BTR2SQL

Migrating a Multi-company Btrieve Application Using BTR2SQL

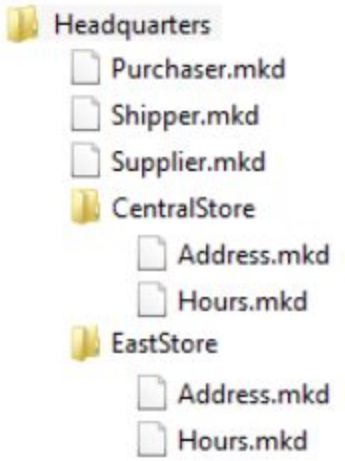
Introduction

This document explains how to use Mertech's BTR2SQL product to migrate a multi-company Btrieve/Pervasive.SQL application to a SQL backend.

BTR2SQL's migration utility converts applications running on Btrieve data files to a SQL server backend, without compromising applications' speed and stability or requiring any source code changes. BTR2SQL does this by reading the application's DDFs to create a SQL database and then storing the SQL table schema in .INT files. The BTR2SQL driver then uses these .INT files to connect to and open the converted SQL tables.

Applying BTR2SQL to Multi-company Migrations

Our customers commonly migrate applications that use a multi-directory structure, as shown in the example below. These applications store general information in a top directory and region-specific information in lower directories.



However, on SQL, different tables with the same names (such as the multiple Address and Hours tables pictured above) collide if stored in the same database. Given this problem, the question arises: What is the best way to migrate a multi-directory Btrieve database to a SQL backend?

There are several ways to successfully migrate multi-directory applications, including:

- Attaching a custom prefix or postfix to your table names during migration.
- Migrating tables to multiple databases.

- Migrating tables to multiple servers.
- Migrating tables to multiple schemas.

We'll examine each of these options, including their pros and cons, later in this paper. But before we do, you need to know a little more about BTR2SQL.

An Overview of BTR2SQL

We'll examine your four multi-company migration options in the next section. But before we can do that, we need to discuss some details of how BTR2SQL works, so your options make sense.

The BTR2SQL Migration Utility reads your DDF files and uses them to map your existing tables' structure. The utility then uses this information to create new, SQL-compliant tables on your SQL server, copy your existing data to the new tables, and store the table structure in a series of intermediate (.INT) files. The BTR2SQL database drivers later use these .INT files to connect to and open the correct converted SQL table.

The BTR2SQL drivers use three files to ensure your Btrieve application operates as expected: .INT files, mds.ini, and mds_global.ini. Let's talk about each briefly.

.INT Files

An .INT file contains all the information BTR2SQL needs to connect to, open, or recreate a converted SQL table. A partial .INT file is shown below.

```
CREATED Sat Apr 19 18:05:00 2014 by SQLBTR 5.1.6402.1
DRIVER_NAME SQL_BTR
SERVER_NAME GATEWAY\sqlexpress
DATABASE_SPACE_NAME tempdb
TABLE_NAME CLASS
SCHEMA_NAME dbo
FILEGROUP_TABLE
FILEGROUP_INDEX
FILEGROUP_TEXT
LOGICAL_RECORD_LENGTH 14
PAGE_SIZE 4096
FILE_FLAGS 512
FILE_DDF_FLAGS 64
PERMANENT_INT NO
MAX_ROWS_TO_QUERY 100
LOCAL_CACHE YES
NUMBER_DF_FIELDS 11
PRIMARY_INDEX 1
TRANSLATE_OEM_TO_ANSI 0
IGNORE_NULL_VALUES 1
TRIM_STRING_FIELDS 1

FIELD_NUMBER 1
FIELD_NAME ID
FIELD_NATIVE_TYPE 15
FIELD_NATIVE_LENGTH 4
FIELD_NATIVE_OFFSET 0
FIELD_NATIVE_FLAG 0
FIELD_DEFAULT_VALUE 0
FIELD_AUTOINCREMENT_TYPE YES
FIELD_INDEX 1

FIELD_NUMBER 2
FIELD_NAME NAME
FIELD_NATIVE_TYPE 0
FIELD_NATIVE_LENGTH 7
FIELD_NATIVE_OFFSET 4
FIELD_NATIVE_FLAG 1
FIELD_INDEX 2

INDEX_NUMBER 1
INDEX_NUMBER_SEGMENTS 1
INDEX_SEGMENT_FIELD 1
INDEX_SEGMENT_FLAG 258

INDEX_NUMBER 2
INDEX_NUMBER_SEGMENTS 2
INDEX_SEGMENT_FIELD 2
INDEX_SEGMENT_CASE IGNORED
INDEX_SEGMENT_FLAG 1298
INDEX_SEGMENT_FIELD 3
INDEX_SEGMENT_CASE IGNORED
INDEX_SEGMENT_FLAG 1282
```

The .INT file's general settings, which you configure while performing your migration (using the BTR2SQL Migration Utility), are listed first. Field and index definitions follow (FIELD_NUMBER 1..n, INDEX_NUMBER 1..n) . These definitions provide BTR2SQL with the information needed to access the corresponding SQL table on your SQL server at runtime.

A couple additional notes about .INT files, before we move on:

- BTR2SQL creates differently constructed .INT files for different types of SQL servers, and for different versions of the same server. **As such, you should not attempt to re-use or duplicate .INT files when switching from one SQL server to another. The same holds true if you plan to target multiple, distinct (different make, model, or version) SQL servers.** If you need to access your application from two distinct servers, you need to perform the BTR2SQL migration twice, once targeting each server.
- Later, we'll walk you through modifying an .INT file's PERMANENT_INT setting, to aid in replicating your application's multi-directory structure on SQL. **You should not make any other changes to an .INT file, outside of modifying the PERMANENT_INT setting as instructed.** Any other changes could result in unpredictable application errors.

Mds.ini

BTR2SQL uses a mds.ini file to store and retrieve your server and database name, server login information, table prefix and postfix values, and, crucially, the location of your .INT files. The mds.ini file uses a number of optional tokens to store this information:

Token	Contains	Allows You To
INT-Folder	The path to your .INT files.	Consolidate your .INT files in one directory.
Server	Your server name.	Identify one target backend for a group of files, instead of storing that information in each corresponding .INT file or passing it in at runtime.
Database	Your database name.	
Schema	Your schema name. (This token is not used in Oracle.)	
UseTrustedConnection	A setting that allows you to rely on Windows authentication when accessing your SQL server. This can be set to Yes or No, and is set to No by default. If this setting is set to Yes, the User and Password tokens described below are ignored.	Store your SQL server login information, so users are not prompted to log in to your server when accessing your application.

User	The username used to log in to your SQL server.	
Password	The password used to log in to your SQL server. This is a 128-bit encrypted value obtained with the command line tool MdsEncryptPassword, which can be found in the BTR2SQL bin folder.	
Table-prefix	Text to prepend to table names during migration.	Create unique table names when migrating multiple directories with the same file names to your target server.
Table-postfix	Text to append to table names during migration.	

These tokens will help us perform the multi-company migrations outlined below, either by migrating files to different databases, servers, or schemas (using the INT-Folder, Server, Database, and Schema tokens), or by modifying similar table names to use unique prefixes and postfixes (using the INT-Folder, Table-prefix, and Table-postfix tokens).

A partial mds.ini file is shown below. You can view a full sample mds.ini file by following the path <programfiles>\Mertech Data Systems\DB Drivers\Btrieve\bin. Note that entries listed under a specific heading (for example, [SQL_BTR] and [ORA_BTR] below) are used only by that database and driver.

```
[SQL_BTR]
INT-Folder=..\IntFiles_MS
Server=sqlsrv\sql2008
Database=testdb
Schema=dbo
UseTrustedConnection=no
User=mydomain\me
Password=4c8fe10aad3
Table-Prefix=store1_
Table-Postfix=

[ORA_BTR]
INT-Folder=..\IntFiles_ORA
Server=orclsrv
User=me
Password=4c8fe10aad3
Table-Prefix=store1_
Table-Postfix=
```

Mds_global.ini

While mds.ini sets information specific to each data folder, mds_global.ini contains global settings that control things like tracing, locking, and application performance.

A partial mds_global.ini file is shown below. You can view a full mds_global.ini by following the path <programfiles>\Mertech Data Systems\DB Drivers\Btrieve\bin. Note that, again, entries listed under a specific heading (for example, [SQL_BTR] and [MYS_BTR] below) are used only by that database and driver. If a value is left empty, the driver's default value is used.

```
[SQL_BTR]
;Tracing
TRACE_ON=c:\trace\sql_btr.tra
TRACE_LEVEL=0
TRACE_FLUSH=no
;Locking
SERVER_SIDE_LOCKING=yes
SQL_LOCKS=no

[MYS_BTR]
;Location of the debug trace file
TRACE_ON = c:\trace\mys_btr.tra
TRACE_LEVEL = 0
TRACE_FLUSH=no
;Locking
SERVER_SIDE_LOCKING=yes
SQL_LOCKS=no
```

Note that where you ultimately store the mds_global.ini file is up to you. BTR2SQL uses the first mds_global.ini file found on the usual System Path. We recommend placing the mds_global.ini file in the same location as your BTR2SQL access DLL and license (.CFG).

Multi-company Migration Options

Now that you know a little more about BTR2SQL, and some of the BTR2SQL-created files involved in a multi-company BTR2SQL migration, we're ready to start talking about your migration options. Below, we'll briefly discuss each option, laying out the pros and cons of each.

After we've finished, we'll provide you with links to a detailed example of each option, so you can get started on your own multi-company migration.

Migrating Each Folder Using a Different Table Prefix or Postfix

If you choose this option, you'll be adding a unique prefix or postfix to each table you migrate, to note which are used by which directory. Then, you'll be creating and updating mds.ini files to match this structure, as well as migrating certain .INT files, so your application functions as expected.

PROS:

- This option is the easiest to implement, as there's less administrative overhead and maintenance to handle after your migration.
- This option is flexible and works the same across all supported SQL backends.

CONS:

- If you have a lot of directories, and tables within those directories, your table list can become very large after adding prefixes/postfixes.
- If you select this option, it will be harder to assign users directory access than it would be if you select one of the other options.
- If you're using Oracle, the 30 character limit on table identifiers can cause issues when attempting to assign prefixes/postfixes.

Migrating Each Folder to a Separate Server or Database

If you choose this option, you'll be migrating each of your company-specific directories to a separate server or database, so similarly named tables don't conflict with each other.

PROS:

- This option allows you to easily load balance your application by distributing data across multiple computers, CPUs, hard disks, etc.
- If you select this option, you can assign user credentials based on each server or database.

CONS:

- Your transactions won't cross multiple "customers," since the driver will have a unique connection to the database for each data folder.
- This option adds the administrative overhead of backing up and maintaining additional hardware and software.

Migrating Each Folder to a Separate Schema (Users on Oracle)

If you choose this option, you'll be migrating each of your company-specific directories to separate schemas (users in Oracle), in much the same way you'd be migrating to multiple databases or servers if you chose the option above. However, migrating to separate schemas on Oracle allows you to more granularly control user access.

PROS:

- There is a clean distinction between "sets" of data.
- You can assign different user credentials to each schema, more granularly controlling user access.
- Using schema-based migration, you can maintain relatively small table lists in each schema.

CONS:

- The additional schemas you create using this option add additional administrative overhead.

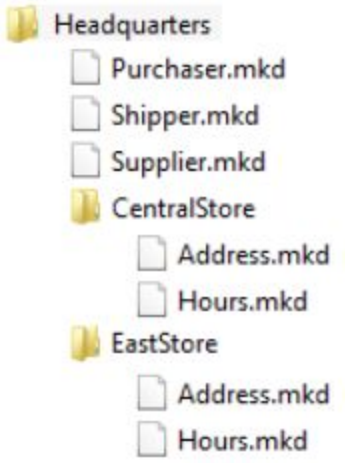
Again, adding a table prefix or postfix during migration is the easiest and most-used option. However, if you have a lot of tables in each of your folders, your table list could become overwhelming after migrating using prefixes/postfixes. For example, if you start with 300 data files, and you have four directories, you'll end up with 1200 tables.

If the prefix/postfix option would create an unmanageable number of tables for your application, or you want to assign user access at a more granular level using one of the other options, you might not want to use prefixes/postfixes. Otherwise, we highly recommend migrating your multi-directory application using table prefixes/postfixes.

Now, you're ready to see what an example migration using each option entails. If you already know which option you'd like to examine in more detail, use the following links to jump to that option:

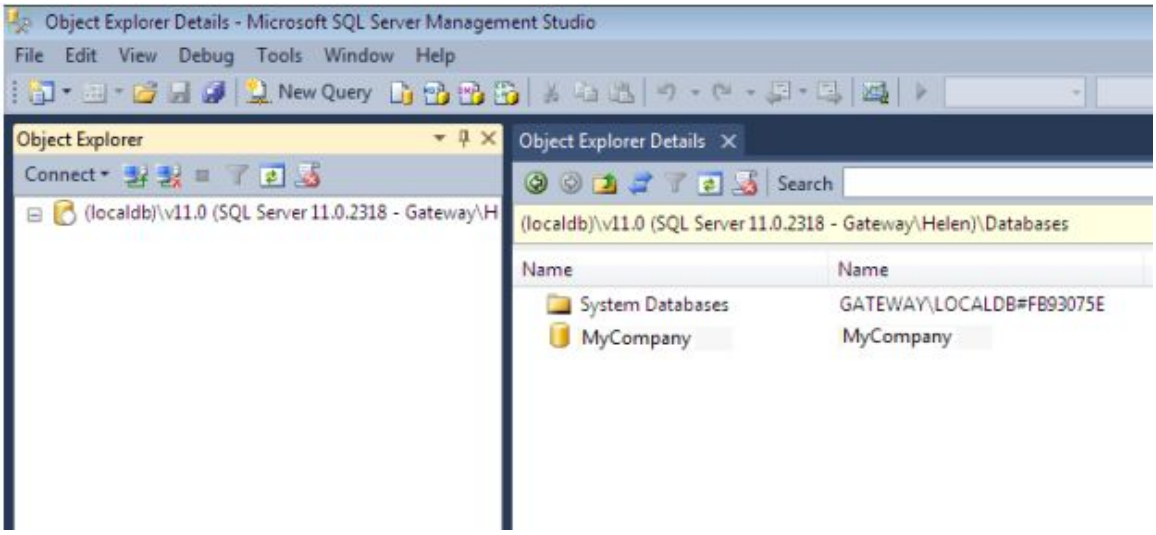
Migrating Using Table Prefixes or Postfixes

This example migration uses the Btrieve database layout shown below and assumes the layout of both the CentralStore and EastStore directories' Address and Hours tables is the same. This example also uses Microsoft SQL Server as the target SQL server.



Create the Required Database on Your SQL Backend

Because this migration will use table prefixes to differentiate between directories, you should create only one database on your SQL backend. We'll migrate all your tables to this database.



Prepare Your Mds.ini Files

To streamline future maintenance and avoid users having to manually log in to your SQL server every time they access your application, we will enter the target backend and login information in the mds.ini file.

We will also modify the INT-Folder setting so we can consolidate your .INT files after migration. We'll modify this token now, while we're here, but we'll also place a semicolon (;) in front of it, so the token is not used during the migration. We'll remove the semicolon later in the process.

Finally, for this migration, we'll use the Table-prefix token to add unique table prefixes to the tables in each directory. This prefix will be used during migration and at runtime.

1. In your Headquarters directory, create an mds.ini.

```
[SQL_BTR]
;INT-Folder = .\intfiles_headquarters
Server = <servername>
Database = MyCompany
User = <username>
Password = <encryptedpw>
Table-prefix = HQ_
```

2. In your CentralStore directory, create an mds.ini.

```
[SQL_BTR]
;INT-Folder = ..\intfiles_stores
Server = <servername>
Database = MyCompany
User = <username>
Password = <encryptedpw>
Table-prefix = Central_
```

3. In your EastStore directory, create an mds.ini.

```
[SQL_BTR]
;INT-Folder = ..\intfiles_stores
Server = <servername>
Database = MyCompany
User = <username>
Password = <encryptedpw>
Table-prefix = East_
```

Migrate All Your Btrieve Tables to the MyCompany Database

We'll now perform the BTR2SQL migration, migrating your files from your Btrieve database to SQL.

1. From the Windows Start menu, select Mertech's ISDBC Drivers for Btrieve > Migration Utility. The Login window appears.
2. In the Database field, select the MyCompany database. Then, select the Login button.
3. Select File > Select File.DDF and open the Headquarters DDF file. A dialog box appears.
4. From the dialog box, select the files you want to migrate (Purchaser, Shipper, Supplier).
5. Right-click to show the pop-up menu and select Convert to MSSQL (or your server). The Convert Data Files dialog box appears.
6. Be sure to NOT de-select the Get Server Name from Login, Get Schema from Login, and Get Database Name from Login checkboxes. These checkboxes must remain selected so information about your target backend is not stored in your .INT files.
7. Select OK. The Migration Utility runs, using the table prefix you added in your Headquarters mds.ini file to prepend HQ_ to the Purchaser, Shipper, and Supplier tables.
8. Select Reports > Migration Report. Examine the migrate.txt file for errors.
9. Select File > Select File.DDF and open the CentralStore DDF file. A dialog box appears.
10. From the dialog box, select the files you want to migrate (Address, Hours).
11. Right-click and select Convert to MSSQL (or your server).
12. Be sure to NOT de-select the Get Server Name from Login, Get Schema from Login, and Get Database Name from Login checkboxes. These checkboxes must remain selected so information about your target backend is not stored in your .INT files.
13. Select OK. The Migration Utility runs, using the table prefix you added in your CentralStore mds.ini file to prepend Central_ to the Address and Hours tables.
14. Select Reports > Migration Report. Examine the migrate.txt file for errors.

15. Select File > Select File.DDF and open the EastStore DDF file. A dialog box appears.
16. From the dialog box, select the files you want to migrate (Address, Hours).
17. Right-click and choose Convert to MSSQL (or your server).
18. Be sure to NOT de-select the Get Server Name from Login, Get Schema from Login, and Get Database Name from Login checkboxes. These checkboxes must remain selected so information about your target backend is not stored in your .INT files.
19. Select OK. The Migration Utility runs, using the table prefix you added in your EastStore mds.ini file to prepend East_ to the Address and Hours tables.
20. Select Reports > Migration Report. Examine the migrate.txt file for any errors.

Consolidate the .INT Files

Since we did not store information about the target database in the .INT files, the .INT files in the CentralStore and EastStore directories are identical. To simplify maintenance down the road, we'll keep only one copy of these .INT files. Likewise, to further facilitate maintenance, we can move the Headquarters .INT files to a dedicated .INT directory.

1. Create a directory in the Headquarters folder to hold the Headquarters .INT files: Headquarters\intfiles_headquarters.
2. Move the Purchaser.INT, Shipper.INT, and Supplier.INT files to Headquarters\intfiles_headquarters.
3. Edit the Purchaser.INT, Shipper.INT, and Supplier.INT files to change the PERMANENT_INT setting to YES.
4. Create a directory in the Headquarters folder to hold the regional .INT files: Headquarters\intfiles_stores.
5. Copy the Address.INT and Hours.INT files from either the CentralStore or EastStore directory to Headquarters\intfiles_stores.
6. Edit the Address.INT and Hours.INF files to change the PERMANENT_INT setting to YES.
7. Delete the Address.INT and Hours.INT files from the CentralStore and EastStore directories.

Finalize the Mds.ini Files

To finalize your mds.ini files, you must uncomment the INT-Folder token in each file by removing the semicolon (;) you added earlier.

Run the Migration Validator

After performing your migration and finalizing your files, you must run the Migration Validator to verify that all files were transferred correctly from Btrieve to SQL.

To run the Migration Validator, go to the Windows Start menu, then select Mertech's ISDBC Drivers for Btrieve > Migration Validator. Migration Validator is a command-line utility, so the Command Prompt will open.

Run the Migration Validator script by typing the following in Command Prompt, replacing the <database location> and <Mertech dll> tags with the path to your database and BTR2SQL DLL, respectively:

```
MigrationValidator <database location> [-dll <Mertech dll>] [-table
<tablelist>] [-maxRows <n>] [/varOnly] [/debug]
```

For our example migration, we'd run the following script:

```
MigrationValidator "c:\MyApp\Headquarters" -dll "c:\Program Files
(x86)\Mertech Data Systems\DB Drivers\Btrieve\bin\sql btr.dll"
```

If the Migration Validator finds a file mismatch, a message appears detailing the issue.

Substitute the Btrieve Access DLLs

Finally, you must replace the wbtrv32.dll and/or w3btrv7.dll that currently resides in your PVSW\BIN folder with the Mertech drives found in the <programfiles>\Mertech Data Systems\DB Drivers\Btrieve\deploy\<subfolder> directory. This last step will ensure Windows correctly opens your application.

IMPORTANT: Be sure to save a copy of your original Pervasive DLLs before you replace them. Many Pervasive tools require these DLLs. For instance, if you need to correct your table definitions later, you must restore your original Pervasive DLLs to do so.

Post-Migration Directory and Database Structures

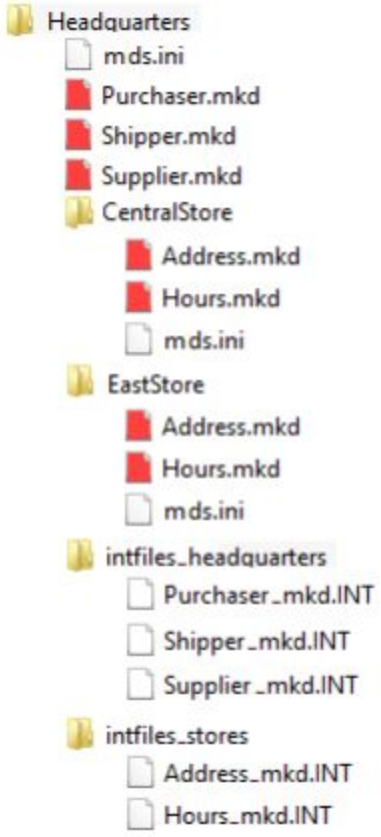
Your directory and database structures should now look like below.

Directory Structure

Notice that, in the picture below:

- The .INT files are in separate dedicated folders, one for headquarters and one for each regional store.

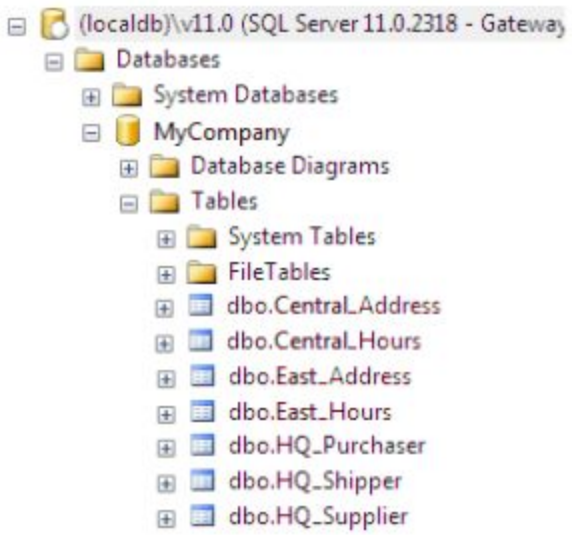
- Data files, marked in red, are no longer needed.
- Each data folder has a mds.ini file that contains the location of the .INT files, target database, login information, and table-prefix.



Database Structure

Notice that, in the picture below:

- All tables have been migrated to one database.
- Tables from the headquarters and each regional store are identified by a prefix.



What Happens at Runtime

When your application loads, the BTR2SQL driver initializes and reads global settings from the mds_global.ini file. These settings toggle tracing and set other performance and locking parameters that affect the entire application.

When your application issues a B_OPEN command to open a file (for example, CentralStore\Address.mkd), the BTR2SQL driver:

1. Looks for the associated .INT file (Address_mkd.int) in the CentralStore directory.
2. When the driver can't find that file (as expected, based on our migration strategy above), the driver looks for an mds.ini file.
3. The driver finds and opens the mds.ini file, which contains:
    - The .INT file location: INT-folder = ..\intfiles_stores.
    - Information on the target backend: Server = <servername> and Database = MyCompany.
    - Login information: User = <username> and Password = <encryptedpw>.
    - Your table prefix: Table-prefix = Central_.
4. The .INT file (CentralStore\..\intfiles_stores\Address_mkd.INT) defines the table name as Address. The driver uses this, plus the Table-prefix in the .INI file, to open the Central_Address table in the MyCompany database.

Note: If you store both your .INT and mds.ini files in the data directory, settings in the .INT file and mds.ini file are merged. If a setting appears in both files (for example, the Server and Database tokens), the values in the .INT file take precedence. Only one mds.ini file is allowed per data directory.

Creating a New Company

As time goes by and your company grows, you'll likely want to add new stores to your application. To do so, you must:

1. Create the new company folder (for example, WestStore).
2. Create an mds.ini file in that folder.

```
[SQL_BTR]
INT-Folder = ..\intfiles_stores
Server = <servername>
Database = MyCompany
User = <username>
Password = <encryptedpw>
Table-prefix = West_
```

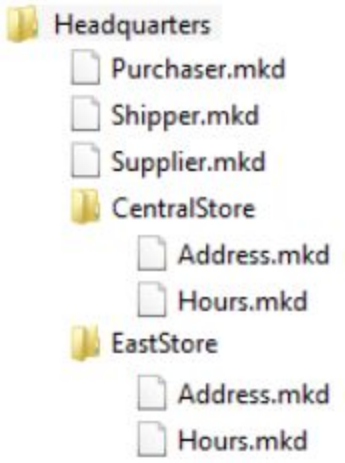
3. Create the new tables using B_CREATE commands.

```
B_CREATE(WestStore\Hours.mkd)
B_CREATE(WestStore\Address.mkd)
```

The driver reads the mds.ini file in the WestStore folder and uses the INT_Folder token to locate the .INT file. The driver uses the table template in the .INT file to create the tables in the WestStore database. The driver prepends the *West_* prefix when creating the new tables in the MyCompany database.

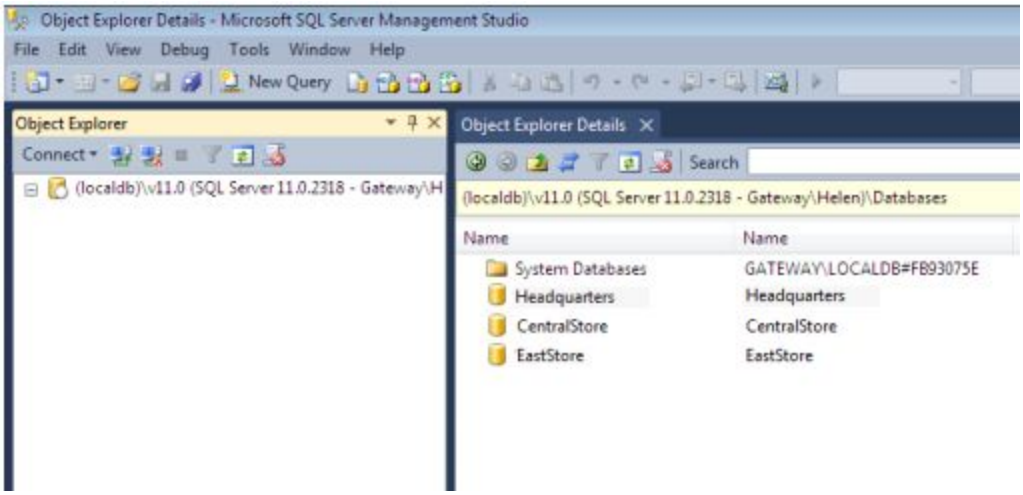
Migrating to Separate Databases

This example migration uses the Btrieve database layout shown below and assumes the layout of both the CentralStore and EastStore directories' Address and Hours tables is the same. This example also uses Microsoft SQL Server as the target SQL server.



Create the Required Databases on Your SQL Backend

To avoid duplicate table names on your target server, we'll migrate each of the folders above to their own database. So, you must create separate databases for the Headquarters, CentralStore, and EastStore folders (as shown below).



Migrate Each Folder to Its Own Database

We'll now perform the BTR2SQL migration, migrating your files from your Btrieve database to SQL. As you perform the migration, you'll migrate each directory to its own database.

1. From the Windows Start menu, select Mertech's ISDBC Drivers for Btrieve > Migration Utility.
2. In the Database field, select the Headquarters database. Then, select the Login button.
3. Select File > Select File.DDF and open the Headquarters DDF file. A dialog box appears.
4. From the dialog box, select the files you want to migrate (Purchaser, Shipper, Supplier).
5. Right-click and select Convert to MSSQL (or your server). The Convert Data Files dialog box appears.
6. Be sure to NOT de-select the Get Server Name from Login, Get Schema from Login, and Get Database Name from Login checkboxes. These checkboxes must remain selected so information about your target backend is not stored in your .INT files.
7. Select OK. The Migration Utility runs, migrating files from the Headquarters folder to your new Headquarters database.
8. Select Reports > Migration Report. Examine the migrate.txt file for errors.

9. Select File > Select Database. When the Select Database window appears, select the CentralStore database from the Database field. Then, select OK.
10. Select File > Select File.DDF and open the CentralStore DDF file. A dialog box appears.
11. From the dialog box, select the files you want to migrate (Address, Hours).
12. Right-click and select Convert to MSSQL (or your server).
13. Be sure to NOT de-select the Get Server Name from Login, Get Schema from Login, and Get Database Name from Login checkboxes. These checkboxes must remain selected so information about your target backend is not stored in your .INT files.
14. Select OK. The Migration Utility runs, migrating files from the CentralStore folder to your new CentralStore database.
15. Select Reports > Migration Report. Examine the migrate.txt file for errors.
16. Select File > Select Database. When the Select Database window appears, select the EastStore database from the Database field. Then, select OK.
17. Select File > Select File.DDF and open the EastStore DDF file. A dialog box appears.
18. From the dialog box, select the files you want to migrate (Address, Hours).
19. Right-click and select Convert to MSSQL (or your server).
20. Be sure to NOT de-select the Get Server Name from Login, Get Schema from Login, and Get Database Name from Login checkboxes. These checkboxes must remain selected so information about your target backend is not stored in your .INT files.
21. Select OK. The migration utility runs, migrating files from the EastStore folder to your new EastStore database.
22. Select Reports > Migration Report. Examine the migrate.txt file for any errors.

Consolidate the .INT Files

Since we did not store information about the target database in the .INT files, the .INT files in the CentralStore and EastStore directories are identical. To simplify maintenance down the road, we'll keep only one copy of these .INT files. Likewise, to facilitate maintenance down the road, we can move the Headquarters .INT files to a dedicated .INT directory.

1. Create a directory in the Headquarters folder to hold the headquarters INT files: Headquarters\intfiles_headquarters.
2. Move the Purchaser.INT, Shipper.INT, and Supplier.INT files to Headquarters\intfiles_headquarters.
3. Edit the Purchaser.INT, Shipper.INT, and Supplier.INT files to change the

PERMANENT_INT setting to YES.

4. Create a directory in the Headquarters folder to hold the regional .INT files: Headquarters\intfiles_stores.
5. Copy the Address.INT and Hours.INT files from either the CentralStore or EastStore directory to Headquarters\intfiles_stores.
6. Edit the Address.INT and Hours.INT files to change the PERMANENT_INT setting to YES.
7. Delete the Address.INT and Hours.INT files from the CentralStore and EastStore directories.

Prepare Your Mds.ini Files

To streamline future maintenance and avoid users having to manually log in to your SQL server every time they access your application, we will enter the target backend and login information in the mds.ini file.

We will also modify the INT-Folder setting to reflect the .INT file consolidation we performed above.

1. In your Headquarters directory, create an mds.ini.

```
[SQL_BTR]
INT-Folder = .\intfiles_headquarters
Server = <servername>
Database = Headquarters
User = <username>
Password = <encryptedpw>
```

2. In your CentralStore directory, create an mds.ini.

```
[SQL_BTR]
INT-Folder = ..\intfiles_stores
Server = <servername>
Database = CentralStore
User = <username>
Password = <encryptedpw>
```

3. In your EastStore directory, create an mds.ini.

```
[SQL_BTR]
INT-Folder = ..\intfiles_stores
Server = <servername>
Database = EastStore
User = <username>
Password = <encryptedpw>
```

Run the Migration Validator

After performing your migration and finalizing your files, you must run the Migration Validator to verify that all files were transferred correctly from Btrieve to SQL.

To run the Migration Validator, go to the Windows Start menu, then select Mertech's ISDBC Drivers for Btrieve > Migration Validator. Migration Validator is a command-line utility, so the Command Prompt will open.

Run the Migration Validator script by typing the following in Command Prompt, replacing the <database location> and <Mertech dll> tags with the path to your database and BTR2SQL DLL, respectively:

```
MigrationValidator <database location> [-dll <Mertech dll>] [-table
<tablelist>] [-maxRows <n>] [/varOnly] [/debug]
```

For our example migration, we'd run the following script:

```
MigrationValidator "c:\MyApp\Headquarters" -dll "c:\Program Files
(x86)\Mertech Data Systems\DB Drivers\Btrieve\bin\sql btr.dll"
```

If the Migration Validator finds a file mismatch, a message appears detailing the issue.

Substitute the Btrieve Access DLLs

Finally, you must replace the wbtrv32.dll and/or w3btrv7.dll that currently resides in your PVSW\BIN folder with the Mertech drives found in the <programfiles>\Mertech Data Systems\DB Drivers\Btrieve\deploy\<subfolder> directory. This last step will ensure Windows correctly opens your application.

IMPORTANT: Be sure to save a copy of your original Pervasive DLLs before you replace them. Many Pervasive tools require these DLLs. For instance, if you need to correct your table definitions later, you must restore your original Pervasive DLLs to do so.

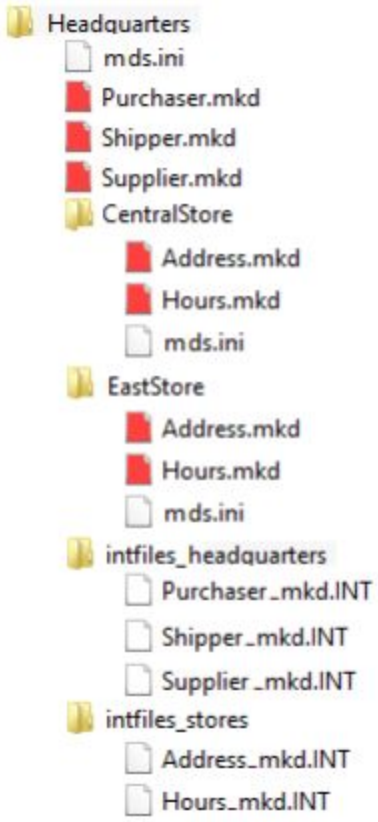
Post-Migration Directory and Database Structures

Your directory and database structures should now look like below.

Directory Structure

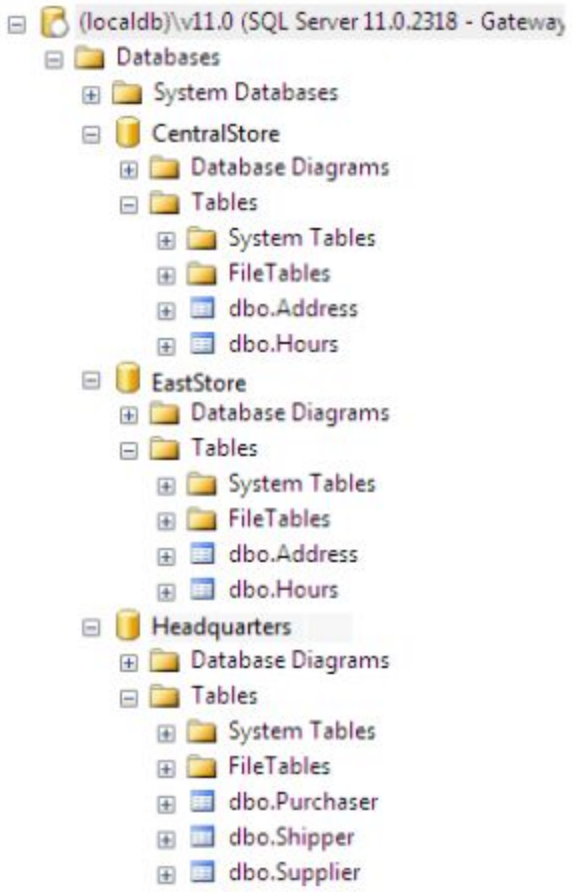
Notice that, in the picture below:

- The .INT files are in dedicated folders, one for your headquarters and one for each regional store.
- The data files, marked in red, are no longer needed.
- Each data folder contains a mds.ini file that contains the location of the .INT files, information on your target database, and login information.



Database Structure

Notice that, in the picture below, a separate database exists for your headquarters and for each regional store.



What Happens at Runtime

When your application loads, the BTR2SQL driver initializes and reads global settings from the mds_golbal.ini file. These settings toggle tracing and set other performance and locking parameters that affect the entire application.

When your application issues a B_OPEN command to open a file (for example, CentralStore\Address.mkd), the BTR2SQL driver:

1. Looks for the associated .INT file (Address_mkd.int) in the CentralStore directory.
2. When the driver can't find that file (as expected, based on our migration strategy above), the driver looks for an mds.ini file.
3. The driver finds and opens the mds.ini file, which contains:
    - The .INT file location: INT-folder = ..\intfiles_stores
    - Information on the target backend: Server = <servername> and Database = CentralStore
    - Login information: User = <username> and Password = <encryptedpw>
4. The BTR2SQL driver reads the corresponding .INT file

(CentralStore\..\intfiles_stores\Address_mkd.INT) and uses the table structure, target backend, and login information to open the Address table found within your CentralStore database.

Note: If you store both your .INT and mds.ini files in the data directory, settings in the .INT file and mds.ini file are merged. If a setting appears in both files (for example, the Server and Database tokens), the values in the .INT file take precedence. Only one mds.ini file is allowed per data directory.

Creating a New Company

As time goes by and your company grows, you'll likely want to add new stores to your application. To do so, you must:

1. Create the new company folder (for example, WestStore).
2. Create an mds.ini file in that folder.

```
[SQL_BTR]
INT-Folder = ..\intfiles_stores
Server = <servername>
Database = WestStore
User = <username>
Password = <encryptedpw>
```

3. Create a new SQL database (for example, WestStore). To do so, either use SQL modeling tools, or call the BTRV API_B_SQL_EXECUTE executable within your application to execute a CREATE DATABASE command. (You can find more information on the BTRV API_B_SQL_EXECUTE executable in the BTR2SQL SDK.)
4. Create the new tables using B_CREATE commands.

```
B_CREATE(WestStore\Hours.mkd)
B_CREATE(WestStore\Address.mkd)
```

The driver reads the mds.ini file in the WestStore folder and uses the INT_Folder token to locate the .INT file. The driver uses the table template in the .INT file to create the tables in the WestStore database.

Migrating to Separate Servers

The process used to migrate to multiple SQL servers is similar to the process used to migrate to multiple databases, which is described in detail in the Migrating to Separate Databases section. As such, the following instructions are condensed and include links to relevant, detailed instructions from the Migrating to Multiple Databases section where necessary.

1. Create the required servers on your SQL backend.
2. Migrate each folder to its own server, following the instructions in the

Migrate Each Folder to Its Own Database section. In steps 2, 9, and 16, select a different server, rather than a different database, for each folder.

3. Consolidate your .INT files.
4. Prepare your mds.ini files, specifying the correct database and a different server for each folder.

```
[SQL_BTR]
INT-Folder = .\intfiles_headquarters
Server = <servername>
Database = CorrectDatabase
User = <username>
Password = <encryptedpw>
```

5. Run the Migration Validator.
6. Substitute the Btrieve access DLLs.

Migrating to Separate Schemas

The process used to migrate to multiple schemas (users in Oracle) is similar to the process used to migrate to multiple databases, which is described in detail in the Migrating to Separate Databases section. As such, the following instructions are condensed and include links to relevant, detailed instructions from the Migrating to Multiple Databases section where necessary.

1. Create the required schemas/users on your SQL backend.
2. Migrate each folder to its own server, following the instructions in the Migrate Each Folder to Its Own Database section. In steps 2, 9, and 16, enter a different username, rather than a different database, for each folder. The entered user must have object creation rights.
3. Consolidate your .INT files.
4. Prepare your mds.ini files, specifying the correct database and a different schema and login for each folder.

```
[SQL_BTR]
INT-Folder = .\intfiles_headquarters
Server = <servername>
Database = CorrectDatabase
User = <username>
Password = <encryptedpw>
```

5. Run the Migration Validator.
6. Substitute the Btrieve access DLLs.

Contact Information

If you'd like to know more about Mertech's products, please visit our website, mertechdata.com, or contact us at:

Corporate Head Office

Mertech Data Systems, Inc.
18503 Pines Blvd. Suite 312
Pembroke Pines, FL 33029 USA
Tel: +1.954.585.9016
Fax: +1.866.228.1213
Email: sales@mertechdata.com

California Office

Mertech Data Systems, Inc.
7621 N. Del Mar Ave., Suite 101
Fresno, CA 93711 USA

Copyrights and Trademarks