test link

Release Notes

c-treeACE V11.0 Release Notes

Audience

Developers

Subject

c-treeACE V11.0 Release Notes

FairCom®

Contents

1. Introduction

We are pleased to deliver another c-treeACE release to our valued maintenance customers. Many improvements and corrections have been implemented to make our latest release the best ever.

This document lists the corrections we have implemented in this release. These changes are grouped by product area.

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 is packed with new features—from simple enhancements to major breakthroughs. The new and updated features are listed in the V11 Update Guide (https://docs.faircom.com/doc/v11ace/).

2. Notable Compatibility Changes

This section lists important known compatibility changes from prior releases.

It is important to review these issues and ensure that your application continues to correctly function after upgrading to this latest release, V2.

2.1 FairCom Server - Change defaults for V11 release

FairCom Server now uses the following default values for these configuration options when they are not specified in *ctsrvr.cfg*:

```
CHECKPOINT_FLUSH        17
LOG_SPACE               120 MB
CHECKPOINT_INTERVAL     10 MB
LOG_TEMPLATE            2
COMMIT_DELAY  2 (1 on Linux)

; Data and index cache size
DAT_MEMORY              100 MB
IDX_MEMORY              100 MB

; Sort memory for index rebuild
SORT_MEMORY             100 MB

; Maximum status log size of 32 MB, keeping one prior copy
CTSTATUS_SIZE           -32000000
```

These options are new to V11, and these are their defaults:

```
; Flush updates for transaction files to file system as soon as possible in the background
TRAN_DATA_FLUSH_SEC     60
TRAN_INDEX_FLUSH_SEC    60

; Flush updates for non-tran files to file system as soon as possible in the background
NONTRAN_DATA_FLUSH_SEC  IMMEDIATE
NONTRAN_INDEX_FLUSH_SEC IMMEDIATE
```

The following options have been added as convenience options. However, you should review them as they may change behavior in your particular installations in subtle ways.

```
; Limit JVM memory
SETENV                  DH_JVM_OPTION_STRINGS=-Xms100m -Xmx300m

; Suppress dynamic dump logging of backed up file names to CTSTATUS.FCS
CTSTATUS_MASK           DYNAMIC_DUMP_FILES

; Log final SNAPSHOT stats to SNAPSHOT.FCS on shutdown for baseline metrics
DIAGNOSTICS             SNAPSHOT_SHUTDOWN
```

2.2 Correct Error Messages Now Returned by fc_create_user Procedure

Calling **fc_create_user** could have resulted in several unusual error messages, due to an overlap of the SA_ADMIN API and c-treeACE SQL error codes. For example, calling this procedure with an empty user name previously returned error code **-18008**, which was reported as message, "CT - A partitioned file can only have one open instance at a time per connection." The SA_ADMIN return codes used by this function have been replaced with c-tree error codes. In the example case, the error code is now correctly returned as **-17450** with message "CT - Invalid user id."

2.3 SQL - Changed error message for error -20139

The error message for error **-20139** has been updated to be more meaningful. The message now states:

```
error(-20139): Index on long fields not supported
```

2.4 SQL - BINARY fields not padded with 0x00

This is a behavior change that affects how the values of BINARY fields are stored in V11 and later.

Prior to V11, the server was not padding BINARY fields with 0x00 when the values were less than the defined length. Because of this, both the values stored on disk and the values returned may have been shorter than the defined length (basically BINARY and VARBINARY behaved in the same manner). This behavior has been corrected in V11 and later so that BINARY fields are properly padded with 0x00 both on disk and when retrieved. Existing data is not touched on disk, however the values returned in SQL are now padded.

The previous behavior can be restored by using the keyword `SQL_OPTION NO_BINARY_PAD` in *ctsrvr.cfg.*

This is only documented for a select few applications that may be impacted by the V11 change in behavior. Consult FairCom support should have you concerns about your current binary data prior to V11.

USE OF THIS OPTION WILL LIKELY RESULT IN UNEXPECTED BINARY FIELD BEHAVIOR.

2.5 Automatic FairCom DB API Batch Buffer Resize

The **ctdbSetBatch()** function takes a parameter *buffeLen* which is the size of the buffer used internally by FairCom DB API to handle batch operations.

```
CTDBRET ctdbDECL ctdbSetBatch(CTHANDLE Handle, CTBATCH_MODE mode, VRLEN targetLen, VRLEN bufferLen)
```

A value of 0 for this parameter was an indication that the default value size should be used. The default buffer size is calculated as the size of the fixed portion of the record multiplied by 128.

In this release and later, when *bufferLen* is set to 0, the default buffer size is calculated as described above, and logic is activated to perform automatic buffer resize if the buffer is not large enough to contain one record.

When *bufferLen* is not 0 and the buffer is not large enough to contain at least one record, the error **BTBZ_ERR** (429) is returned. In this case, it is possible to activate the logic to automatically resize the buffer by adding the new *CTBATCH_AUTORESIZE* FairCom DB API batch mode to the *mode* parameter.

2.6 Proper positioning of ctdbSeekRecord with active record sets

ctdbSeekRecord() logic was modified such that it positions into the set if a record handle has an active criteria set. Prior to this change, if a record handle had an active criteria set, **ctdbSeekRecord()** was not positioning into the set.

2.7 Server process exit code more informative

The process return code was reviewed and made to be more informative. Now, an external server stop request should return **0** on successful shutdown. A shutdown not triggered by external request will return a non-zero value.

2.8 Physical read of variable-length transaction controlled file skips records added by a third-party transaction not yet committed

A feature introduced in V8.14 affected the behavior of **NXTVREC** in physical order when it encountered the "record space" for a new record that had not been committed and was written in a space that was not being reused. Instead of reporting a **VFLG_ERR** (error 158), that new feature skips the uncommitted "record space" (unless the reading is by the transactor, which would see the uncommitted record).

Details

The change affects how variable-length records are internally marked during transaction processing of pre-image space. The behavior prior to V8.14 marked the record header in a way that was considered invalid, causing a **VFLG_ERR** (158) error. The newer behavior sees the record as a deleted record (actually a pending insert) and skips to the next record as in a "read committed" transaction isolation.

This change is limited to the internal handling of header marks for newly added variable-length records.

Indexed files are not affected by these changes because pending key inserts are handled differently under transaction control.

These changes do not include changes to the physical files, record structures on disk, or other transaction control.

Reverting Back to the Old Behavior

These changes can be reverted back to the original (prior to V8.14) behavior using the keyword:

`COMPATIBILITY NO_INIT_VSPACE` (https://docs.faircom.com/doc/`ctserver/57570.htm`)

Changes in the Latest Revision

The changes introduced in the V11 release address the issues with the earlier change as follows:

If the reader has requested acquiring locks on the records that it reads, the physical read acquires a lock on that record and respects the lock before proceeding (earlier the lock was not respected) producing one of the following outcomes:

If...	Then...
Locking the record fails with error 42 (**DLOK_ERR**) because another connection has the record locked and the reader requested non-blocking locking.	The physical record read function returns error 42.
The record is committed (already, or when the lock is released).	The record read proceeds as usual, and the record that would have been skipped is returned to the caller.
The record is deleted or is a “resource.”	The physical record read function continues scanning the data file, reading the next record.
The record header contains an invalid record mark.	The physical record read function returns error 158 (**VFLG_ERR**).

Further changes have been introduced to reduce the occurrence of the 158 (**VFLG_ERR**) without skipping any record by changing the record header marker management during record addition.

`COMPATIBILITY NO_VFLG_ERR` can be used to disable this new handling and restore the earlier (post V8.14) behavior for the rare circumstance in which the old behavior is desired.

2.9 Linux File System Performance and Safety

Review the FairCom website for important information about c-treeACE performance and integrity on Linux platforms: **Linux Caching Considerations** (https://docs.faircom.com/doc**/ctserver/linux-caching-considerations.htm**).

2.10 COMMIT_DELAY Configuration Now Defaults to 1 ms on Linux Systems

The default FairCom Server configuration file, ***ctsrvr.cfg***, included in c-treeACE Linux packages has been modified to improve performance. The `COMMIT_DELAY` setting now defaults to a value of 1 (ms). (*Previously, COMMIT_DELAY was disabled for Linux platforms, see note below.*) One ms has been shown to produce much better performance with *TRNLOG* files on Linux systems.

Note: `COMMIT_DELAY` had been disabled in an earlier correction for a Linux bug, which was preventing synchronous writes from being flushed to disk. That setting degraded *TRNLOG* performance under those circumstances and a default setting of 1 ms is again recommended with c-treeACE V10.3.1.21834 (Build 140307) or later.

2.11 Relaxed COMPATIBILITY FORCE_WRITETHRU Defaults

`COMPATIBILITY FORCE_WRITETHRU` enables the *WRITETHRU* filemode for all non-transaction files (which includes *PREIMG* files). The performance impact of *WRITETHRU* depends on whether `COMPATIBILITY PREV610A_FLUSH` is also enabled. See the descriptions in `COMPATIBILITY FORCE_WRITETHRU` and `COMPATIBILITY PREV610A_FLUSH`.

These options provide a good balance between update performance and recoverability of data in the event of an abnormal FairCom Server termination. They affect non-transaction files as follows:

- Non-transaction files that are neither created nor opened with the *WRITETHRU* filemode have their updates written to the FairCom Server cache and eventually written to the file system cache and to disk.
- Non-transaction files that are created or opened with the *WRITETHRU* filemode have their updates flushed to the file system cache.

Be sure to understand the impact of these file modes with respect to your file transaction mode in use, and type and vulnerability of your application data.

2.12 New Extended Data Types Support

The range of values available for new c-treeACE data types has been extended.

This support required changing the format in which the record schema (*DODA*) is stored on disk and in memory. It affects server, client, and standalone code.

Backward compatibility has been preserved as much as possible:

- The record schema is stored in the new format only if the record schema uses data types in the new range.
- A FairCom Server that supports the new range of data types sends the appropriate record schema format to the client based on whether the client supports the new range of data types. If the record schema uses a data type in the new range and the client does not support the new record schema format, the server fails the operation with error **DTPC_ERR** (1010, "This file uses data types that your client library does not support. Update your client library").
- A client that supports the new range of data types sends the appropriate record schema format to the server based on whether the server supports the new range of data types. If the record schema uses a data type in the new range and the server does not support the new record schema format, the client fails the operation with error **DTPS_ERR** (1011, "This client library uses data type support that your server does not support. Update your c-tree Server").
- A file whose record schema is stored in the new format has a bit set in the file's header to indicate that it uses the new format. A FairCom Server or standalone library that does not support extended data types will fail to open a file whose record schema is stored in the new format with error **FREL_ERR** (744, "file requires unavailable feature"), and a message is logged to *CTSTATUS.FCS* to indicate which DEF_MASK bit triggered the error. The bit that indicates support for the new c-treeACE data type range is 0x40000, so the error message

will look similar to the following (the message will include the 0x40000 bit in addition to any other feature bits that the server does not support):

```
FREL_ERR(744) unsupported DEF_MASK bits: 00040000x for file...
```

2.13 Dynamic Dump Stream Files No Longer Segment by Default

Nearly all of today's modern filesystems support file sizes larger than 2GB. Dump streams from the c-treeACE Dynamic Dump backup are frequently much larger than 2GB and, by default, are written into 2GB segments. This is proving to make management of the backup process more cumbersome. As a result, the Dynamic Dump backup option `EXT_SIZE` script option now defaults to `NO`, such that the dump backup is written to a single file. Simply specify an extent size in your dump script files if you wish to again segment your files.

2.14 Auto-Numbering Replication Defaults Changed

In this release, changes were made to the FairCom Server defaults for replication behavior on *SRLSEG* and identity fields.

Compatibility Note: This is a change in behavior, but many of the FairCom Server configuration files included in FairCom products already contain this option so the behavior is already in effect in those cases. However, the FairCom RTG package did not include these options, so this will change the default behavior for FairCom RTG COBOL and FairCom RTG BTRV.

The default values have been changed for the following options:

REPL_SRLSEG_ALLOW_UNQKEY now defaults to YES

This option allows a *SRLSEG* index to be used as a replication unique index. Without this option, a data file whose only unique index is a *SRLSEG* index would not qualify for replication. By changing this default to YES, the *SRLSEG* index can be used as a replication unique index, which is the commonly-expected behavior.

REPL_SRLSEG_USE_SOURCE now defaults to YES

This option uses the serial number value from the source FairCom Server when adding a record to a replicated file. This option applies to c-tree's asynchronous (source/target) replication model, in which a Replication Agent replicates changes from a source FairCom Server to a target FairCom Server. By changing this default to YES, records added to the target server will contain the *SRLSEG* value from the record on the source server, rather than the target server generating its own *SRLSEG* value for the new record. This default option corresponds to the behavior that is most likely to be expected.

REPL_SRLSEG_USE_MASTER now defaults to YES

This option uses the serial number value from the master FairCom Server when adding a record to a replicated file. This option applies to c-tree's synchronous (local/master) replication model, in which an update to a record on a local FairCom Server triggers an update to the record in the associated table on the master FairCom Server. By changing this default to YES, records added to the local server will contain the *SRLSEG* value from the record on the master server, rather than the local server generating its own SRLSEG value for the new record. This default option corresponds to the behavior that is most likely to be expected.

REPL_IDENTITY_USE_SOURCE now defaults to YES

This option uses the identity field value from the source FairCom Server when adding a record to a replicated file. This option applies to c-tree's asynchronous (source/target) replication model, in which a Replication Agent replicates changes from a source FairCom Server to a target FairCom Server. By changing this default to YES, records added to the target server will contain the identity field value from the record on the source server, rather than the target server generating its own identity field value for the new record. This default option corresponds to the behavior that is most likely to be expected.

REPL_IDENTITY_USE_MASTER now defaults to YES

This option uses the identity field value from the master FairCom Server when adding a record to a replicated file. This option applies to c-tree's synchronous (local/master) replication model, in which an update to a record on a local FairCom Server triggers an update to the record in the associated table on the master FairCom Server. By changing this default to YES, records added to the local server will contain the identity field value from the record on the master server, rather than the local server generating its own identity field value for the new record. This default option corresponds to the behavior that is most likely to be expected.

2.15 “Add Unique Keys First” Feature Applied to ctADD2END Files

Variable-length data files using our *ctADD2END* feature now automatically enable a "add unique keys first" optimization. Enabling this optimization prevents space from being allocated in the data file that is not reused if a record ADD operation fails with a duplicate key error.

2.16 Maximum LIST_MEMORY Setting Increased to 10 MB

The maximum value for the `LIST_MEMORY` server administrator keyword has been changed from 1 MB to 10 MB.

2.17 Maximum Index Members per File (MAXMEMB)

The default for the `MAXMEMB` (the number of index members per single index file) has been updated to 127 allowing a larger number of segments per index.

We now define ctMAXMEMB in *ctopt1.h* instead of in *ctopt2.h*, because *ctport.h* references ctMAXMEMB and it is included before *ctopt2.h*.

We also define MAXMEMB to ctMAXMEMB in *ctopt1.h*.

2.18 Maximum Number of Indexes per Data File (MAX_DAT_KEY) Default Increased to 64

Occasionally data files require a large number of indexes. Commencing with c-treeACE V10.3, the number of indexes default limit was increased from 32 to 64. Customers using the FairCom Server can use the `MAX_DAT_KEY` keyword to change the limit on the number of indexes per data file.

If this limit is too low, the typical error code that would be seen is error 107, **IDRK_ERR** "too many keys for ISAM data file."

This value affects the amount of memory that is allocated to store ISAM index state information. It can be increased without a major impact on performance unless FairCom Server is being run in an environment with very little memory (e.g., certain embedded applications).

Note: In the *standalone* model, `MAX_DAT_KEY` is a *compile-time* setting. If this value is changed, the code must be recompiled with the new setting.

2.19 Maximum Number of Open Files per User (MAX_FILES_PER_USER) Default Increased to 32767

The default FairCom Server `MAX_FILES_PER_USER` value has been increased from 2048 to 32767. We increased the default value to avoid file open operations failing with error **46** due to the lower limit. Generally, developers expect a single connection can open as many files as FairCom Server's `FILES` limit, but this was not the case due to the previous per user default limit.

2.20 Allow a Single Byte or SByte to be passed as a BINARY Value in ADO.NET

The logic has been updated to allow a single Byte or SByte to be passed as a BINARY value in addition to Byte[]. Other types now throw an exception.

2.21 c-treeACE Memory Allocation Limit Disabled

Prior to this modification, FairCom Server's `LMT_MEMORY` configuration option defaulted to 128MB, meaning that a single memory allocation could not exceed 128MB. However, this limit restricted the size of variable-length records from FairCom Server and limited the amount of memory that could be allocated with the `RECOVER_MEMLOG` recovery option.

This limit has been *disabled* by default. If `LMT_MEMORY` is not specified, FairCom Server does not place a limit on the size of a single memory allocation. If desired, `LMT_MEMORY` can be specified in *ctsrvr.cfg* setting a desired maximum allocation size.

Note: This modification results in a behavior change.

This change also applies to standalone mode: `#define ctMEMLMT` is set to zero by default, meaning no memory allocation size limit. If desired, c-treeACE can be compiled with *ctMEMLMT* defined to a non-zero value.

2.22 c-treeACE SQL SETENV limit raised to 8192

c-treeACE SQL server did not start when *ctsrvr.cfg* contained a `SETENV` configuration referencing an environment variable evaluating to more than 1024 bytes.

Previously two limits were enforced in setting environment variables in *ctsrvr.cfg*:

1. While parsing *ctsrvr.cfg* - Limit of 1024 bytes to hold the content of *ctsrvr.cfg* and expand the environment variables referenced (an error was returned if the expanded value did not fit)
2. When storing the variable name and content for SQL usage - Limit of 50 bytes for the environment variable name and 255 bytes for the content (name/value truncated if it did not fit)

Limit #1 has now been increased to 8192

Limit #2 has been removed with dynamic memory allocation.

The net effect is a practical limit is 8192 bytes.

2.23 c-treeACE SQL Stored Procedure Server-side Debugging Options

In V11 and later, server-side stored procedure execution logging can be turned on and off using the TPESQLDBG environment variable. TPESQLDBG is an array of 'Y'/'N' characters that determine which debug options are enabled.

Prior to this revision, this was achieved by setting the 11th element (offset 10, `java_debug`) in the TPESQLDBG environment variable. The logging produced in this way is quite verbose and included information that was useful only to FairCom Technicians.

The interpretation of TPESQLDBG (and other methods setting the same debug features) has been modified as follows:

- TPESQLDBG (which was an array of 12 'Y'/'N' characters) has been expanded to 13 elements.
- The element at offset 12 is a new element named `stp_logging` that controls the **log()** method used in stored procedures to generate log messages.

Note that activating element #11 (`java_debug`) activates internal logging and **log()** method for ANY stored procedure language, not just Java.

2.24 Java Stored Procedure Runtime Classes no Longer Require ctreedbs in Path

Java stored procedure runtime support on the server has been enhanced such that, for ease of compilation and execution, it is no longer necessary to have the *ctreedbs* shared library/DLL in the path for libraries.

2.25 PHP - Components Now Match non-Thread-Safe Defaults for Windows IIS PHP Installations

The default PHP installation for Microsoft Windows IIS is not thread-safe, which implies that any PHP extensions should also be compiled as "not thread-safe." Previously, c-treeACE PHP components were compiled and distributed as thread-safe, which were incompatible with default PHP IIS installations. For better default compatibility, c-treeACE SQL PHP components are now compiled and distributed as non-thread-safe.

2.26 c-treeACE SQL JDBC Socket Timeout Defaults to 0

The default c-treeACE SQL JDBC socket timeout has been changed from 30 minutes to 0 (no timeout). This timeout can be modified by setting the `sock_timeout` property explicitly at connection time:

```
Properties info = new Properties();
info.setProperty("user","ADMIN");
info.setProperty("password","ADMIN");
info.setProperty("sock_timeout","30000");
conn = DriverManager.getConnection ("jdbc:ctree:6597@localhost:ctreeSQL", info);
```

2.27 c-treeACE JDBC Java 1.5 Compatible Driver Availability

c-treeACE JDBC requires JRE 1.6 or later. A Java 1.5 compatible version of the JDBC driver is available upon request for applications still tied to this legacy Java framework.

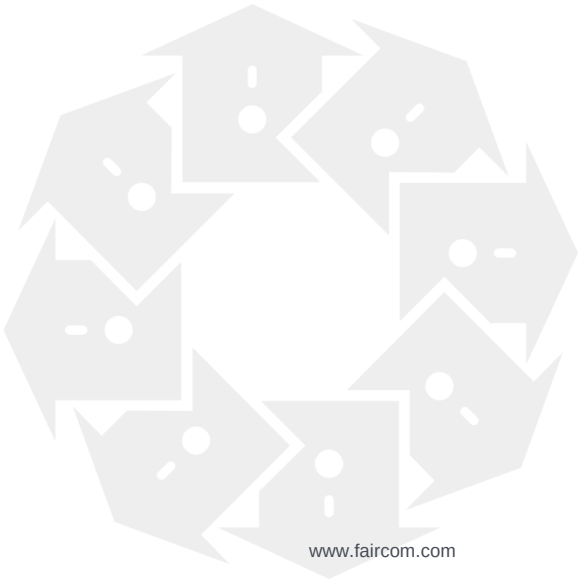
2.28 Windows Servers Now Statically Linked with ZLIB Compression Libraries

c-treeACE uses the open source ZLIB library as a data compression option. These were previously picked up as a shared library. ZLIB libraries are now statically linked into Windows build of c-treeACE servers intended for general distribution.

Note: Some customer OEM agreements prevent us from distributing open source code. In these cases provisions are made to not statically link with ZLIB and a compatible ZLIB .DLL will need to be provided on the environment where compression is desired.

3. Core Engine Updates and Corrections

Improvements have been made throughout the c-treeACE product for this release. This section lists many improvements that address areas in the core engine.



3.1 Critical Production Updates

Corrected Unhandled Exception When File Password Included in Open File Call

When using the c-treeACE Administration utility, **ctadmn**, to connect to FairCom Server, if a file password was specified for *FAIRCOM.FCS*, FairCom Server terminated with an unhandled exception. This could also happen on any call to open a file that specified a non-empty file password on a system that requires properly-aligned memory accesses. The logic has been modified to correct this problem.

Corrected Read and Write Errors after Connection Termination

Using **ctadmn** or c-treeACE Monitor to terminate a SQL or ISAM connection to c-tree Server that is using the TCP/IP communication protocol can cause errors reading and writing files or an unhandled exception when reading data from the socket. *CTSTATUS.FCS* may show errors such as the following:

```
 - User# 00285 External kill request posted against user #89
 |ODBCUSER|SQL:LIVE|
 - User# 00178 ctsave failed: system code = 9  lc = 38  fd = 946
 - User# 00178 ./live.dbs/appkcomp.dat
 - User# 00178 Error on close file. locale:20  sysiocod:9  uerr_cod: 24
 - User# 00178 ./live.dbs/appkcomp.dat
 - User# 00047 WRITE_ERR: ./live.dbs/rlavllog.dat at 0:0x  sysiocod=9 bufsiz=8192 bytes written=0[0]
ioLoc=0: 37
 - User# 00047 FAILED TRAN IO: ctwrtbuf...: 37
 - User# 00047 ./live.dbs/rlavllog.dat: 37
 - User# 00047 Dumped stack for server process 29098172, log=1, loc=60, rc=0
 - User# 00047 O47 M1 L60 F2720 P0x (recur #1) (uerr_cod=37)
```

The logic has been modified to correct these errors.

Prevent FairCom Server WRITE_ERR Termination with Open Transactions Aborted by Quiesce

The FairCom Server was found to shut down due to a write error on a transaction-controlled file in the following precise set of circumstances:

1. A client begins a transaction and updates a transaction-controlled file.
2. While the transaction is still active, the server is quiesced using the **ctquiet** utility's "full consistency" option.
3. The quiesce aborts the client's transaction.
4. If the client then closes its connection while the quiesce is still active, the server shuts down due to a write error on the client's file.

FairCom Server logged the following error messages to *CTSTATUS.FCS* and shut down:

```
Thu Feb 13 09:34:05 2014
 - User# 00014 WRITE_ERR: vcusti at 0:4000x  sysiocod=6 bufsiz=8192 bytes written=0[0]  ioLoc=0: 37
Thu Feb 13 09:34:09 2014
 - User# 00014 FAILED TRAN IO: ctwrtbuf...: 37
Thu Feb 13 09:34:09 2014
 - User# 00014 vcusti: 37
Thu Feb 13 09:34:11 2014
 - User# 00014 Dumped stack for server process 67504, log=1, loc=60, rc=0
Thu Feb 13 09:34:11 2014
 - User# 00014 ctQUIET: unblocking call with action: 9e00x
Thu Feb 13 09:34:11 2014
 - User# 00014 ctQUIET: end of unblocking call
Thu Feb 13 09:34:13 2014
 - User# 00014 O14 M1 L60 F23 P4000x (recur #1) (uerr_cod=37)
```

The logic has been modified to correct this situation.

Corrected Unexpected FairCom Server Internal Error 7495 Crash

In an unusual situation, FairCom Server could crash during commit or abort when a memory allocation occurred. There were two possible symptoms depending on the platform:

1. On a high/low system: FairCom Server terminated with an unhandled exception.
2. On a low/high system: FairCom Server terminated with internal error **7495**.

The memory allocation failure was already detected and handled, however, its handling caused an unexpected internal state in the server, which later caused the observed symptom.

The logic has been modified to correct this situation.

Prevent c-tree Server Unhandled Exception During Update of Compressed Record

c-tree Server terminated with an unhandled exception when updating a compressed record in a *HUGE* transaction-controlled file (*ctTRNLOG* or *ctPREIMG*) at the logical end of file after turning off ISAM key buffers if the transaction had a previous update to the record followed by a savepoint.

The significant factors were:

- the record was compressed
- prior to the update where the crash occurred, the same transaction updated the record and then established a savepoint

Compressed records no longer cause this exception.

Inconsistent FPUTFGET Header Locking for Non-HUGE Index Files and Variable-Length Data Files

A variety of errors were encountered in *FPUTFGET* mode when data was updated. The errors included **69**, **527**, **42**, **160**, **30**, **519**, and internal error **233**. For non-huge index files and

variable-length data files, two connections could be updating a header at the same time, causing one's updates to be lost. This could cause the logical end of file value to be smaller than expected, causing space to be reused for index nodes or variable-length data records. The logic has been modified to correct these problems.

Avoid FairCom Server Termination with Internal Error 8987 When Using UNBUFFERED_IO Configuration Option

After enabling the UNBUFFERED_IO configuration option for a data file, FairCom Server terminated with internal error **8987**. The logic has been corrected to eliminate this occurrence.

Prevent Unhandled Exception When a Single Connection Opens a File More than 1024 Times

FairCom Server terminated with an unhandled exception after a single connection opened a file more than 1024 times. The logic has been modified so that attempting to open a file more than 1024 times in a single connection will now fail with error code **COFM_ERR** (1103) and will preserve the proper co-file state. If a `-1` co-file link is found when closing the file (which is now unexpected), the code avoids using the invalid co-file link.

Corrected Prime Cache Thread Unhandled Exception When Opening File Pending Delete

An unhandled exception occurred when a prime cache thread opened a file that was pending delete. The logic has been modified so that the server returns an error in that situation.

Corrected Errors When Changing a Temporary Index Condition

If a condition was set on a temporary index by one client and then another client changed the condition, an unhandled exception, an internal error, or an **NKEY_ERR** may have occurred. An example of the internal error:

```
Fri Apr 11 16:08:15 2014
 - User# 00217 7491: Memory return out of bounds...
Fri Apr 11 16:08:15 2014
 - User# 00217 Limit:8000000x Return:899c5360x
Fri Apr 11 16:08:15 2014
 - User# 00217 Unexpected internal c-tree(R) error #7491 (uerr_cod=0)
```

The logic has been modified to correct this.

Deadlock Corrected in Data Cache Retrieval Function

A deadlock was found in a data cache retrieval function. This could cause c-treeACE Server to hang during normal operation on a call to request a file's mutex when retrieving data from the

data cache. (This condition only occurred in c-treeACE Server v10.3.0.) The logic has been corrected to eliminate this problem.

Unhandled Exception When Accessing Pruned Memory Index Node

FairCom Server caused an unhandled exception when accessing a memory index node that had been pruned from the tree. The logic has been corrected to eliminate this problem.

3.2 Useful Updates

License File Handling Improvements

V10 introduced a new licensing file for enabling connections, CPU usage and features. Several modifications have been made refining usage of this license object.

License file processing now case-insensitive on Windows

The license file extension (*.lic*) was found to be case-sensitive on Windows. For example, an upper-case file extension (*ctsrvr*.LIC*) was causing an error **960**, license file not found, when it should have caused no issue. The logic has been modified to correct this.

Unexpected missing license file message

A situation was found in which the server would not start due to missing license, although the license was present in server directory. A message was written to *CTSTATUS.FCS* similar to the following:

```
Wed Jan 14 02:00:46 1970
 - User# 14424840       CTDLL_LOAD: Failed to load module libssl.so: libssl.so:
Wed Jan 14 02:00:46 1970
 - User# 00001  LICENSE ERROR: License initialization failed: Missing license file
Wed Jan 14 02:00:46 1970
 - User# 00001  Could not initialize server. Error: 960
Wed Jan 14 02:00:46 1970
```

License handling has been corrected to prevent this message.

License file handling memory-allocation fixed

Setting the FCSRVR_LIC environment variable to point to the directory where a license file was rather then pointing to the license file itself, c-treeACE on Linux failed to start with the following error:

```
LICENSE ERROR: License initialization failed: Memory allocation error (2)
```

This situation has been rectified and the error message now indicates more significant information:

```
License file size error (is FCSRVR_LIC set to a dir instead of the license file?)
```

SESSION_TIMEOUT Improvements

For 64-bit c-treeACE servers, a timeout allows each thread to detect and perform its own disconnection in case of a timeout.

- If `SESSION_TIMEOUT` is negative, it is ignored.
- If `SESSION_TIMEOUT` is less than 5, it is set to 5 so that the minimum SESSION_TIMEOUT value is 5 seconds.

Specify Shared Memory Keys on Unix

When more than one FairCom Server was run on a Unix system, the shared memory keys used by different servers could have the same value, which prevented connections to the servers. In addition, it was possible for unrelated applications to collide with default keys generated by c-treeACE servers.

To address this key collision, it is now possible for an administrator to specify specific shared memory keys for ISAM and SQL shared memory communication protocols ensuring the keys do not match existing keys already in use on the system.

New FairCom Server configuration options are available to directly specify a shared memory key. SQL and ISAM each require separate shared memory support

```
SHMEM_KEY_ISAM <isam_shared_memory_key>
SHMEM_KEY_SQL <sql_shared_memory_key>
```

Shared memory key values can be specified in either decimal or hexadecimal format. For example:

```
; Set shared memory key for ISAM connections to the specified decimal value:
SHMEM_KEY_ISAM 12345

; Set shared memory key for ISAM connections to the specified hexadecimal value:
SHMEM_KEY_ISAM 0xabcd
```

These server configuration options support specifying an environment variable, whose value is substituted for the configuration option value when the server starts up. For example, if the environment variable `MY_ISAM_KEY` is set to a numeric value such as 12345 or 0xabcdef before starting the server process, then the following option can be specified in the server configuration file to use this environment variable value for the `SHMEM_KEY_ISAM` configuration option value:

```
SHMEM_KEY_ISAM %MY_ISAM_KEY%
```

Compatibility Notes:

When these configuration options are *not* used, FairCom Server uses the old method of assigning shared memory keys so its shared memory communication protocol is compatible with old clients. FairCom Server now writes the shared memory key to the shared memory resource file and new clients know to read the shared memory key from this file. If a new client finds no shared memory key in the file, it uses the old method to assign a shared memory key so it is compatible with an old server.

The shared memory resource file is named */tmp/ctreedbs/<server_name>* for ISAM, and */tmp/ctreedbs/CTSQL_<sql_port>* for SQL, where */tmp/ctreedbs* is the default shared memory directory. It can be changed using the `SHMEM_DIRECTORY` configuration option.

An old client will not be able to connect to a new server using shared memory if the server uses the `SHMEM_KEY_ISAM` or `SHMEM_KEY_SQL` configuration option to specify a shared memory key that differs from the shared memory key that the old method would generate.

c-tree Server Name and Port Displayed in Windows System Tray Balloon

As an aid when developing with c-treeACE servers, when starting up on Windows using the tool-tray option, the c-tree Server name and port are now displayed in a system tray balloon for convenience. This feature requires c-tree Server to be run on Windows with the CONSOLE TOOL_TRAY option. It is not available when the Server is run as a Windows service.

Optional c-treeACE READ_ERR Diagnostic Logging

Rare, but unexpected **READ_ERR** (36) messages have been reported. When the operating system denies a read request from the file system, normally, a system error code is reported such as "permissions denied" or otherwise. However, rare and sporadic cases have been reported where this additional error return was zero (0). Good practice dictates we should determine why the OS file system is denying a read operation for any reason. A diagnostic option is now available to generate additional context information, along with a server stack trace when this condition occurs.

```
DIAGNOSTICS READ_ERR
```

This option enables additional diagnostic logging of read errors. When this option is in effect and an unexpected read error occurs (for example, a **READ_ERR** with a 0 *sysiocod* value), FairCom Server logs the following message to *CTSTATUS.FCS* and creates a process stack dump of the FairCom Server process (on systems that support that ability):

```
Mon Sep 14 12:23:19 2015
 - User# 00020  READ_ERR: loc 5 file <FILENAME> offset <OFFSET> iosize <READ_SIZE_IN_BYTES> syserr
<SYSTEM_ERROR_CODE> [physical file size <FILE_SIZE_ON_DISK>]
Mon Sep 14 12:23:21 2015
 - User# 00020  Dumped stack for server process 20788, log=1, loc=0, rc=0
```

`DIAGNOSTICS READ_ERR` can be enabled and disabled at runtime by calling **ctSETCFG()** or using **ctadmn** (menu option **10, Change Server Settings**, then option **9, Change a DIAGNOSTICS option**).

Allow Connection Termination Requests to Interrupt Batch Operations

If a request was made to terminate a connection that was performing a batch insert or a batch update operation, the operation continued to run to completion. Batch update operations are now interrupted and stopped.

Delete Node Queue Messages Now Suppressed by Default

The delete node thread logs the following message to *CTSTATUS.FCS* when it finds a discrepancy between a delete node queue entry and the current index file on disk having the same filename:

```
"ctdnode: could not process empty node"
```

For example, if an index file is deleted and recreated, a delete node queue entry for the original index file will not match the file ID values in the new index file.

As another example, if an index file is blocked using the file block feature, the delete node thread will not be able to access the file. Because this is a normal message, we now suppress it by default.

If desired, the message can be re-enabled by adding the option `DIAGNOSTICS NODEQ_MESSAGE` to *ctsrvr.cfg*.

Suppress SSL Library Loading Error Messages on Server Start Up

The following message is no longer logged to *CTSTATUS.FCS* until a feature that requires SSL support is called, and the library is not available:

```
- User# 00001  CTDLL_LOAD: Failed to load module libeay32.dll: %1 is not a valid Win32 application.
```

Currently, the only feature that requires SSL support is LDAP authentication. If LDAP authentication options are used in *ctsrvr.cfg* and the SSL support library (*libeay32.dll* or *libssl.so* on Unix systems) is not available, then FairCom Server logs this message to *CTSTATUS.FCS*.

Suppress Logging "file is opened without mirror" When MIRRORS NO is Specified

Logging of the message `Following file is opened without mirror...` to *CTSTATUS.FCS* is suppressed when the configuration file specifies `MIRRORS NO`, which disables the use of mirrors.

Permit Failed FairCom Server ctThrdInit() Calls to Return to Caller Instead of Exiting Process

With the c-tree database engine DLL model, a failed **ctThrdInit()** call due to insufficient memory resulted in exiting the process in prior releases. In this release and later, it is possible for an application to take its own actions when **ctThrdInit()** fails. When the database engine shutdown function is called due to an error occurring during a **ctThrdInit()** call, the function will return instead of exiting the process. The error code is returned to the caller.

Path Separator Now Automatically Appended to TMPNAME_PATH Directory

A SQL update failed with error **-20014** (Missing input parameter). The logic was attempting to create a file in the `TMPNAME_PATH` directory specified in *ctsrvr.cfg.* The directory did not exist and the option value did not end with a path separator. (If the option value had ended with a path separator, FairCom Server would have detected that the directory did not exist when it started and it would have logged an error message to *CTSTATUS.FCS* and terminated.)

The logic has been modified to automatically append a path separator to the `TMPNAME_PATH` option if it is not an empty string and does not already end with a path separator. This allows FairCom Server to detect whether or not the specified directory exists.

c-treeACE Professional SDK Build Improvements

PATH_MAX undefined on Linux

The *limits.h* header is now included on Linux for the `PATH_MAX` definition.

Building Server DLL with UNIFRMAT support

Building c-treeACE Server DLL using the PRO package when UNIFRMAT support was enabled failed due to unresolved symbols. UNIFRMAT is now properly supported in all makes.

Compiling FPUTFGET Library with virtual file support off

Compiling an FPUTFGET library with virtual file support off (by adding `#define NO_ctFeatVIRTUAL_FILES` in *ctoptn.h*) failed due to unresolved symbol `ctswitch`. Current makefiles have been adjusted to prevent this.

Errors were seen when compiling a c-tree client after upgrading to c-treeACE V10.3.0. Many of these errors concerned a conflict with c-tree's *ctssl.h* and OpenSSL's *bio.h* & *dh.h*. A sample of the errors include:

```
/usr/include/openssl/bio.h:600: error: new declaration â€˜BIO* BIO_new_file(conschar*, const char*)â€™
/home/rgb/linux.v2.6.x86.32bit/include/ctssl.h:83: error: ambiguates old declaration â€˜void*
BIO_new_file(const char*, const char*)â€™
/usr/include/openssl/dh.h:182: error: new declaration â€˜DH* DH_new()â€™
/home/rgb/linux.v2.6.x86.32bit/include/ctssl.h:88: error: ambiguates old declaration â€˜CTDH* DH_new()â€™
/home/rgb/linux.v2.6.x86.32bit/include/ctssl.h:92: error: previous declaration of â€˜ULONG
ERR_get_error()â€™ with â€˜C++â€™ linkage
/usr/include/openssl/err.h:260: error: conflicts with new declaration with â€˜Câ
```

Errors compiling c-tree when OpenSSL headers are included

These errors have been eliminated by moving the function prototypes for OpenSSL functions to the section of the header file that is compiled only when statically linking with OpenSSL.

Delete old winsock header and library

When using the Borland compiler, `#include <winsock.h>` in *ctappx.c* is bringing in the header file *ctree\source\ntree\sockets\winsock.h*. This causes errors due to conflicting definitions with the compiler's *winsock.h*. This header file and the winsock library have been removed from that

directory because they are no longer used. The versions of these files provided by the compiler are now used.

Remove AG_LOCK macro definition from version compatibility headers

AG_LOCK is defined in a header file on AIX, which caused errors compiling c-tree because c-tree's v6 to v7 header defines AG_LOCK. To eliminate this conflict, the AG_LOCK and ctAG_LOCK definitions (which are not currently required) have been removed from these headers.

3.3 Dynamic Dump and Restore Updates

Non c-tree Files Now Back Up to Correct Directory

If the `!DUMP` keyword used a relative path and the server has a `LOCAL_DIRECTORY` specified then NONCTREEFILES could potentially back up to the wrong directory during the dynamic dump. The logic has been corrected to include any `LOCAL_DIRECTORY` in the path before copying.

Proper Dynamic Dump Subdirectory Creation on Unix

Missing destination directories for a Dynamic Dump were not being created for the subdirectories of non-c-tree files. This problem was limited to Unix systems; it works correctly on Linux and Windows. The logic has been changed to correct this behavior on Unix systems.

Dynamic Dump Error now Returned to ctdump with !BLOCK_RETURN Option

When **ctdump** was used to perform a dynamic dump backup using the `!BLOCK_RETURN` option, if the dynamic dump failed after starting to write to the dump output file, the **ctdump** utility might report success even though the dynamic dump failed. The logic has been corrected to eliminate this problem.

Flush ctdump Utility Filesystem Output Before Exiting

The **ctdump** utility did not flush its standard output to the file system before exiting, so redirecting the output of **ctdump** to a file produced an empty file. The logic has been modified to correct this.

!CLNIDXX Script Option no Longer Crashes ctrdmp Restore Utility on 64-bit Systems

Running **ctrdmp** with the *!CLNIDXX* option in a restore script causes **ctrdmp** to crash with a segmentation fault. The logic has been modified to correct this.

ctrdmp Stabilized During Recovery Phase

An isolated crash was experienced in **ctrdmp** during the recovery phase. The logic has been corrected.

ctfdmp, ctldmp, and ctrdmp Utilities Now Display Version Information on Startup

The **ctfdmp**, **ctldmp**, and **ctrdmp** utilities now display the c-treeACE version used to compile them when they are run. Examples:

```
# ctfdmp
c-treeACE(tm) Version 10.5.0.24817(Build-140501) Transaction Log Forward Roll Utility

# ctldmp
c-treeACE(tm) Version 10.5.0.24817(Build-140501) Transaction Log Dump Utility

# ctrdmp
c-treeACE(tm) Version 10.5.0.24817(Build-140501) Dynamic Dump Restore Utility
```

3.4 Transaction Control and Recovery Strengthened

Avoid Server Shutdown When Finding Outdated Transaction Log

When KEEP_LOGS was disabled (`KEEP_LOGS -1`), an unexpected .FCA log file caused the server to shut down when the corresponding active log became inactive. A similar issue was fixed in an earlier revision by renaming the problem file and retrying. The case involving `KEEP_LOGS -1` has now been fixed.

Index Member Key Counts Now Properly Updated During Automatic Recovery

After automatic recovery, the header for the number of key values in an index member could be significantly different than the actual number. This problem was the result of a change to our internal management of available file control blocks in V10.

During automatic recovery, the checkpoint entries with the index header count information could result in the member index information being skipped because the host index had not yet been opened. The logic managing the index member header information after the host was opened did not work correctly unless *both* of these conditions were true:

- `SKIP_MISSING_FILES` was *not* set to `YES`.
- `COMPATIBILITY NO_AUTO_SKIP` was in the configuration.

Logic has been modified to correct this problem.

Ensure Correct Log Update for Very Large Transactions

A single large transaction log write (for example, a checkpoint holding a large transaction spanning millions of record deletes; potentially a single very large data record image) caused the log to be extended. In certain cases where the write was very large, the transaction log extension was smaller than required, which resulted in the log write extending past the physical EOF. This caused later log extensions to partially overwrite this log region with 0xff, as all log extensions are written at the physical EOF. Error 75 may have been seen.

Note: Using the log template feature should prevent this bug from occurring. All existing logs must be removed when the `LOG_TEMPLATE` keyword is added or `LOG_SPACE` is changed, or potentially vulnerable log extensions could still occur. `LOG_TEMPLATE` is a default configuration option as of V9.

Checkpoint Inconsistency Error Prevented with Deferred OPNTRAN Feature with Superfiles

In very rare circumstances, during a SQL database copy operation, FairCom Server was found to log the following message to *CTSTATUS.FCS* and terminate or become non-responsive:

```
CHKPNT memory inconsistency AN: 751
```

This obscure issue was introduced in c-treeACE V10. Logic has been modified to prevent this situation.

Correct Data File Counts Now Maintained After Resource Update Rollback

A situation was found in which the internal logic did not agree with the actual number of active fixed-length data records.

If a resource update to a fixed-length data file (whether stand-alone or a member of a *SUPERFILE*) occurred after a save point **(TRANSAV()**) and was subsequently rolled back to the save-point (**TRANRST()**), a request for pre-image space caused the internal count to be decremented. Logic has been modified to correct this problem.

Incorrect IICT Behaviors Corrected

IICTs were sometimes causing incorrect behavior such as missing key values, incorrect index header key counts, and the inability to read a record updated in the outer transaction while the IICT was active. The logic has been revised to correct this behavior.

In certain situations, the use of IICTs could cause incorrect behavior such as missing key values, incorrect index header key counts, and allowing a file to be closed even though it has changes pending in an active transaction. To correct this, the changes that were made previously have been extended to apply to index members.

When FairCom Server was started after terminating with an unhandled exception, automatic recovery was found to occasionally fail with error 96 if an IICT was used with an outer transaction that used the defer begin (ctDEFERBEG) transaction mode.

Note: This error was much less likely to happen when using the file-specific IICT than the generic IICT, because the file-specific IICT is a very short-lived operation (only for the duration of a single ISAM record add, delete, or update operation).

The logic has been modified to correct this error.

When FairCom Server was started and automatic recovery needs to run, in certain unusual situations automatic recovery failed with error **96** if IICTs were used. The logic has been modified to correct this issue.

Prevent LEOF_ERR Errors After Forward Roll

Reading a record from a c-treeACE data file failed with error **LEOF_ERR** after performing a forward roll on the file. This occurred because the transaction file number (also known as a "log handle") had exceeded 2GB.

Note: c-treeACE allows log handles up to 4294963200. After that limit is reached, c-treeACE shuts down and the log handle value must be reset: After ensuring that c-treeACE was shut down cleanly so that the transaction log files are no longer needed, reset the log handle by deleting the transaction log files. Prior to reaching this limit, c-treeACE logs warning messages that the system is approaching the limit when the log handle exceeds 4227858432.

The logic has been modified to eliminate this issue.

3.5 Server Engine Updates

Memory Index Node Resources Now Properly Freed

When a memory file index node is freed, the resources for that node (index semaphore or reader/writer lock and temporary key expansion buffers, if allocated) are not freed. The logic has been modified to free resources before freeing a memory index buffer.

Improved Server Stability When Using VSS For Backup

After a VSS backup, c-tree Server terminated with an unhandled exception. The problem requires a specific sequence of events to occur, so it is not expected to be a common occurrence.

Note: Anyone who is using the VSS feature should update their VSS writer DLL to a version that contains this correction.

Non-HUGE Reads Past 4GB Now Correctly Return Error

Data cache processing did not detect an attempt to read or write past the 4GB limit of a non-HUGE file. Logic has been added to detect this situation and return a **READ_ERR** or **FULL_ERR** for read and write operations, respectively.

ITIM_ERR (160) Corrected When CT_STRING UNCSEG Segment Is Not Null Terminated

Error 160 (**ITIM_ERR**) was seen while reading Unicode records when the key contained a *UNCSEG* segment on a *CT_FSTRING* field, the content was not terminated with `'\0'`, and *kseg* was configured as follows:

```
pkdef->kseg_ssiz == ctKSEG_SSIZ_COMPUTED
pkdef->kseg_styp == ctKSEG_STYP_PROVIDED
```

or

```
pkdef->kseg_ssiz == ctKSEG_SSIZ_COMPUTED
pkdef->kseg_styp == ctKSEG_STYP_UTF8.
```

The **ITIM_ERR** persisted after rebuilding the index.

Logic has been corrected to eliminate this error.

Note: It is possible that error 160 will occur on a file where it previously did not appear because the key formation has been changed. Error 160 may be seen on Unicode key segments that are using *CT_FSTRING* or *CT_FUNICODE* without null terminators. A rebuild should solve these occurrences.

IKRS_ERR (109) Corrected When Client Library Supports Fewer Key Segments Than Index Definition

Calls to find records by key value were not returning the expected results when the index used more than the default maximum number of key segments (12). The server was using the `MAX_KEY_SEG` option to support more than the default number of key segments, but the client was not recompiled to support a larger number of key segments. The client's call to get the segment information from the server returned only the key segments that the client could use based on its `MAX_KEY_SEG` compile-time option, so when forming keys for that index, segments beyond that limit were ignored and the key was improperly formed.

The logic has been changed so the function that returns key segment information now returns error 109, **IKRS_ERR**, ("too many ISAM key segments") if the information for all the segments does not fit into the buffer supplied by the client. When the call to get the key segment information fails with error 109, the client returns that error to the caller (except when the error occurs during a call to close a file, in which case the client code ignores the error and continues closing the file).

FNOP_ERR (12) Corrected When Reopening Deferred Closed File with Alternate Path

If a file's close is deferred (for example because the file is *TRNLOG* and has been updated and the transaction is still pending at the time the file is closed), and then the file is reopened using a different path specification (meaning that the file was originally opened with a full path and then reopened using a relative path, or vice-versa), the attempt to reopen the file fails with error **12** and *sysiocod* **-8**.

The logic has been modified to correct this situation.

RRED_ERR (407) Corrected When Rebuilding VARLEN FPUTFGET Indexes

A situation was discovered in which an index rebuild of a variable-length file in *FPUTFGET* mode would fail with error **407**. After the rebuild failed, subsequent opens of the data file failed with error **407**.

This problem was noticed when a variable-length file rebuild was done in multi-user standalone (*FPUTFGET*) mode and the logical end of file value in the data file's header was smaller than the correct logical end of file value.

Logic has been modified to correct this error.

Note: Once the file has been damaged by the rebuild, simply using the rebuild utility with this bug fix won't fix the damage. Restoring to a saved copy of the file is the best way to recover from this problem on a particular file.

IERR_COD (923) Corrected When Compacting VARLEN TRNLOG Data and COMPATIBILITY LOCK_EXCL_TRAN is in Use

When FairCom Server was configured to use the `COMPATIBILITY LOCK_EXCL_TRAN` option, a file compact operation on a variable-length *TRNLOG* data file may have failed with error **923**, and the following message was logged to *CTSTATUS.FCS*:

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

Logic has been corrected to eliminate this problem.

DMAP_ERR (957) Corrected On Overlapping ISAM File Open/Close Calls

An ISAM file open (such as **OPNRFIL**, **OPNRFILX**, **OPNIFIL**, or **OPNIFILX**) could fail with a **DMAP_ERR** (957) error if the timing was right when the ISAM file open/close calls overlapped with **OPNRFILX** calling **CLSFIL** on data file. This situation only occurs under specific timing and overlapping of ISAM open and close operations by multiple connections. Logic has been modified to prevent this error.

DLOK_ERR (42) Corrected for Memory File Add or Update Failures

Adding or updating records in a memory file could potentially fail with error **42** when other connections were reading and deleting records from the same shared memory file. Error **123** could also be seen under similar circumstances when reading memory records from variable-length data files. These errors were only seen in circumstances where a sufficient volume of data and users generated the specific timing necessary for these errors to occur. To resolve this issue, a more stringent internal tracking mechanism is now used to ensure this timing hole cannot occur.

FairCom has fully exercised this new code in our lab and it has passed all of our quality assurance tests. We recommend thorough application testing with this new code line to ensure proper application execution, in addition to watching for any memory growth over time (which could indicate a memory leak).

While addressing errors **42** and **123**, error **160** was occasionally seen due to multi-user interference if `ITIM_RETRY_LIMIT` was too low. To avoid error **160**, the choice of value to use for the `ITIM_RETRY_LIMIT` keyword should be on the order of the number of concurrent connections that are reading and updating the memory file.

MHDR_ERR (549) Corrected With Mirrored File Opens

An attempt to open a mirrored file failed with error **549**, causing FairCom Server to log the following messages to *CTSTATUS.FCS*:

```
Fri Jun 26 16:09:04 2015
 - User# 00018 The primary in the following pair appears to be the most current: 549
Fri Jun 26 16:09:04 2015
 - User# 00018 .\codes.dat|.\lms\codes.dat
Fri Jun 26 16:09:04 2015
 - User# 00018 .\codes.dat|.\lms\codes.dat:549
```

Logic has been revised to correct this.

RENF_ERR (67) Corrected During File Compact

On Unix systems, file compact failed with **RENF_ERR (67)** (Could Not Rename File) when the `TMPNAME_PATH` keyword was set to a different drive than the data file that was being compacted. (On Solaris, the sysiocod system error code was 18: EXDEV, cross-device link).

This problem was caused by the file compact function renaming the original data file to a temporary name using the path specified in `TMPNAME_PATH`. If the `TMPNAME_PATH` keyword was set to a directory on a different disk from the original file, it could cause an error because some Unix systems do not support renaming a file across disks.

Logic has been changed to use the path of the original data file for the temporary filename regardless of the `TMPNAME_PATH` keyword.

Errors 128, 150, and Possible Hangs Corrected With Batch Inserts or Updates Containing IDENTITY Fields and TCP/IP Connections

A batch insert or update on a data file containing identity field caused client calls to fail with error **128** or **150**. In some cases it would cause a hang. This error was seen when using the TCP/IP communication protocol; when using shared memory, no error occurs. Logic has been corrected to avoid an out-of-sync communication buffer.

Sort Error 484 Corrected During Sort File Create With Large Number of Keys

Creating an index on, or rebuilding an index of, a large data file could fail with error 484. c-tree logged the following messages to *CTSTATUS.FCS*:

```
Could not create new segment: unexpected nonmatching segment...: 683
sw012132.49X.002: 683
FAILED IO: ctwrtbuf...: 674
sw012132.49X.002: 674
Error on close file. locale:11  sysiocod:0  uerr_cod: 674
sw012132.49X
```

Logic has been modified to eliminate this problem.

Improved Variable-Length Data File Space Management

Variable-length data file deleted record space management has been improved to avoid the possibility to have space that is never reused.

Modified Rebuild Callback Event Handling

RBLCB_LOG now returns counter value

This is a change to a previous modification of support for rebuild callback mode *RBLCB_LOG*. The counter value is now returned to the client, which is used as an error indicator within the callback implementation. This value is returned in the `comm` buffer `qblg` member as is the case for other callback events. With this change both *RBLCB_MSG* and *RBLCB_LOG* return this value in the counter parameter.

Note: This changes the behavior of a *RBLCB_MSG* callback, which previously returned a default value of 0 for the counter parameter. Applications that expect the prior behavior may need to be modified to accommodate this change.

Cancelation sets RCBK_ERR in standalone

Logic has been changed so that, when the rebuild callback returns an error, the error is set to **RCBK_ERR** in standalone models. When **srvrblcbfnc()** is the callback function, its error code is returned.

Correct Key Value Now Updated with Physical Order Read and ISAM Key Buffers Disabled

When ISAM key buffer support was disabled and automatic ISAM key buffer refresh on update was enabled, reading a record in physical order and then updating it may update the wrong key value. This typically caused the update to fail with error **2** or error **3**. In some cases, the update succeeded but then a subsequent read of the record failed with error **160**.

Reading a record in physical order refers to calling **FRSREC()** or **FRSVREC()** with a data file number or calling **REDIREC()** or **REDIVREC()**.

Logic has been modified to correct this problem: When ISAM key buffer population is disabled, a physical record read now marks the key buffers as having their population disabled, so that a record update refreshes the key buffers as intended.

Named User Counts Now Correctly Applied at Group Level

When using the named user licensing attribute and user accounts that were members of the same group were connected at the same time, the following error was seen: Error **967** ("Logon is denied because this user account has reached its maximum number of concurrent logons").

For example, consider the case in which the named user limit was set to 3 and user accounts A and B were both members of group G. If both A and B had connected and then either A or B attempted to connect again, the login failed with error **967**, even though both A and B were below the named user limit of 3.

Logic has been modified to correct this error.

Correct File Block Behavior with Unix File Access

When **ctFILBLK()** was used to block access to a c-tree file on a Unix system, a client was sometimes able to open the file by specifying the path to the file in a different way than the path was specified in the file block call. For example, if **ctFILBLK()** was called with the filename *ctreeSQL.dbs/test.dat* and then the file was opened with the filename *./ctreeSQL.dbs/test.dat*, the file open was allowed, even though the file should have been blocked.

When the file was unblocked, if a client had been allowed to open the file, the file unblock detected this unexpected situation and logged these messages to *CTSTATUS.FCS*:

```
Fri May  1 10:27:44 2015
 - User# 00017  ctFILBLK: Unexpected error during unblock: a user has file open...
Fri May  1 10:27:44 2015
 - User# 00017  home/fctech/bin/ace/sql/data/ctreeSQL.dbs/admin_test.dat
Fri May  1 10:27:44 2015
 - User# 00017  ctFILBLK: Unexpected error during termuser()...: 24
Fri May  1 10:27:44 2015
 - User# 00017  home/fctech/bin/ace/sql/data/ctreeSQL.dbs/admin_test.dat: 24
```

Logic has been updated to correct for this situation.

UNIFRMAT Builds Now Able to Open V9 Created Files

Due to a change in file open logic, when a UNIFRMAT version of c-treeACE V10.3 attempted to open V9 files the file open failed with error 198 (**BIFL_ERR**). The **ctunf1** utility also failed. Logic has been modified to correct this.

3.6 Memory Usage Stabilization

Fixed Memory Leak When Creating Superfile Member with TRANPROC Support Disabled

A memory leak occurred in a c-treeACE library built with transaction support off when creating a superfile member. The leak occurred when calling **CreateIFileXtd** to create a member file in a super file (the creation of the super file itself does not leak). The logic has been modified to correct this.

Fixed Memory Leak When OPNIFIL Fails with Error 124

An **OPNIFIL()** call that failed with error **124** leaked memory if the data file had r-tree first or last field names set. The memory leak was resolved by freeing the memory before the function returns an error.

Fixed Memory Leak When Closing File without Freeing Range

Server memory usage was seen to increase without bound while running certain SQL queries. The problem was due to allocating a range, **ALCRNG()**, and then closing the file without freeing the range, **FRERNG()**. The logic has been modified to ensure that all allocated ranges are freed when the file is closed.

3.7 Communications Layer Fixes

Errors Ignored When IP Address Return for Host System Fails

FairCom Server is able to ignore some errors getting the IP addresses for the host system when it starts. Sometimes FairCom Server failed to start with error **891** (most commonly seen on Unix systems). The following message was logged to *CTSTATUS.FCS*:

```
- User# 00001  Failed to get IP address for host 'hostname': gethostbyname() returned error code 1
```

This error occurred when the system's host name (as shown by the **hostname** or **uname** *-a* commands) was not present in the */etc/hosts* file and could not be resolved by DNS.

FairCom Server now ignores this error except in situations that require getting the IP addresses for the host system (when using the "local connections only" and "node-based licensing" options).

Error 133 and Long Connect Times Now Avoided During Many TCP/IP Disconnects

When using the TCP/IP protocol, if many clients disconnect at the same time and then many clients try to connect, the clients' connection attempts may fail with error 133. This was caused by a 10 millisecond delay when closing a TCP/IP connection, which caused disconnects and connects to take longer than usual. This delay has been removed as it is not required.

Shutting Down with Connected Clients on Unix Now Frees Shared Memory Resources

When shutting down FairCom Server on a Unix/Linux system, shared memory resources (inter-process semaphores and shared memory regions) might not get freed for connections that exist at the time FairCom Server is shutting down.

The logic has been modified to signal the shared memory connections to stop when shutting down. The state of the shared memory connections is preserved so the thread that owns the connection is able to free the resources.

Allow Start of Unix-Based c-treeACE Server When Shared Memory Key Files Are Deleted

In an unusual situation when shared memory key files had been deleted, FairCom Server failed to start with the following errors in *CTSTATUS.FCS*:

```
SQLSHAREMM: Failed to bind/listen on Unix domain socket for shared memory: 98
FSHAREMM: Failed to bind/listen on Unix domain socket for shared memory: 98
```

The logic has been corrected to eliminate this problem.

Memory Leak in ISAM Unix Shared Memory Protocol Corrected

An ISAM client process leaked memory each time it connected to FairCom Server. A buffer that was allocated for an ISAM client shared memory connection on a Unix system was not freed when the connection was closed. The logic has been corrected to free the memory allocation when we dispose of the connection.

File Transfer with Unix Shared Memory No Longer Generates Client Exception

When using the file transfer function with the Shared Memory protocol on a Unix system, an unhandled exception sometimes occurred in the c-tree client library. The logic has been modified to correct this.

3.8 FairCom DB API API Fixes

Avoid Error 4108 on Compressed Files During Batch Find

A problem was seen in FairCom DB API using batch operations on compressed files. This problem can cause a number or different effects, such as seeing **Error 4108**, retrieving the wrong record, and data corruption on insert. This logic has been changed to correct this problem.

Create Table Now Updates sysindexes in Server DLL Model

A simple query was reported as running slowly. Indexes did not appear in sysindexes. Logic has been corrected to fix this problem.

ctdbAlterTable Now Retains Row-Level Security Information

A problem was causing the row-level filters to be lost during a TRUNCATE SQL operation. Logic has been modified to correct this problem.

Avoid CTDBRET_CANTCHKUID Error During ctdbConnect()

When connecting to a database after compacting its superfile, there have been reports of a possible occurrence of error **-21129** in SQL or error **4129** in FairCom DB API. The logic has been modified to resolve this problem.

Proper Errors Now Returned by ctdbGetIndexByUID

A case was found in which a system table contained incorrect table/index information, such as some indexes had a wrong UID. SQL tried to open an index scan, look for the index which it did not find, but no error was set. It would then continue with a NULL index pointer and crash. The logic has been corrected. In case of an error, it now returns NULL and index handle and sets the error code to **CTDBRET_NOSUCHINDEX**.

Prevent TNON_ERR (71) From ctdbRemoveTable

Calling the FairCom DB API **ctdbRemoveTable()** function to delete a table occasionally failed with **TNON_ERR** error 71. Logic has been modified to correct this error.

ctdbRenameTable Now Correctly Renames Index With Same Name as Table

A situation was discovered in which **ctdbRenameTable** did not rename an index that had the same name as the table. This issue has been resolved so that any index file having the same physical name (excluding the extension) as the data file will be renamed together with the data file.

Identity Field Now Properly Processed in ctdbInsertBatch

After inserting record using FairCom DB API and batches (**ctdbInsertBatch** calls) on tables with an identity field, although the record buffer was properly populated with the identity value, the field was marked as NULL. This caused it to not appear in c-treeACE Explorer (ISAM or SQL). The logic has been modified to correct this situation.

ctdbFreeRecord Memory Leak Corrected When Record Set is Active

If **ctdbFreeRecord()** was called while the record had an active record set, memory holding the partial key was leaked. Logic has been modified to correct this.

Note: An application that explicitly called **ctdbRecordSetOff()** before freeing the record handle would free this memory.

Correct Length Now Always Set for CT_2STRING With setFieldAsBlob and setFieldAsString

Calling **ctdbSetFieldAsBlob()** on a *CT_2STRING* field and passing a blob with size 65537 (which is larger than this filed can accommodate) resulted in success but the field contained only 1 byte and its size was actually 1. The same problem occurred, with different thresholds, on *CT_PSTRING* and *CT_4STRING*.

The logic was modified to add length check so that calling any **ctdbSetFieldAs*** on *CT_[2|4|P]STRING* and setting a value with length greater than the maximum allowed for the field type will result in a **CTDBRET_TOOBIG** error.

Field Callbacks Added for CHAR and VARCHAR Fields

For optimization purposes, the SQL engine handles CHAR and VARCHAR slightly differently from other field types. A side effect of this optimization is that field-level callbacks that may be in place were skipped on CHAR and VARCHAR columns.

Now the code has been changed to detect if field callbacks are in place on each single field, and for fields with callback in place, the optimized way of handling CHAR and VARCHAR columns is not used so the callback takes place.

Identify Legacy FairCom DB API Segment Error More Clearly

FairCom DB API operations on a legacy file may fail with error **4014** (Unknown field name) where it is not clear what the problem is. A new error code was added for the case when an absolute byte offset cannot be calculated for a legacy segment mode:

CTDBRET_INVLEGACYSEGMENT, error 4146: Unable to compute legacy segment offset.

ctdbRenameTable Now Renames Tables When Working Within SESSION_CTREE Mode

Prior to release V11, the FairCom DB API **ctdbRenameTable()** function was able to rename tables only with a valid database handle involved in the process, which made this function unusable when working with *CTREE* session modes.

In release V11 and later, new functionality has been implemented in **ctdbRenameTable()** so that, if a table handle that directly refers to a session is passed to the function, the rename is done at the *CTREE* level. If the first parameter is a database handle or a table handle referring to a database, the function continues to work as it has in the past.

3.9 Core System Updates

BATSETX no Longer Fails on First Call with a Buffer too Small

Calls to **BATSETX** did not return expected results with a small buffer. In some cases an application that used a buffer too small to hold the first record to be returned would not have an error indication. Logic has been modified to correct this situation.

Note: It is not unusual for a buffer to be too small to hold all the records being requested by the batch. However, when the first record does not fit, no record will be returned and no progress will be made on the batch request. Another call to **BATSETX** will need to be made with a larger buffer.

In V11 and later, a new mechanism has been provided to return the buffer size required to avoid the **BTBZ_ERR**. This new behavior only occurs if the **BATSETX** call that failed with **BTBZ_ERR** has **BAT_BTBZ_ERR** OR-ed into the **BATSETX** mode parameter. For example, the following **BATSETX** call would behave as usual if no **BTBZ_ERR** was returned, but the second LONG integer return value (request[1]) would have the minimum buffer size requirement for the subsequent **BATSETX** call to succeed (instead of *bavail*):

```
LONG request[5] = {0};
pVOID buffer[8192];
NINT rc;

rc = BATSETX(datno,request,buffer,8192,BAT_GET | BAT_PHYS |
    BAT_RET_REC | BAT_BTBZ_ERR);
```

Batch Update Now Correctly Recognizes BAT_RET_BLK Record Format

When using batch update to add variable-length records that were read using a batch physical read with the *BAT_RET_BLK* option, the batch update failed with error **SDAT_ERR**. Batch update logic has been modified to recognize the *BAT_RET_BLK* record format.

Blocking Record Read Improvements

BLKIREC() was not immediately checking the next record when the current record failed the blocking condition. Instead it was waiting for the next change. The logic has been corrected so that, when a record fails the blocking condition in **BLKIREC()**, it immediately attempts to read the next record or previous record, depending on the opcode passed to **BLKIREC()**.

When a record did not pass the **BLKIREC** filter, it attempted to read the next record. If the current record was not read in full, the next record read could fail with error **893**. In this situation it is desirable to return the length of the current record to the caller so the caller can read the current record in full, then call **BLKIREC** again. The client code now passes the record length back to the caller in case of error **893** on a **BLKIREC** call.

When **BLKIREC** tried to read the next record because the current record failed the filter condition, it changed the opcode to read next. When **BLKIREC** needed to wait on the system queue after such a retry, it needed to restore the opcode to the original opcode so that it performed the original record read operation after notification informed **BLKIREC** that the file had changed. The logic has now been modified to behave accordingly.

A call to **BLKIREC()** that specified a blocking condition was leaving a record locked if it acquired a lock when reading the record and then found that the record did not meet the blocking condition. The logic has been modified to correct this.

Client TFRMKEY API No Longer Writes Non-Zero High Word Record Offsets into Key Value

When a c-tree client's **TFRMKEY()** function was called for an index that supports huge (8-byte) file offsets and allows duplicates, **TFRMKEY()** sometimes set the first four bytes of the record offset at the end of the key value to an unexpected non-zero value rather than setting it to the documented value of zero. Logic has been modified to correct this.

GETIFIL No Longer Fails Due to Alignment Adjustments

On some platforms, an index may have an incorrect name when opening an existing table with indexes or you may see a segmentation fault when opening an existing table with indexes.

When calling **GETIFIL()**, you must check that the returned length is greater than or equal to the input length because the output buffer (`bufptr`) will hold invalid information. For some systems, two calls to **GETIFIL()** may be required to get the correct *IFIL* structure size, as alignment requirements may not be considered when the input length is less than the packed structure size.

Corrected Error Handling for ctVERIFYidx and ctVerifyFile APIs

The following issues were addressed:

1. When **ctVERIFYidx** was canceled by the callback returning an error, it failed with **VRFY_ERR** rather than the documented **VFYTRM_ERR**.
2. **ctVERIFYidx** reported false errors on an index. Repeating the call eventually showed no errors.
3. The **ctvfyidx** utility failed with **LERR_ERR** when the *-excl* option was set.

These issues were corrected. As a result, **ctVERIFYidx** now requires the file to be open in *ctEXCLUSIVE* mode or in *ctREADFIL* (read only) mode to ensure it does not incorrectly report errors if other users make changes while the verification is running. Opening with *ctSHARED* will fail with **LERR_ERR**.

Note: A call is made to **CTFLUSH** on the index prior to leaf verification to ensure all changes are on disk. In some cases, such as for *SUPERFILE* members, **CTFLUSH** can fail. An error message is logged, but the verification will continue. In this case, a **VRFY_ERR** could indicate the existence of updated index nodes in cache. The application should ensure that the data has been properly flushed.

The **ctvfyidx** utility now defaults to *ctREADFIL*. It uses *ctEXCLUSIVE* when the *-excl* option is specified.

Older versions of **ctvfyidx** will only work with the *-excl* option when connecting to newer servers.

Corrected ctVERIFYidx Reports For Non-ctPREIMG Files With Key Marks

In an unusual situation, a corrupt index was reported for a *ctPREIMG*/*ctTRNLOG* index file that contained key marks if the file mode was changed to non-tran before **ctVERIFYidx()** was run.

```
Verifying index leaf nodes and checking key values...
  0% complete.
Leaf Node 12c00x retrieval disagreement (i 23 ct_elm 22 ct_fnode
12c00x)[00(
<9991>...100% complete.
Index page scan finds entries=136  header=133
Scanned entries include 59 entries no longer part of index.
Internal Index Verify: Problems Discovered
```

If the *ctTRNLOG* filmod was enabled in this file's header, the call to **ctVERIFYidx()** succeeded. The logic has been corrected to resolve this issue.

Error IAIX_ERR (608) Corrected When Compacting or Rebuilding Files Containing SRLSEG or SCHSEG Segment Modes

After a file that used the *SRLSEG* or *SCHSRL* segment modes was compacted, a subsequent compact or rebuild operation failed with **IAIX_ERR** error 608 (*IIDX* attributes do not match file). An ISAM open of the data file typically worked properly other than the 608 error. The logic has been changed to eliminate the error situation.

Note: This change will prevent the 608 error from occurring on files created after the change has been applied. However, a data file that has already been created with the *SRLSEG* or *SCHSRL* segment modes may still experience this error when compacted. In this case, the 608 error can be avoided by specifying the *updateIFIL* option in the *tfilno* field of the *IFIL* structure when performing the compact or rebuild (e.g., use **ctcmpcif** or **ctrbldif** and specify *-updifil*). If you have a file that is already in this state (with those fields set to zero), you will need to use this workaround to correct it.

Improved Handling of Encryption Attributes During File Compact and Rebuild

The file compact utility (**ctcmpcif**) and the **CompactIFile()** API function now better handle file encryption The following improvements were made:

1. The compact API function and utility can optionally change encryption attributes. To use this option when calling the compact API function, use the **setIFILoptions()** macro to set the *setencryptIFILoption* bit in the *tfilno* field of the IFIL structure whose address you pass to the compact API function, and call **ctSETENCRYPT()** with the desired encryption attributes before calling the compact API function. For example:

```
NINT rc;

myifil.tfilno = setIFILoptions(setencryptIFILoption);
rc = ctSETENCRYPT(ctAES32, NULL, 32);
rc = CMPIFIL(&myifil);
```

2. The **ctcmpcif** utility now supports an option to specify the encryption cipher for the data and index files created by the compact operation. Usage:
   -encrypt=cipher - Create the compacted file using the specified cipher:
   - for AES, use *aes16*, *aes24*, or *aes32*
   - for Blowfish, use *blf8*, *blf9*, ..., or *blf56*
   - for DES, use *des8*, *des16*, or *des24*
   - for Twofish, use *twf16*, *twf24*, or *twf32*
   - for no encryption, use none
   - for the deprecated CAMO option, use *camo:MY_ENCRYPTION_KEY* **Note**: CAMO or "Camouflage" is an older, legacy method of hiding data, which is not a standards-conforming encryption scheme, such as AES. It is not intended as a replacement for Advanced Encryption or other security systems.

   Note: If an index file does not exist, the original data file's encryption attributes are used when creating that index file.

   To change the encryption attributes of a file using the compact API function from a client, you must add the option `CHANGE_ENCRYPTION_ON_COMPACT YES` to *ctsrvr.cfg*. Otherwise, the operation fails with error **NSUP_ERR** (454, not supported).
3. Compact now always deletes and recreates index files after saving resources from the original index file. This provides the expected behavior of (possibly) reducing the size of the index file.

Client Library Exception Corrected When ctWNGV is NULL

An exception was generated in the client library an internal global variable, *ctWNGV*, was NULL. For example, if an application using the c-tree multi-threaded client library created a thread using a system thread creation function like **pthread_create()** and then called **CLIFIL()** without first calling **ctThrdAttach()**, **CLIFIL()**'s internal call would raise an exception because *ctWNGV* was NULL. Logic has been modified to eliminate this exception.

ctGetFileUsers() Now Returns Correct User File Number for Multiple File Open Instances of a Single Connection

The **ctGetFileUsers()** function returned the wrong user file number when a connection had multiple open instances of a file. When a connection had a file open more than once, a call to **ctGetFileUsers()** returned the same user file number for all open instances of the file by that connection. Logic has been modified to correct this.

Identity Field Support Now Available in LOCLIB model with Single-User TRANPROC

Identity field support was disabled in the *LOCLIB* model with single-user *TRANPROC*. **NSUP_ERR (454)** was seen when trying to use identity field functions in the *LOCLIB* model whose local-side library was the single-user with transaction support model. Logic has been modified to correct this.

3.10 Interim Build Modifications

Unhandled Exception Fixed for Transaction-Controlled Index File Under Heavy Update Activity in V10.4

Recent versions of FairCom Server (V10.4.0 and V10.4.1) were found in testing to occasionally terminate with an unhandled exception when under heavy load involving update activity on a transaction-controlled index file. This was due to enhancements introduced in V10.4 for additional cache performance optimizations. The logic has been modified to correct this occurrence.

Unhandled Exception Fixed When Shutting Down FairCom Server DLL on System That Does Not Support Memory Tracking

An exception was seen in applications that used the FairCom Server DLL on a system that does not support the memory tracking feature (the memory tracking feature is supported only on Windows and Linux). When the application shut down the database engine, an unhandled exception occurred when freeing memory. The logic has been modified to eliminate this problem.

Note: This was only seen when using the FairCom Server DLL. The standard FairCom Server does not have this problem.

This bug only affects V10.4.0 and later.

Incorrect Index Member Key Counts Corrected When Using KEEPOPEN_LIST

This revision is important for customers who are in the following category:

- You are using the `KEEPOPEN_LIST` option with *non*-TRNLOG files.

  and
- You are using a FairCom Server version between V10.3.1.29082 and V10.3.1.30809 or between V10.4.0.29802 and V10.4.0.30809.

An index member file that was kept open using the `KEEPOPEN_LIST` feature may have shown an incorrect key count in its header.

(As a workaround for this issue, it is possible to disable the `KEEPOPEN_LIST` flush feature by adding the option `KEEPOPEN_FLUSH NO` to *ctsrvr.cfg* and restarting FairCom Server. The workaround is not necessary after upgrading to 10.3.1.30810 or a later version that contains this fix.)

Automatic Recovery Issues Corrected with Aborted Index Operations Introduced in V10.4

In certain rare situations after an automatic recovery, two conditions could exist that could cause data-integrity problems:

- Key values whose Adds were aborted may appear in the index.
- Key values whose Deletes were aborted may be missing from the index.

A performance enhancement introduced in V10.4 changed the way key-level locks and associated abort-node list entries were resolved. Key-level locks and abort-node list entries permit index nodes to be updated by transactions without locking the nodes for the duration of each transaction.

The problem situation occurs when *all* of the following are true:

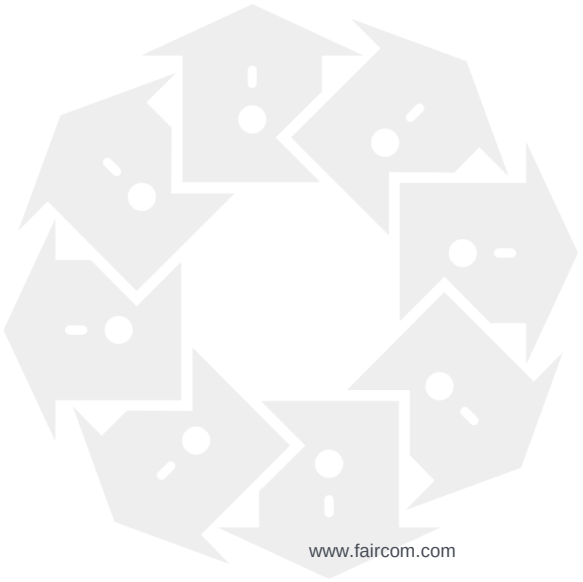
- An index is under transaction processing control (has *TRNLOG* file mode); *and*
- A node split causes key-level locks to be moved to the new sibling node; *and*
- Some of the key-level locks moved to the new node are for aborted transactions; *and*
- The new node is written to disk; *and*
- The system crashes before a checkpoint is written containing the abort node list entries for the new node.

The logic has been modified in this version to eliminate this problem. After installing this version, it is important to *rebuild any indexes that might be impacted by this situation*, or to be safe, simply rebuild all indexes under transaction processing control.

Note: **If you choose NOT to upgrade your c-tree Server** - Adding `COMPATIBILITY NO_UPDMARKS` to existing V10.4 servers will disable this performance enhancement and eliminate the automatic recovery problem. Once this keyword is added, it is important to rebuild any indexes that might be impacted by this situation, or to be safe, simply rebuild all indexes under transaction processing control.

4. c-treeACE SQL Updates and Corrections

In this release, the SQL interface was the recipient of numerous improvements, which are listed in this section.



4.1 Critical Production Updates

Prevent c-treeACE SQL Termination When Accessing LONG Types

The server was found to stop unexpectedly with the following CTSTATUS message:

```
Unexpected internal c-tree(R) error #7495 (uerr_cod=0)
```

This situation was sometimes seen when using the **ctdbSeekRecord()** API or **SQLGetData()** type calls handling long or blob types. In those circumstances, it was seen when another user updated the same record just before the current user attempted to read it. This problem may affect any FairCom DB API or c-treeACE SQL record read.

The logic has been modified to prevent this from occurring.

Prevent Crash With ON Clauses Containing Sub-Queries

A server crash was seen when executing an obscure query that involved a sub-query in the ON clause. The logic has been updated to correct this.

Prevent Crash When Creating Dynamic Indexes

A very rare situation was discovered in which a server crash could result if a dynamic index created by the temporary table manager was exactly large enough to spill from memory to a disk swap file. This error only occurred at a precise byte boundary. One extra byte, or one less byte, would prevent this issue from occurring. Logic has been revised to correct this problem.

Prevent SQL PANIC Condition with Selected Queries

A server panic forced a server shutdown when running a query that updated a field calculating its value based on a query result that joined a second table with the row being updated. Logic has been modified to correct this.

Prevent Crash During Outer Join Optimization

In rare cases, c-treeACE SQL was seen to issue **terr (7497)** during join optimization. Logic has been modified to correct this.

Prevent Unhandled Exception for SQL Insert Statements Involving ROWID

When a SQL insert operation selects ROWID (and optionally, some column values) from a table and omits one or more columns of the target table, c-treeACE SQL terminated with an unhandled exception. Logic has been modified to correct this behavior.

Prevent Potential Crash While Reading LONG VARCHAR Data

A situation was discovered that may cause c-treeACE SQL to crash or memory corruption if the buffer provided is smaller than the LONG VARCHAR or Unicode field. Logic has been modified to correct this problem.

Prevent Crash From ODBC SQLCursor() Call With NULL

If a NULL value was passed to the **SQLCursor()** constructor, the server crashed. Logic has been corrected to handle this situation and return an error if a NULL value is passed.

Crashes and Hangs Corrected if an SQL Query Contains Many INNER JOINS Based on Equality

c-treeACE SQL could crash or hang if a query contained a large number of INNER JOINS with criteria based on equality. The SQL optimizer has logic to check join predicates and when two predicates share a common value for EQUALs, the OPTIMIZER can infer additional EQUALs. If we have join predicates like P1=P4 and P6=P4 (with any valid terms commutation like P1=P4 and P4=P6) the OPTIMIZER infers P1=P6.

Logic has been corrected to ensure that the number of predicates does not exceed the maximum size allotted for the list of all predicates.

Prevent Large Number of IN Values Causing a Server Crash

In rare situations when processing a query containing a very large number of constant entries in an IN operator, a stack overflow and a server crash were sometimes seen. The logic has been thoroughly analyzed and changes have been made to eliminate this issue.

New logic speeds up execution when the number of entries for the IN operator is large (>1000) by avoiding the use of the IN operator on indexes.

LONG Type Stability Improvements

Logic has been modified to fix possible crash and security holes during LVC/LVB handling. These modifications improve stability and security.

Ensure SQL Database Creation at Server Startup Completes Before Allowing ISAM Connections and Server Shutdown

If c-treeACE SQL received a shutdown request while it was still creating the SQL dictionary, the server would hang with a message similar to the following logged to *CTSTATUS.FCS*:

```
 - User# 00001   c-tree(R) V10.4.0.46477 Server Is Operational       -SN 08000003
 - User# 00017   Server shutdown initiated
 - User# 00017   Communications terminated
 - User# 00017   Perform system checkpoint
 - User# 00001   WRITE_ERR: L0000001.FCS at 0:28000x  sysiocod=6 bufsiz=8192 bytes written=0[0]  ioLoc=0:
37
 - User# 00001   ctwrtlog: sysiocod: 6
 - User# 00017   Server shutdown completed
 - User# 00001   ctwrtlog:    error: 37
```

Logic has been modified such that, at server startup, the ISAM logon threads wait until SQL database creation has finished.

Avoid Infinite Parsing Recursion

A complex query was improperly processed by the query engine, causing a server crash. The logic has been corrected to eliminate this problem.

Avoid Crash When Database Name Exceeds Maximum Length

A stack buffer overrun was seen in c-treeACE SQL Server when a SQL client connected to a database whose name exceeded the maximum SQL database name length. The logic has been corrected by truncating the copied name to the maximum name length.

Correct Handling of Recursive Subquery Clauses

A server crash was seen when running a query of the form `SELECT * FROM (<subquery>);` where `<subquery>` was a recursive query. c-treeACE SQL has been modified to identify this situation and returns the expected syntax error.

Corrected Direct SQL Update of LONG Fields

The Server could potentially become unstable when **ctsqlExecute()** was called more than once with LVARCHAR or LVARBINARY parameters. The logic has been changed to properly re-initialize the internal structures for LONG fields, thereby correcting this problem.

Prevent MM Subsystem Crash

The logic has been modified to correct a Server crash due to an obscure, unexpected internal state in the internal Memory Manager.

Note: The MM temporary table manager is disabled by default in V10.4 and later and replaced with a much improved alternative subsystem.

Corrected Infinite Parser Loop

In a certain, unusual situation, a specially formatted query entered an infinite loop during parsing. The logic has been modified to correct this error.

Corrected Buffer Overruns When Using Direct SQL Interface

A server crash was experienced due to a buffer overrun when erroneously calling one of the following functions on a column of a type different than the type the function returns (e.g., calling **ctsqlGetSmallInt** on a big int):

- ctsqlGetNumeric
- ctsqlGetSmallInt
- ctsqlGetInteger
- ctsqlGetReal
- ctsqlGetFloat
- ctsqlGetDate
- ctsqlGetMoney
- ctsqlGetTime
- ctsqlGetTimeStamp
- ctsqlGetBit
- ctsqlGetBigInt
- ctsqlGetNumericAsString

Partial Sort Table Scan No Longer Eliminates Potentially Significant Records

When a partial scan of Sort Table was executed, it eliminated potentially significant records that had not been sorted yet. This issue arose in V10.3 and greater and affected SQL queries when using TOP with SKIP and Order By. The logic has been modified to correct this.

Corrected Result Sets Returned with LEFT OUTER JOIN and TOP and SKIP Conditions

An optimizer issue caused the return of a wrong portion of the recordset when specifying a `TOP n SKIP m` clause on a query with outer joins. The logic has been corrected to resolve this issue.

Corrected Query Results with Literals Over 2048 Characters

A SQL query with a restriction depending on a literal value over 2048 characters did not return matching rows. This issue had been corrected by increasing the maximum pushdown expression size to the maximum SQL field size (8192).

Correct Results Now Returned When Subquery Contains SKIP

A query with a SKIP in a subquery produced the wrong result. For example, the following queries should all return the same number of rows:

```
select c.cust_id,c.name,(select top 1 cust_id from cust where ref_id=c.cust_id) as ref from cust c;
select c.cust_id,c.name,(select top 1 skip 1 cust_id from cust where ref_id=c.cust_id ORDER BY cust_id)
as ref from cust c ORDER BY c.cust_id;
select c.cust_id,c.name,(select top 1 skip 2 cust_id from cust where ref_id=c.cust_id ORDER BY cust_id)
as ref from cust c ORDER BY c.cust_id;
```

The logic has been modified to correct this.

4.2 c-treeACE SQL APIs

ADO.NET Provider Exception Classes Made Serializable

All **Exception()** derived classes in the ADO.NET provider have the [SerializableAttribute] to make them serializable in V11 and later.

ADO.NET Provider - Improved Handling of NUMERIC Type

The ADO.NET driver did not handle some NUMERIC values with precision above 28-29 because it was using the decimal class during the type conversion.

The decimal class is an approximate type and NUMERIC is exact. For more about these classes, see MSDN (http://msdn.microsoft.com/en-us/library/364x0z75.aspx).

Logic has been modified to avoid converting NUMERIC to decimal types.

ADO.NET Provider - Changed TINYINT Handling from byte to sbyte

A TINYINT output parameter was returned with an underlying type of byte. Attempting to retrieve a negative TINYINT resulted in an Overflow exception thrown in **ExecuteNonQuery**. TINYINT handling has been changed from byte to sbyte.

ADO.NET Provider - Charset Option in Connection String Is not Longer Ignored

The ADO.NET provider supported an option in the connection string to specify the charset to be used to interpret CHAR and VARCHAR columns. This option was ignored when the connection was closed and then re-opened specifying a different charset.

ADO.NET Provider - Improved Connection Pooling

The ADO.NET Provider's Connection Pooling feature returned a socket error if the server was restarted and a pooled connection was reused with a call to **Open()**. Logic has been modified to correct this behavior.

ADO.NET Provider - CtreeSqlDataReader Close() No Longer Aborts Automatic Transactions

When using automatic transactions, ADO.NET aborted calls to a stored procedure that inserted, updated, or deleted records and returned a recordset. **CtreeSqlDataReader Close()** used to roll back the automatic transaction causing the observed behavior. The issue has been fixed by changing the close method to always commit the transaction, preserving any insert, update, or delete operation in the stored procedure.

ADO.NET Provider - FcSQLException Constructor Now Thread-Safe Preventing Initialization Errors

The initialization in the **FcSqlException()** constructor called by **BuildTable()** was not thread-safe. Logic has been changed to correct this.

ADO.NET Provider - Last Byte Now Returned from LONG Columns with Length Less than Three

The ADO.NET provider was stripping the last byte from LVARCHAR columns with a length less than 3 characters. For example, a value of 'AA' might be returned as 'A', although 'AAA' behaved correctly. The logic has been changed to correctly handle these situations.

ADO.NET Provider - OUT Stored Procedures Parameters Now Correctly Set

When executing a stored procedure with OUT/INOUT parameters and no resultset using **ExecuteReader()**, output parameters were not set. Logic has been modified to correct this behavior.

ADO.NET Provider - OUT/INOUT SP parameters not set to NULL

The ADO.NET provider was not properly setting IN and INOUT stored procedure parameters to null when the underlying value was null. The logic has been modified to correct this.

Note: This is a behavioral change that may affect existing programs, which may use the returned values without checking for null. Before the change this did not cause any problem since the value was never null; now trying to access the value may throw an exception if the value is null.

ADO.NET Provider - Prevent Reusing a Disposed CtreeSqlConnection

An issue was encountered with code similar to the following:

```
DbConnection conn
...
if(conn == null)
 conn = new CtreeSqlConnection();
...
using (conn)
{
 doStuff(conn);
}
...
doStuff(conn); // conn disposed by "using"
```

The **using()** was calling **conn.dispose()** and then the disposed connection was re-used. A disposed `CtreeSqlConnection`, and any disposed object in general, must not be reused. Prior to this change, the driver logic allowed this to work normally except that **Commit()** and **Rollback()** had no effect.

Because disposing a `CtreeSqlConnection` also disposes the underlying `DbConnection`, which could have other possible side effects, c-treeACE now throws an exception if **Open()** is called on a disposed `CtreeSqlConnection`.

ADO.NET Provider - Removed Unexpected Assert Exceptions

The ADO.NET provider's CtreeSQLException class had an Assert instruction that was throwing an unneeded Assert exception in certain cases (such as an "Invalid Statement" exception). Logic has been modified to correct this behavior.

ADO.NET Provider - SqlDataReader.GetSchemaTable() Now Returns Correct Information

The **SqlDataReader.GetSchemaTable()** method returned wrong information for IsKey and IsUnique.

From Microsoft documentation: “IsUnique is true if the Column is of type timestamp”. c-tree ACE SQL does not have “timestamp” in the same way MS-SQL has and so “IsUnique” is now always set to false.

IsKey should be set to “true” on columns composing a set of columns in the rowset that, taken together, uniquely identify the row. The ADO.NET driver is not able to properly identify such columns without a major performance impact so instead of returning incorrect information, the driver has been modified to always return false.

ADO.NET Provider - Prevent Mixing Transaction Contexts

This modification adds a check to avoid assigning a transaction to the wrong connection and throw an exception when this situation occurs. This prevents you from assigning a transaction from a different connection, for example:

```
DbConnection conn1
DbConnection conn2
DbCommand cmd1 = conn1.CreateCommand()
Dbcommand cmd2 = conn1.CreateCommand()
cmd1.Transaction = conn2.BeginTransaction()
```

ADO.NET will now throw an exception in the `cmd.Transaction set{}` if the new transaction is not derived from the expected connection.

ADO.NET Provider - Improved Exception Handling

If an exception was thrown in **AutoAddParameters**, it was not converted to a higher level **CtreeSqlException** and, therefore, may not be caught. Proper conversion logic was added to avoid this problem. This is a change in behavior because an exception is now caught.

ADO.NET Provider - Corrected -20123 Error When Updating Rows with LONG Fields

An unexpected **-21023** error (Table does not exist) was returned when executing an Update statement with parameters on an LVC (or LVB) column and no records matched the Where clause. Logic has been updated to properly handle this situation so no error is generated.

ADO.NET Provider - Corrected LVARCHAR Empty String Handling

The ADO.Net driver returned a NULL string for an LVARCHAR column containing a string with length 0. Logic has been modified to distinguish between an empty string ("" where length is 0) and NULL (where the value is undefined).

ADO.NET Provider - Improved Handling of FLOAT, DOUBLE, and REAL Data Types

The handling of FLOAT, REAL, DOUBLE SQL data types in the ADO.NET driver has been improved (in particular the floating point precision was not respected). This change included adding a FLOAT_CLEANUP definition to the Visual Studio project.

ADO.NET Provider - Corrected TimeStamp Values Returned

A problem in c-treeACE SQL Explorer was caused by the **FcValue.GetString()** method returning an incorrectly formatted value for timestamp columns. Logic has been corrected.

ODBC - Infinite Loop Corrected in Client Driver

The ODBC driver entered an infinite loop in **SQLPutData()**. Logic has been changed such that the driver detects the infinite loop, breaks it, and returns an error sqlstate **22026** (String data, length mismatch).

ODBC - SQL_C_DEFAULT Now Correctly Maps BIGINT to 64-bit Integer

When fetching BIGINT columns as SQL_C_DEFAULT, the result was incorrect. SQL_BIGINT was mapped to SQL_C_CHAR when SQL_C_DEFAULT native storage was specified.

The behavior has been modified to change the mapping to SQL_C_SBIGINT.

Note: This is a change of behavior.

ODBC - SQLBindCol with NULL Indicator and ROWSET Size > 1 No Longer Crashes Client

An application crash was experienced when calling an ODBC Fetch function when using ROWSET size > 1 and binding a NULL length indicator. Logic was modified to correct this.

ODBC - SQLExecDirect with Parameters Set to SQL_DATA_AT_EXEC No Longer Generates Client Crash or Syntax Error

When a parameter was bound with SQL_DATA_AT_EXEC and **SQLExecDirect** was called, either a Syntax error (**-20003**) occurred or the application crashed with an invalid memory read access fault when **SQLParamData** was called. This problem affected V10.1 and later ODBC drivers only. Logic has been modified to correct it.

ODBC - SQL_ROWSET_SIZE Attribute Now Returns Correct Number of Rows

ODBC 2.*x* applications that called **SQLExtendedFetch** and set the SQL_ROWSET_SIZE statement attribute to a value greater than 1 could see extra rows returned. The logic has been changed to correct this behavior.

ODBC - SQLGetData Memory Overwrite Corrected

If **SQLGetData** is called with a *bufferlength* argument of zero on data types LVARCHAR or LONG NVARCHAR **SQLGetData** will write 1 byte before the buffer. Logic has been corrected to eliminate this situation.

ODBC - SQLGetInfo Returns Corrected Information

SQLGetInfo was found to return incorrect information. **SQLGetInfo(***SQL_MAX_INDEX_SIZE***)** now returns 1024 rather than 0 (unknown/unlimited).

ODBC - SQLGetTypeInfo Returns Corrected Information

The following changes were made to **SQLGetTypeInfo** results:

- **TIMESTAMP** -
  Values COLUMN_SIZE, MINIMUM_SCALE, and MAXIMUM_SCALE were changed to reflect that fractional seconds are supported.
- **TIME** -
  Values COLUMN_SIZE, MINIMUM_SCALE, and MAXIMUM_SCALE were changed to reflect that fractional seconds are supported.
- **LVARCHAR** -
  Value COLUMN_SIZE was set to 2GB-1.
  CASE_SENSITIVE was changed to TRUE.
  SEARCHABLE was changed to SQL_UNSEARCHABLE.
- **CLOB** -
  Value COLUMN_SIZE, was set to 2GB-1.
  CASE_SENSITIVE was changed to TRUE.
  SEARCHABLE was changed to SQL_LIKE_ONLY based on the documented behavior.

  Note: The LVARCHAR data type is recommended for CLOB support.
- **BINARY** -
  Value COLUMN_SIZE was set to 8192.
- **VARBINARY** -
  Value COLUMN_SIZE was set to 8192.
- **LVARBINARY** -
  Value COLUMN_SIZE was set to 2GB-1.
  SEARCHABLE was changed to SQL_UNSEARCHABLE.
- **BLOB** -
  Value COLUMN_SIZE was set to 2GB-1.
  SEARCHABLE was changed to SQL_UNSEARCHABLE.

  Note: The LVARBINARY data type is recommended for BLOB support.

ODBC - SQLNativeSql Now Returns Correct Statement Length

An ODBC application failed with the following message: "Insufficient base table information for updating or refresh." The **SQLNativeSql** function was updated to correctly evaluate string length of the input statement when ctSQL_DYN_STMT is active.

ODBC - SQLSetCursorName Corrected Buffer Addressing

An application crashed with an invalid memory read violation because SQLSetCursorName we reading past the end of a buffer in certain situations. Logic has been modified to eliminate this problem.

JDBC - Character Set Can Now Be Specified in the Connection URL

In V11 and later, special "high-bit" characters (ASCII > 127) can be properly displayed in c-treeACE SQL JDBC-based tools. The JDBC driver encodes all strings as UTF-16 because that is the format used by Java. Prior to this change, the source was assumed to be UTF-8. It is now possible to specify the character set in the URL so it can be handled properly.

The character set can be specified in the connection URL as follows:

```
jdbc:ctree:port@host_name:db_name?characterEncoding=encoding
```

The URL string has the following components:

- *jdbc:ctree* - An identifying protocol and subprotocol string for the c-treeACE SQL JDBC Driver.
- *:port* - The port number associated with the c-treeACE SQL server on the host system.
- *@host_name* - Name of the server system where the database resides.
- *:db_name* - Name of the database.
- *?characterEncoding=encoding* - Replace *encoding* with a valid Java encoding name (e.g., *US-ASCII*, *ISO-8859-1*, *UTF-8*, etc.).

JDBC - Allow Disabling Socket Timeout

The JDBC connection dropped after 30 minutes when executing long-running queries. The socket timeout defaults to 30 minutes and can be changed with the Driver Properties. Setting `sock_timeout=0` now results in infinite timeout (previously a setting of 0 resulted in the default of 30 minutes).

JDBC - Corrected Error -26049 (Invalid column number) During SELECT ... FOR UPDATE Query

Running a SELECT ... FOR UPDATE query with the JDBC driver was failing with error **26049** (Invalid column number) because of an internal JDBC driver issue. The JDBC driver has been modified to correctly handle the query.

JDBC - Corrected Exception When Character Set Not Provided In URL Connection String

When a JDBC connection URL specified either the user or the password but did not specify the characterset to use, the JDBC driver threw an exception. The issue has now been fixed.

JDBC - Driver.connect Method Now Compliant with Incorrect URL Handling

The Driver.connect method should return null when the JDBC URL is inappropriate for the driver instead of throwing an exception. In multi-database environments, this will prevent the DriverManager from connecting to any databases registered after c-tree. The logic has been corrected to return a NULL if the URL is unacceptable.

JDBC - LVARCHAR Empty String Now Correctly Returned

The JDBC driver returned a null string for an LVARCHAR column containing a string with length 0. Logic has been modified to return the correct empty value.

JDBC - Improved Type Conversion Error Message

An exception message, "Invalid conversion error from 9 to 93", has been improved. The internal numeric values have been replaced with type names in the exception messages. The above message becomes "Invalid conversion error from TIME to TIMESTAMP."

JDBC - DatabaseMetaData.getTable Now Conforms to JDBC Standards

The **DatabaseMetaData.getTable()** method was shown to be non-conforming to expected JDBC behavior.

- A third-party reporting package did not show tables in its domain designer. This product was calling **DatabaseMetaData.getTable()** passing a type that our driver does not support. Logic has been modified such that unknown types are ignored, making it behave as other JDBC drivers.
- **DatabaseMetaData.getTable** returned an empty result when looking for valid and invalid table types. Logic has been modified to correct this error and return information for the valid types.

JDBCjava.sql.Connection.isValid Now Returns Correct State

JDBC exposes the **java.sql.Connection.isValid** method, which returns "true" if the connection had not been closed and was still valid. The implementation of this method in c-treeACE SQL JDBC incorrectly returned "false." This behavior has now been corrected.

PHP - Components Now Match non-Thread-Safe Defaults for Windows IIS PHP Installations

The default PHP installation for Microsoft Windows IIS is not thread-safe, which implies that any PHP extensions should also be compiled as "not thread-safe." Previously, c-treeACE PHP components were compiled and distributed as thread-safe, which were incompatible with default PHP IIS installations. For better default compatibility, c-treeACE SQL PHP components are now compiled and distributed as non-thread-safe.

PHP - Integer Values Now Correctly Handled

The PHP interface was not correctly reading integer values from tables, which caused it to show unusually large values. Logic has been corrected.

Direct SQL - ctsqlSetParameter Memory Corruption Corrected

A client application crash was seen because of memory corruption when using **ctsqlSetParameter(...)** with a *TPE_DT_CHAR* parameter. Logic has been modified to correct this.

Direct SQL - Corrected ctsqlNumericToString and ctsqlStringToNumeric Unicode Handling

A problem was discovered converting a string into numeric and vice versa using the **ctsqlStringToNumeric** or **ctsqlNumericToString** functions when the library is compiled with Unicode support. The logic has been modified so that the two functions consider if the string passed in (or in output) is ASCII or UTF16 depending on the Unicode compilation switch.

DSQL - ctsqlExecute() Unhandled Exception Corrected

A client application crash was occasionally experienced when the **ctsqlExecute()** function was called without a previous **ctsqlPrepare()** call or after a **ctsqlExecuteDirect()** call. Logic has been modified to eliminate this crash.

4.3 Multiple “Internal Error” Conditions Corrected

Several “Internal Error” conditions were reported using c-treeACE SQL. “Internal Errors” are reported when an inconsistency within the SQL engine is generated and further processing cannot be determined. Reported issues included the following:

- A query with multiple (>2) UNIONs in the WHERE clause failed with error **-20000** (SQL internal error). It caused a panic situation reported as follows:

```
- User# 00019 : PANIC - PSR blk::chk_tree 1 PID 3428
```

- An internal error (or panic) was seen when executing a query with a subquery containing both left and right joins.
- A SQL query sometimes failed with error **-20000** due to an unexpected situation while searching a dynamic index. The logic has been modified to correct this.
- A query failed with an internal error when no N[VAR]CHAR field was involved in the query. This was due to a problem with the internal technology to handle sorting and temporary tables performing string conversions that were failing. The problem was resolved.

These have all been reviewed and corrected.

4.4 Corrected Memory Leak When Error Occurs Creating Table with IDENTITY Field

When a table with a SQL IDENTITY field was created and the table creation failed, allocated memory for the identity field name in the FairCom DB API table structure was leaked. The logic has been modified to free the memory for the identity field name.

4.5 ALTER TABLE Now Checks REFERENCES Permissions When Adding Foreign Key Constraints

The following type of SQL statement succeeds, regardless of the users permissions on `your_table`:

```
ALTER TABLE mytable ADD CONSTRAINT fk_ref FOREIGN KEY (x) REFERENCES your_table(y)
```

ALTER TABLE was failing to check if the user had REFERENCES permission on `your_table`, which is required to create such a constraint. This logic has been corrected.

4.6 SYSDATE Default Field Values Now Allowed With ALTER TABLE

It was not possible to set a default field value to SYSDATE (and others) using alter table. For example, the following failed with a FairCom DB API invalid date error when the table contained records:

```
alter table mytbl add mycol date default SYSDATE;
```

The alter table logic considered SYSDATE to be a literal value for the field. FairCom DB API tried to set the new field value to the default using the SYSDATE literal, which was not a valid date.

Alter table logic has been modified to allow this.

4.7 Unix Client Shared Memory Connections No Longer Leak Memory

A c-treeACE SQL client process leaked a small amount of memory each time it connected. Memory allocated for a SQL client shared memory connection on a Unix system was not freed when the connection closed. Logic has been corrected to free these memory allocations at appropriate times.

4.8 AIX Clients No Longer Drop Inactive Shared Memory Connections

On AIX systems, a c-treeACE SQL client using shared memory communication dropped its connection after 10 seconds of inactivity. Logic has been updated to maintain the connection.

4.9 Avoid Connection Errors If First COMM_PROTOCOL Module Fails to Initialize

When the first `COMM_PROTOCOL` listed in ***ctsrvr.cfg*** failed to load (e.g., because the DLL was missing), c-treeACE SQL connections would fail with error code **-17046**. Logic has been modified to correct this error.

4.10 Correct Error Messages Now Returned by fc_create_user Procedure

Calling **fc_create_user** could have resulted in several unusual error messages, due to an overlap of the SA_ADMIN API and c-treeACE SQL error codes. For example, calling this procedure with an empty user name previously returned error code **-18008**, which was reported as message,

"CT - A partitioned file can only have one open instance at a time per connection." The SA_ADMIN return codes used by this function have been replaced with c-tree error codes. In the example case, the error code is now correctly returned as **-17450** with message "CT - Invalid user id."

4.11 Correct Return of LVARCHAR Data When Using CT_STRING Data Type

ADO.Net (and some other APIs) returns the LVARCHAR buffer with a length that is 1 byte larger than it should be when the underlying c-tree data type is *CT_STRING* (which happens only for imported tables). The extra byte is set to `'\0'`. If the underlying c-tree data type is *CT_4STRING* or *CT_2STRING* this extra byte is not present. Logic has been modified to correct this.

4.12 Correct VARCHAR Data Now Returned in Complex Queries

An issue was found in the 32-bit c-treeACE SQL V10.3 (or later) running complex queries in which a VARCHAR set to an empty string was retrieved as garbage instead of the proper content. LONG column handling has been modified to correct this.

4.13 Proper NULL Values Now Returned for LVARCHAR Columns When Sorting for ORDER BY

A query projecting an LVARCHAR field and having an Order By clause sometimes returned data instead of NULL for LVARCHAR values set to NULL.

4.14 LONG VARBINARY Support Added for ORDER BY Clauses

In V11 and later, a query projecting an LVARBINARY column using an ORDER BY clause that requires sorting in memory may result in projecting null values. Prior to this change, LVARCHAR was the only supported long type. The logic has been updated to handle LVARBINARY.

4.15 Query Speed Improved From Dynamic Index Usage

A query that ran fast on V9.5 was found to run dramatically slower in V10. Comparing the V9.5 execution plan with the V10 execution plan, it is clear that V10 tends to build a Dynamic Index (DIDX) instead of using an existing index to execute a join.

In V10 logic has been introduced to better optimize joins when the restriction imposed by join predicate can be executed using only a few columns of a multi-column index. In some cases it may be more convenient to create a DIDX and use it instead of doing partial key searches on the existing index. However, the logic was using a DIDX in cases where it was not preferred.

The logic has been refined to perform better checks before deciding to use a DIDX.

4.16 Corrected TRUNCATE Scalar Function Result

A query was returning the wrong numeric value. After running the following commands:

```
CREATE TABLE MYNUM(fld NUMERIC(10,3));
INSERT INTO MYNUM VALUES(-0.003);
INSERT INTO MYNUM VALUES(0.003);

SELECT TRUNCATE(fld,0) FROM MYNUM;
```

The following was incorrectly returned:

```
TRUNCATE(FLD,
-------------
:.0000000E+125
-:.000000E+125
2 records selected
```

Truncate logic has been updated to correct this error.

4.17 Corrected Rounding of Sub-Query Numeric Values

A SELECT statement with a sub-query returned rounded numeric values as if the scale was 0 instead of returning the proper value with the proper scale. The logic has been corrected to set the scale value appropriately.

4.18 Empty Column Names No Longer Returned When Column Length Is 64-characters

ISQL, GUI tools (Java and .NET versions), JDBC, and other APIs returned an empty column name for a column when its name was exactly 64-characters long. Logic has been modified to allow a proper return.

4.19 Case-Insensitive CONTAINS Now Properly Matches Uppercase Data

A query using the `CONTAINS` predicate with `ctoption(icontains)` did not return some rows that had case matches. The logic has been modified to handle this situation correctly.

4.20 MAX Scalar Function on Field With Descending Index No Longer Returns MIN Value

In certain situations, a query using `max()` would return the wrong result when an index existed on `(f2,f1)` and `f1` had the opposite ordering (ASC/DESC) of `f2`. A query similar to the example below would return the wrong result:

```
SELECT max(f1) from si142076 where f2 = '     3'
```

The logic has been corrected to fix this problem.

4.21 SUSER_NAME and USER_NAME Scalar Functions Improved

The **SUSER_NAME** scalar function could cause server instability and the **USER_NAME** scalar function incorrectly returned the same result as the **USER** function. These functions have been rewritten to fix these problems. They now both correctly return the name of the user who is currently logged in.

Notice that the **USER** scalar function returns the name of the user under which it is currently running, which may be different from the login user during stored procedure execution. The name will be either the login user or (if the function is called within a stored procedure) the owner of the stored procedure.

The following scalar functions were also reviewed:

- **UID**: This function is not supported and always returns **-20133**.
- **USER_ID**: This function is not supported and always returns **-20133**.
- **USER_NAME**: This function returns the name of the user who logged in to the SQL server. When called passing a user ID, as supported by the syntax, it returns **-20133**.
- **SUSER_NAME**: Identical to **USER_NAME**.

4.22 Corrected GRANT of Column Permissions for non-DBA Users with Table Grant Permissions

A GRANT of column-level UPDATE or REFERENCES permissions failed unexpectedly for a non-DBA user who had table-level grant permission. The logic has been modified so that this operation is allowed if table-level permissions are sufficient for GRANT.

4.23 REVOKE GRANT OPTION Now Correctly Removes Column Permissions

REVOKE GRANT OPTION did not remove column-level permissions. The following should remove the specified grant option from rootuser and remove all specified permissions that rootuser granted to others (child users), but still allow rootuser to access the table:

```
REVOKE GRANT OPTION FOR permission ON table FOR rootuser CASCADE
```

For table-level permissions this was done correctly. For column-level permissions, child users only had their grant option revoked rather than access permission.

The logic has been modified to correct this.

4.24 Corrected Index Creation for Selected Imported Tables

A situation was found in which CREATE INDEX failed with error **-21045** (CTDB Unknown index number) on tables imported with the **ctsqlimp** *-n* option. Logic has been corrected to account for this situation.

4.25 Optimizer Improvements for Field Type Constraints

New logic has been added to improve optimizer strategies for execution plan construction. This logic avoids unnecessary table scans when the scan condition is known to be false because of comparing a field with a value out of the range of possible values for that field. For example:

```
f1 < -200
```

when `f1` is a tinyint (valid values -128...+127).

4.26 Error -20134 No Longer Returned When Setting LONG Fields to NULL

SQLExecute failed with error **-20134** when setting a LONG field to NULL using the ODBC driver and the **SQLBindParameter()** function. The logic has been modified to correct this.

4.27 Error -20133 No Longer Improperly Returned When Setting UID in DEFAULT Clause

A CREATE TABLE or ALTER TABLE with DEFAULT UID succeeded, but the UID function was not supported. SQL logic has been corrected such that a CREATE or ALTER that tries to set DEFAULT UID on a column now returns **-20199**. The behavior of existing tables that already have a DEFAULT UID is unchanged.

4.28 Error -20142 No Longer Returned with UDF Execution After Table Import

An error **-20142** was seen when using a Java user-defined function (UDF) after importing a table. The logic has been modified to correct this.

4.29 Corrected DH_REBUILD_SEL_CUTOFF Handling

- Enabling `SETENV DH_REBUILD_SEL_CUTOFF` caused queries with more than 50 parameters to fail with **-20006** (Column not found).
- A syntax error was returned by a valid query when parameters had certain values and `DH_REBUILD_SEL_CUTOFF` was enabled. This error in the parsing of character parameters has been corrected.

Both of these conditions have been corrected.

5. Replication Agent Updates

Enhancements listed in this section have been made to the first member of the FairCom Advanced Module Series:

The FairCom Replication Agent

5.1 Improved Replication Agent Handling of Compressed Records

Replication Agent stopped with the following error when it attempted to decompress a record image for a compressed record file:

```
ctrepl: Failed to set old record image in operation list entry: 938
```

This and the following related compressed record handling issues have been corrected.

- Replication Agent terminates with error **26** when closing a file.
- Replication Agent terminates with the error "Failed to retrieve file information when processing compression resource: **101**".
- Replication failed with error **946** when a compressed record file was created and then a record was modified before closing and reopening the file. *CTSTATUS.FCS* showed the message:

```
Compression support has not been initialized for file: 946
```

5.2 Maintain Replication Agent Connection During Dynamic Dump of Target Server

The Replication Agent sometimes shut down with error **78** logged to *ctreplagent.log* when a dynamic dump was performed on the target server. The resulting transaction abort from the dynamic dump no longer halts replication.

5.3 Prevent Replication Agent Termination Related to Locked State Record on Target Server

If the Replication Agent finds its state record locked on the target server, it logged error **827** and kept running, but later the agent logged error **100** and sometimes terminated due to that error. The logic has been modified to correct this.

5.4 Improved Error Handling for Replication Agent HTRN_ERR (520)

When files are copied to a target replication server, index files contain transaction high-water marks that can conflict with new transaction numbering of incoming replicated transactions. It is very possible error **HTRN_ERR** (520, high transaction mark error) may be observed in the replication exception log when this occurs. It is likely the first transaction that replicated to this file fail with this error.

The Replication Agent will now attempt to handle **HTRN_ERR** errors by aborting and retrying the transaction. Retries are not attempted if the Replication Agent is using the following option:

`exception_mode operation`.

Note: With this change, it is no longer necessary to run a **CLNIDXX** operation (using the **ctclnidxx** utility, the `!CLNIDXX` dynamic dump restore option, or the `AUTO_CLNIDXX` *ctsrvr.cfg* option) prior to accessing the target server's copy of the file.

5.5 Avoid Replication Agent Halt if Target Connection is Lost During Checkpoint Processing

The Replication Agent shut down if it lost its connection to the target server when processing a checkpoint entry. The following message was logged to *ctreplagent.log*:

```
Failed to update replication agent state on data target: 127
```

When processing a checkpoint entry that it read from the source server, the Replication Agent attempted to update its current position on the target server. Because the target server had shut down, the Replication Agent terminated. The Replication Agent has been modified to detect the loss of connection and reconnect, which corrects this issue.

5.6 Support Partial Record Rewrite in Local/Master Synchronous Replication

Local/master Synchronous replication is used when one server is a designated master server with multiple local servers. In this model, the master server is always updated along with the local server where the update originated. This is done in a distributed all or nothing transaction. Other local servers, not participating in the transaction, then receive the update via replication from the Replication Agent.

When replication was using the local/master model, some partial variable-length record updates were not applied to the master copy of the data. The logic has been modified to prevent this.

5.7 Replication Agent Stability Improvements

A couple error situations were reported where Replication Agent would unexpectedly terminate with one of the following messages to *ctreplagent.log*:

```
ctrepl: Failed to retrieve item from operation list: 101

ERR: Failed to retrieve next change: 2
```

The source FairCom Server status log (*CTSTATUS.FCS*) may have also contained the following message:

```
ctrepl: Failed to add entry to node list when processing LLOGOFF entry: 2
```

Conditions leading to these errors are now appropriately handled allowing replication to continue.

5.8 Avoid ICUR_ERR When Using Replication Agent Administrator

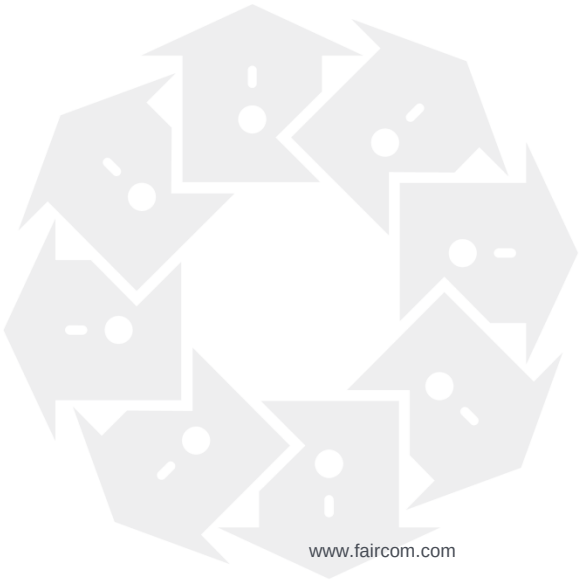
A situation was encountered when running the Replication Agent Administrator utility with the *getlog* switch (**repadm** *-getlog*). An operation failed with error 100 (**ICUR_ERR**). The logic has been corrected to fix this.

5.9 Corrected ctReplSetPosByTime Error 76

Error 76 (**TBAD_ERR**) was seen when directly calling the Replication API funciton **ctReplSetPosByTime()**. This was encountered when unrelated transaction logs existed on disk with log numbers that sequentially followed the active log. The logic has been modified to open the correct log in this situation and to stop scanning with the active log, thereby effectively ignoring the unrelated logs.

6. Utility Improvements

c-treeACE ships with a full complement of utilities to aid in development, administration, and system management. The changes to those utilities are listed in this section.



6.1 “Change DIAGNOSTICS” Menu Option Restored for ctadmn Administrator Utility

The **ctadmn** menu option **Change Server Settings** did not display one of the documented options: **Change a DIAGNOSTICS Option**. The utility has been modified to correct this.

6.2 Flush ctstat Statistics Utility Output Before Pausing

The **ctstat** utility has been enhanced to better handle cases in which an administrator redirects its output to a file. The utility now flushes standard output after each wait interval (determined by the *-i* setting, which defaults to 1 second), so that the **ctstat** output is written to the file immediately.

6.3 ctstat -filelocks and -userlocks Options Now Show Lock Information When a Large Number of Locks Are Held

When **ctstat** is run with the *-filelocks* or *-userlocks* option, **ctstat** does not display the locks for the file if a large number of locks are held on the file. The logic has been modified to correct this.

6.4 ctinfo Now Reports Mismatch Between Data and Index Filemode When Data File Has No Indexes

When **ctinfo** was run on a data file that uses transaction control and the data file does not have any associated indexes, **ctinfo** prints the following warning message:

```
WARNING MISMATCH IN TRAN/NOTRAN FILE MODES!
Data and index should both have tran if one does.
```

This message does not apply when a data file has no indexes. The logic has been modified to suppress the warning message in that situation.

6.5 Compact and Rebuild Utilities Now Correctly Update IFIL Resource When -updifil Option is Specified

An issue was discovered when **ctrbldif** was run with the *-updifil* option and the data file name was specified with a directory name (for example, **ctrbldif** */datadir/test.dat -updifil*). If data and index file names in the IFIL had the same path as each other, the utility displayed a message

saying that the index name was being changed but the IFIL resource was not updated. The logic has been modified to correct this.

6.6 cttrnmod No Longer Terminates with Unhandled Exception When Data File Uses Extension Other Than .dat

The **cttrnmod** utility terminated with an unhandled exception when the specified data file used an extension other than *.dat*. The logic has been modified to correct this behavior.

6.7 ctquiet Utility Now Accepts Passwords Over 16 Bytes

A connection failed with error **451** because **ctquiet** was hard-coded to limit passwords to 16 bytes (the limit before V10). The logic has been updated to conform to our current password limits (currently 64 bytes).

6.8 ctsqlcdb Now Returns Non-zero on Error

The **ctsqlcdb** utility incorrectly returned zero in some error cases. This utility has been changed so that it now always returns 2 if an error occurs.

6.9 ctsqlutl Column Rename Improved

After using **ctsqlutl** to rename a column, a query on the table may result in odd behaviors if the table was already queried before the column rename. The logic has been modified to correct this.

6.10 ctcv67 Now Includes Support for Partial Key Distinct Counts

The **ctcv67** utility failed with error **IAIX_ERR** (608) when converting a non-huge file to huge. The logic has been modified to correct this error.

6.11 ctstap - Single-Threaded Version of Multi-Threaded Test

In V11 and later, a new test program has been added: **ctstap**, which is a single-threaded version of **ctmtap**. This test program is useful for checking that concurrent add, delete, and update operations on a file by multiple *FPUTFGET* processes work properly.

Description

ctstap is FairCom's Singlethreaded API sample program. This sample program is an IFIL version of ctmark. As in ctmark, the main routine runs a trial of database operations.

Supported options:

- *a<action>*, where <action> is one of the following values:
  - *A* - add data
  - *D* - delete data retrieved in random order
  - *G* - retrieve data in random order (this is the default action)
  - *S* - call STPUSR() and terminate immediately
  - *T* - run cttestfunc() routine
  - *U* - update data retrieved in random order
- *e<cipher>* - Encrypt the files using the specified cipher, where <cipher> is the numeric value of one of the cipher types listed in ctport.h. For example, for AES32 encryption specify e62. Use with the fc option.
- *fc* - Create the data and index files. If not specified, open existing files.
- *g<transaction aborts>* - Approximate number of transactions to be aborted. For example, a value of 2 implies that about 20% of the transactions will be aborted. Defaults to 1 (10% aborted transactions).
- *h<filename>* - Use the specified file name (without extension).
- *ic* - Create files that use record compression. Use with the fc option.
- *ih* - Create files that use 8-byte offsets. Use with the fc option.
- *in* - Create files without extended header. Use with the fc option.
- *is* - Create segmented files. Use with the fc option.
- *iv* - Create variable-length record files. Use with the fc option.
- *L* - Use a local connection, otherwise use a client connection. This option is supported only in the LOCLIB model.
- *mc* - Create files with offset compression in index nodes. Use with the fc option.
- *md<data cache_size in MB>* - Set the data cache size in megabytes.
- *mi<index cache size in MB>* - Set the index cache size in megabytes.
- *mm* - Create files with optimized transaction marks. Use with the fc option.
- *mN* - Create files with no transaction processing. Use with the fc option.
- *mt* - Create files with transaction processing without logging. Use with the fc option.
- *mT* - Create files with transaction processing with logging. Use with the fc option.
- *mta* - Create files with auto switch to ctPREIMG mode. Use with the fc option.
- *mTa* - Create files with auto switch to ctTRNLOG mode. Use with the fc option.
- *mxa* - Create files with variable length index node format. Use with the fc option.
- *n<number of iterations>* - Number of iterations of the specified action per trial. Defaults to 100 if not specified.
- *od<data cache size in bytes>* - Set the data cache size in bytes.
- *oi<index cache size in bytes>* - Set the index cache size in bytes.
- *op<page size in bytes>* - Set the page size in bytes.

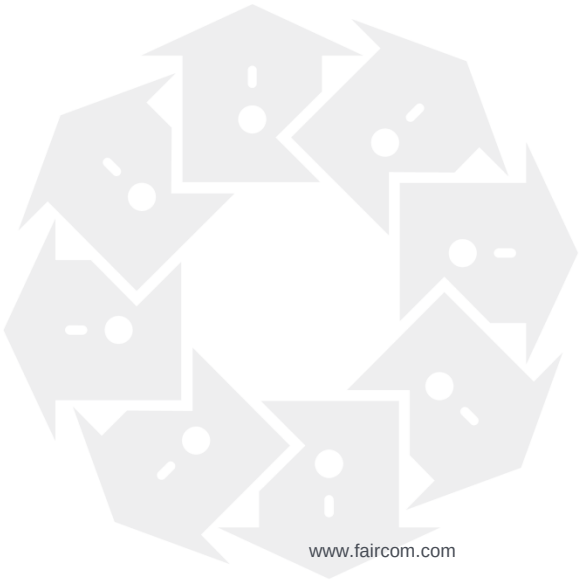
- *ox* - Open the files in exclusive mode.
- *p<password>* - Use the specified user password.
- *q* - Quiet mode - don't print progress messages.
- *r<seed>* - Random number seed (0-65535); defaults to 1
- *s<server name>* - Name of FairComDB Server. Defaults to FAIRCOMS.
- *u<user name>* - Use the specified user name.

7. FairCom Typographical Conventions

Before you begin using this guide, be sure to review the relevant terms and typographical conventions used in the documentation.

The following formatted items identify special information.

Formatting convention	Type of Information
Bold	Used to emphasize a point or for variable expressions such as parameters
CAPITALS	Names of keys on the keyboard. For example, SHIFT, CTRL, or ALT+F4
FairCom Terminology	FairCom technology term
FunctionName()	c-treeACE Function name
Parameter	c-treeACE Function Parameter
`Code Example`	Code example or Command line usage
utility	c-treeACE executable or utility
filename	c-treeACE file or path name
`CONFIGURATION KEYWORD`	c-treeACE Configuration Keyword
CTREE_ERR	c-treeACE Error Code



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

8. Index

D

E

F

G

I

J