2012.10.23

Sales Order: Peachtree Imports Using the Data Mirror

Project: mw_Pt_SampleImports.sln

Preparation Steps

We recommend that you *do not attempt imports unless you are a journeyman or journey-woman software engineer.* Your Peachtree Imports Using the Data Mirror subscription entitles you to a ½ hour consult. More than that is billed hourly.

Initial Setup: See the project settings for the default settings. The same settings are available from the mw_Pt_SampleImports.exe.config in the install folder. Minimally review the SQL instance & database name. Adjust for your site. See settings box below for more details.

Target Peachtree Company: The Peachtree company specified in your DataMirrorForPT SQL database is the target for the imports. Sample sales orders will be taken from that data. Imports will go into that company.

Application User Interface: A single form is displayed when you start the application. It has 4 buttons.

Button1: Add Required SOJ SQL Objects

(Run only ONCE) Be sure you have specified your SQL instance & database name correctly for your site.
SQL tables, views & procedures are used by this application. Step 1 is to add the needed SQL objects to your DataMirrorForPT database. The SQL scripts reside in the project resources. Each template will have a set of SQL objects. Sales Orders has 5 SQL objects. Inspect sb_IterateResources in cls_Add_SQLObjects.vb if you want to familiarize yourself with the method used to add SQL objects.

Button 2: Make SOJ XML Sample File

In order to import an xml file, you will need the basic structure of the xml file. One way to get this is to build a SQL view that uses the data you want to export but *pulls the data from a Peachtree database that already has sales orders*. 2 SQL views (aaVw_SOJ_Hdr & aaVw_SOJ_Det) have the sample import data that I used for my import sample. *(You do NOT want to import ALL your old sales orders. I filter out all but the last 2 transaction using the SQL view)* These 2 SQL views are placed in a dataset oDS and filled by the data adapter oDA. See cls_MakeSampleXML.vb. I also added a relationship oRel so that the way the data is displayed in the xml is nested. I also added sb_IterateDS(oDS) to assist with inspection of the dataset data; and sb_IterateTables_nFields for inspection of the data structure AFTER import (see oDS_Test.ReadXml). These test/inspection aides are commented out so uncomment as needed to do your own inspections.

Use this step to export a sample of data you intend to import. Examine the xml created in the process & use the structure to assist in building the xml files for actual import by your application. Be careful with potential nulls & empty strings: see the sales rep & sales tax code handlers. Freight charges also requires care.

Import Steps

Button 3: Read SO XML Files & Save to SQL

Uses the class cls_GetXML.vb to iterate the xml files in a folder specified in the project settings, save the sales order header & details to SQL tables for processing. Inspect the dataset DS_4SalesOrders.xsd regarding the target SQL objects. The data in Peachtree has not been modified up to this point. Be sure you have a test company & appropriate Peachtree data backups before going to the next step.

Button 4: Save SQL Sales Orders to Peachtree

Uses the class cls_SalesOrders.vb to iterate the sales order header & detail records in the SQL tables. Also validates the ID's & adds associated indexes, validates tax & subtotal calculations (fn_ValidateSOItemIDs), & displays discrepancies if any. If the data is validated, the sales order is marshaled for Peachtree import (fn_SaveSalesOrder). Each sales order is wrapped in a transaction so that a failed import is rolled back.

The ship-to address is pulled from the Peachtree address table. If no ship-to address is available, then the bill-to address is used.

The sales order detail records must be in a specific order: summary, tax, line items, then freight (each record is marked by code comments). At the end, for the audit trail, a call must be made to cls_SupFunctions-fn_AddAuditTrailRecord. This will allow the Data Mirror to see the transactions. The 5th argument in cls_SupFunctions.fn_AddAuditTrailRecord is the [window name]; no variations are allowed in the [window name]. In the example "Sales Orders" is the exact [window name] used by Peachtree. *If there are any extra spaces or spelling changes, the audit record will be ignored by the Data Mirror. The data will be imported, just not mirrored.*

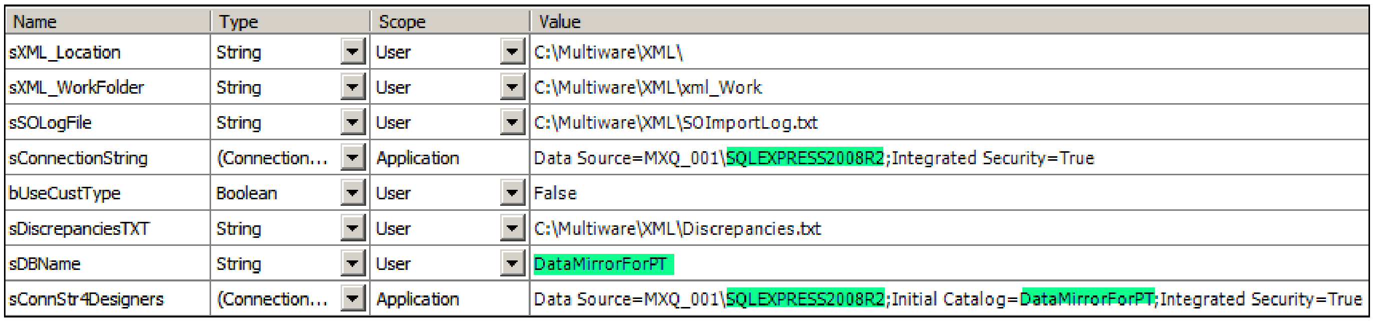
Example Window Names
"Credit Memos"
"General Journal Entry"
"Inventory Adjustments"
"Quotes"
"Payments"
"Purchase Orders"
"Receipts"
"Sales/Invoicing"
"Sales Orders"
"Vendor Credit Memos"

For each import an end transaction completes the import. If any part of the import fails, the transaction is aborted. This ensures that Peachtree does NOT get partial imports.

All the needed support functions are in the class cls_SupportFunctions.vb. Required *structures & globals* are stored in the standard module std_Globals.vb. Lookups are all in the class cls_Lookups.vb.

For this example, I did not import customer records but all the plumbing is in cls_Customers.vb. If you need to handle new customers or updates to customer records, see the class cls_Customers.vb. All the company functions are in the class cls_CompanyInfo.vb.

All the settings I used are self-explanatory and set in the project properties.



Name	Type	Scope	Value
sXML_Location	String	User	C:\Multiware\XML\
sXML_WorkFolder	String	User	C:\Multiware\XML\xml_Work
sSOLogFile	String	User	C:\Multiware\XML\SOImportLog.txt
sConnectionString	(Connection...	Application	Data Source=MXQ_001\SQLEXPRESS2008R2;Integrated Security=True
bUseCustType	Boolean	User	False
sDiscrepanciesTXT	String	User	C:\Multiware\XML\Discrepancies.txt
sDBName	String	User	DataMirrorForPT
sConnStr4Designers	(Connection...	Application	Data Source=MXQ_001\SQLEXPRESS2008R2;Initial Catalog=DataMirrorForPT;Integrated Security=True

- **Location of xml sales order files:** XML files in this folder are iterated for processing
- **Work folder for processed xml files:** xml files are moved here for actual processing
- **Sales order import log:** Lists the SO# applied to the imported SO with totals
- **SQL Connection string:** Connection string without the database name
    - Specify SQL Server instance name.
- If you use Customer Type, set True: Use if there is special handling for customer types
- **Discrepancies Log:** List any discrepancies found during data validation
- **Database Name:** Used to specify the name of the SQL Server database
- sConnStr4Designers is used by designers which need a connection string. You probably won't need to touch it. If you are going to modify the dataset in the designer, specify your SQL instance & database name.

With the SQL connection string & the database name you can target multiple Peachtree companies on multiple servers. The target Peachtree company files are selected from the database specified by the SQL connection string & the database name.

Be very careful. The power in this example should be self-evident. With just a little carelessness you can make a real mess. We recommend that you use backups extensively. We also recommend that you *do not attempt imports unless you are a journeyman or journey-woman software engineer.* Your Peachtree Imports Using the Data Mirror subscription entitles you to a ½ hour consult. More than that is billed hourly.

Consider having one of our senior software engineers build your application. Send customer support a requirements specification & we will provide an estimate.