test link

c-treeACE

V11.5 Release Notes

Audience

Developers

Subject

V11.5 Updates and Changes to the c-treeACE Database Technology

FairCom®

Contents

1. Introduction

This document lists the corrections we have implemented in this release. These changes are grouped by functional area. Be sure to review the Critical Fix Chapters, as follows:

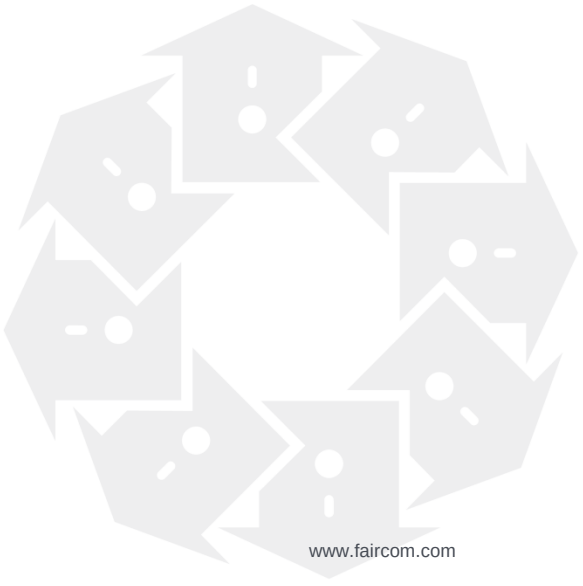
- **c-treeACE Server Critical Fixes** (page 3)
- **c-treeACE SQL Server Critical Fixes** (page 14)

Note, c-treeACE SQL Server users should review both the *c-treeACE Server* (page 3), and the *c-treeACE SQL Server* (page 14) chapters, as well as *FairCom Server* (page 17) and *c-treeACE SQL* (page 25).

We also recommend everyone review **Notable Compatibility Changes** (page 57).

You may have received some of these changes in a prior “interim” delivery. The best way to ensure that you get the latest set of changes is to update to this, our latest release.

V11.5 is packed with new features—from simple enhancements to major breakthroughs. The new and updated features are listed in the *V11.5 Update Guide*.



2. FairCom Server Critical Fixes

2.1 Corrected checkpoint processing to persist required transaction logs for recovery

This fix corrects a potentially serious issue involving the logging of the following message to *CTSTATUS.FCS*:

```
Index buffer on commit node list without update flag...
```

Background information: Checkpoint processing is a critical component of transaction control. During this phase, a known good state of the database is created and persisted. There are several key phases of operations during checkpoint processing:

- Determining cache page vulnerabilities
- Flushing updated cache page information to disk
- Writing critical recovery state information to persisted transaction logs
- Determining which transaction logs to retain (only logs required for recovery should be persisted)

It is possible during checkpoint processing that while walking internal updated cache page lists a c-tree thread can update an entry after that position in the list has already been examined. This condition is allowed, as no mutex is held during list processing to maintain performance with very high transaction rates.

In the V8 release of c-tree Servers, this condition was not always properly checked. A buffer could be left on the list, even though it had been properly flushed. The following status log message was noted immediately preceding such an event:

```
Index buffer on commit node list without update flag...
```

However, this potentially resulted in not releasing any transaction logs, as the server determined they were needed for recovery of this buffer page. The accumulation of large numbers of transaction logs could result in consuming all disk space.

Corrective action was introduced toward the end of the V8 line of c-tree Servers and included in V10 lines. This correction detected and marked such unexpected updated buffer states such that they would trigger appropriate transaction log release.

However, with V10 servers under high transaction load, the diagnostic message continued to be noted and now the number of active log files decreasing to 4 and possibly not increasing after that point:

```
Index buffer on commit node list without update flag...
The number of active log files decreased to: 4
```

Further, during a maintenance window, a c-treeACE server was improperly halted. During automatic recovery, error **96** occurred: a transaction log was determined to be required for recovery, however, did not exist. In this case, it was for an index file buffer, which could be recovered by an index rebuild of the data file.

It has been determined that these automatic recovery errors could occur for either a data file or index file phase of recovery. If the data phase recovery fails, data may not be recoverable. If the

data recovery phase succeeds, data is not lost. If the index recovery phase fails, changes to indexes may be lost. Indexes can be rebuilt from the existing data files in this case.

This issue has been corrected in all c-treeACE Servers with build dates after 160527 (May 27, 2016). It was determined that the initial checkpoint operation correctly handled the condition. However, a subsequent checkpoint failed to consider the same state information causing the calculated log requirements to be out of sync with what was actually required. A proper check condition now correctly releases only the logs no longer necessary for recovery, persisting all required recovery logs.

Note: In this case of a successful data recovery phase and failed index recovery phase, using the server keyword `TRANIDX_LOPN_ERR_CONTINUE YES` in *ctsrvr.cfg* allows automatic recovery to succeed. However, indexes should be rebuilt ensuring they are in sync with the data.

2.2 V11.2 unhandled exception on first update to member index, or error 14 on member index after automatic recovery

An unhandled exception can occur in c-treeACE Server V11.2.* only, in either of these two situations:

1. An index file was opened read-only then closed but was kept open using the `KEEPOPEN_LIST` option and was then reopened and an index member was updated in such a way that the host index was not also updated (for example, read a record and update a field that is used in a member index and while the field is not used in the host index).
2. It is also possible for a transaction-controlled index to fail to open with error **14** after automatic recovery if FairCom Server terminated abnormally after the index file was opened and an index member was updated without updating the host index (which is very unlikely).

The logic has been modified to eliminate these problems.

2.3 Protection for internal server threads still running at end of server shutdown

When FairCom Server shuts down, it waits for a period of time for internal threads to terminate. If the time limit is reached and a thread hasn't terminated, the server logs a message to *CTSTATUS.FCS* indicating this situation (for example, "ctDISKFULLaction thread still running").

A potential problem was that, if internal threads are still running at the end of server shutdown, the server still freed some resources that the threads might be using (such as memory suballocator lists). **This could cause the server to crash at the end of shutdown.**

To avoid this possibility, we now track the number of threads that are still running, and if at least one thread is still running, we skip freeing the resources.

2.4 Server Crash - Failed memory allocation on startup could crash server

In a very specific situation following a failed memory allocation, a server crash was experienced at startup. Additional allocation checks within the server startup sequence now avoid the unexpected null memory reference.

2.5 Possible Server exception with CTSTATUS_SIZE option

FairCom Server could terminate with an unhandled exception when `CTSTATUS_SIZE` is specified and either a dynamic dump started or a log template copy occurred and then failed (typically due to a lack of sufficient disk space). This logic has been corrected.

2.6 Server terminated abnormally when ctQUIET() was called after a quiesce previously failed to reopen a file

FairCom Server experienced an abnormal shutdown (possibly causing FairCom Server to hang) when **ctQUIET()** was called after a previous **ctQUIET()** call could not reopen files when undoing the quiesce. Messages in CTSTATUS.FCS:

```
Unexpected error during ctQUIET system level file open...
Unexpected ctCLOSE_FDflg3...
mbclosx: close (myfile.dat) failed with error= {9}
Unexpected error during ctQUIET system level close...: -198
ctsave failed: system code = 9 lc = 38 fd = -1
```

The logic has been modified to correct this problem.

2.7 ctrdmp terminated with unhandled exception during recovery phase

The dynamic dump restore utility, **ctrdmp**, occasionally terminated with an unhandled exception during its recovery phase. This was more likely to happen when there was a large amount of transaction log activity included in the dynamic dump, which is common when a large (time-consuming) backup is performed when updates are being made to the database. The logic has been modified to eliminate this problem.

2.8 Failed call to ctDeferredIndexControl() could hang c-tree Server connections

FairCom Server connections could hang after a call to the c-treeACE Deferred Index Logic (new in c-treeACE V11.0) function call **ctDeferredIndexControl()** with opcode of *DFKCTLsetcallback* failed. Stack traces showed hung threads in a call to request a file's deferred index reader/writer lock. The logic has been improved to correct this problem.

2.9 Unhandled exception in Blocking Locks: BLKIREC()

An unhandled exception in the FairCom Server could be caused in a call to **BLKIREC()**. This issue only occurs if a FairCom Server internal system queue is being closed at the same time the **BLKIREC()** is getting called, which is a rare event. The logic has been corrected to eliminate this problem.

2.10 Crash in REDVREC following fileblock

The ISAM function to re-read a variable-length data record, **ReReadVRecord** (**REDVREC**), could cause a FairCom Server crash if a **ctFILBLK()** call was made prior to the **ReReadVRecord()** call, and the **ctFILBLK()** call failed because the file specified wasn't found. The logic has been corrected to eliminate this problem.

2.11 Unhandled exception when a second callback is added to a file

A record update callback raised a c-treeACE Server exception due to an unexpected call. When a new record update callback was added, a call was made to the RUCB_DATASCAN callback so that operations could be done on records that existed before the new callback was added. If this was not the first callback added to the table, the wrong callback was executed. This issue has been corrected by modifying the logic to ensure the correct callback is called.

2.12 Crash after error opening file with conditional index

A crash has been fixed that could occur for a file with a conditional index resource if an error occurred during a c-tree ISAM-level file open after the conditional index resource was retrieved.

2.13 ctdump may truncate dump stream file if DAVL_ERR (442) occurred

A situation was encountered when the **ctdump** utility was run with a new option added in c-treeACE V11 to receive the dump stream data from the server (by using the *-o* option with **ctdump**, or using the `DDOPT_RECVSTREAM` option in *ctsrvr.cfg*). If an error such as error **442** occurred that was normally not a fatal error (it simply indicates that some of the specified files could not be included in the dump, which can be normal if files are missing), the **ctdump** utility did not close the dump stream file that it is writing. In this case, the dump stream file might be truncated, possibly resulting in an incomplete backup file. The logic has been modified to handle this situation correctly.

2.14 Dynamic dump restore terminated with internal error when restoring segmented files

The dynamic dump restore utility, **ctrdmp**, sometimes terminated with an internal error such as **7491** or **7495** when restoring segmented files. The logic has been modified to correct this.

2.15 Dynamic Dump with !PROTECT option sometimes caused FairCom Server to hang

When a dynamic dump was performed on a non-transaction-controlled, variable-length data file and the `!PROTECT` dump script option was used, FairCom Server may hang if updates were made to the data file while the dynamic dump was backing it up. The logic has been modified to correct this.

2.16 Automatic recovery crash when redoing PRMIIDX operation for data file with compressed records

c-treeACE terminated with an unhandled exception during automatic recovery if recovery redid a PRMIIDX operation on an index of a data file that used compressed records. The logic has been modified to eliminate this problem.

2.17 Automatic recovery failed with error 12

An obscure issue involving a non-transaction-dependent file create as part of automatic recovery in conjunction with the deferred indexing logic added in V11.0 could lead to an automatic recovery failure with error **12**. If `DIAGNOSTICS TRAN_RECOVERY` was active during the automatic recovery, the output in *RECOVERY.FCS* showed the transaction redo phase of automatic recovery redoing the create of a non-transaction-dependent file. Example entry:

```
tranred: 1-0252e6 #39 O20 F0051-000 P0000-570271e3x   CRETRAN: ctredocre1: <>
```

This issue was identified internally by the FairCom QA team and has not been reported in the field. The logic has been corrected to eliminate this problem.

2.18 Automatic recovery failed with L69 error

In a special circumstance, FairCom Server's automatic recovery failed with an **L69** error. *CTSTATUS.FCS* showed this message before the error:

```
WARNING: zero log-entry file handle (tfil)
```

Transaction-controlled files are assigned a zero transaction file number until it is assigned a transaction file number. If a checkpoint includes that file in its list of open files, it can cause automatic recovery to fail due to the zero transaction file number.

The checkpoint now checks if the file has a zero transaction file number when determining if the file is to be included in the checkpoint's list of open files. If the transaction file number is zero, the file is not included in the checkpoint.

2.19 RECOVER_MEMLOG option could cause automatic recovery to fail with errors 55, 96, etc.

When using the `RECOVER_MEMLOG` configuration option, automatic recovery could fail with an error such as 55 or 96. The logic has been corrected to eliminate this problem.

Workaround: If you are using `RECOVER_MEMLOG` prior to this revision and see automatic recovery failing, try restarting without the `RECOVER_MEMLOG` option.

2.20 Auto Restore Point quiesce no longer causes threads to remain blocked after quiesce fails with error 843

The interaction between an auto Restore Point quiesce that failed with error **843** and other threads performing checkpoints may cause threads to remain blocked until a later call to **ctQUIET()** unblocks them. The logic has been modified to correct this.

2.21 Exception when undoing transaction-dependent file create

When a create of what appeared to be a transaction-dependent data file failed, a c-treeACE Server exception occurred when attempting to undo the file create. The file control block that was sent to the c-treeACE Server was corrupt in the single instance where this situation occurred. The logic has been enhanced to issue a controlled shutdown of the server in this unlikely situation.

2.22 FairCom Server controlled shutdown CHKP_ERR (529)

When an application wrote large records (large meaning greater than the c-treeACE Transaction Log file size (set by the `PAGE_SIZE` keyword) to a transaction-controlled (ctTRNLOG) c-tree data file and c-treeACE Server was configured to use log templates, and the length of written record hits a precise size boundary, c-treeACE Server could terminate with error **529**. This issue has only been seen in FairCom's internal QA department.

2.23 Server deadlock involving aged cache page flush

Under very precise circumstances, FairCom Server could hang if a checkpoint found an unexpected state of a data cache page or index buffer at the same time the page in question was fully aged and the buffer was ready to be flushed. The logic has been modified to correct this situation.

2.24 Successive BLKIREC() calls caused memory leak

When a client calls **BLKIREC()**, FairCom Server memory use may increase. This is typically only noticeable if many calls to **BLKIREC()** are made by clients. The logic has been updated to correct this.

2.25 Server crash due to orphaned cache page when out of disk space

Errors were occasionally seen in obscure situations where FairCom Server could not write an updated data cache page to a non-transaction-controlled c-tree data file when physically closing the file (for example if the data file was created with a zero extension size and the disk is full).

1. The cache page could be written to the wrong file; or
2. When an attempt to write the page was made (say by a background flush thread or **CTFLUSH()** call for all files), the write failed with an error that was considered fatal, causing FairCom Server to shut down with a "FAILED TRAN IO" error message in *CTSTATUS.FCS*.

The following example shows messages from *CTSTATUS.FCS* resulting from this problem. *mark.dat* is a non-transaction data file and *cxd04.dat* is a ctTRNLOG data file. Note the write error on *mark.dat* due to no disk space, and the write error on *cxd04.dat* due to an invalid file descriptor (because *cxd04.dat* was opened and then closed before the flush of the page was attempted):

```
Mon Nov  7 10:51:14 2016
 - User# 00014 WRITE_ERR: mark.dat at 0:8000x  sysiocod=112 bufsiz=32768 bytes written=0[0]  ioLoc=0: 37
Mon Nov  7 10:51:18 2016
 - User# 00014 FAILED IO: ctwrtbuf...: 37
Mon Nov  7 10:51:18 2016
 - User# 00014 mark.dat: 37
Mon Nov  7 10:52:09 2016
 - User# 00014 Error on close file. locale:11  sysiocod:0  uerr_cod: 37

Mon Nov  7 11:02:15 2016
 - User# 00005 WRITE_ERR: cxd04.dat at ffffffff:ffffffffx  sysiocod=6 bufsiz=32768 bytes written=0[0]
ioLoc=0: 37
Mon Nov  7 11:02:25 2016
 - User# 00005 FAILED TRAN IO: ctwrtbuf...: 37
Mon Nov  7 11:02:26 2016
 - User# 00005 cxd04.dat: 37
```

The cache flushing logic has been modified to handle the situation properly and eliminate this problem. However, it is always advisable to ensure you have sufficient disk space for c-treeACE controlled data files and the system files (*L*.FCS* files, etc.) created by the c-treeACE Server process.

Note: By default, FairCom Server treats a write error on a transaction-controlled (ctTRNLOG) file as a fatal error and it immediately shuts down in this situation. As a result, the changes described here primarily affect **non-ctTRNLOG data files**. Also, an error writing an index node to disk when physically closing the index file was already properly handled.

2.26 Server Exception - Correct unhandled exception in compression logic

Calling a fixed-length record read function, such as **FRSREC()** or **EQLREC()**, in client/server mode on a data file that supports compressed data records sometimes produced the following symptoms:

- FairCom Server could terminate with an unhandled exception in the RLE decompression function.
- The function could fail with error 938 (**AREC_DCM**), error 634 (**NLEN_ERR**), or it could return invalid data.

The logic has been corrected.

2.27 Server Exception or read incorrect ending of compressed record

A call to FairCom Server to read a record from a variable-length data file in compressed data record files when the record image spans cache pages by 4 or fewer bytes could return wrong data in the last 1 to 4 bytes, or fail with **ITIM_ERR** (160). In some cases, an unhandled exception could also occur, which will bring the FairCom Server down. The logic has been altered to correct this.

2.28 Server - Exception when Replication Agent read past end of transaction log

FairCom Server terminated with an unhandled exception when Replication Agent attempted to read past end of a transaction log. This can happen if a Replication Agent is using an old transaction log and offset reference that does not correspond to the current state of the transaction logs. Prior to the exception occurring, this message was logged to *CTSTATUS.FCS*:

```
ctrepl: READ_ERR at code location 5: lognum=1 logpos=91964148 redamt=1048576 nread=0 err=0
```

The logic has been changed to protect against this situation.

2.29 Server hang while dynamic dump waited for quiet transaction state

When a dynamic dump was performed without the *!DELAY* option, FairCom Server could hang if a connection that already had an active transaction attempted to begin a transaction. The logic has been modified to correct this.

2.30 Server deadlock in record update callback

In a very rare timing-related issue, FairCom Server V11.*x* could hang when a connection was committing an update to a record in the same file to which another connection was adding a permanent index using the *ctNO_IDX_BUILD* or *ctQUEUE_IDX_BUILD* options. The logic has been modified to correct this.

2.31 Server unhandled exception when using ctstat -funcfile file

FairCom Server experienced an unhandled exception when using **ctstat** *-funcfile*. When using the **ctstat** *-funcfile* option to log per-file function timings to the file *SNAPFUNC.FCS*, FairCom Server occasionally terminated with an unhandled exception. The logic has been updated to correct this problem.

2.32 Server crash when field starting in fixed portion and ending in variable portion

A server crash was seen when inserting records into an imported table field when a field was a CT_4STRING that has the first 4 bytes in the fixed portion of the record and the rest in the variable and len 0 in the DODA. The logic has been modified to correctly handle this improperly coded DODA situation.

2.33 Server crash due to invalid license

A FairCom Server crash was seen when starting the FairCom Server with an invalid license file. Additional license file size checks were added to avoid unintended null memory references.

2.34 Server unhandled exception when two connections rebuild a compressed data record file at the same time

Rebuild of a compressed record data file sometimes caused FairCom Server to terminate with an unhandled exception if two connections attempted to rebuild the same file at the same time. The logic has been corrected to eliminate this problem.

2.35 Server - Exception at shutdown when DISK_FULL_ACTION active

When the `DISK_FULL_ACTION` option was used, FairCom Server sometimes reported an exception while it was shutting down. The logic has been modified to correct this problem.

2.36 Server unhandled exception when r-tree report's c-tree initialization fails

In rare conditions, FairCom Server terminated with an unhandled exception when running an r-tree script if the r-tree script failed to initialize properly. The following types of messages were logged to *CTSTATUS.FCS*:

```
- User# 00019  rtree: startup failure
- User# 00019  W32 Exception handler invoked before ctcatend.
```

The logic has been modified to resolve this issue.

2.37 Mac OS X server exception when shutting down fails

An unhandled exception was seen when shutting down FairCom Server on Mac OS X. The logic has been corrected to eliminate this problem.

2.38 Server handles failed startup with SNAPSHOT_SHUTDOWN

If server initialization gets sufficiently far before failing, and keyword `DIAGNOSTICS SNAPSHOT_SHUTDOWN` is enabled, the Snapshot operation is called. This may encounter memory structures that have not yet been initialized, resulting in a memory access violation. This sequence of events is now properly detected to eliminate hangs or crashes.

2.39 RenameIfil() possible file loss on Unix/Linux fixed

During a file rename operation, Unix/Linux systems could have been unable to open a file due to **DMAP_ERR** (957) with a *CTSTATUS* message similar to the following:

```
User# 00055 DMAP_ERR: index file myFile.idx (474) for data file myFile.dat (506) is already associated with data file myFile.dat (148)
```

On Unix/Linux systems, a **RenameIfil()** could succeed even if the new name already existed on disk, causing the original file to be deleted. The **DMAP_ERR** may occur if the original file was open and in use at the time of the rename.

The logic on Unix/Linux systems has been modified so that, if the new file name already exists, the **RenameFile()** and **RenameIFile()** functions will now fail with **RENF_ERR** (67).

Note: This is a behavior change. Server keyword `COMPATIBILITY RENAME_OVERWRITE` allows reverting to the prior behavior.

2.40 Server Exception - Correct unhandled exception in compression logic

Calling a fixed-length record read function, such as **FRSREC()** or **EQLREC()**, in client/server mode on a data file that supports compressed data records sometimes produced the following symptoms:

- FairCom Server could terminate with an unhandled exception in the RLE decompression function.
- The function could fail with error 938 (**AREC_DCM**), error 634 (**NLEN_ERR**), or it could return invalid data.

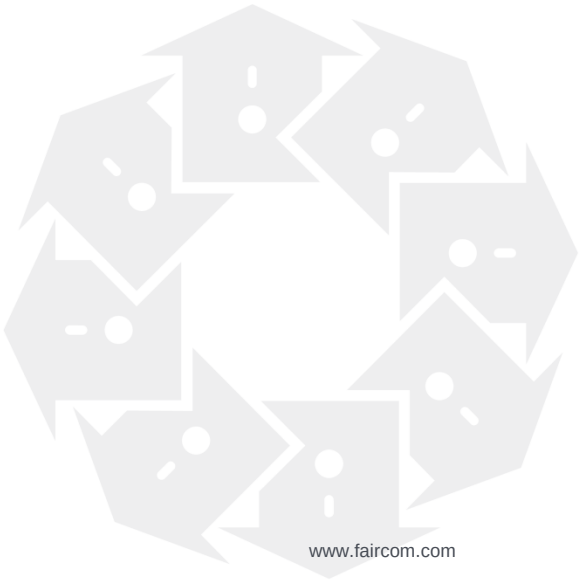
The logic has been corrected.

2.41 Server Exception or read incorrect ending of compressed record

A call to FairCom Server to read a record from a variable-length data file in compressed data record files when the record image spans cache pages by 4 or fewer bytes could return wrong data in the last 1 to 4 bytes, or fail with **ITIM_ERR** (160). In some cases, an unhandled exception could also occur, which will bring the FairCom Server down. The logic has been altered to correct this.

2.42 REDREC() reads wrong last 4 bytes of uncompressed fixed-length record in compressed file

A call to **REDREC()** to read a record from a compressed fixed-length data file can read the wrong value for the last four bytes of the record. This can only happen for a record that isn't compressed. The logic has been modified to resolve this issue.



3. c-treeACE SQL Server Critical Fixes

3.1 SQL parser consumed all system memory and crashed the server

When trying to execute a query having subqueries, a case switch with joins caused the SQL parser to consume all system memory and eventually **crashed the system**.

Note: This serious server issue requires the server to be updated if queries having "case then" (searched case) are used.

3.2 Exception when identity field references out-of-range field number

c-treeACE SQL Server terminated with an unhandled exception after a query failed with SQL error **-21017** (or FairCom DB API error 4017, **CTDBRET_NOTFIELD**). This was an unexpected situation in which the identity field referenced an out-of-range field number. The logic has been modified to handle this situation. If the identity field number is out of range for the table's field definitions, the following error is returned: **CTDBRET_NOTFIELD**.

3.3 c-treeACE SQL Server Unicode crash

The Linux Unicode version of c-treeACE SQL Server crashed following a query such as the following:

```
"select max(transaction_date) from mytable"
```

The use of MAX and MIN aggregate functions over indexed Unicode columns has been corrected to prevent this from occurring.

3.4 c-treeACE SQL Server crash following failed Java initialization on AIX

If the c-treeACE SQL Server Java keywords in *ctsrvr.cfg* were active, and improperly set, the following was logged at startup: `JVM Init Error: Could not find ctree/sqlsp/SQLDA class`. The server then crashed on the first SQL connection. This crash was observed on POWER CPU (AIX & Linux) only. The logic has been modified to correct this.

3.5 c-treeACE SQL Server exception when building a key for a Unicode key segment

c-treeACE SQL Server caused an exception when building a key for a Unicode key segment using the source size provided option on a length-counted string data type (CT_PSTRING, CT_2STRING, CT_2UNICODE, or CT_4STRING) if the record buffer holds only part of the field. The logic has been modified to correct this problem.

3.6 Possible exception or infinite loop in co-file support

During use of the c-treeACE SQL co-file support (an advanced feature providing enhanced locking control for files opened multiple times in the same connection), an exception or an infinite loop was possible when c-treeACE SQL Server was opening a file. The following message could be logged to *CTSTATUS.FCS*:

```
WARNING: unexpected number of co-files in usrfilupd:
```

The logic has been updated to eliminate this potential problem.

3.7 Server crash with self-referencing constraints on table with long field

A server crash was seen when inserting a record into a table with long fields having a self-referencing foreign key (a foreign key that referenced the primary key of the same table). This issue has been now resolved.

3.8 SQL subquery in case clause crash

A server crash was seen when running a query having Subquery in Searched Case clause within the WHERE condition. The logic has been modified to correct this.

3.9 SQL parser crash fixed

A **server crash** was experienced when running a query with a sub-query having a scalar function that had another sub-query with no FROM clause as its argument. The logic has been corrected to eliminate this problem.

3.10 Fixed crash running query with Order By and a sub-query with Group By on multiple columns

A server crash was occasionally seen while running a query with Group By in a sub-query on multiple fields and Order By in the outer query. The problem occurred in a very specific optimization case that rarely occurs. It was triggered by a Group By in a sub-query with another

sort above it in the main query. When the Group By was on more than one column, the error case was triggered. The logic has been corrected to eliminate this problem.

3.11 Fixed memory leak on each SQL connection

A small memory leak was occurring on each SQL connection. The logic has been corrected to eliminate this problem.

3.12 SQL Bit String conversion fix

A SQL Bit String conversion was occasionally causing heap corruption. The logic has been modified to correct this problem.

3.13 SQL handling of binary literals larger than field size

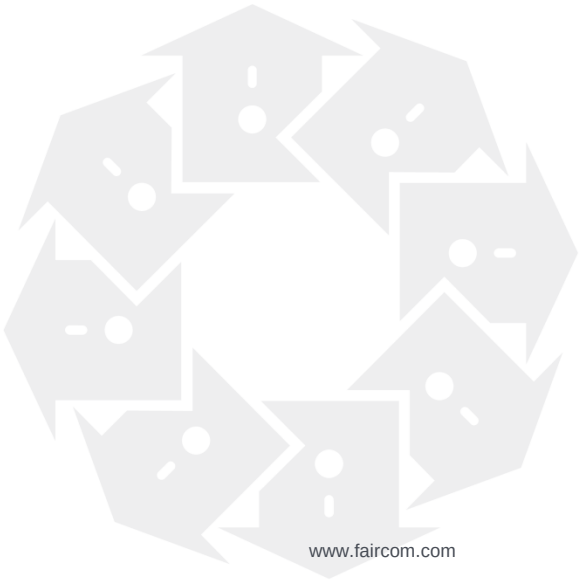
A server crash was seen with a SQL statement that used binary literals that were larger than the field size, for example:

```
create table tsttbl ( col1 binary , col2 varbinary(2), col3 char(4),col4 char(4));
!echo _delete from table
delete from tsttbl where col1 < x'aabb';
```

The logic has been modified to handle this properly.

3.14 DSQL - Server crash when calling ctsqlGetBlob on a field with any data type other than a long

Using DSQL, a **server crash** due to stack corruption was seen when calling **ctsqlGetBlob** on a field that had a type that was not long. The logic has been updated to correct this.



4. FairCom Server Changes

4.1 File copy function call may fail with error 965 (BCOD_ERR)

A call to the FairCom Server file copy function sometimes unexpectedly failed with error 965 (**BCOD_ERR**). The logic has been corrected to avoid this error.

4.2 Record update callback function file open

When a data file had a record update callback function in effect and a call was made to open the file, an error sometimes caused the data and index files to remain open. The logic has been modified to correct this.

4.3 V11 client may lose connection to FairCom Server when pstack was run on server

When the **pstack** utility was run on the FairCom Server process on a Unix system, the V11 client sometimes lost its connection to the FairCom Server. The logic has been modified to correctly handle this situation.

4.4 Server shutdown failed with error 452

When shutting down FairCom Server on Windows without `CONSOLE NO_PWRDWNPASSWORD` and `CONSOLE NO_SHUTDOWN_PROMPT` in *ctsrvr.cfg*, the server prompted for the ADMIN user name and password, but then failed to shut down with error **452**. The logic has been corrected to eliminate this problem.

4.5 FairCom Server ISAM counter values on Linux

FairCom Server's ISAM counters were not properly initialized on Linux. This has been corrected.

4.6 FairCom Server no longer logs TLOG_ERR with trantyp of 0x4e

In certain situations, FairCom Server was found to log the following message to *CTSTATUS.FCS*:

```
Thu Aug 31 10:40:48 2017
User# 00024 TLOG_ERR: usrtr=77479 usrty=0x20000210 trantyp=0x4e tranfil=342
```

This was an errant message and should not have been logged. The logic has been modified to prevent this situation from occurring.

4.7 One-time memory leak in background file flush threads

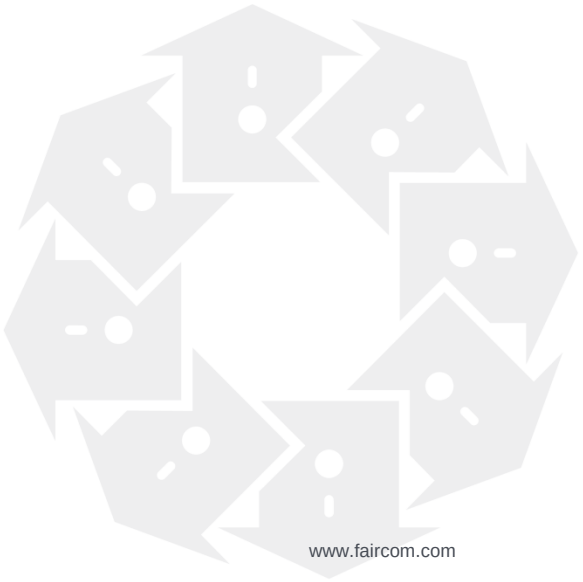
A one-time memory leak was discovered in FairCom Server when the background file flush threads were enabled. This memory leak has been resolved.

4.8 Improved detection of invalid server configuration option values

FairCom Server checks for allowed values in the configuration file (*ctsrvr.cfg*). In the case of server configuration options that accept `YES` and `NO`, it treated alternatives, such as `OFF` and `ON`, as `NO`. The option check has been changed to allow only `YES` and `NO` (case-insensitive). Any other value will generate an error. The affected server configuration options are:

```
ADMIN_ENCRYPT
ADVANCED_ENCRYPTION
ALLOW_MASTER_KEY_CHANGE
AUTO_CLNIDXX
AUTO_MKDIR
AUTO_TRNLOG_LIGHT
CHANGE_ENCRYPTION_ON_COMPACT
CHECK_CONFIG
CHECKPOINT_MONITOR (also allows DETAIL)
CHECKPOINT_PREVIOUS
DEADLOCK_MONITOR
ENABLE_TRANSFER_FILE_API
GUEST_LOGON
INHERIT_FILE_PERMISSIONS
KEEPOPEN_FLUSH
LDAP_SSL
LOCK_HASH_NOSHRINK
```

```
LOG_ENCRYPT
MIRRORS
PERMIT_NONTRAN_DUMP
PREIMAGE_DUMP
PREIMAGE_HASH_NOSHRINK
RECOVER_DETAILS
RECOVER_SKIPCLEAN
RECOVER_TO_RESTORE_POINT
REPL_IDENTITY_USE_MASTER
REPL_IDENTITY_USE_SOURCE
REPL_SRLSEG_ALLOW_UNQKEY
REPL_SRLSEG_USE_MASTER
REPL_SRLSEG_USE_SOURCE
SECURE_LOGONS_ONLY
SESSCHG_ENABLE
SKIP_CTADDWORK
SKIP_INACCESSIBLE_FILES
SKIP_MISSING_FILES
SKIP_MISSING_LOG_MIRRORS
SKIP_MISSING_MIRRORS
STARTUP_BLOCK_LOGONS
SUBSYSTEM COMM_PROTOCOL_SSL SSL_CONNECTIONS_ONLY
SUBSYSTEM COMM_PROTOCOL_SSL VERIFY_CLIENT_CERTIFICATE
SUPPRESS_LOG_FLUSH
SUPPRESS_LOG_SYNC
TRANIDX_LOPN_ERR_CONTINUE
UNBUFFERED_LOG_IO
USE_EMPTY_LIST
VSS_WRITER
XTDKSEG_FAILED_DEFAULT_OK
```



4.9 Socket timeout TRSP_ERR (809)

Applications that use c-tree's client-side socket timeout feature, including the Replication Agent (by using the `socket_time` configuration option), were found to shut down with error **809** even though the call to **ctReplGetNextChange()** did not exceed the specified socket timeout. The logic has been modified to correct this issue.

Note: This modification affects the Replication Agent and any client application that uses c-tree's client-side socket timeout feature.

4.10 V11 COMMIT_DELAY on Windows change

Slow transaction commit performance was seen on Windows, with c-treeACE V11.0 - V11.2 Servers when there was one committer and another connection had an active transaction.

It was found that the effectiveness of `COMMIT_DELAY` could be improved when it calculated a sleep time that was not zero but was less than one millisecond, which equates to a zero sleep time on Windows. To improve the effectiveness of `COMMIT_DELAY` in that situation, a minimum delay of one millisecond was used on Windows.

Prior to this change, the `COMMIT_DELAY` sleep time calculation occurred even when only one connection was committing a transaction if at least one other active transaction existed.

With this change, `COMMIT_DELAY_MINIMUM` now only affects the `COMMIT_DELAY` sleep time if more than one connection is committing at the time that the `COMMIT_DELAY` sleep time is calculated.

4.11 DISK_FULL_ACTION environment variables fix

When an environment variable name was specified in the value of the server configuration option `DISK_FULL_ACTION`, it was ignored. For example:

```
DISK_FULL_ACTION shutdown %VOL1% 1G
```

`%VOL1%` was not replaced with the environment variable value. The logic has been modified to correct this.

4.12 COMPATIBILITY RENAME_OVERWRITE ignored on Windows

The `COMPATIBILITY RENAME_OVERWRITE` keyword was ignored for c-treeACE Windows Severs. The logic has been modified to correct this.

4.13 FairCom Server now writes automatic restore points at the specified interval

When the `AUTO_RESTORE_POINT` server configuration option was used with the `CHECKPOINT YES` option, restore points were sometimes written more frequently than the specified transaction log interval. This did not cause any specific issues, other than a very slight performance degradation. The logic has been modified to correct this behavior.

4.14 Automatic directory create option not working for TRNLOG transaction-dependent file creates

The automatic directory creation option (*ctAUTOMKDIR* in `xcreblk splval` field) was not working for TRNLOG transaction-dependent files. The file create failed with error **17** instead of creating the missing directories.

A workaround was available: Disable file system metadata flushing by adding `COMPATIBILITY NO_FLUSH_DIR` to *ctsrvr.cfg* and restart c-tree Server.

The logic has been modified to correct this problem.

4.15 Connection hangs when reading records due to abandoned commit write locks

A call to FairCom Server to read a record on a transaction-controlled file without acquiring a record lock could sometimes hang. This hang occurred as an internal lock intended only for the transaction commit phase was not appropriately freed upon final commit cleanup and record reads were in a delay loop waiting for this transient lock state to clear. The logic has been corrected.

4.16 PREIMG file create fail with DOTX_ERR (955) during dynamic dump

File create of a PREIMG file sometimes failed with a **DOTX_ERR** (955) when a dynamic dump was running and `PREIMAGE_DUMP YES` was specified in *ctsrvr.cfg*. The logic has been modified to eliminate this problem.

4.17 Error 160 after automatic recovery or forward roll on huge TRNLOG fixed-length data file

In a very specific case, a record read failed with error **160** on a fixed-length huge (meaning a c-tree Huge Enabled file, which can grow beyond 4GB and has the extended header block) TRNLOG data file after automatic recovery or forward roll.

The number of keys in the index matched the number of data records, but some of the key values did not match the key values formed from the data record image. The key values all differed in the same way: only the key byte that was built over the byte at offset 8 differed from the expected value, and the data record had a value of 0x0 for that byte, while the key value had a non-0x0 value.

The specific situation where this was seen was as follows: A record was deleted and at least one other deleted record (call that record A) was already present on the delete stack. In another transaction, the deleted space was reused and the new record happened to have the same value in the byte at offset 8 of the record as the low-order byte of the already-deleted record (A). In addition, the next 7 bytes of the new and old images also matched. This caused the transaction log entries to not record the new value at byte 8 of the record. That situation caused automatic recovery or forward roll to leave that byte set to 0x0. However, the keys were added using their expected values from the transaction logs, leading to error **160** when reading the record by a key that used that byte of the record.

The logic has been modified to correct this problem.

4.18 Automatic Recovery - "ctfsize failed" message in CTSTATUS.FCS

A message was incorrectly logged to *CTSTATUS.FCS* during automatic recovery:

```
Tue Dec 01 12:19:54 2015
 - User# 00001  tranrcv:     Checking index delete stack.
Tue Dec 01 12:19:54 2015
 - User# 00001  tranrcv: Recomposing index file FAIRCOM.FCS!USER.idx:
Tue Dec 01 12:19:54 2015
 - User# 00001  ctfsize failed: system code = 6  lc = 3  fd = 0
Tue Dec 01 12:19:54 2015
 - User# 00001  FAIRCOM.FCS!USER.idx
```

The logic has been modified to eliminate logging this improper message.

4.19 FairCom Server keeps all transaction logs when deferred index thread starts with nonexistent log 1

FairCom Server did not properly remove its transaction logs if it deferred indexing was being used and the deferred indexing state file was created when transaction log #1 did not exist. For example, a server may have been updated from V10 to V11, so transaction logs existed (but not log 1) and the file *DFRKSTATEDT.FCS* did not exist. In this case, the following messages are observed in *CTSTATUS.FCS*:

1) at server startup:

```
Wed Mar 09 15:44:30 2016
 - User# 00010  Thread Start: Tran File Deferred Indexing Thread
Wed Mar 09 15:44:30 2016
 - User# 00010  L0000001.FCS: 96
Wed Mar 09 15:44:30 2016
 - User# 00010  DeferredIndexer: ERR: Failed to set log position to log #1, offset 0: 96
Wed Mar 09 15:44:30 2016
 - User# 00010  DeferredIndexer: ERR: Failed to get log position: 96
```

2) during server operation:

```
Thu Mar 10 01:59:00 2016
 - User# 00168  The number of active log files increased to: 53205
Thu Mar 10 01:59:00 2016
 - User# 00168  Deferred Indexer: Scan
```

The logic has been modified to correct this.

4.20 Node split in transaction-controlled index file could cause error

A node split in a transaction-controlled index file that did not allow duplicates could cause the key to not be retrievable due to a performance optimization that was first implemented in c-treeACE V10.4. In very rare circumstances, when this optimization is active (V10.4 and newer) updating a record could fail if a number of obscure conditions are true. A specific key update failing with error 4 on a transaction-controlled index file is a symptom of this issue, or an **EQLREC()** or **EQLKEY()** could fail to find a key in the index even though it existed.

This problem is less likely to occur in V11 because of an additional optimization, which adds a check during node split that causes the node to be cleaned up when a large number of key marks exists in the node that is going to be split.

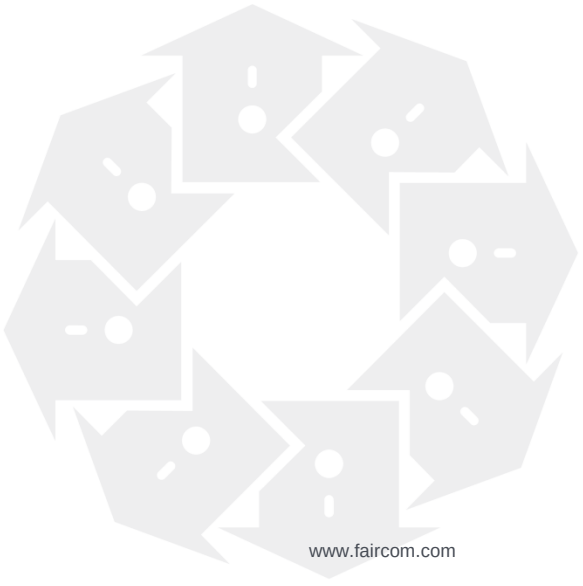
Workaround: Add `COMPATIBILITY NO_UPDMARKS` to *ctsrvr.cfg* and restart FairCom Server is a field workaround for this issue as it disables this optimization.

The logic has been modified to correct this situation.

4.21 Transaction commit improvements

When a record update callback was configured to run at transaction commit time and the record update callback function attempted to perform a transaction-controlled update operation, the commit could fail with error **99**. The logic has been improved to eliminate this error. This also has the advantage that any changes to transaction-controlled files made by the record update callbacks will be part of the committing transaction.

Note: This is a **Behavior Change** because changes made by record update callbacks were not part of the commit before this revision.



5. c-treeACE SQL Server

5.1 Dropping a column sometimes caused problems with the IDENTITY field

The IDENTITY attribute was sometimes lost when performing an Alter Table if the identity field was not the first field of the table. This could cause problems properly identifying the IDENTITY field. The logic has been modified to keep track of the IDENTITY field when columns are dropped.

5.2 SQL cursor position not properly maintained

A problem with cursor scanning was seen when using NEXT or PREV scan operator to retrieve the first of the last row of a record set. This problem has been identified and fixed. The fix is a server-side fix, which applies to all APIs providing cursor capability.

5.3 SQL DECODE scalar function

The code that determines the result type, and in particular the precision (and scale in case of numeric types) was setting the scale to 0. The logic has been modified to set the precision and scale at the maximum value among the various arguments, without losing integer digits.

5.4 DECODE scalar function did not properly set the scale of its result

The DECODE scalar function in V11.* prior to V11.5 may not properly set the scale for numeric values. This caused a SQL statement similar to the following to return incorrect results:

```
SELECT DECODE('A','A',1.25,'B',2.52,3.12)
```

The above statement returned `1` instead of `1.25`.

This behavior has been reverted to match the correct behavior of V10.

5.5 Linux Unicode server failed to enable SQL TCP-IP socket

The following message was logged when the Unicode c-treeACE SQL Server started and connections to SQL were not available: "Daemon:listen failed: Bad file descriptor." The logic has been corrected to eliminate this error.

5.6 Query with predicate "X OR EXISTS(Y)" yielded incorrect results

A SQL query with a predicate of the form:

```
X OR EXISTS (Y)
```

yielded different results from a query of the following form:

```
EXISTS(Y) OR X
```

The logic has been improved to correctly handle this query.

5.7 ctsqlimp imports indexes with ISAM null key check causing wrong query result

The following SQL queries returned different results:

```
select count(*)
from site_account where nodehh_idfk1 not in (select pk from nodehh where nodehh_node_id=1501);
```

and

```
select count(account_owner_identity_role_idfk1 )
from site_account where nodehh_idfk1 not in (select pk from nodehh where nodehh_node_id=1501);
```

This issue occurred only on imported tables and only when there was one or more index with the IFIL inulkey member set to 1 (or **ctdbGetIndexNullFlag()** returned *YES* for the index).

This problem has been corrected by modifying the **ctsqlimp** utility so that it skips conditional indexes, temporary indexes (checks already present), and also indexes with c-tree NULL key check activated.

5.8 Corrected SQL database upgrade conversions

When updating to a new c-treeACE SQL Server, a database connection resulted in the following message:

```
(-20006, 'Column proc_rt not found/specified')
```

A SQL database upgrade was skipped due to an error during the database version test at server startup. This test is now properly enabled.

5.9 SQL database conversion for Unicode databases

Database upgrade conversion did not run for Unicode databases. The symptom was a **proc_rt error** (-20006) returned on SQL connection. This has now been fixed.

5.10 Reduce file descriptor usage

A SQL query failed with error **-20014** "Missing input parameters" for a query with no parameters. *CTSTATUS.FCS* contained:

```
- User# 01968 Error!!!
  Time: 26.01.2016 08:33:00
  Type: FCError
  Function: LTStore::CreateNewFile-2
  Code: -25008
  Message: FC_ERR_IOOPEN - Open file failed.
```

The logic has been modified to reduce file descriptor usage.

5.11 c-treeACE SQL Server ignored "preserve cursor" client request

When the ODBC driver DSN was configured with "preserve cursor ON", the ODBC driver behaved as if the configuration was "preserve cursor OFF." This was a server-side issue, which has now been fixed.

5.12 SQL yielded wrong query result

A query returned the wrong result when it contained a subquery with a Group By (or an Order By) and the outer query contained a Where clause. The logic has been corrected to return the proper result.

5.13 Selecting from a view that referred to a Select statement with right outer join gave wrong results

Selecting from a view that referred to a Select statement that had a right outer join returned incorrect results. The logic has been modified to fix this problem.

5.14 Parameterized query failed if one parameter was a 8192-char string

A parameterized SQL query failed with an overflow error if one of the parameters was a string containing a full 8192 characters. The logic has been corrected to eliminate this error.

5.15 Alter Table on table with identity field failed with error -17749

The following SQL statements caused error **-17749**:

```
create table hot(f1 integer, f2 smallint, f3 bigint, f4 integer, f5 NUMERIC(32,4), f6 VARCHAR(1000), f7
VARCHAR(256), f8 VARCHAR(5000), f9 integer);
insert into hot values(1,1,1,1,1,REPEAT('a',1000),REPEAT('b', 256), REPEAT('c',5000), 77);
commit;
ALTER TABLE hot ADD f10 INTEGER IDENTITY(1,1);
ALTER TABLE hot DROP COLUMN f1;
error(-17749): CT - bad parameter value
```

The logic has been modified to correct this. In addition, the checking for default values has been improved.

5.16 Alter Table VARBINARY default error

When using a "regular" SQL Alter Table (not a "hot alter table"), the following failed:

```
CREATE TABLE Hot(id BIGINT PRIMARY KEY, allow_update BIT, f1 INTEGER );
INSERT INTO Hot (id,allow_update,f1) VALUES (1,0,1319664638);
--works
ALTER TABLE hot add f26 BINARY(22) DEFAULT '2-28-2000 12:01:11.001';
select CASE f26 WHEN '2-28-2000 12:01:11.001' THEN 'PASS' ELSE 'FAIL' END as T23 from Hot WHERE id = 1;
ALTER TABLE hot add f27 VARBINARY(22) DEFAULT '2-28-2000 12:01:11.001';
select CASE f27 WHEN '2-28-2000 12:01:11.001' THEN 'PASS' ELSE 'FAIL' END as T24 from Hot WHERE id = 1;
error(-21025): CTDB - Internal error

and "PANIC - SEG read_record_buffer SQL data format error -20052 reading .\ctreeSQL.dbs\admin_hot field
3 offset 19351 PID 18768"
```

The logic has been corrected.

5.17 SQL - DEFAULT NUMERIC, BIGINT values

The following SQL statement failed:

```
CREATE TABLE Hot(id BIGINT PRIMARY KEY, allow_update BIT, f1 INTEGER );
INSERT INTO Hot (id,allow_update,f1) VALUES (1,0,1319664638);
ALTER TABLE hot add f13 BIGINT DEFAULT 7777777777777777;
select f13,CASE f13 WHEN 7777777777777777 THEN 'PASS' ELSE 'FAIL' END as T10 from Hot WHERE id = 1;
ALTER TABLE hot add f14 NUMERIC(32,4) DEFAULT 77777777777777777777777777.7777;
select f14,CASE f14 WHEN 77777777777777777777777777.7777 THEN 'PASS' ELSE 'FAIL' END as T11 from Hot WHERE
id = 1;
ALTER TABLE hot add f15 MONEY(30) DEFAULT -77777777777777777777777777.99;
select f15,CASE f15 WHEN -77777777777777777777777777.99 THEN 'PASS' ELSE 'FAIL' END as T12 from Hot WHERE
id = 1;
```

The logic has been modified to correct this.

5.18 c-treeACE SQL Server race condition fixed

A race condition existed if SQL import actions occurred simultaneously with the first full SQL client database connection. The logic has been modified to eliminate this problem.

Workaround: Before making any administrative SQL database changes (such as SQL import or dropping a database), ensure a regular SQL client has previously connected after server startup.

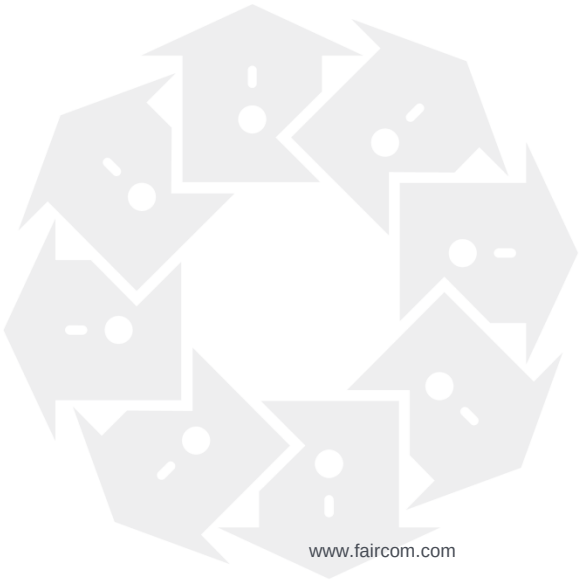
5.19 Unknown indexes identified because CreateFilesetHost renamed templates' internal indexes

Unknown indexes were identified during SQL import because **CreateFilesetHost** renamed some of the templates' internal indexes causing them to appear to be user indexes. The logic has been modified to correctly handle this process.

5.20 Unable to connect to SQL on Windows Server 2003 or XP

Clients were unable connect to c-treeACE SQL V11 over TCP/IP for servers running on Windows 2003 or Windows XP, typically receiving error **-30096** from any client-side application that tried to connect to the c-tree Server. IPv6 support was not correctly being disabled by `SQL_OPTION NO_IPV6` on very old versions of Windows (versions prior to Vista and Windows Server 2008). Connecting to SQL via TCP/IPv4 on these Windows versions required disabling IPv6 support via `SQL_OPTION NO_IPV6` in *ctsrvr.cfg*. The logic has been modified so that the keyword correctly disables IPv6 support on those versions of Windows. Therefore, to use c-treeACE V11 or newer on any release of Windows prior to 2008, add the following to the *ctsrvr.cfg* file:

```
SQL_OPTION NO_IPV6
```



6. Core Engine

6.1 Communications Layer

Client may hang after failed call to ctTempDir() when using TCP/IP protocol

When using the TCP/IP communication protocol, a c-tree client may hang in a call to FairCom Server after a call to **ctTempDir()** returns. The client is waiting on a socket read. The logic has been modified to correct this problem. Similar changes were made to the following functions to ensure that they never return more data than the client expects:

- ctTempDir()
- ctCopyFile
- ctReplCheckFileFilter
- ctReplGetFileName
- ctReplSetFileFilter

Shared memory connection attempt hanged on Solaris 9 and earlier systems

A client that attempted to connect using the shared memory communication protocol on a Solaris 9 or earlier system was found to hang. TCP/IP connections to the same server succeeded. The logic has been modified to handle shared memory connection on these systems.

Shared memory error 128 on AIX when multithreaded process used single-threaded library

Shared memory error **128** was seen on AIX when a multithreaded process used a single-threaded c-tree client library. In a separate instance, when a Java application used the c-tree single-threaded client library on AIX, a c-tree call failed with error **128** when using the shared memory communication protocol. The logic has been modified to correct these problems.

Error 133 connecting to FairCom Server using shared memory when server was busy

Error **133** was sometimes seen when connecting to FairCom Server using shared memory even though the server was running. This was most likely to happen when the server was busy performing multiple file creates. The logic has been modified to correct this situation.

Communication fixes and modifications

Minor changes have been made in the area of communication including the following:

1) When the broadcast feature was used, the server shut down soon after starting up with the following message in *CTSTATUS.FCS*:

```
Tue Jan 19 22:42:08 2016
 - User# 00022  O22 M3 L80 F0 P2710x (recur #1) (uerr_cod=0)
```

The logic has been modified to correct this.

2) The broadcast example in the c-treeACE sample program **ctixmg** did not work because it passed back the address of a stack variable. This has been corrected and the broadcast example in **ctixmg** has been enabled. When server name of "−" is specified, **ctixmg** looks for the server using broadcast.

3) GetServerInfo and GetServerInfoXtd were unresolved when linking with a multi-threaded client DLL that used the IPv6 protocol. This has been fixed in the c-treeACE **mtmake** build utility.

4) The NetBIOS protocol has been removed from being a supported protocol on Windows (client and server). **mtmake** has been changed so that when IPv6 is not requested as a supported protocol, the server makefile/project file doesn't compile the IPv6 source modules.

SQL IPv6 support fixes

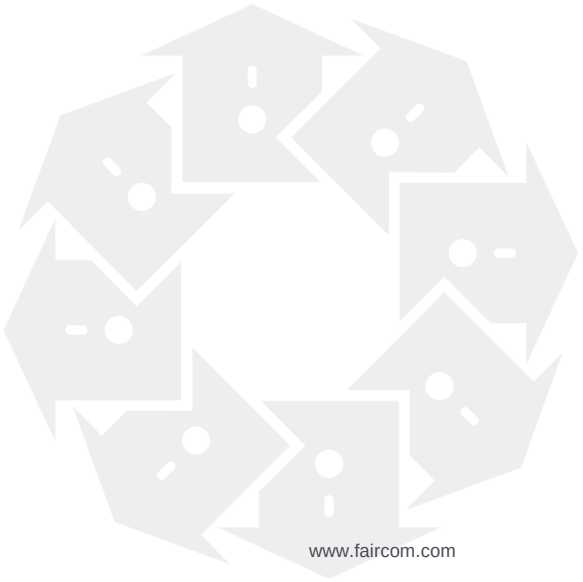
The introduction of IPv6 support on Linux caused two problems:

- On systems that do not support IPv6, the SQL TCP/IP socket was not being created, so it was not possible to connect to the server.
- On systems supporting IPv6, it was impossible to connect to SQL using IPv4.

In both cases, IPv6 support can now be disabled at runtime using the `SQL_OPTION NO_IPV6` keyword solving the problem. Prior to this revision made in June 2016 (c-treeACE Servers with a build date of 160601), the keyword was not effective.

SQL connect failed with error -17749 using LDAP

A SQL client connection attempt failed with error **-17749** when using LDAP authentication. The logic has been modified to correct this.



6.2 Memory Management

Server memory counters reported incorrect values

Server memory counters, SYSCFG and ctSNAPSHOT, reported incorrect values for memory usage in very active systems. The logic has been modified to correct this.

6.3 Indexing

Exception in client-side library when ALCRNG() is called with an out of range index file number

A call to **ALCRNG()** in the c-tree client library may fail with an unhandled exception if the index number (keyno) is outside the supported range of file numbers for the connection. The logic has been modified to check for out-of-range keyno and return error **BKEY_ERR** (80) if the keyno is out of range.

Error 519 or wrong index key counts after automatic recovery (if member updated; host not)

In V11.2 and later, but not including V11.5, it was possible for a key retrieval operation on an index to fail with error **519** or for an index member's key count to be incorrect after the index went through automatic recovery. This only happened if the index contained one or more members in addition to the host index, and if the index file was opened and only members (not the host index) were updated (e.g., by adding and/or deleting keys to the members without also changing keys in the host index, and then c-tree terminated abnormally before the index file was physically closed). The logic has been modified to correct this problem.

CREIFILX8 failed to create non-existent directory on Unix when ctAUTOMKDIR was specified

CREIFILX8 failed to create non-existent directory on Unix when *ctAUTOMKDIR* was specified or the `AUTO_MKDIR YES` keyword was enabled. The logic has been modified so that *ctAUTOMKDIR* works correctly on Unix.

Record update failed with NKEY_ERR (894) when using null key

In c-treeACE V11, an ISAM record update on a file that has an index that uses the null key value option failed with error **894** when the record did not have a key for that index due to the null key option. The logic has been modified to correct this error.

Adding index to a compressed file could fail with READ_ERR (36)

Adding an index to a compressed record data file could fail with error **36**. This error was typically noticed when repeatedly adding indexes to the file. The logic has been corrected to eliminate this error.

Unhandled exception when assembling key value if index definition specified a negative segment length

An unhandled exception was seen in c-tree when creating an index on a string field with a length count if the key segment definition specified a negative segment length. The logic has been corrected to eliminate this problem.

Error 894 (NKEY_ERR) - Deleting record failed if index used null key feature and key was null

Error 894 (**NKEY_ERR**) was seen when deleting a record in a file that used the null key index attribute and the key was null. This issue was created by a new feature added in V11.0. This issue is fixed in c-treeACE V11.2 and later.

Error 14 when opening index with additional members that was copied after quiesce with flush

In V11.2 up to, but not including V11.5, after flushing files and quiescing FairCom Server, a member index may unexpectedly have its update flag still set, causing it to fail to open with error 14 (**FCRP_ERR**). The logic has been modified to correct this.

PRMIIDX/RBLIIDX QTOC_ERR (953) with auto restore points

When automatic restore points were used, a call to **PRMIIDX()** or **RBLIIDX()** could fail with error **953**. The logic has been modified to correct this.

6.4 Dynamic Dump and Backup/Restore

Incorrect key counts after dynamic dump restore

An index that was restored from a dynamic dump backup sometimes had an incorrect key count in its header after the dump restore completed successfully. The logic has been corrected to eliminate this problem. We also applied changes to improve performance of dump restore based on our observations of real-world sample data.

Forward Roll failed for TRANDEP creates

Without the *!SKIP* option, a forward roll failed with error **12** if TRANDEP creates were encountered during the roll. The logic has been modified to properly handle this case.

Dynamic dump restore terminated with internal error when restoring segmented files

The dynamic dump restore utility, **ctrdmp**, sometimes terminated with an internal error such as **7491** or **7495** when restoring segmented files. The logic has been modified to correct this.

Incremental forward roll reported successful termination when restore point not found

ctfdmp reported "FWD: Successful Roll-Forward Termination at Restore Point" when one or more necessary transaction logs were missing and `!RP` was specified. This has been corrected.

ctrdmp - Dump restore failed with error 12

In rare circumstances, running **ctrdmp** (c-treeACE Restore Backup Utility) on a backup failed with error **12** logged to *CTSTATUS.FCS*. The situation was caused by a transaction-dependent file create log entry for a data and index file that was NOT included in the backup, which is fairly uncommon. The logic has been modified to properly handle this situation.

For additional context, recall that a dynamic dump restore consists of the following major phases:

1. Extract individual files from the backup file.
2. Run a recovery on the transaction controlled files.
3. Rollback to the beginning of the dump, as files were copied at different times, so changes made during the dump may be inconsistently applied.

During recovery, a transaction-dependent file create log entry was encountered for a data and index file that was NOT included in the backup, so these files were skipped and not created. Then an index member creation log entry was encountered to add an additional logical index to this skipped file, which required the host index be opened, but the file did not exist. **ctrdmp** was unable to determine that this was an expected state, and treated the missing file as a fatal error.

6.5 Multi-User Standalone (FPUTFGET)

Rare occurrence of "Retried Hdr I/O failed" Messages in High Concurrent FPUTFGET Environments now Eliminated

In *FPUTFGET* mode, "Retried Hdr I/O failed" messages were seen in *CTSTATUS.FCS* when accessing non-huge files with extended headers being updated by many processes at the same time.

The logic has been modified to correct this problem.

Multi-threaded FPUTFGET compile error with Embarcadero

Compiling a multi-threaded FPUTFGET with the Embarcadero compiler failed in *ctbtrd.h*. The logic has been changed to eliminate this error.

Multi-threaded FPUTFGET on Unix - Incorrect lock release with recursive lock support

In the multi-threaded FPUTFGET operational model on Unix systems, a call to release a record lock could release a lock held by another thread when recursive lock support was enabled. This could cause data to become inconsistent, leading to error **158** or **160**. In another situation, faulty unique file ID generation could cause data consistency errors. The logic has been corrected to handle these situations.

FPUTFGET header updates result in errors

Errors such as 30, 69, and 527 were seen on c-tree data and index files in *FPUTFGET* mode.

When closing a file that has its update flag set, there is potential for corruption of the header information in *FPUTFGET* mode. A loop was not correctly locking the header before writing a new value in rare situations. This caused updates to the header to be lost, resulting in a variety of errors. The logic was modified to correct this situation. Note that this only applies to FPUTFGET operations.

FPUTFGET now treats read lock requests as write lock requests

Because FPUTFGET does not support read locks, it now promotes read lock requests to write lock. Before this upgrade, the function that acquired a read lock, did nothing but return success. This could cause problems for an application that used **LKISAM()** with the *ctREADREC* or *ctREADREC_BLK* mode in the FPUTFGET model because no locks were actually acquired.

In FPUTFGET mode, **LKISAM()** now converts read lock requests (*ctREADREC* and *ctREADREC_BLK* lock modes) to write lock requests (*ctENABLE* and *ctENABLE_BLK* lock modes).

FPUTFGET - Corrected variable-length record add duplicate key error on SRLSEG index

In the multi-user standalone (FPUTFGET) operational model, adding a record to a variable-length data file that used data compression or Hot Alter Table sometimes failed with a duplicate key error on the data file's SRLSEG index. The logic has been improved to eliminate this problem.

6.6 Caching

PRIME_CACHE and PRIME_INDEX caused only data or index file to be read into cache

In an obscure situation when the FairCom Server `PRIME_CACHE` option was used on a data file and the `PRIME_INDEX` option was used on an index file that was associated with that data file, only the data file or the index file was read into cache. The logic has been modified to correct this issue.

7. File Management

7.1 FairCom Server displays correct file descriptor requirement value

Prior to c-treeACE V11.2.3, when the `FILE_HANDLES` configuration option was used in *ctsrvr.cfg* on a Unix system, but the operating system file descriptor limit was too low to accommodate the specified number of file descriptors, FairCom Server would log the following message to *CTSTATUS.FCS* and fail to start up:

```
 - User# 00001  ERROR: The hard limit on file descriptors available to this process (<descriptor_limit>)
is lower than the database engine's file descriptor requirement (<required_descriptors>). Either increase
the hard limit, or decrease the FILES or CONNECTIONS settings.
```

This message is incorrect in two ways:

1. The `<required_descriptors>` value that is included in the message is the total of the `FILES` and `CONNECTIONS` settings rather than the `FILE_HANDLES` value.
2. The message indicates that `FILES` or `CONNECTIONS` should be decreased rather than `FILE_HANDLES`.

Starting with V11.2.3, the code distinguishes between these two cases: if `FILE_HANDLES` determines descriptor limit, then FairCom Server displays the following message, using the `FILE_HANDLES` value as the `<required_descriptors>` value:

```
 - User# 00001  ERROR: The hard limit on file descriptors available to this process (<descriptor_limit>)
is lower than the database engine's file descriptor requirement (<required_descriptors>). Either increase
the hard limit, or decrease the FILE_HANDLES setting.
```

Otherwise, the server displays the original message, using the `FILES` and `CONNECTIONS` settings as the `<required_descriptors>` value.

7.2 UQID_ERR (463) on file open

An unusual error that sometimes occurred during a physical open of a c-tree file could cause the connection to hang when closing that file. For example, if the server was quiesced and the quiescing thread attempted to open a file, the file open could fail with error 463 (**UQID_ERR**), because the internal index file used for file uniqueness checking had been closed by the quiesce. That could cause a call to logoff or close the file to hang. The logic has been modified to correct this.

7.3 Avoid infinite loop on a corrupted index file

In rare cases, a corrupted index could result in the FairCom Server background node deletion thread entering an infinite loop while holding locks on multiple index nodes, which could cause operations on this index to hang. We now ensure this operation will eventually terminate.

7.4 PLOW_ERR (712) on record Add over multi-segment partition key

A record Add sometimes failed with error 712 (**PLOW_ERR**) on a partitioned file if the partition rule was defined on a key that consisted of more than one key segment. The logic has been modified to correct this.

7.5 Superfile member error 14 after being restored from Dynamic Dump

Error **14** was seen when opening members of a superfile that was restored from a dynamic dump due to prior modifications of the file redirection logic (`!REDIRECT`). This issue affected c-treeACE V11.2 servers with build dates between October 2016 and February 2017. The logic has been modified to correct this.

7.6 Rebuild or compact failed with FUSE_ERR (46)

Due to a new optimization in V11.0, a rebuild or compact function call sometimes failed with error **46** if files were open that had an overlapping range of file numbers. The logic has been modified to prevent this situation, thus eliminating the potential for this error.

7.7 Rebuild failed with error 775 on replicated file if a connection had the file open

A rebuild or compact of a replicated file sometimes failed with error **775** if a connection had the file open. The logic has been corrected to eliminate this problem.

7.8 Compacting file with Extended field types

Compacting a file using Extended field types CT_TIME_MS or CT_TIMES_MS (time & timestamp with millisecond support) caused the schema to be corrupted in the new file. This could cause a variety of unexpected side effects. This issue has been resolved in this line.

7.9 CreateFilesetHost: Resource copy failed

The c-treeACE FILESET function **CreateFilesetHost()** failed with error **36** on a Huge file. The logic has been modified to correct this.

7.10 File compact or index rebuild sometimes failed with FULL_ERR (39) or KLOD_ERR (58)

File compact or index rebuild sometimes failed with error **39** or **58** when using server rebuild optimizations if the data file contained a sufficient quantity of records to cause the temporary sort work file to exceed 4GB. This was seen when using FairCom Server V11.2 or later with rebuild optimizations enabled. In one case, the following message was logged to *CTSTATUS.FCS*:

```
- User# 00022 CTSORT: write to sort work file F:\tmp\sw014163.00X at offset 4294965308 for 65490 bytes
failed: error 39 system error code 0
```

Workaround: Two workarounds are available: 1) Add `MAX_REBUILD_QUEUE 0` to *ctsrvr.cfg* to disable the rebuild optimizations and restart c-tree Server, or 2) rebuild using a standalone (non-client/server) rebuild utility.

The logic has been modified to correct this problem (the workarounds are not necessary if FairCom Server is updated to use V11.5 or later).

7.11 ctVerifyFile() sometimes failed with error 160 on a 64-bit memory file or error 123 on a huge file

A call to **ctVerifyFile()** sometimes failed with error **160** on a 64-bit memory file or error **123** on a huge file. The logic has been modified to eliminate this problem.

7.12 Rebuild or compact failed with error 608 on file with SCHSRL key segment

Rebuild or compact sometimes failed with **IAIX_ERR** error 608 when the file had a SCHSRL key segment. The logic has been modified to correct this error.

7.13 Possible buffer overflow generating temporary file names on Windows

An exception occurred when generating file names for temporary file names on Windows due to a buffer overflow. The logic has been corrected to address this issue.

7.14 Rebuild/compact with errorOnCorruptIFILoption enabled causes VLENGTH files to fail with DCPT_ERR (1107)

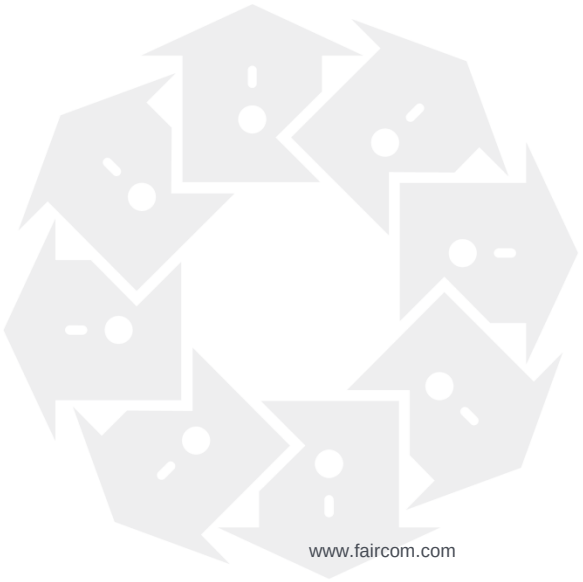
A rebuild or compact with *errorOnCorruptIFILoption* enabled caused all VLENGTH files to fail with **DCPT_ERR** error 1107. The logic has been modified to correct this.

7.15 IERR_COD (923) compacting a transaction-controlled data file

Error **IERR_COD** (923) was seen when compacting a transaction-controlled data file if `COMPATIBILITY LOCK_EXCL_TRAN` was specified in *ctsrvr.cfg*. This is an internal error, caused by a recent optimization to the internal FairCom Server Transaction Logging. In this particular case, with compression and this compatibility switch in place, an internal deadlock could occur. If this happened, error **923** and the following message was logged to *CTSTATUS.FCS*:

```
DEFER_OPNTRAN: file's semaf already acquired
```

The logic has been modified to eliminate this error.



8. Fixes for Extended Features

8.1 Record update callback function could unexpectedly change the current record offset

Following a record add or update that called a record update callback function, the current record offset might differ from the offset of the added or updated record. This could cause **ctdbWriteRecord()** to return error 767 (**FALG_ERR**), or it could cause application errors due to using the wrong record offset. The logic has been enhanced to prevent this.

8.2 Fixed-length record update or delete in IICT failed with error 114 if the outer transaction added the record

An update or delete of a fixed-length record in IICT failed with error **114** if the record was added by the outer transaction. In this situation, the record update or delete should not be allowed, because a standard restriction of IICT is that it does not permit a file that was updated by the outer transaction to be updated in an IICT. In this case, the update or delete should fail with error 935 (**IICT_FIL**) instead of error **114**. The **RWTREC()** and **DELREC()** logic has been changed to check if the file has been updated in the outer transaction before checking if the record is marked deleted.

8.3 Time handling in expressions

Filters applied to CT_TIME fields do not behave properly and some functions dealing with time have been found to return incorrect results. This problem was introduced when we added support for time with milliseconds in V11. The logic has been improved to properly handle time with and without milliseconds.

8.4 Batch Processing

ctdbEndBatch() call sometimes freed record locks even with CTBATCH_LOCK_KEEP option

A call to **ctdbSetBatch()** with the *CTBATCH_LOCK_KEEP* option did not prevent **ctdbEndBatch()** from releasing locks on records acquired by a batch read. Here is an example that demonstrates this behavior:

```
if ((rc = ctdbSetBatch(pRecord,CTBATCH_GET | CTBATCH_PHYS | BAT_RET_BLK | CTBATCH_LOCK_WRITE |
CTBATCH_LOCK_KEEP,0,rbufsiz))) {
 printf(""Error: Failed to initialize batch: %d\n"",
  rc);
 goto err_ret;
}
rc = ctdbNextBatch(pRecord);
rc = ctdbEndBatch(pRecord);
/* locks have been released--unexpected */
```

You can use the **ctstat** utility's *filelocks* option to check that the locks were still being held after the **ctdbEndBatch()** call. For example:

```
ctstat -filelocks myfile.dat -h 1 -i 2 -u ADMIN -p ADMIN -s FAIRCOMS
```

The logic has been modified to address this issue.

Physical order batch retrieval with record filter returned too few records

A physical order batch retrieval with a filter did not always return all the expected records. When called using a physical order batch retrieval combined with a record filter (mode parameter included both BAT_PHYS and BAT_VERIFY), **DoBatchXtd()** (used by **ctdbSetBatch()** and related APIs) sometimes terminated when the first record excluded by the filter was encountered. In some cases, an internal error was not properly cleared, causing the batch call to return with fewer records than expected. The logic has been modified to correct this.

ctdbInsertBatch may cause heap corruption

The use of **ctdbBatchInsert** was found to cause heap corruption in rarely encountered situations involving variable-length records if the number of records was very close to the size of the batch buffer. The logic has been rewritten to correctly handle this situation.

8.5 Sequence Create error

Sequence Create operations could fail with **SEQINC_ERR** (907) or **FBEG_ERR** (553) for some values of initial sequence and increment, even though the values should have been allowed. The logic has been modified to correct this.

8.6 PartitionAdmin() returns RSYN_ERR (747)

In a client/server environment, a call to **PartitionAdmin()** could fail with error **RSYN_ERR** (747, partition rule syntax error), even though the partition rule was properly formed. The logic has been corrected.

9. Relational c-treeACE SQL

9.1 SQL

SQL query that generates dynamic index corrected

In a very rare and specific case (an equality search was done on an index allowing duplicates and additional boundary conditions were satisfied), a particular SQL query returned incorrect results. A variable initialization within internal sort logic was found to be incorrect. This condition is now properly checked and logic has been modified to prevent this problem.

SQL queries never returned when synonym pointed to non-existing table

When "t" was a synonym for the table "owner.t" and "owner.t" was not present, queries on "t" never returned. This situation could arise only when using **ctsqlimp** options in previous releases. SQL queries no longer infinitely loop on this unexpected condition. This situation returns error code **SQL_ERR_NOTBL**.

SQL table open error 4120 (or -21120)

Attempting to sqlize (or open in SQL) tables that had indexes on fields mapped to Boolean, date, time, timestamp, or that used UTF string encoding in SQL failed with error **4120** or **-21120**. The logic has been modified to correct this.

SQL STORAGE_ATTRIBUTES parsing issue

The last attribute in the STORAGE_ATTRIBUTE option of CREATE TABLE was ignored if it was not terminated with a semicolon ("`;`"). For example, `'nothuge'` was ignored in the following statement:

```
CREATE TABLE ... STORAGE_ATTRIBUTE 'nothuge'
```

The logic has been modified to be in compliance with SQL guidelines, so that the semicolon is required to separate multiple attributes, but it is not strictly required at the end of the list.

Internal SQL error on REPEATABLE READ query

A query was failing with SQL internal error (**-20000**) when run at the REPEATABLE READ isolation level. The logic has been modified to correct this.

ctscmp failed with error 608 during index rebuild

Due to a recent change in V11.0, rebuilding a superfile member with SRLSEG failed with **IAIX_ERR** (608) during index rebuild. The logic has been fixed in V11.5.

SQL query with left outer join and parameters now parses correctly

A situation was discovered in which a query using a left outer join for which the "describeparam" call returned the wrong number of parameters and incorrect information about the parameters. This could occur in ODBC and potentially any other relational API. When a query with a left outer join returns only columns from the left table in the resultset, the outer join may be superfluous and can be eliminated from the query, in such a case the parameters affecting the right table were removed as well hence the problem.

The logic has been modified so that, when a left outer join that contains parameters is optimized away, the parameter information is not lost.

SQL query with Left Outer Join fixed

A SQL query with a Left Outer Join failed with internal error (**-20000**) due to an unexpected situation during cost evaluation. The Join cost evaluation logic has been improved to correct this problem.

9.2 c-treeACE JDBC

Unexpected JDBC 26046 (closed resultset) exception calling getWarning()

A c-treeACE JDBC application that worked properly with earlier FairCom RTG releases simply looping through a resultset failed with error code **26046** calling **getWarning()** after fetching the last row. Recent modifications to the c-treeACE JDBC driver introduced this problem, which has been now solved.

c-treeACE JDBC ResultSet.isAfterLast incorrectly returned true with the isAfterLast() positioning method

In c-treeACE JDBC, **ResultSet.isAfterLast** incorrectly returned `true` in these cases:

- A new result set after reusing a prepared statement.
- Records remain to be fetched and calling **next()** returned more records.

A proper false value is now returned in these usages.

JDBC catalog handling in DatabaseMetaData was not compliant with JDBC standard

The return values from these functions was not correct and was causing issues using third-party reporting tools, such as Jasper Reports. The returns have been corrected as follows:

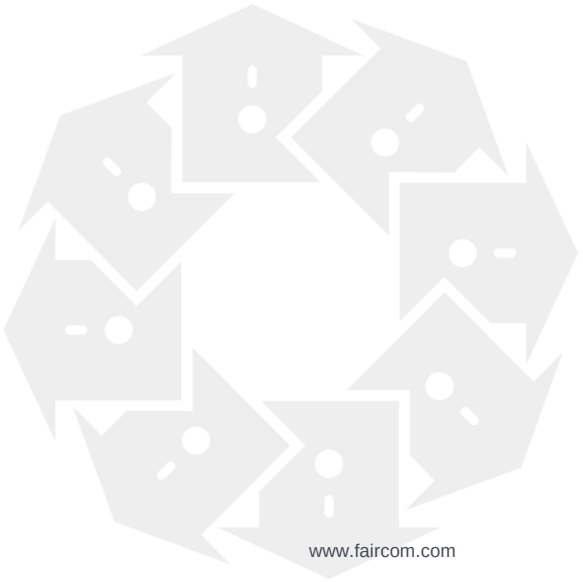
- **DatabaseMetaData.isCatalogAtStart()** returns `false` indicating that c-tree does not support specifying catalog in fully-qualified table names (FQN).
- **DatabaseMetaData.getCatalogSeparator()** returns **""**.
- **DatabaseMetaData.getMaxCatalogNameLength()** returns `0`.

JDBC Connection to set warnings instead of failing with exception on method returning a resultset

This revision resolved a discrepancy between the JDBC driver specification and the JDBC API documentation about the behavior of the following methods:

- **createStatement**
- **prepareStatement**
- **prepareCall**

The previous logic followed the JDBC API documentation and failed with an exception because these methods were not supported. The logic has been modified to follow the JDBC specification, which says to set a warning if the operation is not supported.



9.3 ADO.NET

ADO.NET Provider may inappropriately retain SQL cursors for ExecuteNonQuery statements

The ADO.NET provider sometimes inappropriately retained SQL cursors for **ExecuteNonQuery()** statements. The logic has been modified to correct this behavior.

ADO.NET returned wrong number of decimals

The .NET version of the c-treeACE SQL Explorer truncated DOUBLE values. The ADO.NET driver was truncating the number of decimals to just two places. The logic has been modified to correct these issues.

ADO.NET Entity Framework provider issues with DDL generation

The ADO.NET Entity Framework provider exhibited an issue with DDL generation. When a user tried to use "Generate Database from Model" in Visual Studio, a few column types were not properly handled and caused an exception. The logic has been modified to correct this.

ADO.NET error when reading non-Unicode varchar fields from Unicode FairCom Server

An error was seen when retrieving varchar columns content with the ADO.NET driver and Unicode server: **Error: -30100 - EOF while reading chars**. The logic has been modified to correct this error.

ADO.NET client-side crash fixed

A possible client-side crash was seen when connecting with the ADO.NET provider using Shared Memory on Windows. The logic has been corrected to fix this.

ADO.NET EF6 provider support for V11 servers

The ADO.NET EF6 provider was returning an error on c-treeACE V11 servers saying "This server version is not supported." The logic has been updated to indicate that V11 servers are supported.

ADO.NET connection pooling

An application using ADO.NET thread pooling could sometimes freeze. The logic has been enhanced to better handle thread pooling.

9.4 ODBC

Nodename + Process ID used to identify ODBC applications

The ODBC API does not have an interface to specify extra identifying information that our server can display to identify a specific application using the ODBC driver. By default, c-tree Server set the "nodename" to "SQL:<dbname>" when a new connection was made.

It is now possible to display a client process ID connected to the server. On the server side, when `SQL_OPTION PID_IN_NODEID` is specified in **ctsrvr.cfg**, the client process ID is appended to the default nodename as "SQL: PID <client pid>". If the client application does not pass the PID information, the original default nodemane is still used. The default node name can be overwritten at any time after connection by a client calling the **fc_set_nodename** (https://docs.faircom.com/doc**/sqlops/56378.htm**) built-in stored procedure.

This nodename string is displayed for all SQL client APIs and tools except for JDBC-based APIs/tools.

Log statements for DESCRIBE operation

The ODBC driver has been updated so that statements sent by the DESCRIBE operation are now being logged. The logging logic has been modified to always specify the operation the server is executing to make it easier to reconcile statements logged more than once.

ODBC SQLTables call failed with syntax error

A SQL database tool failed with a syntax error after connection via ODBC. In an ODBC program, the following call fails with "syntax error":

```
SQLTables(hStmt "%, SQL_NTS, "", SQL_NTS, "", SQL_NTS, "", SQL_NTS)
```

The logic has been modified to correct the syntax error.

ODBC driver allowed "%" as catalog in ODBC SQLTables and SQLColumns calls

The FairCom ODBC driver failed with the following call:

```
SQLTables ("%",SQL_NTS.....)
```

The ODBC driver logic has been updated to improve the specific check for the first parameter being `""` or `"%"`.

9.5 c-treeACE PHP

PHP driver improvements

The c-treeACE PHP driver has been improved to eliminate issues in the following areas:

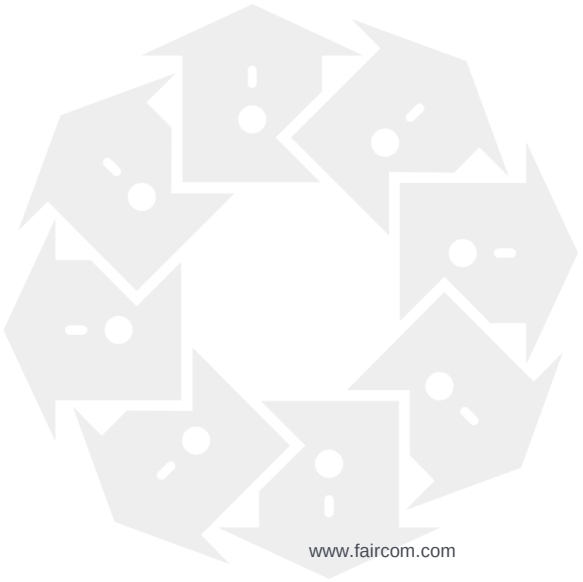
- Values can be now inserted to LVARCHAR columns with **PDO::PARAM_STR** parameter binding.
- Binding NULL as a parameter for several datatypes.

- Rollback of an explicit transaction (autocommit was always in effect unless it was explicitly disabled).
- **BindParam()** binding by reference.
- BIGINT parameters on big-endian systems.
- Binding parameters to non-LONG columns with **PDO::PARAM_LOB**.
- Binding parameters to datatypes REAL, DOUBLE, NUMERIC, and MONEY.
- Binding parameters to BINARY and VARBINARY types.

Additionally, "memory usage" issues have been addressed.

9.6 Embedded SQL function dh_conv_data() causes client crash

A client program calling the Embedded SQL function **dh_conv_data()** caused a SIGSEGV Segmentation fault. The logic has been correct to address this problem.



10. High-Velocity c-treeACE APIs

10.1 FairCom Low-Level

Unexpected Deleted Data with low-level calls in client/server model

In the client/server model, when using the low-level functions **RETREC()** and **RETVREC()** (**ReleaseData** and **ReleaseVData**), unexpected record offsets beyond the 4GB boundary could be marked deleted and available for reuse because the high word set by **ctSETHGH()** was not being set properly. The logic has been improved to correct this. Note this is a rare situation as the low-level API imposes additional network overhead, so very few applications are expected to be impacted by this issue.

10.2 c-treeACE ISAM

ISAM-level client call returned error 128

An ISAM function call in the c-tree client library sometimes returned a communication error such as **128** when the `isam_err` state variable was zero. This was seen when a client call failed with a communication error. The logic has been corrected so that the error is returned properly.

ADDREC DELFLG_ERR (31) following recovery

In extremely rare situations, it was possible for the delete stack to become corrupted on a data file after it went through automatic recovery. In these obscure scenarios, ADDREC could return a **DELFLG_ERR** error (31) following recovery. ADDREC and NEWREC failed with error 31 until the table was rebuilt. Instead of returning **DELFLG_ERR**, c-tree now discards its list of deleted records and retries the operation. This should prevent the error and allow the call to proceed normally. The following is logged to *CTSTATUS*:

```
WARNING: an invalid delete stack has been reset to zero for fixed length data file
```

This message indicates that all the space occupied by deleted records in that file at that point in time is lost and will not be re-used until the file is rebuilt or compacted.

Note: This is a Compatibility Change.

ITIM_ERR (160) Reading records another connection is deleting

When reading records from a fixed-length record transaction-controlled data file, the read sometimes failed with error **160** if another connection was deleting the records. This error could occur even if the reader requested a read or write lock on the record. The logic has been modified to correct this.

Workarounds:

1. Add `ITIM_RETRY_LIMIT 100` to *ctsrvr.cfg*. This increases the error **160** retry limit.

   or

2. Change the application's record deletion code to commit the transaction with mode ctKEEP, and then call **LKISAM(*ctFREE*)**. This causes the deleted records to remain locked until later in the transaction, reducing the likelihood of the reader encountering error **160**.

ITIM_ERR (160) or wrong record on Next Record with ISAM key buffers disabled

After disabling ISAM key buffers, reading a record using an index, then re-reading the record, if an update then caused the record to move, a call to read the next record could fail with error **160** or return the wrong record.

Note: This issue only affected applications that use the option to disable ISAM key buffers with code that supports automatically refreshing the ISAM key buffers when needed. This feature is enabled in V10.4.1 and later.

The logic has been modified to eliminate this problem.

Next Record fix using co-files

After updating a key value in client/server mode, a call to read the next record from an index could return the wrong value in certain, very specific situations.

The following specific steps are required to reproduce this issue:

1. Open a variable-length TRNLOG data file and its associated index file at ISAM level two times in one connection (this is referred to as "co-files"). The data file has two records that we will refer to as record "A" and record "B."
2. Call **FRSVREC()** on the first instance of the index to read record "A".
3. Call **FRSVREC()** on the second instance of the index to read record "A".
4. Call **RWTVREC()** on the second instance of the data file to update record "A", causing the record to move without changing the key values.
5. Call **NXTVREC()** on the first instance of the index. We expect this call to return record "B", but it returns record "A".

To improve the efficiency of next key and previous key operations, FairCom Server maintains "current low-level key" state variables for each open index file. It also has logic to identify situations in which it must ignore those variables and perform a full tree search for the next or previous key.

The logic has been enhanced to correctly handle this situation: The logic has been extended so that, when adding a key when co-files are in use, it will check if any of the co-files are positioned on this same index node and perform the appropriate key search.

FRSREC() failed with error INOT_ERR (101)

A call to **FRSREC()** to read a record from a non-huge variable-length data file having a data record length defined to be zero (which is a very unusual event) failed with error **101**. The logic has been modified to correct this problem.

Incorrect parameter specified in SETFLTR() and SETFLTRN()

The documentation for the ISAM Filters had a minor discrepancy between **SETFLTR()** and **SETFLTRN()** regarding the first parameter:

SETFLTR:

```
COUNT SetDataFilter(COUNT datno, pTEXT condexpr)
```

SETFLTRN:

```
COUNT ctDECL SETFLTRN(COUNT filno, UCOUNT fltnum, UCOUNT fltopts, pTEXT expression);
```

The documentation and the descriptions in the code have been changed to name this parameter *datno* instead of *filno*. The use of the parameter *datno* indicates these functions operate over a data file only. When *filno* is used, it indicates the function can operate over a data file and and index file.

VRLN_ERR (149) in RWTVREC()

RWTVREC() allowed a caller to update a record to a length smaller than the data file's defined minimum record length (fixed-length value). This caused subsequent update attempts to fail with error **149**.

RWTVREC() now specifically checks for this condition and returns error **149** any time the data record length specified by the caller is smaller than the defined data record length for the data file or if the data record length specified by the caller is zero.

PUTHDR mode ctSUSSRLSEGhdr ignored on client side

When ctSUSSRLSEGhdr was enabled in client/server, the client's record buffer was modified after a successful ADDREC/ADDVREC. The logic has been corrected so that ctSUSSRLSEGhdr temporarily disables SRLSEG auto-numbering, which is the expected behavior.

ctstat -isam - Improved accuracy of ISAM counters

The FairCom Server ISAM record read counter value could be smaller than expected on a multiprocessor system when multiple connections were reading records at the same time. The logic has been enhanced to ensure that these counts are accurate.

The ISAM counter state variable has been changed to a distributed set of state variables. The number of counter instances in the set is set by the `DIST_ISAM_COUNTER_BUCKETS` configuration option, which defaults to 8 and has a maximum supported value of 1024. By maintaining multiple counter buckets, and summing the individual buckets prior to displaying the result, the performance impact of this change has been minimized.

10.3 c-treeDB “NAV” API

c-treeDB is an easy-to-use database programming API encapsulating many tedious internal details and supplying powerful control over data record navigation. c-treeDB API layers are now known as "Navigational," or NAV, better articulating this advanced database interface. NAV support includes these programming frameworks: C, C+, Java, .net and, with V12, Node.js (Python will be coming soon!).

Select FairCom DB API functions improperly allowed fields larger than 65535 bytes

The FairCom DB API function **ctdbAddField** takes a VRLEN (signed 4-byte integer) for *FieldLength*. This implies it will accept a value greater than the maximum stored in the DODA, which is 65535. Although no error was generated when this function was called with a length greater than 65535, the actual length stored in the DODA overflowed, resulting in unexpected behavior.

To prevent this, *FieldLength* checks have been added to the following functions:

- **ctdbAddField**
- **ctdbAddVirtualField**
- **ctdbInsField**
- **ctdbInsFieldByName**

These functions now return **CTDBRET_ARGSMALL** (4006) if the length is less than 0 and **CTDBRET_TOOBIG** (4043) if the length is greater than 65535.

ctdbDeleteTable on missing table failed with INOT_ERR instead of CTDBRET_NOSUCHTABLE

ctdbDeleteTable() called on a table that did not exist in the dictionary failed with error **INOT_ERR** instead of **CTDBRET_NOSUCHTABLE**. The logic has been modified to return the proper error, **CTDBRET_NOSUCHTABLE** (4023).

Note: This is a **Behavior Change**.

ctdbGetRecordCount sometimes returned incorrect count

The FairCom DB API **ctdbGetRecordCount()** function sometimes returned an incorrect count when it was called on a table without a $ROWID, $RECBYT, or unique index, and the first index was conditional or supported null keys. The logic has been modified to correct this obscure situation.

ctdbRenameTable may not rename index having the same name of the table

After a FairCom DB API **ctdbAlterTable()** call was used to add a new index, a **ctdbRenameTable()** call failed to rename the default index associated with the table (default index containing rowid or recbyt or even an index created together with the table without assigning a filename). The logic has been modified to handle this situation correctly.

ctdbRenameTable did not properly rename indexes containing full path

An index that had the same name as the table was not renamed by a successful **ctdbRenameTable()** call if the index name contained a full path. The logic has been modified to correctly rename the index in this situation.

ctdbStringTo* functions fail with 4029 (invalid date)

The **ctdbStringToDate** function, and other **StringTo*** time-related functions, failed to identify certain strings as valid. The dash ("-") was not properly handled as a part separator in dates and returned **CTDBRET_INVDATE** (4029). For example, the string "`01-01-2017`" was not considered a valid date. The logic has been modified to correct this behavior so that dashes can be used as separators in dates.

ITIM_ERR (160) reading a record that used Unicode ctKSEG_STYP_PROVIDED key segment

Error **160** could occur when reading a record that was added using FairCom DB API if the record's last field was a length-counted string data type (CT_PSTRING, CT_2STRING, or CT_4STRING) that had an index segment that used the Unicode key segment mode (UNCSEG) with extended key segment source type of ctKSEG_STYP_PROVIDED or ctKSEG_STYP_UTF8, and the field data had zero length.

Other possible symptoms include: error **INOT_ERR** (101) when searching for a key matching a particular value, error **KDUP_ERR** (2) when adding a record, and the wrong collation (ordering) of key values.

The logic has been corrected to handles this situation properly.

Unicode conversion failed with VBSZ_ERR (153)

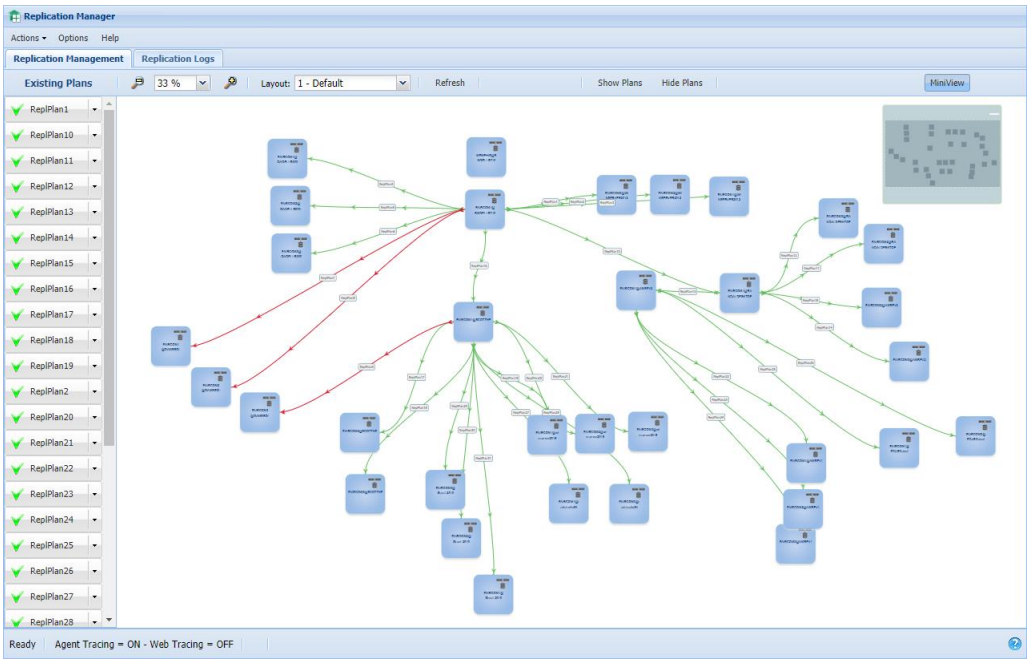
The FairCom DB API functions **ctdb_u16TOu8()** and **ctu16TOu8()** may fail with **VBSZ_ERR** (153), or read past the end of the source array on non-Windows systems. The logic has been modified to eliminate this error.

11. Replication

Now available with FairCom DB V12, FairCom RTG V3, and FairCom Edge V3, the next phase of FairCom replication brings new capabilities to effortlessly tackle the most demanding of replication requirements:

- **Dynamic Data Sync** - The initial synchronizing of data between a source and target is no longer a manual process. Data synchronization is now under Replication Manager control and completely automated when deploying replication plans between servers.
- **Replicated File Operations** - File create, delete and rename operations are now under replication control.
- **Integrated Replication** - The new, advanced version of the Replication Agent is now a plug-in that runs inside the FairCom Database Process.

- **Synchronous Replication** - For mission-critical environments, this ensures data is replicated before returning a commit to the application, eliminating data loss and out-of-sync data.
- **Parallel Replication** - Replication is now multi-threaded for greater speed. You control the level of parallelism to meet your tolerance for latency, and to match the capabilities of your CPU. Parallel replication honors the order of dependent transactions to ensure full ACID compliance.
- **Replication Manager** - This browser-based interface does all the work to set up replication. It backs up source tables, creates tables on the target server, restores tables to the target server, creates the ongoing replication task, starts the replication task, and catches up all activity when replication is paused and restarted.



FairCom Replication has made such great strides in this release, that it deserves its own, dedicated book: **Replication Concepts (https://docs.faircom.com/doc/cluster-for-scalability-availability/).**

11.1 Replication Agent Unique Index improved

Changing the `REPL_SRLSEG_ALLOW_UNQKEY` option on the source server, or using a different `REPL_SRLSEG_ALLOW_UNQKEY` setting on source and target servers could cause errors replicating changes that had been written to the source server's transaction log prior to changing that option. This could happen if the data file had a SRLSEG index, such as a FairCom DB API ROWID index. Record updates and deletes might fail with error **101** or **4**, or the wrong record might be updated or deleted. The logic has been modified to correct this.

11.2 Replication Agent terminated with error 938 decompressing a record image

When reading log entries for record updates on a replicated compressed data record file, the Replication Agent sometimes logged the following message to *ctreplagent.log* and terminated:

```
ERR: Failed to retrieve next change: 938

The source server CTSTATUS.FCS shows this message:

 - User# 00019  ctrepl: Failed to set record image in operation list entry: 938
```

The logic has been modified to eliminate this problem.

11.3 Replication Agent LOPN_ERR (96)

Replication Agent terminated with **LOPN_ERR** (96), "Could not open previous log: 96," in source server *CTSTATUS.FCS*. The same symptom can occur with the deferred indexing transaction file thread.

WORKAROUND: For the Replication Agent, add COMPATIBILITY NO_REPL_DEFER_TRAN to the source server's configuration file.

The logic has been corrected to eliminate this problem.

11.4 Replication Agent failed to start with error -1

After the source server went through automatic recovery and the Replication Agent reconnected, it sometimes failed to start with error **-1**, due to an unexpected condition introduced by the Deferred Index support added in V11.0. The following message was shown in *CTSTATUS.FCS* on the source server:

```
ctrepl: Found tran in redo list with non-ENDTRAN status: -1
```

The logic has been modified to correctly handle this situation.

11.5 Socket timeout TRSP_ERR (809)

Applications that use c-tree's client-side socket timeout feature, including the Replication Agent (by using the `socket_time` configuration option), were found to shut down with error **809** even though the call to **ctReplGetNextChange()** did not exceed the specified socket timeout. The logic has been modified to correct this issue.

Note: This modification affects the Replication Agent and any client application that uses c-tree's client-side socket timeout feature.

12. Administrative Utilities

12.1 ctrdmp BIDX_ERR (527)

A restored backup using **ctrdmp** failed with error **527** and logging the following entries to *CTSTATUS.FCS*:

```
User# 00001 Beginning automatic recovery process
Wed Feb 01 09:56:49 2017
User# 00001 insexp81 @: 235
Wed Feb 01 09:56:49 2017
User# 00001 Unexpected key comparison results: ct_tky 0 ct_tkp -1
Wed Feb 01 09:56:49 2017
User# 00001 ./restore/myfile.idx: 527
Wed Feb 01 09:56:49 2017
User# 00001 Automatic recovery terminated with error: 527
```

c-treeACE Servers efficiently reuse index node space when all keys from a node are deleted from a node. “Empty” nodes are placed on a list and made available for future reuse avoiding expensive file size extensions. During a dynamic dump, this list is written and persisted to the backup file at the beginning of a backup.

However, a rare situation (an index leaf node split occurred while the index file was being dumped with an existing delete node list preserved and a node from the list now reused) could cause a bad node reference on the list when that backup was restored.

Index nodes are now blocked from reuse during the dynamic dump operation to prevent interference with potential subsequent node reuse.

12.2 ctquiet - Add ctQTfailAfterTimeout mode

The **ctquiet** utility has been modified to support the new **ctquiet()** *ctQTfailAfterTimeout* mode. The **ctquite** utility now has a new option:

- *-x* - Quit **ctquiet** if quiet transaction state isn't achieved within *-t* seconds; all files flushed; no transactions are terminated.

12.3 ctsqlimp - Correct SQL column widths for imported CT_MONEY columns in ctslqimp

An imported ISAM CT_MONEY column incorrectly imported as MONEY (4,2) resulting in overflow/underflow errors when querying columns with large values. The logic has been corrected.

12.4 ctstat -i X

When the **ctstat** utility was run using the *-i X* switch, **ctstat** *-isam -i 60 1* printed a line of statistics (in the reported case, always zeros) then waited 60 seconds and exited with no further output.

In general, when using the *-i X* switch, the utility waits *X* seconds before exiting even if there are no more activities to perform.

The logic has been modified so that **ctstat** *-isam -i 60 1* now prints the first line and then exits immediately instead of waiting 60 seconds with no reason.

12.5 cttrnmod UNQK_ERR (775)

When the **cttrnmod** utility is used to change the replication attribute of a c-tree data file and the file cannot be opened at the ISAM level, **cttrnmod** reports error **775** rather than a file open error, which obscures the real problem.

The utility has been modified to error out when changing the replication attribute of a data file if the data file cannot be opened at ISAM level. The utility now also reports the file number and system error code when the file cannot be opened. Example:

```
C:>cttrnmod set repl=off -f files.txt
c-treeACE(tm) Version 11.4.0.27619(Build-170403) Change Transaction Mode Utility
Copyright (C) 1992 - 2017 FairCom Corporation
ALL RIGHTS RESERVED.
Disabling replication for files listed in file files.txt...
  Replicate  Tranmode  Filemode  Filename
  ---------  --------  --------  --------
Error: Failed to open file sx3990.dtc: error 12 on file 1 (syserr 2)
   0 Data Files Updated
   1 Error
```

12.6 Locale support added to dbdump, dbload, dbschema

It was reported the c-treeACE SQL data dump utility, **dbdump**, failed when applied to a table containing UTF-8 characters: **ERROR : Data conversion error**.

These characters were not defined in the local locale and defaulted to 7-bit ASCII. A local locale is now set using the system locale API call so that data conversion routines now handle data outside 7-bit characters. This relies on **setlocale(***LC_ALL,""***)** behavior being supported, which falls back to the locale settings (*LC_CTYPE*, *LC_COLLATE*, etc.) defined in the environment. The c-treeACE SQL **dbdump**, **dbload** and **dbschema** utilities were all updated in this manner.

12.7 ctidmp utility FTYP_ERR(53)

The **ctidmp** utility terminated with error **53** when reading a large dump file. The utility has been modified to correct this error.

13. Notable Compatibility Changes

13.1 Preventing Possible Data Loss with Compact & Rebuild Operations

In most cases, compact and/or rebuild operations perform as expected: scanning your data file for potential invalid record headers, fixing those it finds invalid, and finally writing out a new compressed file and rebuilding indexes. Recently, a case was investigated where our default behavior should not be used due to potential data file corruption.

In a few rare and select cases, FairCom identified situations where compacting a **variable-length and (typically) non-transaction enabled (non-TRNLOG) file**, using the **ctcmpcif** compact utility or any of the **CompactIFile** functions, can fail, potentially, and unexpectedly, also altering your original file being compacted such that further attempts result in data loss.

Typically, it is only expected to see this situation occur with non-TRNLOG enabled files. However, in case #2 discussed below, a valid TRNLOG enabled file data file was found to exhibit this very unusual behavior. This raises an important point – if you encounter situations where you believe you must rebuild TRNLOG enabled files, please contact FairCom, as this is a highly unexpected event and we want to fully understand the context and nature of your data integrity situation. **The recoverable nature of TRNLOG files should always maintain full data integrity through nearly any circumstance.**

We have modified prior behavior to take a much more conservative approach protecting your original data state. Compact and rebuild (which can produce similar results) facilities listed below have been altered such that, when any data file integrity issue is detected, default behavior is now to stop further operations. In addition, no changes are made to existing original data records.

Note: This is a compatibility Change.

Full List of Functions Changed

The following utilities and function calls now default to stopping if an error is encountered:

ctcmpcif – Compact utility
ctrbldif – Rebuild utility
GUI tool - Dr. ctree has compact and rebuild functionality

Long function name	Short function name
CompactIFile()	CMPIFIL()
CompactIFileXtd()	CMPIFILX()
CompactIFileXtd8()	CMPIFILX8()
RebuildIFil()	RBLIFIL()
RebuildIFileXtd()	RBLIFILX()
RebuildIFileXtd8()	RBLIFILX8()

Default Behavior Changes the Original File

Prior to c-treeACE V11.5, the default behavior for compact and rebuild operations over variable-length record data files was to scan for invalid record headers before compacting a file. If the logic found invalid headers, it attempted to fix the headers in the original file. If the operation failed at this point, the original file could become corrupted. Two significant classes of situations are known to arise:

1. In the vast majority of cases, this can only occur if the original data file is already corrupted due to invalid and irrecoverable record headers. Because the original data file is corrupted, restoring a backup is recommended.
2. In some rare cases, it is possible the operation could exhibit vulnerability caused by mistaking old marks or binary record data as valid record header marks. In a very specific case, compact operations can treat space management nodes with inuse length 0, while logically a correct value, as invalid. This case could lead to data loss in a valid file, making it necessary to restore from a backup.

To ensure the second case does not cause data loss, observe our recommendations below and always back up all concerned data files before compacting or rebuilding.

Compact and Rebuild Best Practice

To avoid potential data loss, FairCom strongly recommends only using our latest c-treeACE V11.5 compact and rebuild logic that default to stopping if a corrupt data record header is encountered, and returns error **DCPT_ERR** (1107) rather than attempting to repair the data link in the record header. This allows an administrator to know with greater certainty that serious damage has occurred and they may prefer to recover the file from a backup.

If an administrator prefers allowing c-treeACE to attempt to recover a file, as done prior to V11.5, the following options are available to revert to prior behavior:

- *-repairdata* command-line switch - Use this switch with the **ctcmpcif** and **ctcmpcif** compact utilities, e.g., **ctcmpcif** *-rdata*
- *-repairCorruptIFILoption* option - The *repairCorruptIFILoption* option should be OR-ed into the tfilno of the rebuild (**RebuildIFIL()**) and compact (**CompactIFile()**) functions using the *setIFILoptions()* macro.

  For example:

```
myIFIL.tfilno = setIFILoptions(repairCorruptIFILoption |
updateIFILoption);
```

Best Practices:
1.) It is always important to back up concerned data files before you attempt to compact or rebuild them.
2.) Ensure sufficient disk space is available before starting a compact or rebuild. You will need up to at least 3X the size of the data file plus index files (original files + new files + copy) for a successful compact or rebuild.
3.) Consider keeping a copy of the data file for sufficient time and ensure all expected data is available before removing the backup.

13.2 Failed ctdbCreateTable could leave files on disk

In certain circumstances, a **ctdbCreateTable()** that failed left the table and the index files on the disk. This occurred only in situations in which the failure occurred after the table was created (e.g., during the follow-up operations such as PUDODA and PUHDR). The logic has been altered to identify failures in this stage and remove the newly created files.

Note: This is a **Behavior Change** in the **ctdbCreateTable()** function.

13.3 Save space in variable-length files created by FairCom DB API/SQL

The logic has been improved to save 5 bytes (or 9 bytes on huge files) per record on tables with variable-length records by creating them without the hidden $DELFLD$ field that is useful only in fixed-length records. This is a behavior change in the record structure for files created through FairCom DB API and SQL.

13.4 ctdbCreateMRTTable behavior change

To avoid possible conflicts between an MRT Table (Multi-Schema Tables) definition and its host table, **ctdbCreateMRTTable** was modified in V11.5 to ignore the following creation flags and use the same configuration used on the host table:

- CTCREATE_NODELFLD
- CTCREATE_NOROWID
- CTCREATE_NONULFLD

Furthermore, the optimization to automatically avoid the use of DELFLD on variable-length records has been disabled on MRT Tables.

Note: This constitutes a Compatibility Change.

13.5 FPUTFGET now treats read lock requests as write lock requests

Because FPUTFGET does not support read locks, it now promotes read lock requests to write lock. Before this upgrade, the function that acquired a read lock, did nothing but return success. This could cause problems for an application that used **LKISAM()** with the *ctREADREC* or *ctREADREC_BLK* mode in the FPUTFGET model because no locks were actually acquired.

In FPUTFGET mode, **LKISAM()** now converts read lock requests (*ctREADREC* and *ctREADREC_BLK* lock modes) to write lock requests (*ctENABLE* and *ctENABLE_BLK* lock modes).

13.6 Transaction commit improvements

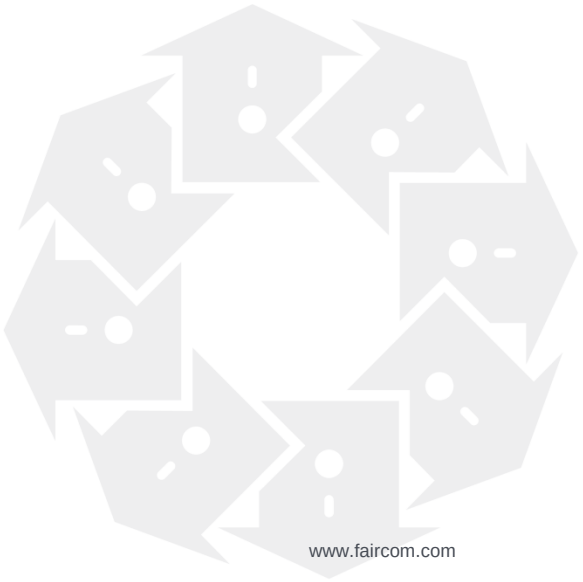
When a record update callback was configured to run at transaction commit time and the record update callback function attempted to perform a transaction-controlled update operation, the commit could fail with error **99**. The logic has been improved to eliminate this error. This also has the advantage that any changes to transaction-controlled files made by the record update callbacks will be part of the committing transaction.

Note: This is a **Behavior Change** because changes made by record update callbacks were not part of the commit before this revision.

13.7 ctdbDeleteTable on missing table failed with INOT_ERR instead of CTDBRET_NOSUCHTABLE

ctdbDeleteTable() called on a table that did not exist in the dictionary failed with error **INOT_ERR** instead of **CTDBRET_NOSUCHTABLE**. The logic has been modified to return the proper error, **CTDBRET_NOSUCHTABLE** (4023).

Note: This is a **Behavior Change**.



Copyright Notice

Trademarks

FairCom DB, FairCom EDGE, c-treeRTG, c-treeACE, c-treeAMS, c-treeEDGE, c-tree Plus, c-tree, r-tree, FairCom, and FairCom's circular disc logo are trademarks of FairCom USA, registered in the United States and other countries.

The following are third-party trademarks: Btrieve is a registered trademark of Actian Corporation. Amazon Web Services, the "Powered by AWS" logo, and AWS are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries. AMD and AMD Opteron are trademarks of Advanced Micro Devices, Inc. Macintosh, Mac, Mac OS, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries. Embarcadero, the Embarcadero Technologies logos and all other Embarcadero Technologies product or service names are trademarks, service marks, and/or registered trademarks of Embarcadero Technologies, Inc. and are protected by the laws of the United States and other countries. HP and HP-UX are registered trademarks of the Hewlett-Packard Company. AIX, IBM, POWER6, POWER7, POWER8, POWER9, POWER10 and pSeries are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. Intel, Intel Core, Itanium, Pentium and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. ACUCOBOL-GT, Micro Focus, RM/COBOL, and Visual COBOL are trademarks or registered trademarks of Micro Focus (IP) Limited or its subsidiaries in the United Kingdom, United States and other countries. Microsoft, the .NET logo, the Windows logo, Access, Excel, SQL Server, Visual Basic, Visual C++, Visual C#, Visual Studio, Windows, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Oracle and Java are registered trademarks of Oracle and/or its affiliates. QNX and Neutrino are registered trademarks of QNX Software Systems Ltd. in certain jurisdictions. CentOS, Red Hat, and the Shadow Man logo are registered trademarks of Red Hat, Inc. in the United States and other countries, used with permission. SAP® Business Objects, SAP® Crystal Reports and SAP® BusinessObjects™ Web Intelligence® as well as their respective logos are trademarks or registered trademarks of SAP. SUSE" and the SUSE logo are trademarks of SUSE LLC or its subsidiaries or affiliates. UNIX and UNIXWARE are registered trademarks of The Open Group in the United States and other countries. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. Python and PyCon are trademarks or registered trademarks of the Python Software Foundation. isCOBOL and Veryant are trademarks or registered trademarks of Veryant in the United States and other countries. OpenServer is a trademark or registered trademark of Xinuos, Inc. in the U.S.A. and other countries. Unicode and the Unicode Logo are registered trademarks of Unicode, Inc. in the United States and other countries.

All other trademarks, trade names, company names, product names, and registered trademarks are the property of their respective holders.

Open Source Components

Like most software development companies, FairCom uses third-party components to provide some functionality within our technology. Often those third-party components are selected because they are a standard in the industry, they offer specific functionality that is easier to license than to develop and maintain in the long run, or they provide a proven and inexpensive solution to a particular business need. Examples of third-party software FairCom uses are the OpenSSL toolkit that provides Transport Layer Security (TLS) for secure communications and the ICU Unicode libraries to provide wide character support (think international characters and emojis).

Some of these third-party components are the subject to commercial licenses and others are subject to open source licenses. For open source solutions that we incorporate into our technology, we include the package name and associated license in a notice.txt file found in the same directory as the server.

The notice.txt file should always stay in the same directory as the server. This is particularly important in instances where your company has redistribution rights, such as an ISV who duplicates server binaries and (re)distributes those to an eventual end-user at a third-party company. Ensuring that the notice.txt file "travels with" the server binary is important to maintain third-party and FairCom license compliance.

1/30/2025

14. Index

T

U

V