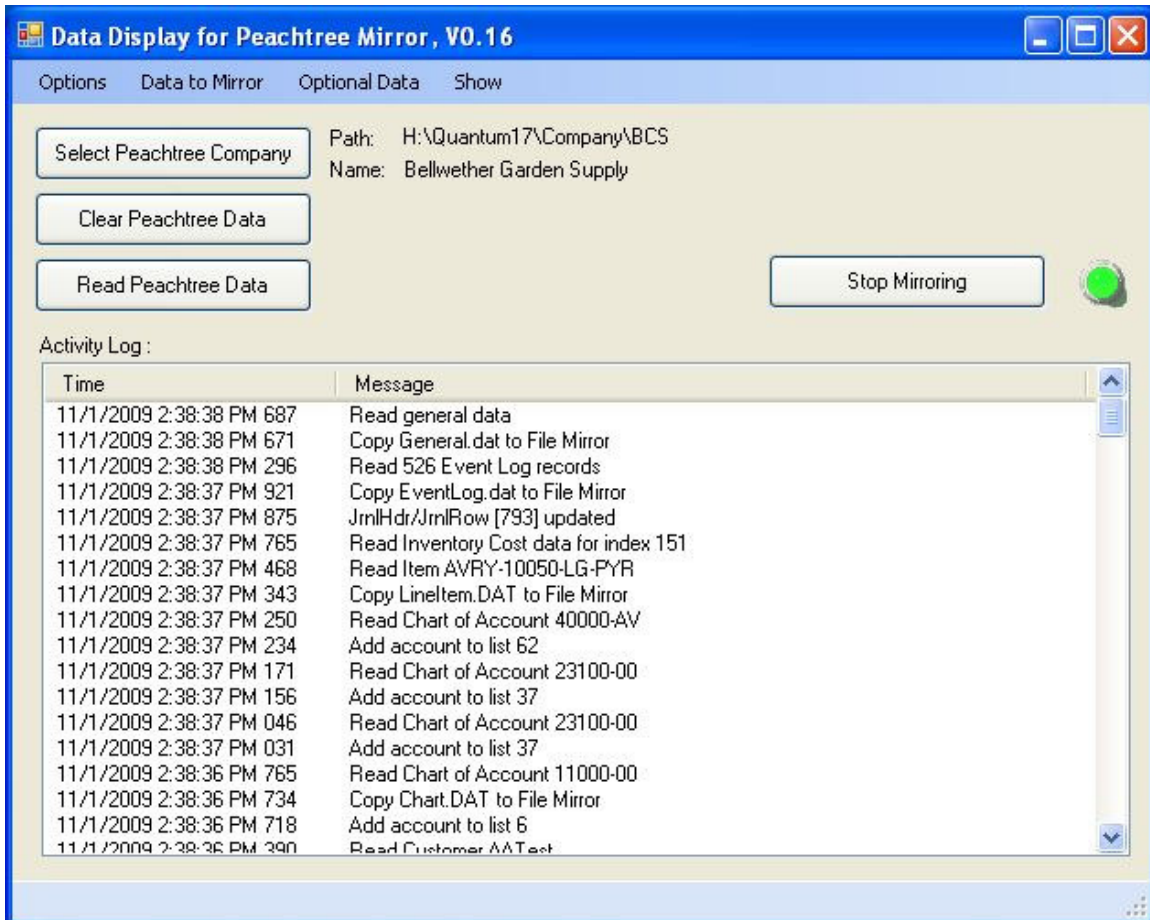
This event is triggered each time an updated value on the progress of an operation is triggered.

### **oControl.StatusProgressDone()**

This event is triggered when the progress of the current application is complete.

## 4. Display Mirrored Data Application

The DataMirrorForPT kit includes the utility DisplayMirroredData, written in VB.NET, that illustrates how to use all the methods and properties of the MirrorControl class. This utility lets you control the data mirroring library without having to write any code. The utility also includes simple forms for showing data in many of the Sql Server data tables.



The first three menu bar items, when selected, call the library to show the corresponding form. The Show menu item can be used to select a display of data in many of the Sql Server database. These data display forms are very simple - similar to what you would see viewing the data in an MS Access table or with Sql Server Management Studio.

The three buttons on the left are for invoking the corresponding methods in the MirrorControl class.

The button on the right will start and stop mirroring. The light to the right of the button will be green when the mirroring state is stable, yellow when it is waiting to make updates, and red when the Sql Server database is being updated.

Activity log entries will be red for errors. Click the description to see full text.

A source listing of the VB.NET can be found in Appendix A.

## 5. Sql Server vs. PawCom Access tables

In general the Sql Server tables in MirroredData.mdf are similar to the tables in the PawCom MS Access application. However, the PawCom tables have a lot of extraneous baggage from older versions of Peachtree. Also the PawCom tables have some fields where Peachtree originally stored IDs but now stores indices, and for these fields the current PawCom has to do an extraneous and time-consuming lookup to find the ID given the index in order to maintain backward compatibility. That is, the structure of the PawCom tables is more consistent with the way the Peachtree records were stored several years ago. The Sql Server tables, on the other hand, are based on the latest structure of the Peachtree records, which has been fairly stable for the last few years.

In the MS Access application for PawCom the default file parameters caused the journal data to be read twice, once for the JrnlHdr/JrnlRow raw data and again for the specific journal views of that data. The MirrorControl object only reads the JrnlHdr and JrnlRow records and the MirroredData Sql Server database contains specific views for the various journals that map exactly into the PawCom specific journal tables.

Plans are to create additional views that map directly into the PawCom tables so that applications that were developed with PawCom will be able to see recordsets in the Sql Server database that look exactly like the current PawCom tables.

## 6. Attaching the Sql Server tables to MS Access

The Sql Server database can be linked to an MS Access application using ODBC. To do this you first need to create an ODBC connection. When using Sql Server Express you can do this by going to Control Panel : Administrative Tools : Data Sources (ODBC). Click the System DSN tab and then the Add button. Select the "SQL Server Native Client 10.0" driver. Name the data source DataMirrorForPT and in the server to connect to box enter [local]\SQLEXPRESS. On the next form select Integrated Windows Authentication.

After creating an ODBC source to the Sql Server database you can link to the tables in MS Access as follows: in design mode go to the menu bar entry "File : Get External Data : Link Tables". Under "Files of Type" select ODBC databases. On the next form select "Machine Data Source : DataMirrorForPT". On the Link Tables form select only the files that begin with "dbo." The other files are internal database system files that Sql Server uses behind the scenes. When you click OK you will see "Select Unique Record Identifier" boxes pop up for each journal you selected. For header tables select NRecord or GUID. For detail tables select GUID. You may also see a box for Customer Sales. If so select CustomerID and Period.

## Appendix A. Source listing for DisplayMirroredData

```
Imports Microsoft.SqlServer.Management.SMO
Imports System.Data

Public Class MainForm

    Dim Mirroring As Boolean = False
    Dim LEDColor As String = ""
    Dim LogIndex As Integer = 1
    Dim FormShown As Boolean = False
    Dim ListIndex As Integer = 0

    Dim WithEvents oControl As New DataMirrorForPT.MirrorControl

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        SetLED("Off")
        ButtonState(False)
        Me.Show()
        Application.DoEvents()
    End Sub

    Private Sub btnSelectCompany_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSelectCompany.Click
        ButtonState(False)
        Dim Changed As Boolean = oControl.SelectCompany()
        ButtonState(True)
        If Changed Then
            Me.tbPath.Text = oControl.CompanyPath
            Me.tbCompanyName.Text = oControl.CompanyName
        End If
    End Sub

    Private Sub btnClearAll_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnClearAll.Click
        Cursor.Current = Cursors.WaitCursor
        ButtonState(False)
        oControl.ClearAllPTData()
        ButtonState(True)
        Cursor.Current = Cursors.Default
    End Sub

    Private Sub btnReadAll_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnReadAll.Click
        ButtonState(False)
        Cursor.Current = Cursors.WaitCursor
        oControl.ReadAllPTData()
        Cursor.Current = Cursors.Default
        ButtonState(True)
    End Sub

    Private Sub mMirrorStateChanged(ByVal State As Short) Handles
oControl.MirrorStateChanged
        Dim myColor As String = ""
        If State = 0 Then
```

```

        myColor = "Off"
    ElseIf State = 1 Then
        myColor = "Green"
    ElseIf State = 2 Then
        myColor = "Yellow"
    Else
        myColor = "Red"
    End If
    SetLED(myColor)
End Sub

Private Sub mLogTransactionAdded(ByVal myTime As DateTime, ByVal
Milliseconds As Integer, ByVal Message As String) Handles
oControl.LogTransactionAdded
    Dim objItem As New ListViewItem
    ListIndex += 1
    objItem = ListView1.Items.Add(Format(ListIndex, "000000"))
    objItem.SubItems.Add(myTime.ToString & " " &
Format(Milliseconds, "000"))
    objItem.SubItems.Add(Message)
    If Message.Substring(0, 6) = "Error:" Then
        objItem.ForeColor = Color.Red
    Else
        objItem.ForeColor = Color.Black
    End If
    Application.DoEvents()
End Sub

Private Sub mStatusText(ByVal Text As String) Handles
oControl.StatusText
    Me.ToolStripStatusLabel1.Text = Text
    Application.DoEvents()
End Sub

Private Sub mStatusProgressInit(ByVal MaxValue As Integer) Handles
oControl.StatusProgressInit
    With Me.ToolStripProgressBar1
        .Visible = True
        .Minimum = 0
        .Maximum = MaxValue
    End With
    Application.DoEvents()
End Sub

Private Sub mStatusProgressUpdate(ByVal Value As Integer) Handles
oControl.StatusProgressUpdate
    Me.ToolStripProgressBar1.Value = Value
    Application.DoEvents()
End Sub

Private Sub mStatusProgressDone() Handles
oControl.StatusProgressDone
    Me.ToolStripProgressBar1.Visible = False
    Application.DoEvents()
End Sub

Private Sub ButtonState(ByVal State As Boolean)

```

```

        btnClearAll.Enabled = State
        btnReadAll.Enabled = State
        btnSelectCompany.Enabled = State
        btnMirrorControl.Enabled = State
        MenuStrip1.Enabled = State
        Application.DoEvents()
    End Sub

    Private Sub SetLED(ByVal Color As String)
        Dim myColor As String
        myColor = Color.ToUpper
        imgLedGreen.Visible = False
        imgLedYellow.Visible = False
        imgLedRed.Visible = False
        If myColor = "GREEN" Then
            If Mirroring Then
                ButtonState(True)
                imgLedGreen.Visible = True
            End If
        ElseIf myColor = "YELLOW" Then
            imgLedYellow.Visible = True
            ButtonState(False)
            btnMirrorControl.Enabled = True
        ElseIf myColor = "RED" Then
            imgLedRed.Visible = True
            ButtonState(False)
        Else
            ButtonState(True)
        End If
        LEDColor = myColor
        Application.DoEvents()
    End Sub

    Private Sub btnMirrorControl_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles btnMirrorControl.Click
        Mirroring = Not Mirroring
        If Mirroring Then
            Me.btnMirrorControl.Text = "Stop Mirroring"
            SetLED("Green")
            oControl.StartMirroring()
        Else
            oControl.StopMirroring()
            Me.btnMirrorControl.Text = "Start Mirroring"
            SetLED("Off")
        End If
    End Sub

    Private Sub ChartOfAccountsToolStripMenuItem_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
    ChartOfAccountsToolStripMenuItem.Click
        FormChart.Show()
    End Sub

    Private Sub OptionalDataToolStripMenuItem_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
    OptionalDataToolStripMenuItem.Click
        oControl.ShowDataToMirror()
    End Sub

```

```

End Sub

Private Sub OptionalDataToolStripMenuItem1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
OptionalDataToolStripMenuItem1.Click
    oControl.ShowOptionalData()
End Sub

Private Sub OptionsToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
OptionsToolStripMenuItem.Click
    oControl.ShowOptions()
End Sub

Private Sub AuditTrailToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AuditTrailToolStripMenuItem.Click
    FormAuditTrail.Show()
End Sub

Private Sub CompanyInfoToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CompanyInfoToolStripMenuItem.Click
    FormCompanyInfo.Show()
End Sub

Private Sub ContactsToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ContactsToolStripMenuItem.Click
    FormContacts.Show()
End Sub

Private Sub CostsToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
    FormCosts.Show()
End Sub

Private Sub CustomersToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CustomersToolStripMenuItem.Click
    FormCustomers.Show()
End Sub

Private Sub EmployeesToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
EmployeesToolStripMenuItem.Click
    FormEmployees.Show()
End Sub

Private Sub EventLogToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
EventLogToolStripMenuItem.Click
    FormEventLog.Show()
End Sub

```

```
Private Sub InventoryCostToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
InventoryCostToolStripMenuItem.Click
    FormInventoryCost.Show()
End Sub
```

```
Private Sub JrnlHdrToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
JrnlHdrToolStripMenuItem.Click
    FormJrnlHdr.Show()
End Sub
```

```
Private Sub JrnlRowToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
JrnlRowToolStripMenuItem.Click
    FormJrnlRow.Show()
End Sub
```

```
Private Sub JrnlSNoToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
JrnlSNoToolStripMenuItem.Click
    FormJrnlSNo.Show()
End Sub
```

```
Private Sub LineItemsToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
LineItemsToolStripMenuItem.Click
    FormLineItems.Show()
End Sub
```

```
Private Sub JobDataToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
JobDataToolStripMenuItem.Click
    FormJobs.Show()
End Sub
```

```
Private Sub PhasesToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
PhasesToolStripMenuItem.Click
    FormPhases.Show()
End Sub
```

```
Private Sub CostsToolStripMenuItem1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CostsToolStripMenuItem1.Click
    FormCosts.Show()
End Sub
```

```
Private Sub SalesTaxCodesToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
SalesTaxCodesToolStripMenuItem.Click
    FormSalesTaxCodes.Show()
End Sub
```

```
Private Sub TaxAuthoritiesToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TaxAuthoritiesToolStripMenuItem.Click
```

```

        FormTaxAuthorities.Show()
    End Sub

    Private Sub TaxTablesToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TaxTablesToolStripMenuItem.Click
        FormTaxTables.Show()
    End Sub

    Private Sub TicketsToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TicketsToolStripMenuItem.Click
        FormTicket.Show()
    End Sub

    Private Sub VendorsToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
VendorsToolStripMenuItem.Click
        FormVendors.Show()
    End Sub

    Private Sub WorkTicketsToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
WorkTicketsToolStripMenuItem.Click
        FormWorkTickets.Show()
    End Sub

    Private Sub MainForm_Shown(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Shown

        Cursor.Current = Cursors.WaitCursor
        mStatusText("Initializing. Please wait ...")
        Application.DoEvents()

        Try
            'The connect method can be used in any of the following
ways:
            ' oControl.Connect() - Connects with
DataSource=SQLExpress, Initial Catalog = "DataMirrorForPT"
            ' oControl.Connect(DataSource, InitialCatalog, , ) -
Connects to the specified Sql Server instance and database
            ' oControl.Connect(DataSource, InitialCatalog, Username,
Password) Connects to the spefied Sql Server
            ' instance and database with Sql Authentication.
            oControl.Connect()
        Catch ex As Exception
            MsgBox(ex.Message, , "Error on oControl.Connect")
        End Try

        If Not oControl.Initialized Then
            Me.Close()
            Exit Sub
        End If

        gblConnectionString = oControl.ConnectionString
        Me.tbPath.Text = oControl.CompanyPath
        Me.tbCompanyName.Text = oControl.CompanyName

```

```

        Mirroring = False
        Cursor.Current = Cursors.Default
        mStatusText("")
        SetLED("Off")
        ButtonState(True)

    End Sub

    Public Sub New()

        ' This call is required by the Windows Form Designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.

    End Sub

    Protected Overrides Sub Finalize()
        oControl = Nothing
        MyBase.Finalize()
    End Sub

    Private Sub ListView1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ListView1.Click

        MsgBox(ListView1.SelectedItems.Item(0).SubItems(1).Text.ToString &
vbCrLf & _

        ListView1.SelectedItems.Item(0).SubItems(2).Text.ToString)
    End Sub

End Class

```