Overview

This document describes the W32 utility functions that have been upgraded to work when the target files are mapped MySQL tables.

The Utility Functions

The utility functions that have been updated include:

Bcopy
Bcreate
Berase
Bfilestat
Bindinfo
Brename

In general, any time a <filename> is used as an argument to one of these functions, it refers to the mapped MySQL file name. For example, if the _cet_files_ table has a row that associates K:/CET/somefile with the MySQL table “custterm”, then “K:/CET/somefile” is what you would use as the argument in the utility function. The W32 runtime (RUNW32.EXE) will perform a lookup in _cet_files_ to determine what table is being referenced.

Please keep in mind that the filename used in OPEN statements and the utility functions is case-sensitive and must match exactly the entry in the “Name” field of the corresponding _cet_files_ row.

Because the functions generally don’t know what the file type is of the file name arguments, they perform operations in a specific order:

- If the filename matches that of a sequential file that exists, the operation is performed on that sequential file, if it makes sense to do so (e.g., Bindinfo only looks at indexed and MySQL files).

- Otherwise, If the filename matches that of a direct file that exists, the operation is performed on that direct file, if it makes sense to do so.

- Otherwise, If the filename matches that of a mapped MySQL file that is present in the _cet_files_ table, the operation is performed on that table and _cet_files_ row entry.

- Otherwise, If the filename matches that of am indexed file that exists, the operation is performed on that indexed file.

Bcopy

The W32 manual describes the Bcopy function in the following manner:

> The Bcopy function may be used to copy a file without having to leave the BASIC environment to perform an operation with a CSI statement. File names may be in either a DOS or THEOS format. If the destination file does not exist, it will be created with the same specifications as the source file. The syntax of the CALL statement is:
>
> **CALL Bcopy("*source-filename destination-filename*")**

The function has been modified to permit a mapped MySQL table for **both** arguments. Other caveats and restrictions include:

- Both arguments must be of the same file type: sequential, direct, indexed or MySQL
- The table copy is performed by duplicating the underlying source table exactly and creating the destination table, including all fields and populated data.
- If the destination table already exists, it will be dropped and recreated.

Bcreate

The W32 manual describes the Bcreate function in the following manner:

> Bcreate operates like the utility described in the previous chapter except that it is executed from within a BASIC program by using a CALL statement instead of a CSI. The syntax of the statement is:
>
> **CALL Bcreate("create-command line")**

For mapped MySQL file names, this function is limited to performing a "<filename> CLEAR". It is not possible to create a MySQL table because not enough information is available to do so (only the mapped filename is specified, not the underlying MySQL table).

The statement CALL Bcreate("K:/CET/somefile CLEAR") will delete all rows from the table represented by "K:/CET/somefile" and will have no effect on the _cet_files_ table.

Berase

The W32 manual describes the Berase function in the following manner:

> Berase is designed to erase a file and eliminate having to perform a CSI statement to execute the DOS DEL command. In either case, the file must be closed before it can be erased.
>
> The syntax of the statement is:

CALL Berase(“filename”)

This function is deletes the row in _cet_files_ that corresponds to “filename” and drops the underlying table that “filename” is mapped to.

Bfilestat

The W32 manual describes the Bfilestat function in the following manner:

> This function may be used to determine if a file or directory exists. Its syntax is:
>
> **CALL Bfilestat(“filename”,addrof(stat$))**
>
> where:
>
> **filename**
>
> The filename in a DOS or THEOS format. If the file is indexed, the extension (.idx or .dat) must be specified as in data\test.idx.
>
> **stat**
>
> A string that contains either a D for directory, R for a file, or a null if the filename does not exist.

This function works in the following way:

- If “filename” is a directory, a value of “D” is returned in stat$
- Otherwise, if there exists a sequential file of that name, an “R” is returned in stat$
- Otherwise, if there is a direct file of that name, an “R” is returned in stat$
- Otherwise, if there is a mapped SQL file of that name, an “R” is returned in stat$
- Otherwise, if there is an indexed file of that name, an “R” is returned in stat$

Bindinfo

The W32 manual describes the Bindinfo function in the following manner:

> This function may be used to find out information about an open indexed file.
> The syntax is:
>
> **CALL Bindinfo(channel%,addrof(count),addrof(rec%),addrof(key%))**
>
> where:

channel

An integer variable or constant representing the channel on which the indexed file was opened.

count

A numeric value which will return the number of records in the file.

rec

The length of the data record. Note that this value has traditionally been the sum of the characters in the data record plus those in the key.

key

The return value is the length of the key.

For a mapped MySQL file, the function returns the current number of rows in the associate table in **count**, the key length which is stored in the associated _cet_files_ row in the variable **rec**, and the record length which is stored in the associated _cet_files_ row in the variable **key**.

Brename

The W32 manual describes the Brename function in the following manner:

The Brename function is designed to eliminate having to perform a CSI statement to execute the DOS RENAME command. In either case, the file must be closed before it can be renamed.

The syntax of the statement is:

CALL Brename("original-filename new-filename")

This function works in the following way:

- If there exists a sequential file with **original-filename** name, it is changed to **new-filename**.
- Otherwise, if there exists a direct file with **original-filename** name, it is changed to **new-filename**.
- Otherwise, if there is a mapped SQL file with **original-filename** name, it changes the value of "Name" in the _cet_files_ row for that mapped MySQL file name to **new-filename.**
- Otherwise, if there is an indexed file with **original-filename** name, it is changed to **new-filename**.

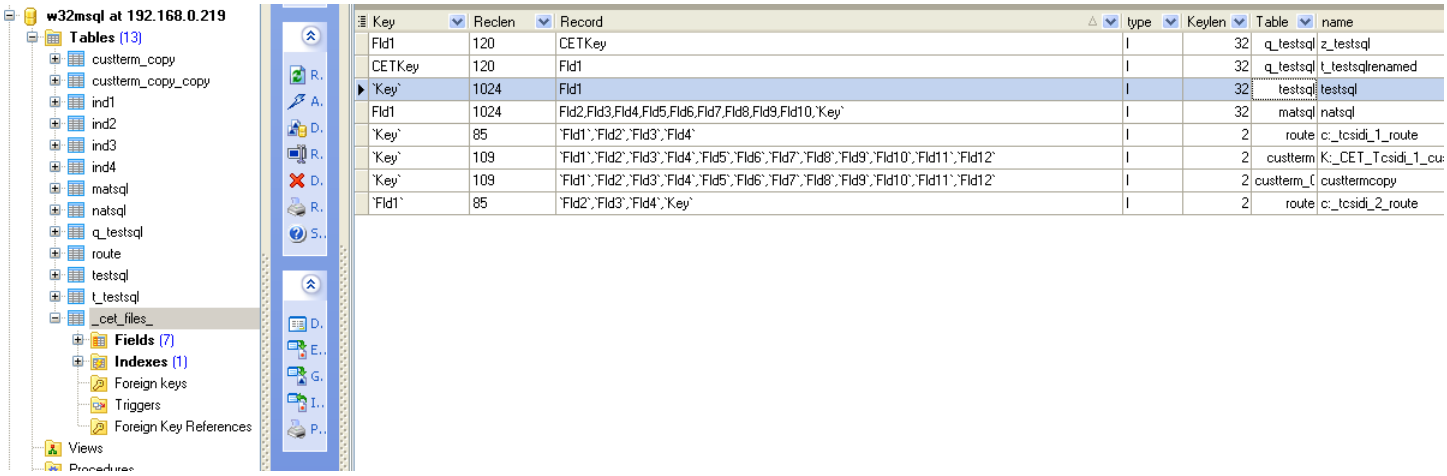
Test Routine

I've written a small test routine to try each of the functions:

```
20 REM -
30 REM -  Test routine for new RUNW32 functions with MySQL support
40 REM -
50 REM -     Bcopy
60 REM -     Bcreate
70 REM -     Berase
80 REM -     Bfilestat
90 REM -     Bindinfo
100 REM -     Brename
110 REM -
120 REM -  We assume that we have a single table with a mapped name
130 REM -  of testsql.  We can test everything else from that.
140 REM
150
160 REM - First, let's see what's in testsql
170
180 PRINT "Opening testsql and listing records"
190
200 OPEN #1:"testsql", update indexed
210
220 WHILE 1
230 READNEXT #1,Key$: Val1$, Val2$
240 IF EOF(1) THEN break
250 PRINT "Key = ";Key$;", Rec Values are ";Val1$;",";Val2$
260 WEND
270
280 CLOSE #1
290
300 REM - Next let's copy testsql
310
320 PRINT 'CALL Bcopy("testsql testsqlclone")'
330 CALL Bcopy("testsql testsqlclone")
340
350 REM - Let's see if it happened, first do a filestat
360
370 PRINT 'CALL Bfilestat("testsqlclone",addrof(stat$))'
380 CALL Bfilestat("testsqlclone",addrof(stat$))
390 PRINT "Bfilestat on testsqlclone returns ";stat$
400
410 REM - Now, let's try renaming this file
420 PRINT 'CALL Brename("testsqlclone testsql2")'
```

```
430 CALL Brename("testsqlclone testsql2")
440
450 REM - Now, let's try deleting this file
460
470 PRINT 'CALL Berase("testsql2")'
480 CALL Berase("testsql2")
490
500 REM - Let's see if it happened, first do a filestat
510
520 PRINT 'CALL Bfilestat("testsql2",addrof(stat$))'
530 CALL Bfilestat("testsql2",addrof(stat$))
540 PRINT "Bfilestat on testsql2 returns ";stat$
550
560 REM - Now let's clear testsql2, first lets make one
570
580 PRINT 'CALL Bcopy("testsql testsql2")'
590 CALL Bcopy("testsql testsql2")
600
610 REM - Now let's see if there are any records there
620
630 OPEN #1:"testsql2", update indexed
640
650 PRINT "Here's the records in testsql2"
660 WHILE 1
670 READNEXT #1,Key$: Val1$, Val2$
680 IF EOF(1) THEN break
690 PRINT "Key = ";Key$;", Rec Values are ";Val1$;",";Val2$
700 WEND
710
720 CLOSE #1
730
740 CALL Berase("testsql2")
    WAIT
750 END
760
```

The MySQL setup is:



The result of the test follows: