QTB VS QTBPRO DATA FILES

INTRODUCTION

The information provided here is for those users who want to have a better understanding of the backend database used to store your data.

Quick Trial Balance (QTB) data files are Microsoft's Visual FoxPro, Version 9, DBF files. While there are a number of advantages to using this format, over the years, many of those advantages have turned into disadvantages. For example, in the past the DBF file format was supported by a number of applications that would facilitate accessing or transferring the data easily. However, as Microsoft continued to make the DBF file format it used proprietary, fewer and fewer companies continued to support it. Even Microsoft's own Excel program will not open a Visual FoxPro, Version 9, DBF file.

When the decision was made to re-architect QTB, it became clear that we would also have to change the format of the data files, since continued use of the DBF format had become a hindrance rather than an advantage.

In the end, we decided that we would use the SQLITE database for our file storage needs. It was lightweight (in database terms), did not require any additional installations of drivers or support files, and widely supported.

If you do a Google search for "sqlite managers", the results will list a host of database managers that can be used to create, read, and write to SQLITE database files.

CONVERTING QTB FILES TO QTBPRO

If there is a downside to changing the data file storage format, it's the conversion process. This is a one-time conversion to get the data in the existing DBF formats to the new SQLITE format.

The conversion process is not only fast, but very simple. QTBPRO allows you to select a QTB data file (using the Open button on the Home screen toolbar) and convert it to the QTBPRO format. The conversion is well tested so this should be a minor inconvenience and once done, should never have to be done again.

There is also an option in the Housekeeping menu that will convert all your data files at once. While this may take more time – this really depends on the number of files, number of books, and number of accounts – it does run without user intervention other than selecting the folder where your data files reside.

DATA FILES – CURRENT YEAR

QTB files are composed of a number of separate files for each client that contains the various information you can enter. There are two parts to the file names.

Prefix – The prefix for each file that comprises a client set is the same for all files. For example, if you assign a file name of ABC, then the prefix for all files in the client set will be ABC.

Extension – The extension of the files designate what data is contained in each file. For example, using ABC as the prefix of the file name, a file named ABC.B1 contains the client information, ABC.B2 contains the account information.

There are many files comprised in a client set and we won't list them all here.

In QTBPRO, there is only ONE file in a client set. So if the file name is ABC, then the file containing all client data is named ABC.QTB

So after converting your data, there will be ONE file for each client set of files that are converted.

DATA FILES – PRIOR YEAR

Prior year files are created whenever you perform a reset for a new year. Prior year files have a prefix and an extension just like current year files, but unlike current year files, there is only one file for each client and each year. For example, if the file name assigned to a client file is ABC and the prior year is 12/31/2013, then the prior year file will be named as follows:

ABC_31-DEC-2013__.QBZ

However, each prior year file is actually a compressed ZIP file containing all the files in a client set. Before opening a prior year file, the QBZ file needs to be uncompressed and then selected.

Prior year files will also be converted when you select a client file.

In QTBPRO, prior year files are no longer compressed and can be selected just like a current file.

Shown below are the data files created by AK. Assume the file name is ABC. Note that the extension of the file identifies what is actually stored in the file.