DATA MIRROR FOR PT

USER'S GUIDE

Multiware Software, Inc.
Jul 21, 2019

Supports Sage 50 2020 or Earlier

Changes are listed in the version history

Table of Contents

1. Introduction 3
2. Prerequisites 4
3.1 Methods 5
ClearALLPTData 5
Connect(DataSource, InitialCatalog, Username, Password) 5
ReadAllPTData 6
SelectCompany 6
ShowDataToMirror 7
ShowOptionalData 8
ShowJournals2Mirror 9
ShowOptions 10
StartMirroring 11
Stop Mirroring 11
3.2 Properties 11
CompanyName 11
CompanyPath 11
ConnectionString 11
Initialized 11
Password 11
TimerEnabled 12
TimerInterval 12
Username 12
3.3 Events 12
oControl.MirrorStateChanged(ByVal State As Short) 12
oControl.LogTransactionAdded(ByVal myTime As DateTime, ByVal Milliseconds As Integer, ByVal Message As String) 12
oControl.StatusText(ByVal Text As String) 13
oControl.StatusProgressInit(ByVal MaxValue as Integer) 13
oControl.StatusProgressUpdate(ByVal Value as Integer) 13
oControl.StatusProgressDone() 13
4. Display Mirrored Data Application 14
5. SQL Server vs. PawCom Access Tables 15
6. Attaching the SQL Server tables to MS Access 15
7. Connection to SQL Server 16
32-bit Windows 16
64-bit Windows 16
DefaultMirrorData.XML 17
If you use Company Name & Folder, READ CAREFULLY 17
SQL Authentication 19
8. Starting the Data Mirror on Startup 20
The “A” Startup Parameter & Scanning the Audit Trail 21
9. Mirroring 2 or More Companies 23
10. Using DataMirrorForPT Data in MS Excel, MS Access or MS Word 23

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 DataMirrorForPT.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, InitialCatlog,,,)
```
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.

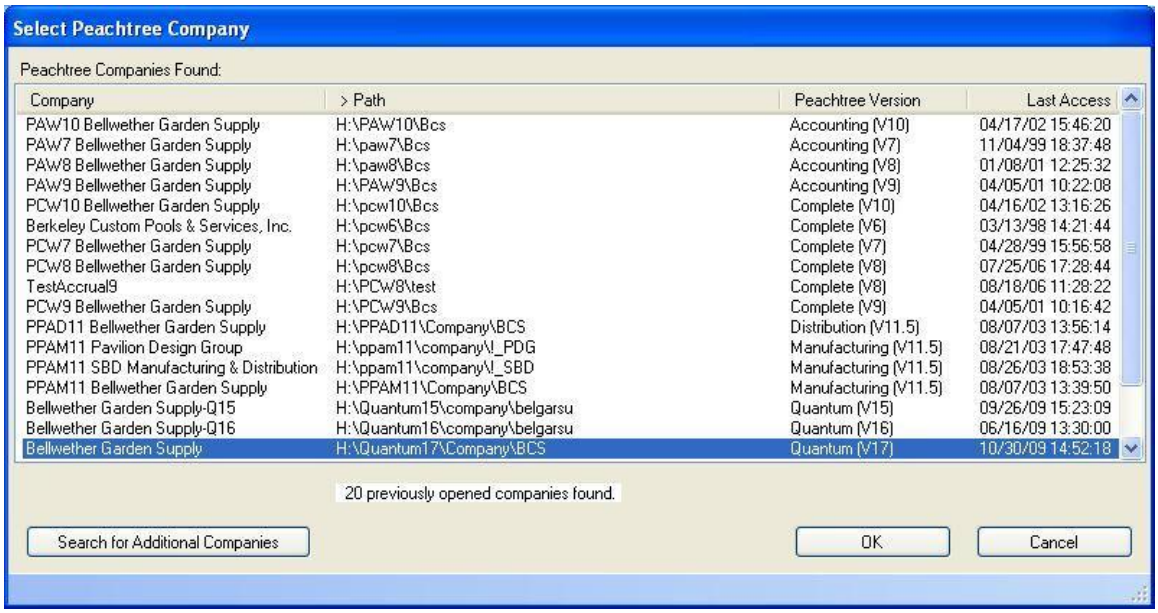
ReadAllPTData

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*.

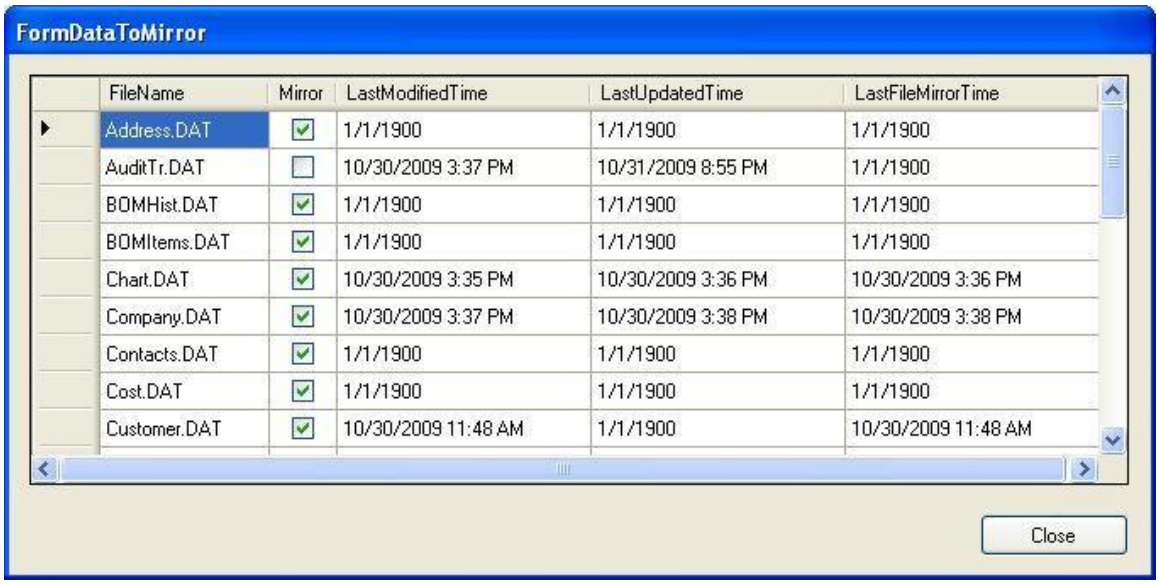
If you do not select a new company or click Cancel the return value will be false. If a problem is encountered that doesn't allow the form to display, an error will be raised.

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 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()
```

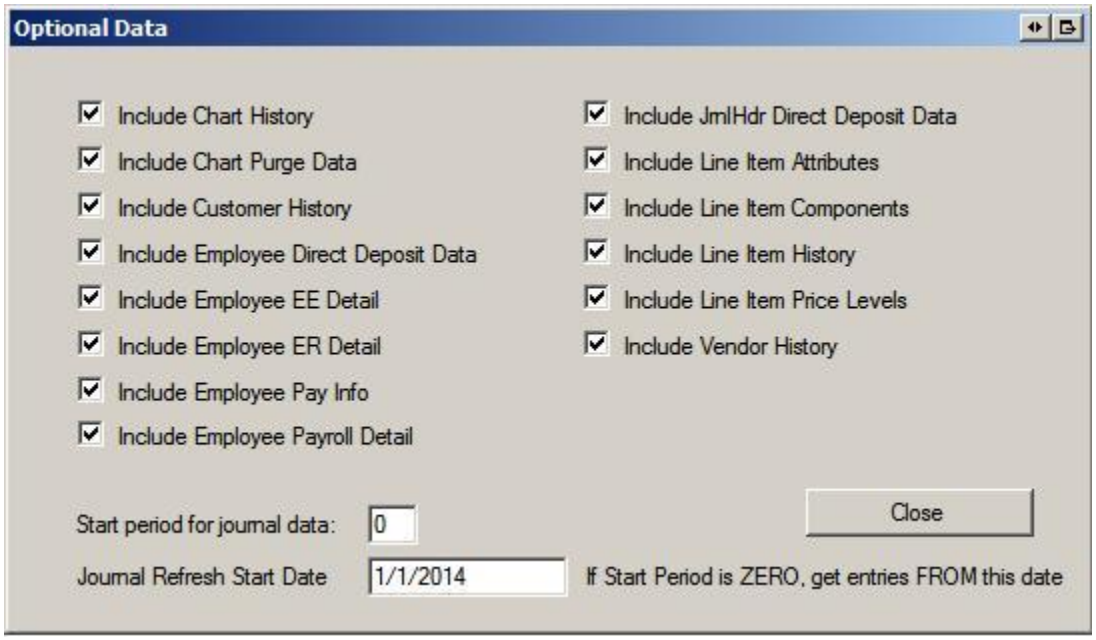
The data on this form is stored in the SQL Server table r. *DataToMirror* If you do not want to display the form above in your application you can edit the SQL Server table directly as desired.

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.

Journal Refresh Start Date allows you to limit journal records to a starting date. So below, instead of reading all journal rows going back to the very first entry, only record entered from 1/1/2014 will be exported. Be careful with the interaction ***between*** Start Period & Start Date. Use the Periods table to get your bearings.



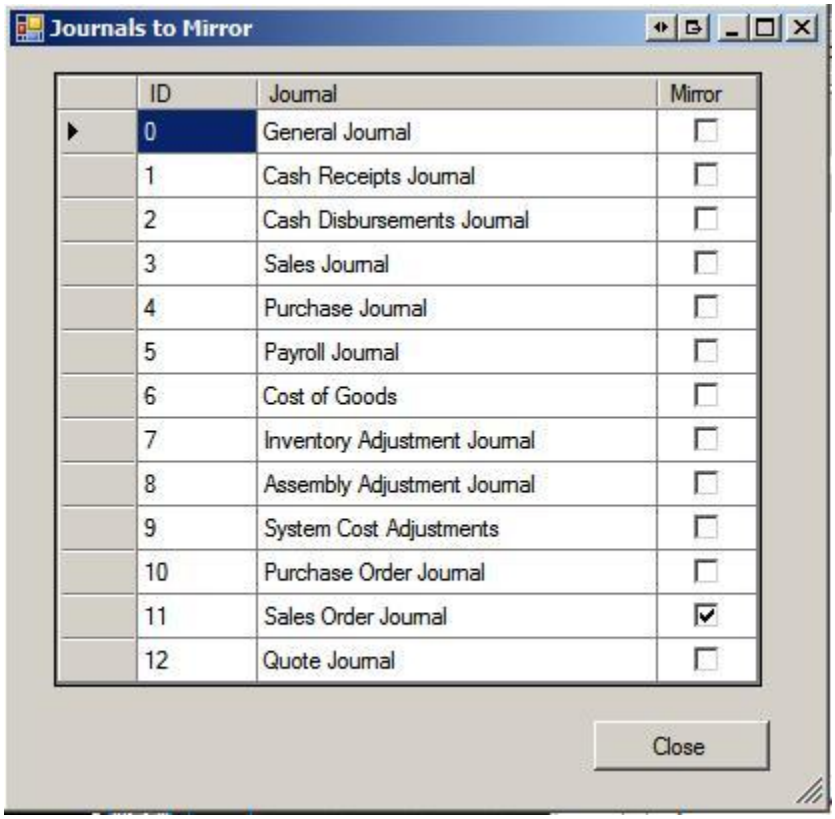
Example:

```
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.

ShowJournals2Mirror

Displays a form that lets you select which journals to mirror. In case where you have a very large database with high volume of transactions, you will get better performance by limiting the journals mirrored. However in limiting the journals you must be careful get the data you need for your application. For example, sales invoicing may need sales orders and cash receipts. Similarly purchases may need purchase orders & cash disbursement.



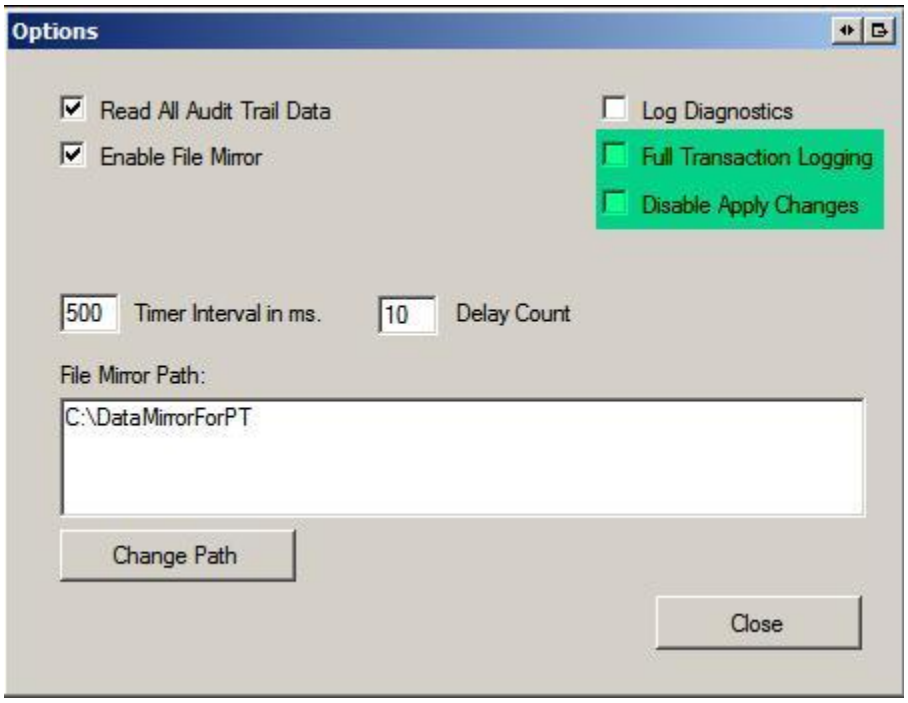
Example:

```
oControl.ShowJournals2Mirror()
```

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

ShowOptions

Displays 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.

SQL Server has 2 recovery models: full & simple. Simple is the default for the Data Mirror however you can override that setting by checking Full Transaction Logging. For large databases, full transaction logging takes lots of resources/disk space.

Checking *Disable Apply Changes* will not allow the users to apply changes only to implemented tables. This may be necessary if the volume of non-mirrored transactions is excessive. For very large databases with very high volume of changes, *Read Peachtree Data* may be the fastest way to ensure your mirrored tables are completely current.

Checking *Log Diagnostics* will enable additional logging of diagnostics. Un-checking *Log Diagnostics* will disable additional logging.

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 default is FALSE or disabled.

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)

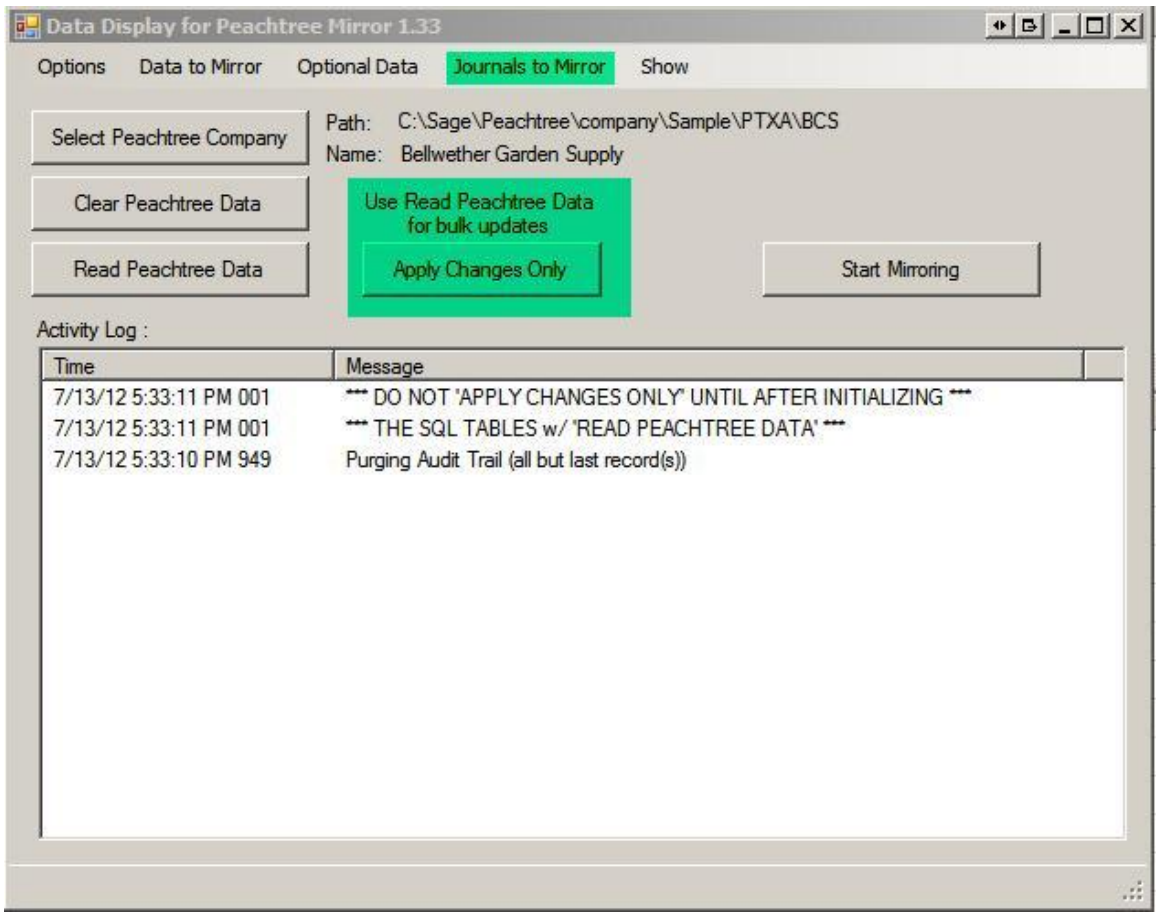
This event is triggered each time an updated value on the progress of an operation is triggered.

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 four 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 middle button [Apply Changes Only] will scan the SQL tables for missing records and add them, records in Peachtree showing a higher LastUpdateCounter (where available) and update them, and records deleted in Peachtree and delete them.

- [Apply Changes Only] applies just to the Customer, Journal and Line Item tables for release 1.33.

- For version 1.35, the [Apply Changes Only] is done for the Journal any time there is a delete recorded in the audit trail.
- For release 1.37.06.2014, [Apply Changes Only] only scans the last 30 days. This means Journal *deletions/changes made before the 30 day window must be addressed with a full Read Peachtree Data*.

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 compatability. 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.

7. Connection to SQL Server

Connection methods vary. The following assumes you have SQL Server & SQL Management Studio installed & some familiarity with both.

32-bit Windows

DataMirrorforPT will query the registry for instances of SQL Server and allow you to choose which one to use for the Mirror database.

64-bit Windows

DataMirrorforPT will be *unable* to navigate to the section of the registry with SQL instances. Use DefaultMirrorData.xml in the DataMirrorforPT (install) folder to specify the SQL Server instance to be used. The following shows the default settings:

DefaultMirrorData.XML

```
<?xml version="1.0"?>
<Data>
        <DataSource>.\SQLEXPRESS</DataSource>
        <InitialCatalog>DataMirrorForPT</InitialCatalog>
        <Username></Username>
        <Password></Password>
        <CompanyName></CompanyName>
        <CompanyFolder></CompanyFolder>
</Data>
```

If you use Company Name & Folder, READ CAREFULLY

Company Name & Folder: You can set company name & folder in DefaultMirrorData.xml.

- Blanks (default setting)
  - Uses what was set last through the [DisplayMirrorData] form, or
  - If newly opened, a blank triggers the [Select Company] form to open which scans for all Sage 50 companies. You then select from that list.

- Company Name & Folder SET
  - If the same as the currently set company, does nothing.
  - If different from the currently set company, same as selecting a new company from the [Select Company] form. Clears all the tables.

Rules:

- Either leave blank or set ***BOTH*** company name & folder.
- Don't set company name ***OR*** folder and expect it to work properly! (Don't leave one blank)
- To be sure you get the right data, use the Sage 50 [Maintain Company] form.
  - Copy the Company Name & Folder from there.
- If you do not ***exactly match the Company Folder***, don't expect it to work!
- DO NOT USE QUOTATION MARKS.
- If you change the company through the [Select Company] form, make sure you adjust DefaultMirrorData.xml.
  - If the Company Name & Folder are set differently in DefaultMirrorData.xml then the Data Mirror will go back to that on the next restart.

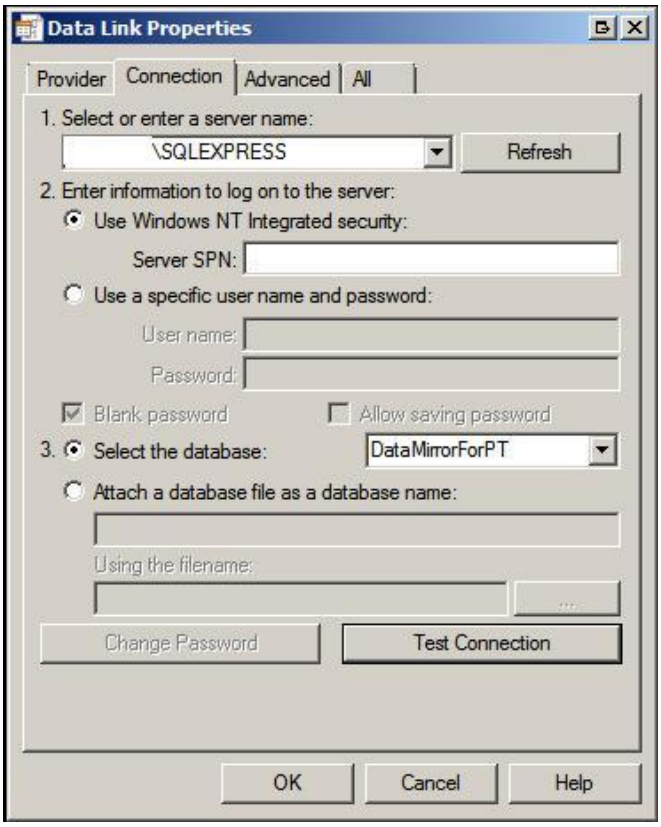
If you are having difficulty connecting, we suggest you use a universal data link file to test your connection independent of the DataMirrorforPT. Simply create a new test.txt on your desktop, change the extension to .udl and then double click test.udl.

On the provider tab, select SQL Native Client or SQL Server Native Client 10.0.

1. If the server drop down does not populate, click refresh, then select a SQL Server instance.

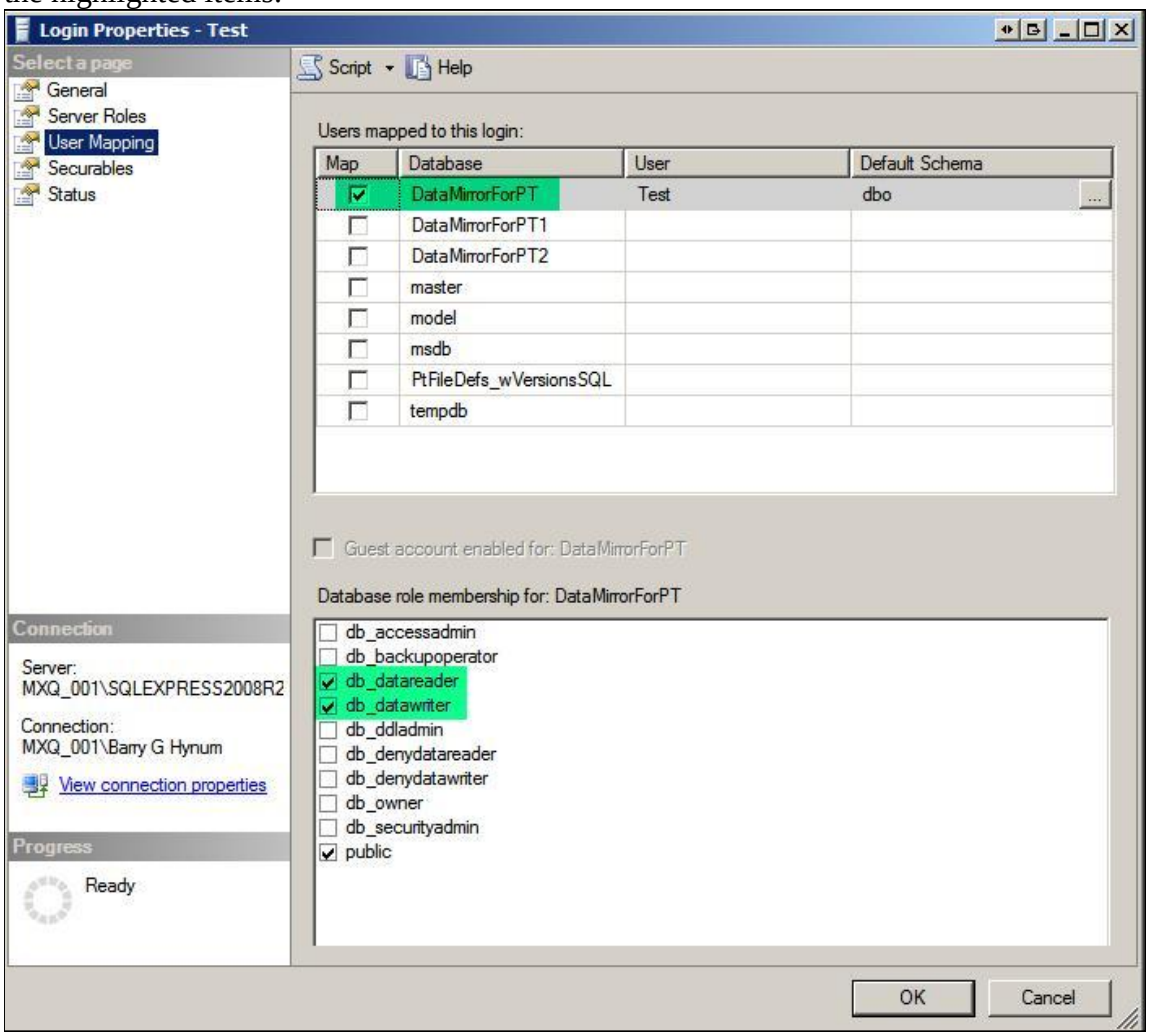
2. Set your authentication method. The default is Windows Authentication but you can use SQL authentication if you choose.

3. Select the database: DataMirrorforPT. Then click test connection. (You may need to create an empty version of the DataMirrorforPT database using SQL Management Studio)

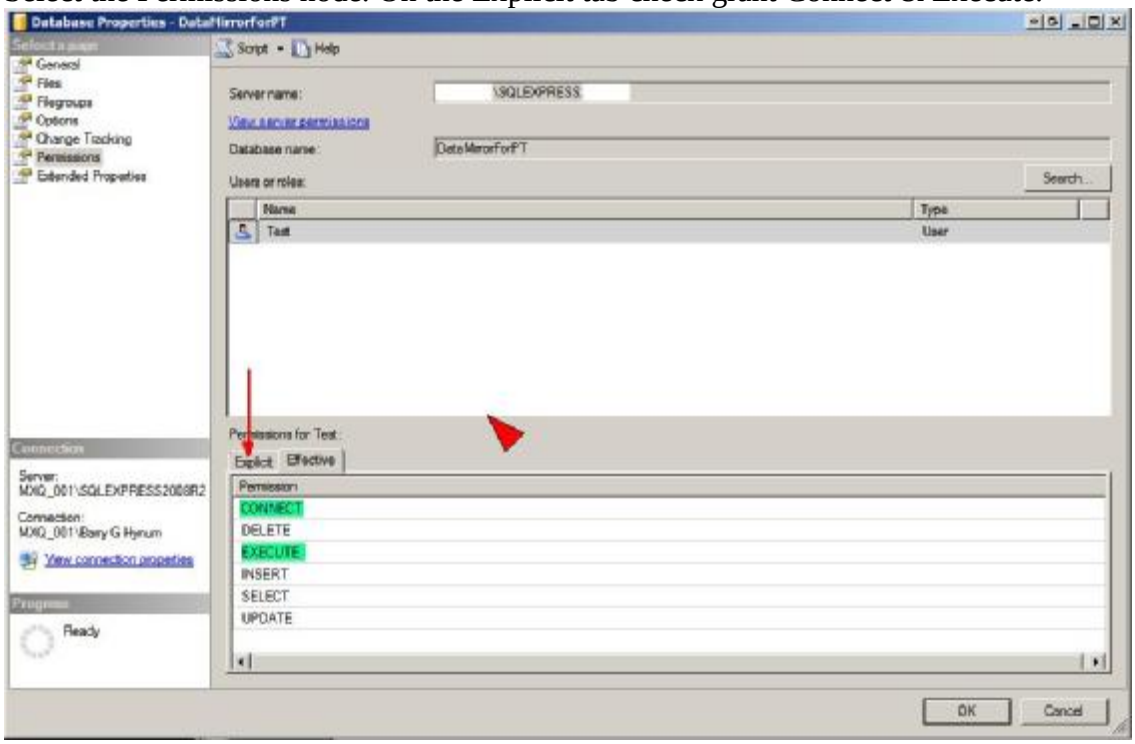


SQL Authentication

If you choose to use SQL authentication, then you will need a user name & password set up using SQL Management Studio. Select the Security node then Logins. Create a new login or use an existing login. Then in the login properties select User Mapping. Check the highlighted items.



Expand the database node then right click DataMirrorforPT to expose the properties. Select the Permissions node. On the Explicit tab check grant Connect & Execute.



Property views will vary by version of SQL Server. Exposing logins in plain text is not considered a safe practice; we recommend Windows Authentication.

8. Starting the Data Mirror on Startup

The DisplayMirroredData application has 5 command line switches:

- A – Scan audit trail for changes since the last DM start up.
- R - Refresh (read) all on Startup, and
- S - Start mirroring on Startup
- C - Apply Changes on Startup (Customers, Line Items & Journals [last 30 days])
- X – Exit application (Quit)

- The switch delimiters are spaces.
- Valid combinations:
    - A X (Scan audit trail & exit)
    - S (Scan audit trail then start mirror)
    - R X (Read all & exit)
    - R S (Read all & Scan audit trail then start mirror)
    - C X (Apply changes & exit)
    - C S (Apply changes & start mirror)
- Be sure to use valid sequences otherwise expect the unexpected.

You can start up the DisplayMirroredData application from ***Task Scheduler*** in the Administrative Tools in the Control Panel. Create a basic task, set the start "When computer starts," set "Start a Program" (navigate to DisplayMirroredData.exe in the Program Files under Multiware), add arguments (no quotes), lastly add the start in folder "C:\Program Files\Multiware\DataMirrorForPT\" (no quotes). (On 64-bit machines, the start in folder will be "C:\Program Files (x86)\Multiware\DataMirrorForPT") (no quotes).

If you wish to use the Data Mirror as a service, please take a look at ***Always Up*** (can turn any executable into a service.)

The company must be set manually at first use. If you want specific tables, set them manually before using startup parameters.

If you start up the DisplayMirroredData application from Task Scheduler the DisplayMirroredData form will display but it can be minimized.

The "A" Startup Parameter & Scanning the Audit Trail

- Originally starting the mirror with the "S" parameter did not scan the audit trail for changes since the last run until a change was made in Sage 50 that triggers the mirror. That has been corrected: now "S" will scan the audit trail for changes THEN start mirroring.
- Originally a manual start of the mirror did not scan the audit trail for changes since the last run until a change was made in Sage 50 that triggers the mirror. That has been corrected: now a manual mirror start will scan the audit trail for changes THEN start mirroring.
- A read-all "R" does a full table refresh on all selected tables. For big databases with large tables, this is inefficient.
- The "A" startup parameter will scan the audit **trail for any changes since the last audit trail record processed.** In other words, "A" is designed to grab only ***CHANGES*** since the last Data Mirror run.
- A X (Scan audit trail & exit) The *transaction log* reflects the session so viewing those records will tell you what happened since the last DM run. If you are only interested in tracking changes, use A X then examine the transaction log to see a list of changes since the last DM run.
    - You can connect either MS Excel or MS Access via ODBC to the transaction log table & any other tables of interest if you are uncomfortable with SQL Server.
    - Important: the transaction log is cleared on a re-start of the data mirror. That means YOU must capture the transaction log BEFORE re-starting the data mirror IF you want a CONTINUOUS record of Sage 50 changes over a period of time.
        - NOTE: a re-start means re-starting the data mirror *application*.
        - Starting & stopping via the mirror BUTTON will not clear the transaction log.

- The data mirror continuously records transactions monitored in the [transaction log] table. The [transaction log] table is only cleared when you stop & re-start the data mirror application. Starting & stopping with the mirror button will not clear the [transaction log] table.

9. Mirroring 2 or More Companies

Any number of companies can be mirrored. Just make a copy of the install folder: i.e. copy “C:\Program Files\Multiware\DataMirrorForPT” to “C:\Program Files\Multiware\DataMirrorForPT2”, edit DefaultMirrorData.xml in the ***copy folder***. Change the InitialCatalog name. The 1st time your run DisplayMirroredData.exe from DataMirrorForPT2, designate the company from which you want to pull data. Note you will need to start each DisplayMirroredData.exe from their respective folders. They may be run side by side.

A separate DataMirrorForPT folder & database are needed for each respective company to be mirrored.

10. Using DataMirrorForPT Data in MS Excel, MS Access or MS Word

A separate document is provided with getting started assistance for using DataMirrorForPT in either MS Excel, MS Access or MS Word, see UsingMirrorData_InExcel_orAccess.pdf in the “C:\Program Files\Multiware\DataMirrorForPT” folder.