test link

Update Guide

c-treeACE V10.3 Update Guide

Audience

Developers

Subject

FairCom's high-performance NAV and SQL database technology.

FairCom®

Contents

1. Introduction

This document describes the new features in c-treeACE V10.3

The FairCom team is pleased to present the V10.3 edition of the c-treeACE database technology. This release benefits from the extensive feedback from customers whose implementations you encounter every day. Whether you are using your credit card, shipping a package, picking up a lottery chance, or buying tickets for your favorite concert, you are supported by the power of c-treeACE—we work closely with the vendors of these commercial applications around the world.

We would like to thank our customers for their support. We owe you the credit for pushing us with your real-world demands.

Precision engineering is what we do. FairCom's exacting engineering results in a precision product for your customers. To achieve this, FairCom provides a unique model not available from traditional database vendors. We start with proven high-quality database technology and the people who write and know this code. Then we add the ability for customers to engage in a close technical relationship with our experienced engineering team. The result is a precision engineered solution for your exacting needs.

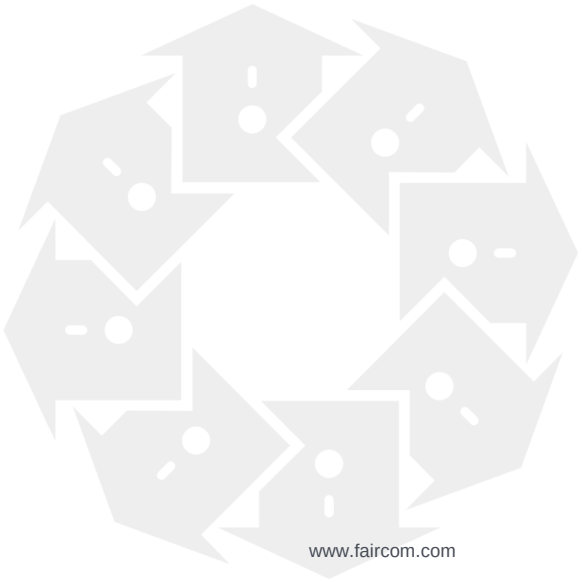
This new release is packed with innovations, improvements, and enhancements.

So, whether you are a new c-treeACE user or a veteran code warrior, we encourage you to take a close look at our latest efforts described in the pages that follow and “Scale Up” to the best work we have yet offered: c-treeACE V10.3.

We thank you for your business.

Sincerely,

The FairCom Team



2. Highlights of c-treeACE 10.3

c-treeACE V10.3 has a strong focus on engineering-level enhancements and continues to broaden our support for developers who choose our platform. FairCom is proud to make available our latest productivity-enhancing features.

2.1 Documentation Updates

Corrections to the printed V10.3 Update Guide:

GetRecordKeyPos() returns pCTOFFSET

The **CTRecord::GetRecordKeyPos()** C++ function (page 44) returns a data type of pCTOFFSET. The printed manual incorrectly shows a data type of LONG.

The **ctdbGetRecordKeyPos()** function (page 42) shows a data type of pULONG8, which, is equivalent to pCTOFFSET in this instance.

ctsqlExecuteBatch() function

The **ctsqlExecute** function is erroneously referred to as **ctsqlExecuteBatch** in the printed version of the documentation. To execute a batch, use the **ctsqlPrepare** and **ctsqlExecute** functions, as described in the *c-treeACE Direct SQL Interface User's Guide*.

Staying up-to-date...

At FairCom, we are constantly working with our customers to bring you the finest product possible. In this rapidly evolving environment, the best place to find up-to-date information, including the latest documentation, is on the FairCom website.

2.2 Performance: Faster than Ever

FairCom continues to make performance improvements to put your application ahead of the competition. The gains are especially apparent when scaling across a large number of concurrent users on multi-CPU systems.

In addition, the V10.3 update to c-treeACE is providing enhancements to the c-treeACE SQL internal temporary memory allocation routines which will improve performance and enhance stability with complex and large queries.

See the section titled **Performance Gains** (page 7) for more information.

2.3 Unix Shared Memory Enhancements

c-treeACE V10 introduced shared memory support on Unix systems to greatly improve the performance of localhost applications. Further improvements have been made to this functionality in V10.3.

See the section titled **Improved and Expanded Shared Memory Support** (page 15) for more information.

2.4 Extended Data Compression Options

V10 introduced compression support; this release makes further advances to help you deal with today's large data files.

- A high-performance Run Length Encoding (RLE) algorithm is now available which outperforms zlib compression in many cases.
- User-defined compression is available using a library you provide. With this feature, FairCom may be able to help you solve special requirements.
- Compressed data files can now be compacted.

See the section titled **Improved Compression Support** (page 13) for more information.

2.5 New Batch Update Functions

c-treeACE provides batch operations that have the potential to greatly improve performance when a group of related records needs to be processed. This release includes improvements to this feature that can result in greater efficiency.

See the section titled **Batch updates** (page 17) for more information.

2.6 c-treeACE SQL Changes

Enhancements to the SQL interface include the improvements listed below to name just a few:

- Removed SQL Statement Length Limits - V10.3 allows statement lengths up to 32MB for your most complex SQL statements.
- c-treeACE SQL system tables are now encrypted by default to protect database meta data.

See the section titled **c-treeACE SQL** (page 34) for more information.

2.7 Enhanced Deployment Flexibility with Environment Variable Support

The list of available c-treeACE options that can be configured through environment variables has been expanded.

See the section titled **Configuration Flexibility with Environment Variable Support** (page 22) for more information.

2.8 Greater Connection Control Flexibility

c-treeACE now supports additional management for concurrent user accounts:

- Limit ISAM vs. SQL Connections
- Configurable c-treeACE SQL Server connection timeout

See the section titled **Greater User Connection Control** (page 30) for more information.

2.9 Latest Platform Support

Continuing support for the latest platforms is included in this release. These include:

- IBM AIX 7.1
- Embarcadero RadStudio XE3 and XE4
- Java 1.7 support

2.10 Cross-Platform Java Tools

This release of c-treeACE includes graphical tools written in Java for cross-platform support. This support allows system administrators to leverage these powerful tools on any platform that supports c-treeACE and Java. Some of the tools have now been combined to simplify their operation. The following tools are available:

- **c-treeACE Explorer** - Allows you to view and edit both SQL and ISAM databases in the same tool
- **c-treeACE Monitor** - Gauges and statistics for visually monitoring your c-tree databases
- **Error Viewer** - An updated version of the error viewer provides look-up and custom editing of error codes

See the section titled **Introducing Cross-Platform Java Tools** (page 52) for more information.

3. Performance Gains

FairCom continues to make performance improvements designed to put your application ahead of the competition. The gains are especially apparent when scaling across concurrent users on multi-CPU systems.

3.1 Improved scalability with distributed data and index cache counters

A read scalability test indicated significant overhead associated with handling the data cache hit and request counters as the number of threads increased. To reduce this overhead, the logic to collect the following statistical information was changed by implementing a pool of counters per each value, thereby reducing contention:

- data cache requests
- data cache hits
- index cache requests
- index cache hits

By default, each of these counters is stored in four separate memory locations. The number of memory locations can be changed by specifying `CACHE_STAT_ARRAY_SIZE <N>` in *ctsrvr.cfg*, where `N` is the number of memory locations for each counter.

The functions that read these statistics, **IOPERFORMANCE()** and **ctSNAPSHOTsysdat()**, have been changed to call functions that sum the instances of the counters and return the sum instead of reading a single counter value from memory.

When the c-treeACE Server starts up, it writes a message to *CTSTATUS.FCS* indicating how many cache stat array entries are in use:

```
Data and index cache statistics array size: 16
```

3.2 Add Unique Key First file mode

In certain test cases, a large percentage of variable-length add operations failed because of duplicate keys. To account for these special cases, a **PUTHDR()** mode, *ctADDUNQFRShdr*, has been added that can enable a new way of managing variable-length record add operations. This allows an add record call to add unique keys before allocating space for the data record from the data file. If a unique key add fails, the **ADDVREC()** operation fails immediately. If it succeeds, the **ADDVREC()** operation continues.

In the test cases where a large percentage of **ADDVREC()** operations are expected to fail because of duplicate keys, up to a 35% performance gain has been observed.

In regular cases where the **ADDVREC()** seldom fails because of duplicate keys, enabling this feature may result in a small loss of performance.

3.3 Skip the backup of Non-Transaction and Pre-image Files

A new configuration option improves backup performance by skipping non-transaction files and pre-image files during a dynamic dump:

```
PERMIT_NONTRAN_DUMP < YES | NO >
```

Default: YES

When set to NO, non-transaction files and pre-image files are skipped during a dynamic dump even if they are included in the !FILES section of the dynamic dump configuration script. The exception is if PREIMAGE_DUMP is set to YES, then pre-image files continue to be included in the dump.

3.4 Improvements under the hood

The remaining improvements listed in this chapter represent changes made to the internal logic that benefit you without making changes to your application or configuration.

c-treeACE SQL large query improvements

The V10.3 update to c-treeACE includes a complete overhaul of the c-treeACE SQL internal temporary memory allocation routines which will improve performance and enhance stability with complex and large queries. This major enhancement to our core product is one more reason developers can be confident in the c-treeACE engine.

Reduced index node contention

During extensive testing with a customer in the financial sector we identified contention with the internal index node logic that only manifested under heavy load from multiple threads all accessing the same index node. We have improved this critical section of the c-tree index-handling logic and are pleased to report a 23% improvement in performance with our internal testing.

Improved c-treeACE SQL ADO.NET Provider fetch performance

The c-treeACE SQL ADO.NET Provider was optimized for much faster record retrieval.

An increased fetch buffer size for the latest Windows systems reduced retrieval times for 190,000 records from approximately 60 seconds to 7 seconds.

c-treeACE SQL on big-endian systems considers index for backward scan

On AIX systems (or any big-endian system) c-treeACE SQL was previously not considering an index as an option for a backward scan. The logic has been modified to correct this situation. This change can provide performance gains for selected queries where a backward scan will prevent a full table scan from being performed.

Field Callbacks improve c-treeACE SQL Types SDK performance

The c-treeACE SQL Types SDK allows developers to access non-standard data types via SQL. For example, many existing date types don't follow c-tree standard date conventions. The c-treeACE Types SDK gives developers a method to convert these dates on the fly for performing SQL inserts and queries.

The SDK has been extended with this release to allow conversion to occur at field access time instead of (or in addition to) record read/write time. This new capability allows performing conversions only when data is actually accessed, avoiding it when not strictly necessary. The

performance benefit of this approach is very evident in those cases where a record is read but the portion of it that requires conversion to occur is not accessed.

Improved performance of physical order batch reads with no locking

The scalability of concurrent physical order batch-read operations on a file was improved by modifying the batch retrieval logic to perform less memory allocation. On concurrent physical order batch retrieval, this resulted in significant performance improvement. Depending on the number of concurrent batch operations, we witnessed performance gains from 2 to 7 times faster.

Improved performance at Transaction Commit

Changes to the loop that checks each of the user's files for a deferred close or delete at the end of a transaction have yielded performance improvements.

Reduced contention of synchronization objects with Slim Reader/Writer Locks

As part of the effort to improve performance based on reducing contention, some synchronization objects used to control concurrency on items that are often read and seldom written were changed from mutex to reader/writer locks.

Key buffer optimization for non-partition key search when Partition File support is in use

Internal c-treeACE buffers are maintained to optimize indexes that do not match the partition index. For a partitioned file, calling **FRSRNG**/**NXTRNG** with a range that matches all key values on a key that has no relationship to the partition key was faster than using **FRSREC**/**NXTREC** on that same index. The range functions are taking advantage of the optimization that maintains key buffers for indexes that do not match the partition index. This optimization has now been enabled for NXTREC operations on a non-partition index.

Optimized FairCom DB API field structure information retrieval performance

The inner code of the following functions has been modified to avoid unnecessary c-tree function calls, obtaining a 30% performance enhancement for these functions:

- **ctdbGetFieldSize**
- **ctdbGetFieldOffset**
- **ctdbGetFieldAddress**
- **ctdbGetFieldDataLength**

ISAM performance enhancement

Performance for ISAM clients has been improved by avoiding unnecessary calls to the server to request an alternate collating sequence for indexes that don't have one.

Improved c-treeACE SQL numeric read performance

Extensive profiling of the server revealed locations where performance could be improved when reading numeric fields and these have been modified accordingly.

Improved c-treeACE SQL performance for queries containing both Group By and Order By clauses

The c-treeACE SQL optimizer is now able to better identify the existence of an index that can be used to perform the Group By and the sort operation at the same time avoiding the overhead of performing the two operations in separate steps.

c-treeACE SQL optimization for non-contiguous ranges

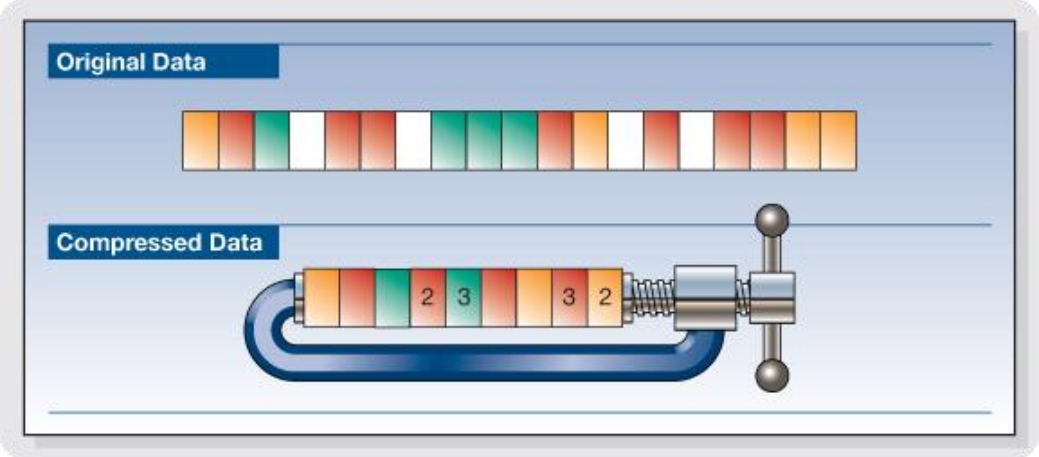
The c-tree index range logic, used by c-treeACE SQL to perform index scans, has been improved to better identify the “end of scan” condition and avoid scanning further keys when it is certain that no more keys in the index can match the search criteria.

4. New and Improved

New and enhanced features improve overall functionality of c-treeACE. Advances include Compression, Shared Memory, Batch Updates, and many others listed in this section.

4.1 Compression

c-treeACE V10 introduced compression support. This greatly reduces disk space requirements for today's extremely large files.



Remember, c-treeACE supports files up to 16 exabytes (18 million terabytes)! When data is sparsely populated, compression can substantially reduce your space needs.

User-Defined Compression

In addition to Run Length Encoding (RLE) and zlib compression, the c-treeACE data record compression feature supports user-definable compression modules. The **ctSETCOMPRESS** function will accept a compression type of *ctCMPRECuser*, which must be accompanied by the name of a DLL that performs the custom compression. Discuss your needs with a FairCom engineer if you need help with special requirements.

For more information, see **ctSetCompress** (https://docs.faircom.com/doc**/ctreeplus/ctsetcompress.htm**) in the *c-tree Plus Programmer's Reference Guide*.

Run Length Encoding (RLE) compression option

A new Run Length Encoding (RLE) compression option has been provided. Configure the RLE compression algorithm in *ctsrvr.cfg*, similar to the keywords for ZLIB or USER supplied compression.

The `CMPREC_TYPE` keyword now supports these options:

```
CMPREC_TYPE  <"ZLIB" | "USER" | "RLE">
```

To enable data compression routines:

```
CMPREC_TYPE      <"ZLIB" | "USER" | "RLE">
CMPREC_VERSION   <a number >= 1>
CMPREC_DLL       <name of DLL>
```

These keywords should be entered in the configuration file in the order shown. A DLL name is required for a `CMPREC_TYPE` of `"USER"`.

Dynamically load zlib

In v10.3.0 and later, c-tree dynamically loads zlib instead of statically linking with it.

If the zlib library cannot be loaded and compression is attempted to be used, c-tree returns error **CMPR_ERR**, sets sysiocod to 5 (indicating a DLL load error), and the first time that c-tree returns this error it logs the following message to *CTSTATUS.FCS*:

```
zlib compression support is not available: <error message>
```

Compressing/un-compressing existing data

It is now possible to enable/disable compression on existing data files programmatically by using the **CMPIFIL()** function or through the use of the **ctcmpcif** utility.

To enable or disable compression when calling **CMPIFIL()**, set the **chgcompressIFILoption** bit in the **tfilno** member of the IFIL structure whose address you pass to **CMPIFIL()**. When this bit is set, **CMPIFIL()** switches the compression state of the file: compressed files become un-compressed, un-compressed files become compressed.

To enable or disable compression when using the **ctcmpcif** utility, command-line options are provided:

- *-compress* - Create the compacted data file with compression
- *-nocompress* - Create the compacted data file without compression

4.2 Shared Memory

c-treeACE V10 introduced Shared Memory support on Unix systems, which greatly improves the performance of localhost applications—in many cases, a 4-8 fold increase in performance vs. TCP/IP. V10.3 makes further improvements to Unix Shared Memory connections.

Shared Memory protocol for SQL connections on AIX, Linux, and Solaris

The Shared Memory protocol is now supported for SQL connections on AIX, Linux, and Solaris (it was already supported for ISAM connections on Unix and both ISAM and SQL connections on Windows).

Shared Memory for c-treeACE SQL ADO.NET provider connections

The Shared Memory communication protocol has been extended to the c-treeACE SQL ADO.NET provider when connecting to a c-treeACE SQL Server on the same machine. This can improve performance 4-8 fold over the default TCP/IP connections. To take advantage of the Shared Memory protocol the *ctsqlshm32.dll* (32-bit) or *ctsqlshm64.dll* (64-bit) must be present. When available, Shared Memory is the default protocol for local connections.

Shared Memory for JDBC

The Shared Memory communication protocol has been implemented for JDBC. A new shared library named *ctsqlshm32* (for 32-bit) or *ctsqlshm64* (for 64-bit) was created to allow the JDBC driver to use the Shared Memory protocol for local connections. This library is native for the platform the JDBC driver runs on; hence different platforms require different “flavors” of the library.

Faster client detection of lost Unix Shared Memory connections

Previously, when a Unix Shared Memory connection was lost and the server did not remove it, the client waited 5 seconds before timing out and checking the state of the connection.

This improvement makes the first client timeout a timed wait of 50 milliseconds, and successive waits (for a long c-tree call such as a rebuild) time out in 5 seconds. Since most c-tree calls will return in under 1 millisecond, a single 50 millisecond timeout will not normally add overhead and

gives the feel of an immediate return in the event a connection has been lost. The Shared Memory header is now checked and avoids a timeout if the server disconnected cleanly.

Improved Shared Memory connect performance on AIX

When many clients connect to a c-treeACE Server at the same time using Shared Memory on AIX, overhead in the system calls caused the connections to be establish more slowly than when using TCP/IP.

The logic has been changed to use a Unix domain (file system) socket for transferring data between the client and server when using shared memory on AIX. The Unix domain socket exists as a file named `/tmp/ctreedbs/<servername>.logon`.

Additional information logged to CTSTATUS.FCS for failed Shared Memory connection attempts on Windows

c-treeACE has been enhanced to log a more descriptive error message to *CTSTATUS.FCS* when a Shared Memory connection attempt fails due to Windows denying the c-treeACE Server process access to the client process. Previously, the message "Connect named pipe failure: -1" was logged to *CTSTATUS.FCS* on Windows.

The logic has been improved to detect when a Shared Memory connection attempt fails because the system denied access to the client process and will log **error 978**.

4.3 Batch Updates

c-treeACE batch operations can greatly improve performance by reducing the number of round-trips to the server when a group of related records must be processed. V10.3 adds further capabilities to this valuable capability.

Previously, batch operations provided capabilities to either Retrieve or Delete records matching a partial key. We have now added the capability to Update a collection of records, each identified by its unique key value. This powerful feature adds a new BAT_UPD option for the mode parameter specified when calling the batch function.

The *Batch Operations* section of the *c-tree Programmer's Reference Guide* provides detailed information about these operations.

BAT_UPD and BAT_UPD_KEY

By default, BAT_UPD extracts the unique key from the record image, reads the record and then uses the record image provided in the input buffer to perform an Update operation. If the extracted key does not exist, a record ADD operation is performed. A mode called BAT_UPD_KEY is provided in which the old key can be specified for situations in which a rewrite might change the unique key used to locate the record.

Overview of Operations Performed

For each record in the input buffer, the batch update operation will do the following:

1. Either begin a transaction or establish a save point if the user already has an active transaction.
2. Build target key on the unique key.
3. Read the record with the requested lock:
   - If the record exists, update it.
   - If the record does not exist, add it.
4. In case of success or failure: Commit or abort the transaction or clear or restore the save point.
5. Release record locks if requested.

Notice that a missing key implies that a new record will be added.

A unique key is required on the file. If the index allows duplicates, the operation will return an error, so it is important to specify which unique index to use.

The function will return the number of records that were added and the number of records that were updated.

Update Modes

This operation provides two update modes:

- BAT_UPD (default) - The batch update call sends a series of record images and the *filno* parameter corresponds to a unique key index.
- BAT_UPD_KEY - The record images are preceded by the key value that uniquely identifies the record to be rewritten or added.

In both cases, variable-length data files have the 4-byte length of the data record as the first field of each entry.

In the default BAT_UPD mode, entries are formatted as shown below:

- For fixed-length data file: `<record image 1> <record image 2> ...`
- For variable-length data file: `<reclen 1> <record image 1> <reclen 2> <record image 2> ...`

When BAT_UPD_KEY mode is used, entries are formatted as:

- For fixed-length data file: `<old key 1> <record image 1> <old key 2>  <record image 2> ...`
- For variable-length data file: `<reclen 1> <old key 1> <record image 1> <reclen 2> <old key 2> <record image 2> ...`

The BAT_UPD_KEY mode should be used when a rewrite might change the unique key used to locate the record. The old key permits the record to be located and read before the rewrite operation.

Batch Updates and BAT_RET_BLK Format

Beginning with c-treeACE V11, batch update now 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.

Additional Modes

Additional modes can be used with BAT_UPD and BAT_UPD_KEY to provide record locking. See **Batch Update operations: BAT_UPD and BAT_UPD_KEY** (page 19).

New batch update mode BAT_UPD_KEPSRL preserves serial number in existing record

A batch update call that includes the BAT_UPD_KEPSRL mode causes the batch update to preserve the serial number in records that are updated by the batch update. Without this option, the serial number value that is supplied in the record image passed to the batch update is used.

This option is useful if a batch update is being done without first reading a record that is to be updated, and so the serial number value is not known.

Batch Update Operations: BAT_UPD and BAT_UPD_KEY

The batch update call to **DoBatchXtd()** (short name **BATSETX**) is formatted as follows:

```
BATSETX(keyno,request,NULL,bufsiz,mode)
```

Where the parameters are:

- *keyno* - Index file number of a unique index associated with the target data file
- *request* –
  - Input buffer comprised of 5 LONG integers followed by the record entries as described above. On input, *((pLONG)request)[2]* holds the number of entries.
  - Output buffer comprised of 5 LONG integers:

    0) successful rewrites

    1) successful adds

    2) reserved for future use

    3) in case of error, offset into the input buffer for the entry causing the error (zero offset is just after the 5 LONG integers)

    4) a 32-bit status word that indicates what operation failed (see **BAT_STT_XXXX** in *ctport.h*):

Symbolic name	Value	Description
BAT_STT_UPDATE	0x00000001	error on rewrite
BAT_STT_ADD	0x00000002	error on insertion
BAT_STT_DELETE	0x00000004	error on delete
BAT_STT_REDIREC	0x00000008	error on reading old record
BAT_STT_EQLKEY	0x00000010	error on equal key
BAT_STT_BUFSIZ	0x00000020	record image area too small
BAT_STT_FRMKEY	0x00000040	error assembling or transforming unique key value
BAT_STT_GTVLEN	0x00000080	error on get var rec length
BAT_STT_GETMEM	0x00000100	error allocating record buf
BAT_STT_CONVERT	0x00000200	error converting rec image

- *bufsiz* - Size of the input buffer region that follows the 5 LONG integers
- *mode* - BAT_UPD or (BAT_UPD | BAT_UPD_KEY).

Error Handling

BATSETX() returns an error code or **NO_ERROR**. This operation will stop if an error occurs. In case of an error, the number of successful Updates and ADDs are returned in elements 0 and 1 of the request output buffer as described above.

Because the operation stops on the first record that fails, adding the number of Updates and number of ADDs gives you the record number that failed.

Transaction Control - If an error occurs when the files are transaction controlled, none of the successful updates will be committed since BAT_UPD starts a transaction if one is not active or a save point if a transaction is active. In the event of an error, it either aborts the transaction or

restores to the save point. For files that are not transaction controlled, the successful updates survive the error.

Status Word Zero - If an error code is returned and the status word is zero, the error occurred during the setup of the BAT_UPD and no attempts were made to update any records.

Limitations

See *Heterogeneous support for BAT_INS and BAT_UPD* (page 21).

Record Locking - BAT_LOK_BLK, BAT_LOK_KEEP and BAT_LOK_ONE

Record locking for BAT_UPD (as well as BAT_INS) is different from batch calls used to retrieve records. The retrieval batches use batch-specific protocols to manage the locks. BAT_UPD (and BAT_INS when records are added one at a time) use standard ISAM API calls to update the files; the locking behaves just as if the client made the ISAM API calls.

BAT_UPD includes support for these modifiers to affect record locking:

- BAT_LOK_BLK: Acquire blocking lock when reading the record.
- BAT_LOK_ONE: Release lock immediately after add/update.
- BAT_LOK_KEEP: Keep locks after batch ends.
- BAT_LOK_WRT: Acquire write lock when reading the record

If BAT_LOK_BLK and BAT_LOK_WRT are included with BAT_UPD, the ISAM lock state is changed to *ctENABLE_BLK*.

- If either a transaction or **LKISAM** is called before **BATSETX** is invoked, the **LKISAM** state will be changed to *ctENABLE_BLK* only if the existing ISAM lock state is *ctENABLE* at the time **BATSETX** is invoked (typically this will be the case).

If BAT_LOK_BLK is included with BAT_UPD, this will cause the lock requests for the internal read operations (that find the existing versions of the records to be updated) to sleep if the record is locked by another user. This will cause the batch update to sleep until such locks are freed.

If BAT_LOK_KEEP is included with BAT_UPD, the data records updated or added to the file by BATSETX will remain locked on return from the **BATSETX** call. If either a transaction or **LKISAM** is called before **BATSETX** is invoked, the locks will remain after **BATSETX** returns whether or not BAT_LOK_KEEP is used. If neither was invoked before BATSETX, then BAT_LOK_KEEP will also cause the ISAM lock state set in **BATSETX** to persist after **BATSETX** returns.

If BAT_LOK_ONE is included with BAT_UPD and the file to be updated is not transaction controlled, then the record is unlocked after each record is updated or added; otherwise the records are unlocked at the end of **BATSETX** unless BAT_LOK_KEEP is in effect.

- If the file is transaction controlled, BAT_LOK_ONE is ignored.

Heterogeneous Support for BAT_INS and BAT_UPD

BAT_INS and BAT_UPD support heterogeneous client/server environments in which the record images formed by the client do not conform to the record images stored on disk. In addition, this modification detects alignment discrepancies between the client's record images and the alignment on disk.

The following are important details related to this new capability in heterogeneous client/server situations:

- BAT_INP_RAW, USERPRF_NDATA and USERPRF_NTKEY change the batch conversion process:

  If BAT_INP_RAW is included in the BATSETX mode, then the collection of records, key values (BAT_UPD_KEY only) and control information such as variable record lengths are *not* converted.

  If BAT_INP_RAW is not turned on, then if USERPRF_NDATA is turned on, the record images are *not* converted, but the record lengths (variable-length data files only) *will be* converted.

  If BAT_INP_RAW is not turned on, then if USERPRF_NTKEY is turned on and if BAT_UPD_KEY is included in the BATSETX mode, the key will *not* be transformed before it is used to find an existing data record.
- If the conversions have not been turned off as noted above, then the server performs the necessary conversions. If the server predates this modification, and conversions are required, the BATSETX fails on the client-side (i.e., no call is made to the server) with **SCNV_ERR** (994).
- If an alignment discrepancy is found, BATSETX fails with **ALGN_ERR** (992).

4.4 Configuration

New configuration features make c-treeACE easier to roll out than ever.

Keyword defaults based on available CPUs and CPU license limit

Previously, the c-treeACE Server's `CACHE_STAT_ARRAY_SIZE`, `DATA_LRU_LISTS`, and `INDEX_LRU_LISTS` options defaulted to 4 and `MEMORY_HASH` defaulted to 1. Now these defaults are based on the number of available CPUs on the system and the CPU limit specified in the license file.

SYSCFG() now returns the values for these settings in the cfgCACHE_STAT_ARRAY_SIZE, cfgDATA_LRU_LISTS, cfgINDEX_LRU_LISTS, and cfgMEMORY_HASH array entries.

Configuration Flexibility with Environment Variables

c-treeACE allows many of its configuration options to include an environment variable name that will be substituted with its value when the configuration file is read. For example, on Windows an administrator might want to set the c-treeACE data directory based on the `PROGRAMDATA` environment variable:

```
LOCAL_DIRECTORY %PROGRAMDATA%\FairCom\V*\bin\ace\sql\data
```

In this example, `%PROGRAMDATA%` is replaced with the value of the `PROGRAMDATA` environment variable (according to the environment variables that are defined in the logon session in which c-treeACE is running).

If you want to specify a literal `%` sign that is not treated as part of an environment variable name, use `%%`. For example, if you want to specify a value of `ABC%DEF`, specify it as `ABC%%DEF`. A literal `%` sign cannot be used in an environment variable name; it always signifies the end of the environment variable name. For example, `%MY%%VAR%` is treated as two environment variables, `MY` and `VAR`, rather than one variable named `MY%VAR`.

The following configuration options support this feature:

- `ADMIN_MIRROR`
- `BROADCAST_DATA`
- `CTSRVR_CFG`
- `DISK_FULL_ACTION`
- `DISK_FULL_VOLUME`
- `DYNAMIC_DUMP`
- `KEY_EXCHANGE_PARAMS`
- `LOCAL_DIRECTORY`

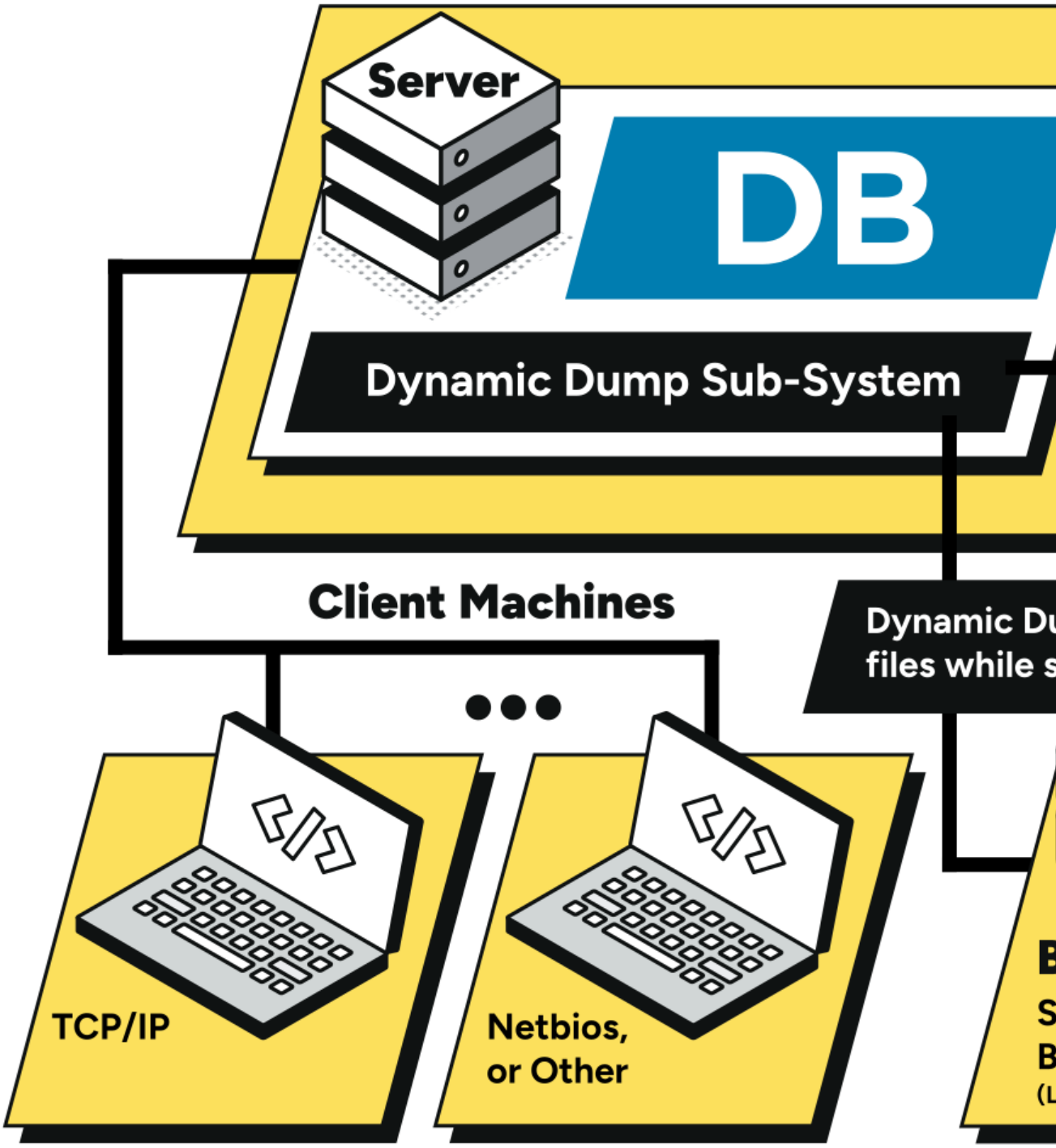
- LOG_EVEN
- LOG_EVEN_MIRROR
- LOG_ODD
- LOG_ODD_MIRROR
- MASTER_KEY_FILE
- MIRROR_DIRECTORY
- NULL_STRING
- PREIMAGE_FILE
- SERVER_DIRECTORY (deprecated)
- SHMEM_DIRECTORY
- SIGNAL_DOWN
- SIGNAL_MIRROR_EVENT
- SIGNAL_READY
- SIGNAL_USER_DOWN
- SIGNAL_USER_READY
- SQL_LOGFILE
- START_EVEN
- START_EVEN_MIRROR
- START_ODD
- START_ODD_MIRROR
- TMPNAME_PATH

License file name can be set with environment variable

At startup, FairCom Server looks for the license file in the current working directory (CWD) . In some cases, the license file may not be in that directory. In those cases, the server was not able to load the license file and start.

A new environment variable, FCSRVR_LIC, can be used to specify the license file. When this variable is specified, the server uses the file specified as the license file. The environment variable should contain a complete file name for the license file.

4.5 Backup/Restore



Backup Defer Interval for Improved Performance

The `DYNAMIC_DUMP_DEFER` option causes the dynamic dump to pause for the specified number of milliseconds each time it writes 64 KB of data to the dynamic dump stream file. For large backups, even the smallest DYNAMIC_DUMP_DEFER value of 1 millisecond adds significant time to the dynamic dump. For example, 100 GB = 1600000 * 1 ms. = 1600 seconds of additional time.

An additional keyword, `DYNAMIC_DUMP_DEFER_INTERVAL`, specifies the number of 64 KB blocks that are written before the `DYNAMIC_DUMP_DEFER` sleep is performed. For example, `DYNAMIC_DUMP_DEFER_INTERVAL` 16 would cause the `DYNAMIC_DUMP_DEFER` sleep to occur after every 64 KB * 16 = 1 MB of data written to the dump stream file.

Note: If a value greater than 5000 is specified for `DYNAMIC_DUMP_DEFER_INTERVAL`, the value is set to 5000. If a value less than 1 is specified, the value is set to 1.

This option can be set by the **ctSETCFG()** API function:

- **ctSETCFG(** *setcfgDYNAMIC_DUMP_DEFER_INTERVAL, "16"* **);**

A new menu option to set this value has been added to option 10 of the c-treeACE Server Administration (**ctadmn**) menu.

Forward Roll Path Redirection

While restoring from a backup with the forward roll utility, **ctfdmp** supports applying file name redirection rules to the file names that appear in the transaction logs. To use this feature, run **ctfdmp** with the *!REDIRECT* option, specifying the name of a text file that contains the redirection rules. For example, the following command indicates that the file *redir.txt* contains redirection rules:

```
ctfdmp !REDIRECT redir.txt
```

Each line in the file contains the portion of the file name to replace and its replacement. Place double quotation marks around a string if it contains spaces. A line that begins with a semicolon is ignored.

Examples

To replace an empty path with *output*\:

```
;Replace empty path with output\
"" output\
```

To replace *Program Files(x86)* with *Program Files*:

```
"Program Files(x86)" "Program Files"
```

To replace *production*\ with *test*\:

```
production\ test\
```

4.6 Logging and Information

Full version and build date available to clients

The c-treeACE Server's "mini" version number, build date, and base build date (if any) are now available to clients. The cfgVERSIONID2 array entry of **SYSCFG()** holds the mini version and build date values and cfgVERSIONID_BASE holds the base build date value. These new values allow a client to display the same full server build date that the c-treeACE Server displays.

The **ctadmn** utility and **ctdbGetProductVersion()** have been modified to include these values in the version string.

The function **ctGetVERSIONID2()** is used to retrieve the mini and build date values from the **SYSCFG()** cfgVERSIONID2 value. **ctGetVERSIONID_BASE()** is used to retrieve the base build date (if any) from the **SYSCFG()** cfgVERSIONID_BASE value.

Enhanced audit capability logs IP addresses for all connections in lock dump log on Windows

c-treeACE now tracks IP addresses for all connections.

When a lock dump is logged using the **ctLOKDMP()** API function, the list of connections now also includes the IP address (in addition to the user name and node name).

Note: This support is currently limited to Windows.

Error codes adjusted for uniqueness

We now enforce the convention that all error codes have unique absolute values. A small number of positive error code values were shared with a negative value. For example, **FNOP_COD(-10)** and **SPAC_ERR(10)** share the absolute value of 10.

Changing to unique error codes caused the following negative sysiocod values to change:

Symbolic constant	Old value	New value
FNOP_COD	-10	-975
FNOR_COD	-11	-976
PRTL_COD	-970	-972
PRTL_FF	-971	-973
LWRT_COD	-972	-974

None of the positive error codes were changed.

Applications checking sysiocod using symbolic constants need to be re-complied to map to the appropriate value. Applications checking sysiocod using error values need to be updated, where necessary, to the new error codes.

Automatic Recovery diagnostic logging option

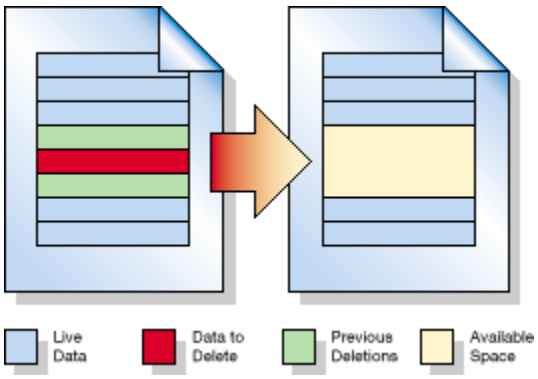
c-tree supports logging details of its automatic transaction recovery operation and in this revision a c-treeACE configuration option is added to enable transaction recovery logging:

`DIAGNOSTICS TRAN_RECOVERY` (off by default) causes transaction recovery log messages to be written to the file *RECOVERY.FCS*.

In the standalone model, this feature is enabled by activating `#define ctDBGRCVR` before compiling the c-tree library. In such cases, log messages are written to the file *RECOVERY.FCS*.

4.7 Improvements in space reclamation in data records

Improvements have been made in the ability to combine deleted space to reduce fragmentation. ctTRNLOG files that do not have a RECBYT index can now attempt to combine the deleted space with adjacent deleted space that already exists.



To enable this behavior, either:

- Add the server keyword `COALESCE_TRNLOG ON` (default is `OFF`)

or

- Set the global variable `ctcoaltran = YES` in a standalone compile.

4.8 Increased default maximum number of segments per key

The default maximum number of segments per key (`ctMAX_KEY_SEG`) has been increased to `16`.

4.9 Enable Windows service pre-shutdown timeout

Windows Vista and later allow a new Service PRESHUTDOWN notification, which gives the service extra time to complete its shutdown. This can help to avoid a lengthy auto-recovery if the server has too many cache pages to flush within the SHUTDOWN limit.

An option has been added to the **ctntinst** utility to set the timeout. To set the timeout:

```
ctntinst -setpsdtimeout <TIMEOUT_IN_SECONDS> <SERVICE_NAME>
```

To view the timeout setting:

```
ctntinst -showconfig <SERVICE_NAME>
```

Note: The **ctntinst** tool is no longer included with FairCom Server as of V10.3. It is recommended to use the standard Windows **sc.exe** command, which is more compatible than this utility.

4.10 Locking table allows different behavior per-file, per-connection

In V10.3, enhancements to the locking table allow the same file to be opened twice by the same thread with different locking applied each time. V11 further enhanced this behavior, as described below.

Starting in V10.3, c-treeACE supports opening the same file multiple times in the same connection assigning a different file number to each file or, in FairCom DB API, a different file handle. This can be useful in situations where you want to allow the same file to be opened twice by the same thread with different locking attributes applied to each thread.

Each of these sibling files is referred to as a "co-file." For example, if the file *customer.dat* is opened in the same connection using file numbers 5 and 10, then we say that file 5 is a co-file of file 10, and vice versa.

In this case there are considerations about how locks interact within the same connection when operating using different co-files. For example, if a write lock is acquired on a record R using file number 5 within the same connection, what is the behavior of trying to acquire a lock on R using co-file number 10?

In this example, before this enhancement, FairCom Server behaved as follows:

> The lock on R issued with co-file number 10 succeed and is considered a "secondary lock", while the lock acquired first (using file number 5) is considered "primary."
>
> The difference in the locks manifests itself during calls to unlock the record: If the primary lock is unlocked first, then the primary lock and all the corresponding locks on co-files are removed. But if a secondary lock is unlocked before the primary lock is unlocked, then only the secondary user lock is removed; and the primary lock is maintained.
>
> Any other connection saw the record locked until the primary lock was released.

This previous behavior has been maintained and it is the system-level default behavior.

It is now possible to configure the behavior choosing among 4 different options:

- **NODIFUSR**: The default as described above.
- **DIFUSR**: Locks on co-files are considered as acquired from a different connection, so the lock on R issued with co-file number 10 will fail.
- **SAMUSR_M**: Locks on record R on co-files are considered as the same lock acquired on the same file, so lock on R issued with co-file number 10 succeeds. As soon as the lock is released in one of the co-files that successfully requested the lock, the lock is released. Therefore, before acquiring the lock on R using file number 10, the lock can be released only

using file number 5, but after acquiring the lock on R using file number 10, the lock can be released either by using file number 5 or 10.

- **SAMUSR_1**: Locks on record R on co-files are considered as the same lock acquired on the same file, so lock on R issued with co-file number 10 succeeds. As soon as the lock is released in one of the co-files (whether or not the lock was requested using the co-file) the lock is released. Therefore, even before acquiring the lock on R using file number 10 the lock can be released either by using file number 5 or 10.

Recursive locks are not supported for co-files. An attempt to open a co-file when recursive locks are pending on the underlying file will fail with the error **MUOP_RCR** (998). An attempt to issue a lock on a co-file with the *ctLK_RECR* bit set in the lock mode will fail with the error **MLOK_ERR** (999).

Read locks behave in a manner consistent with write locks. The notable issues are:

1. With DIFUSR, read locks can be issued for different co-files; and unlocking one co-file's read lock does not remove the read lock from any other co-files that requested the read lock.
2. With DIFUSR, a read lock on a co-file cannot be promoted to a write lock if other co-files have read locks; a non-blocking write lock will fail with **DLOK_ERR** (42) and a blocking write lock will fail with **DEAD_ERR** (86).
3. With SAMUSR_*, read locks can be issued for different co-files, and unlocking one co-file read lock unlocks all the co-file read locks.
4. With SAMUSR_*, read locks can be promoted to write locks as long as no other threads have also acquired read locks.
5. With SAMUSR_1, a read lock on a co-file can be unlocked using another co-file's file number even if no lock has been issued using the other co-file number.

The system-level default can be controlled by using one of the following configuration keywords which sets the behavior accordingly to their names.

- `COMPATIBILITY MULTIOPN_DIFUSR`
- `COMPATIBILITY MULTIOPN_SAMUSR_M`
- `COMPATIBILITY MULTIOPN_SAMUSR_1`

A connection can override the system-level default for all open instances of a file by calling:

PUTHDR(*datno*, *mode*, *ctMULTIOPNhdr***)**

Where *mode* is one of the following:

- ctMULTIOPNnodifusr
- ctMULTIOPNdifusr
- ctMULTIOPNsamusr_M
- ctMULTIOPNsamusr_1

If no **PUTHDR** call is made, the system-level default is used for that connection's instances of the file. When a file is opened, if that connection already has the file open, the newly opened file inherits the MULTIOPN setting of the already-open file instance. An attempt to change the setting so that one instance of the file would be inconsistent with the others will fail with error **MOFL_ERR**. A file's MULTIOPN state can only be changed if it is the first open instance of the file and it has no pending locks.

4.11 File rebuild memory limit - File memory usage keyword for rebuild memory usage

Support was added for `SORT_MEMORY` to specify the maximum memory to use in a rebuild. The `SORT_MEMORY` keyword specifies a memory size in bytes and can use the MB and GB keywords (unlike `MAX_K_TO_USE`).

The maximum `SORT_MEMORY` value is:

- 4 TB - 1 for 64-bit c-treeACE
- 4 GB - 1 for 32-bit c-treeACE

As the `SORT_MEMORY` option is more intuitive, its use is recommended over MAX_K_TO_USE. (MAX_K_TO_USE remains available for backward compatibility). If both SORT_MEMORY and MAX_K_TO_USE are specified in *ctsrvr.cfg*, only the one that is specified last in the configuration file takes effect.

4.12 Database Copy with Virtual Tables and improved performance

The database copy logic, **ctsqlcdb** *-copy*, has been updated to support virtual tables (called "Multi Record Tables" or "MRT tables"). In addition, the copy process was modified to generate fewer, larger transactions to improve performance.

4.13 Greater user connection control

Limit Concurrent Connections by User

c-treeACE now supports limiting the number of concurrent user accounts and a configurable server-wide limit on connections per user account.

The `MAX_CONCURRENT_USER_ACCOUNTS <max_accounts>` configuration option sets the maximum number of user accounts that can connect at one time to c-treeACE.

The `MAX_CONNECTIONS_PER_USER_ACCOUNT <max_connections>` configuration option sets the maximum number of connections for each user account.

If limits are set in the license file, the configuration option can only be used to further reduce the connection limits (they cannot be increased above the license file limits).

The new error code **ALMT_ERR** (984) is returned when logon is denied because the number of distinct user accounts that are allowed to be connected at one time has been reached.

Independent Control of ISAM and SQL Connections

The c-treeACE Server configuration options `MAX_ISAM_CONNECTIONS` and `MAX_SQL_CONNECTIONS` can be used to set the maximum number of ISAM and SQL

connections, respectively. These configuration options can only reduce the limits set in the license file (they cannot increase the limits).

4.14 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

Batch insert operations in BAT_RET_BLK mode now supported

A batch insert on replicated data files with BAT_RET_BLK mode failed with a **REPL_ERR** error because replication did not support this operation. The c-tree Replication Agent now supports BAT_RET_BLK batch inserts on a replicated data file.

Replication Agent - Improved counting of failed operations

The Replication Agent's counting of operations that fail because the file cannot be opened on a target server has been improved. When applying changes fails because a file cannot be opened, the failed operation count is now incremented so that the transaction gets aborted, and when logging the exception records, the appropriate operation counter is incremented.

Log more descriptive error message

The Replication Agent has been improved to log more descriptive error messages when it fails to read the source or target server node ID because the host name cannot be resolved.

Note: The Replication Agent has been changed to redirect its standard error output to *ctreplagent.log* instead of *ctreplagent_stderr.log* so there is only one log file to monitor.

Starting at source server's current log position

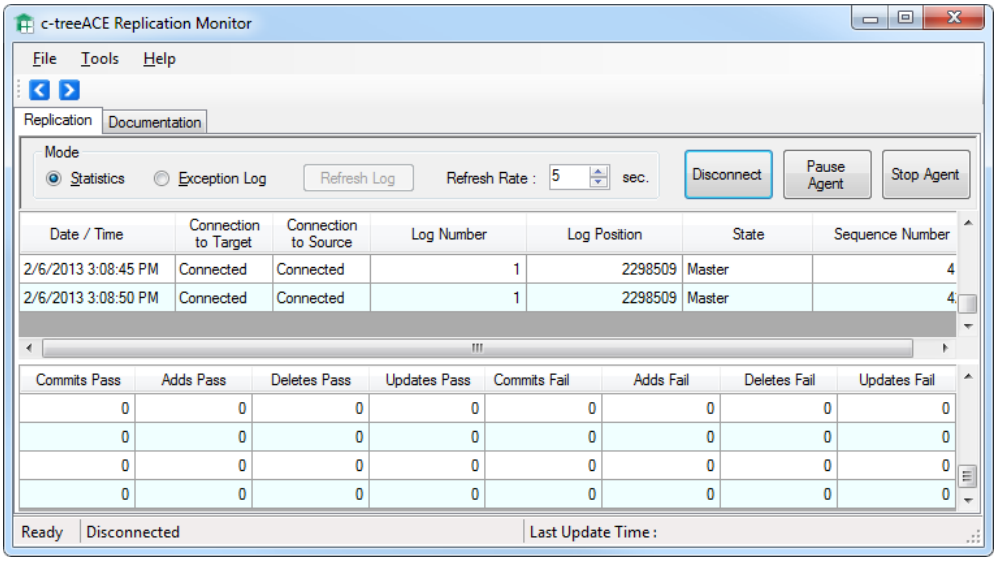
The Replication Agent can now be started at the source server's current log position by creating the file *ctreplagent.ini* (which is the same file that is normally used to override the Replication Agent's starting log position) and writing `#current` on the first line of the file.

The Replication Agent connects to the source server and reads its current log position then initializes the replication read connection to the source server using that log position.

Note: The source server must be V10.0 or later to use this feature.

c-treeACE Replication Monitor - New tool for the Replication Agent

A new graphical utility, the c-treeACE Replication Monitor (**c-treeReplMonitor.exe**), has been created to monitor the FairCom Replication Agent.



5. c-treeACE SQL

The SQL interface continues to be enhanced for greater stability and performance. The improvements listed in this section represent our latest efforts in our SQL subsystem.

5.1 Temporary memory subsystem improved

This update provides enhancements to the c-treeACE SQL internal temporary memory allocation routines which will improve performance and enhance stability with complex and large queries.

5.2 Removed c-treeACE SQL statement size limitation

The maximum size of c-treeACE SQL statements has been greatly expanded. Statements can now dynamically grow up to 32MB, providing for the most complex SQL statements. (Prior versions imposed a restrictive 35,000-character limit on SQL statements.) Statements larger than this are refused by the server.

Note: The client APIs do not impose any limitation on the maximum statement size. However, some tools, such as **ISQL**, **dbload**, **dbdump**, and **dbschema**, still impose a statement-size limitation of 35,000 characters.

5.3 Encryption of c-treeACE SQL system tables

c-treeACE SQL system tables are now encrypted by default to protect sensitive database meta data. The encryption algorithm is AES with a 128-bit key when advanced encryption is in use (meaning `ADVANCED_ENCRYPTION` is active in *ctsrvr.cfg*). When not using the advanced encryption, this data is scrambled with CAMO. 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.

To have system tables in clear text, `SQL_OPTION NO_CRYPT_SYSTBL` must be placed in *ctsrvr.cfg* before the database is created.

5.4 Configurable c-treeACE SQL server connection timeout

The c-treeACE SQL connection timeout is now configurable using the keyword `SQL_CONNECT_TIMEOUT_SEC`. The default is 1 second. A setting of 0 will revert to the previous behavior.

5.5 Built-in procedure now returns full version information

The built-in stored procedure **fc_get_server_version** did not return the "mini" part of the version number as a separate field. The "version" and "build_date" fields of the resultset were not large enough to contain the full string resulting in **error -20052** (overflow error) when executing the stored procedure.

The stored procedure has been updated as follows:

- "version" expanded from 32 to 40 chars
- "build_date" expanded from 8 to 14 char
- "ver_mini" added as last column in resultset - This is a new integer value containing the mini revision number (third revision number component). It is the last column (instead of between the "ver_minor" and "ver_revision") so as maintain backward compatibility with existing code that may assume the column positions.

The output of the stored procedure when launched from ISQL is as follows:

```
VERSION                  VER_MAJOR VER_MINOR VER_REVISION BUILD_DATE   VER_MINI
-------                  --------- --------- ------------ ----------   --------
10.0.1.60592(Build-121010)       10         0        60592 121010              1
```

5.6 SQL_OPTION OLD_DELFLD_LEN

Allows c-treeACE SQL CREATE TABLE and ALTER TABLE commands to use the $DELFLD$ size that was in effect prior to V9.5:

```
SQL_OPTION OLD_DELFLD_LEN
```

In c-treeACE SQL V9.5 the FairCom DB API $DELFLD$ size changed from 4 bytes (or 5 in case of non-HUGE files) to 9 bytes. This change required any C structures describing the record to be changed.

Any table created when this keyword is specified in the *ctsrvr.cfg* will have the 4-byte $DELFLD$ used in earlier releases.

Any table created or altered in structure using ALTER TABLE while `SQL_OPTION OLD_DELFLD_LEN` is in effect will have the $DELFLD$ set to 4 bytes regardless of the size of the original table.

At the FairCom DB API level, the **ctdbAlterTable()** function now takes a new mode, `CTDB_ALTER_V8DELFLD`, to indicate that the table (if re-created) must have the old $DELFLD$ size (`CTDB_ALTER_V8DELFLD` forces `CTCREATE_V8DELFLD` during ALTER TABLE).

5.7 Remove c-treeACE SQL database on failed database copy

If a c-treeACE SQL database copy fails, the new (partially copied) database is now removed from the list of active databases. Any copied files still remain on disk.

5.8 c-treeACE SQL shutdown when Callback Library loading fails

When `SQL_OPTION LOAD_CALLBACK_LIB` was specified, and callback loading failed, the server continued regular operations without any notice.

As the callback library is critical for proper data interpretation, when its loading is required by specifying the above configuration keyword, a failed load now generates a panic condition and terminates the c-treeACE SQL Server operation with the following message logged in *CTSTATUS.FCS*:

```
- User# 00001 : PANIC - TPEUTIL LoadSQLSDK callback library load failed PID 5996
- User# 00001 SQL PANIC, attempting shutdown
```

6. Interface Technology Additions

c-treeACE continues to provide developers with a rich set of APIs. This release broadens and deepens that support.

6.1 FairCom DB API

Table file names of the form *.XXX.XXX

The open table logic in FairCom DB API attempts to determine if file names contain an extension. When a file name of the form *Y.XXX.XXX* was passed to **ctsqlimp**, it set the file name used to open the table to *Y.XXX* and the file extension to *XXX*. When it opened the file, it identified the file name as *Y.XXX* ending with the extension *.XXX*, which was incorrect.

New "extension handling hints" were added so that it is now possible to indicate if the default heuristic should be used, or to specify that the extension is present or missing before calling **ctdbOpen()**. This new flag is set by default to EXT_DETECT indicating to use the heuristic approach. Setting it to EXT_MISSING indicates that the extension is missing.

In addition, modifications were made to **ctsqlimp** to import tables with names of the form **.YYY.XXX* (where *XXX* may or may not be the same as *YYY*).

When a table handle has been allocated on the session (no dictionary support is in use), you can call **ctdbSetExtensionHint(***hTable,...***)** before **ctdbOpenTable()** to indicate:

1. EXT_DETECT: Backward compatible behavior, FairCom DB API will detect if the file ends with "**.***" and remove it from the file name and then add the current table extension.
2. EXT_PRESENT: The file name is specified with file extension, FairCom DB API will detect if the file ends with "**.***" and use what matches as an extension; if no match, the extension is set to "".
3. EXT_MISSING: The file name is specified without extension, FairCom DB API will add the current extension.

Calling **ctdbSetExtensionHint(***hDatabase,...***)** on a database handle allows setting the extension hint for **ctdbAddTable()** and **ctdbAddTableXtd()**.

In addition to the hints described above, it is possible for the **Xtd()** version to use:

- EXT_PRESENT_L - Logical file name contains file extension; physical does not. FairCom DB API will remove the extension "**.***" from the logical name.

- EXT_PRESENT_P - Physical file name contains file extension; logical does not. FairCom DB API will remove the extension "`.*`" from the logical name and store it a the file extension.

Function:

ctdbSetExtensionHint

Set extension hint in the handle to help with proper file name building at table open time.

Declaration

```
CTDBRET ctdbDECL ctdbSetExtensionHint(CTHANDLE Handle, EXT_INFO ExtHint)
```

Parameters:

- *Handle* [IN] - Table Handle.
- *ExtHint* [IN] - The extension hint for **ctdbOpen**, **ctdbAddTable**, **ctdbAddTableXtd**:

  EXT_DETECT - Backward compatible behavior: FairCom DB API will detect if the file terminates with "`.*`" and remove it from the file name and then add the current table extension.

  EXT_PRESENT - File names contain file extension: The file name is specified with file extension; FairCom DB API will detect if the file terminates with "`.*`" and use what matches as extension, in case of no match the extension is set to "" (none).

  EXT_MISSING - File names do not contain file extension: The file name is specified without extension; FairCom DB API will add the current extension.

  For **ctdbAddTableXtd** only:

  EXT_PRESENT_L - Logical file name contains file extension; physical does not.

  EXT_PRESENT_P - Physical file name contains file extension; logical does not.

Returns:

None

ctdbGetRecordKeyPos()

The following new method has been implemented in the FairCom DB API API to retrieve the given record position in the default index: **ctdbGetRecordKeyPos()**

Syntax

```
CTDBRET ctdbDECL ctdbGetRecordKeyPos(CTHANDLE Handle, pCTOFFSET pPosition)
```

Parameters

- *Handle* [IN] - FairCom DB API C API record handle
- *pPosition* [OUT] - Position in the given index

Description

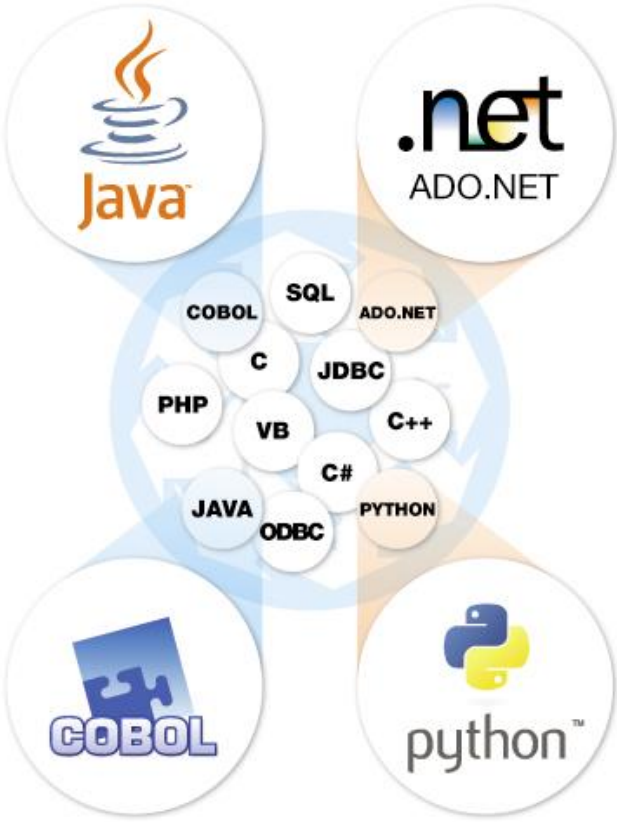
Retrieves the given record position in the default index. The returned position is the Ordinal key position, not a file offset.

Return

CTDBRET_OK on success

6.2 c-treeDB.NET

All tools and assemblies have been updated to support the latest Microsoft .NET 4.0 Framework.



.NET Tools for VS2010 - All projects updated to use .NET Framework v4.0

All the .NET projects for VS2010 have been updated to use the v4.0 .NET framework. If you target the .NET v3.5 framework, it uses the VS2008 compiler 'tool chain' which can result in an internal compiler error in VS2010. The target framework was changed to v4.0 in the **_v10.sln* files to eliminate that error.

See Also

- http://support.microsoft.com/kb/976656

.NET - Removed STRONGSIGN from assemblies

To be more compliant with standard practice for C# programmers, STRONGSIGN has been removed from .NET assemblies. We no longer force the assembly to be signed with the FairCom key. This allows developers to sign with their own key, which they can keep secret.

.NET - New SetEncryption method for FairCom.Isam API

A new **SetEncryption** method for the *FairCom.Isam.dll* makes it possible to enable/disable file encryption from this interface. The following methods, along with the ENCRYPTION enumerator (which lists all possible encryption types), have been implemented:

```
static COUNT SetEncryption(ENCRYPTION mode, int keylen)
static COUNT SetEncryption(array<byte>^ key, int keylen)
```

Although this method is based on the standard FairCom DB API **SetEncryption()** function, it has been modified to better comply with .NET requirements.

Typical calls from C#

Basic encryption:

```
byte[] enc = Encoding.ASCII.GetBytes("faircom");
int err = mCtree.SetEncryption(enc, enc.Length); // Enable
...
int err = mCtree.SetEncryption(enc, 0); // Disable
```

Advanced encryption:

```
int err = mCtree.SetEncryption(ENCRYPTION.DES8, 1);  // Enable
...
int err = mCtree.SetEncryption(ENCRYPTION.DES8, 0); // Disable
```

FairCom.CtreeDB.dll - New GetRecordBuffer(Byte buffer) method

The new **GetRecordBuffer(***Byte buffer***)** method was implemented in the *FairCom.CtreeDB.dll* to provide the ability to return the complete record buffer in a Byte array. The method is of the form:

```
public virtual void GetRecordBuffer(Byte[] buffer)
```

6.3 FairCom DB API C++

The following changes have been implemented in the FairCom DB API C++ API.

CTRecord::GetRecordKeyPos

Syntax

```
CTOFFSET CTRecord::GetRecordKeyPos() const
```

Parameters

- None

Description

Retrieves the given record position in the default index.

Return

Position. The returned position is the Ordinal key position, not a file offset.

Throws a *CTException* exception object if an error occurs.

6.4 FairCom DB API .NET API

The following changes have been implemented in the FairCom DB API JTDB API.

CTRecord.GetRecordKeyPos

Syntax

```
CTOFFSET CTRecord::GetRecordKeyPos() const
```

Parameters

- None

Description

Retrieves the given record position in the default index. The returned position is the Ordinal key position, not a file offset.

Return

Position. Throws a *CTException* exception object if an error occurs.

6.5 FairCom DB API .NET API

The following changes have been implemented in the FairCom DB API .NET API.

CTRecord.GetRecordKeyPos

Syntax

```
long CTRecord.GetRecordKeyPos()
```

Parameters

- None

Description

Retrieves the given record position in the default index.

Return

Position. The returned position is the Ordinal key position, not a file offset.

Throws a *CTException* exception object if an error occurs.

6.6 FairCom DB API Java

c-treeACE V10.0 introduced a FairCom DB API layer for Java, frequently referred to as JTDB. This allows c-tree Java developers direct record access capability for performance when it is needed—up to 40% performance gain over JDBC. Several additions have been made to this new addition to the c-tree family of interface technologies.

Java helper library updated

Many of the FairCom DB API for Java classes have been updated including grid, renderers, and editors.

Improved message to indicate incorrect JTCB JNI DLL found

Improvements have been made to the error message that is displayed if the *mtclijni* DLL is missing or if it is found but was compiled for a different platform (32-bit vs 64-bit). The message now suggests the proper action.

Reduced thread contention with FairCom DB API for Java (JTDB)

A mutex has been moved from the thread to the session in FairCom DB API for Java for both performance and scalability reasons. The following functions are now exported and available for use within the JTDB layer: **ctThrdIni()**, **ctThrdSet()**, and **ctThrdPut()**.

During the use of **ctdb.AllocSession()**, call **ctThrdSet()** or **ctThrdInit()** before **ctdbAllocSession()**.

6.7 SQL Interfaces

PHP - Query timeout support

Query timeout support has been implemented in PHP. The query timeout can be set in *php.ini* by setting the `ctsql.querytimeout` property to the number of seconds. The default is 0 meaning no timeout.

For example, to set a 5-second timeout:

```
[ctsql]
ctsql.querytimeout=5
```

Python updated to use cDecimal Class and new ctsqlapi functions

Profiling analysis revealed ways that the Python driver could be made more efficient in handling numeric fields. Improvements that have been made include:

- Using the `cdecimal` package when available.
- Removing the `_sqlnumeric` class that stored the decimal in native format before conversion.
- Mapped the new **ctsqlGetNumericAsString** and **ctsqlSetNumericParameterAsString** functions and used them to deal with numeric values.

Python improved performance on numeric column retrieval

Changes to the logic the Python interface uses to retrieve numeric values have improved performance of this operation by a factor of 200.

Python Cursor.rowcount returns number of fetched rows

The **Cursor.rowcount** property has been improved so that it now returns the number of fetched rows instead of -1 as it did previously. Both the old and new behaviors are compliant with the DB API 2.0 standard.

Direct SQL - New numeric conversion functions

Most high-level languages have their own numeric/decimal types that usually require converting them into strings in the high-level language and then converting the strings into the proper numeric/decimal types. To aid in this process, two new functions were added that return numeric columns as strings:

ctsqlSetNumericParameterAsString

Declaration

```
CTSQLRET ctsqlDECL ctsqlSetNumericParameterAsString(pCTSQLCMD hCmd, INTEGER index, CTSQLTYPE ptype, BIT isnull, CTSQLCHAR* buffer)
```

Description

Set parameter for all numeric types: CTSQL_NUMERIC and CTSQL_MONEY.

Parameters

- *hCmd* - command handle
- *index* - parameter number you want to set. The value must be greater or equal to zero but less than the parameter count
- *ptype* - type of parameter being passed in buffer. This type must match one of the following types: CTSQL_NUMERIC and CTSQL_MONEY
- *isnull* - indicate if parameter should be NULL or not
- *buffer* - string representing the numeric data

Return

Returns **CTSQLRET_OK** on success.

ctsqlGetNumericAsString

Declaration

```
CTSQLRET ctsqlDECL ctsqlGetNumericAsString(pCTSQLCURSOR hCursor, INTEGER colnumber, CTSQLCHAR* buffer, INTEGER size)
```

Description

Retrieve the value of a numeric field.

Parameters

- *hCursor* - cursor handle
- *colnumber* - number of column
- *buffer* [out] - buffer to receive field data
- *size* - size of buffer

Return

Returns **CTSQLRET_OK** on success.

Direct SQL - ctsqlClearError function

A new function was added to clear the error information returned by **ctsqlGetError()** and **ctsqlGetErrorMessage()**:

ctsqlClearError

Declaration

```
void ctsqlClearError(pCTSQLCONN hConn)
```

Description

Clears the error information returned by **ctsqlGetError()** and **ctsqlGetErrorMessage()**.

Return

Void.

See Also

- **ctsqlGetError()**
- **ctsqlGetErrorMessage()**

6.8 Support for uTFRMKEY in client library

Support for the **uTFRMKEY()** function has been implemented in the c-tree client library. The c-tree standalone, client, and server DLLs now export this function.

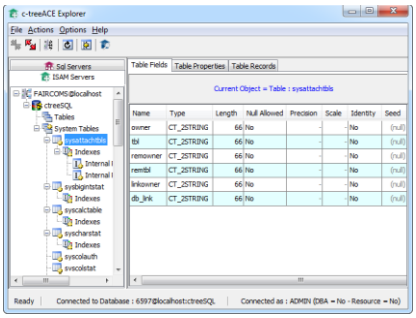
7. GUI Tools

The graphic tools have been refreshed. They are available in Java for cross-platform support. In addition, we have made numerous updates to the original .NET versions of these valuable tools.

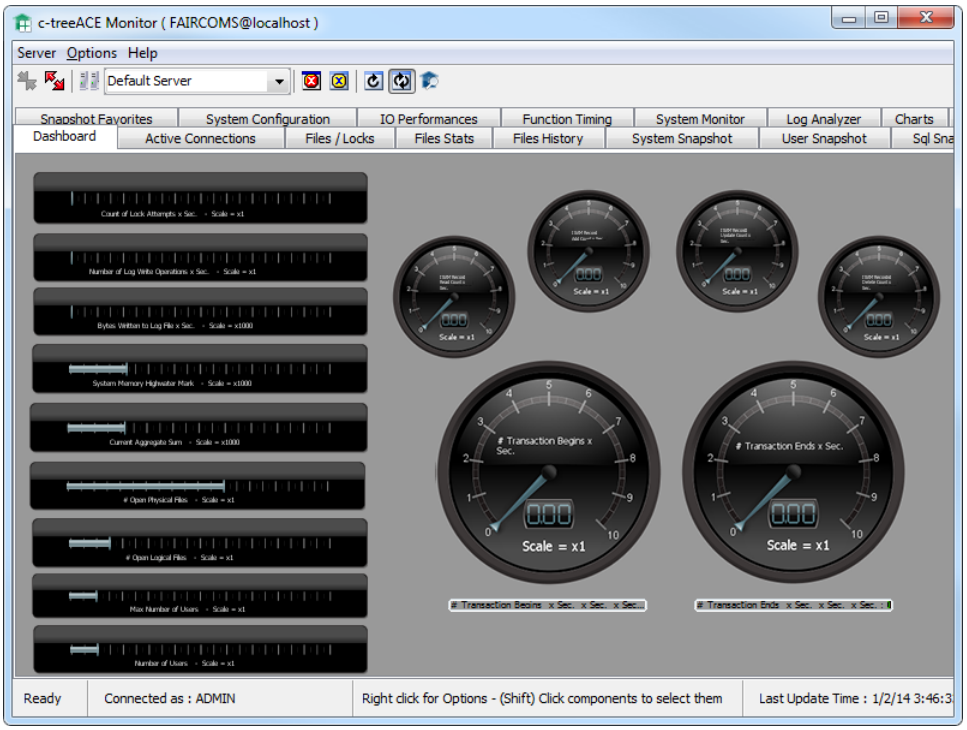
7.1 Cross-platform Java tools

The cross-platform Java versions of the graphical tools have been greatly enhanced. Some of the tools have been combined to simplify their operation. The following tools are available:

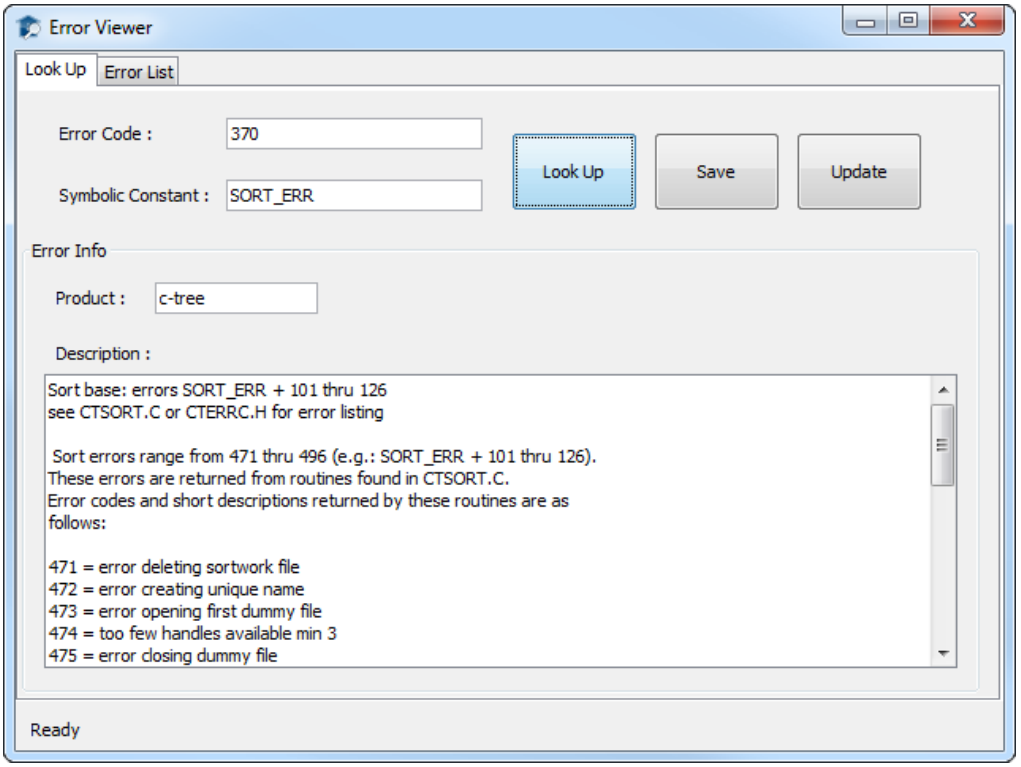
c-treeACE Explorer - View and edit both SQL and ISAM databases from a single tool



c-treeACE Monitor - Gauges and detailed monitoring of your c-tree databases

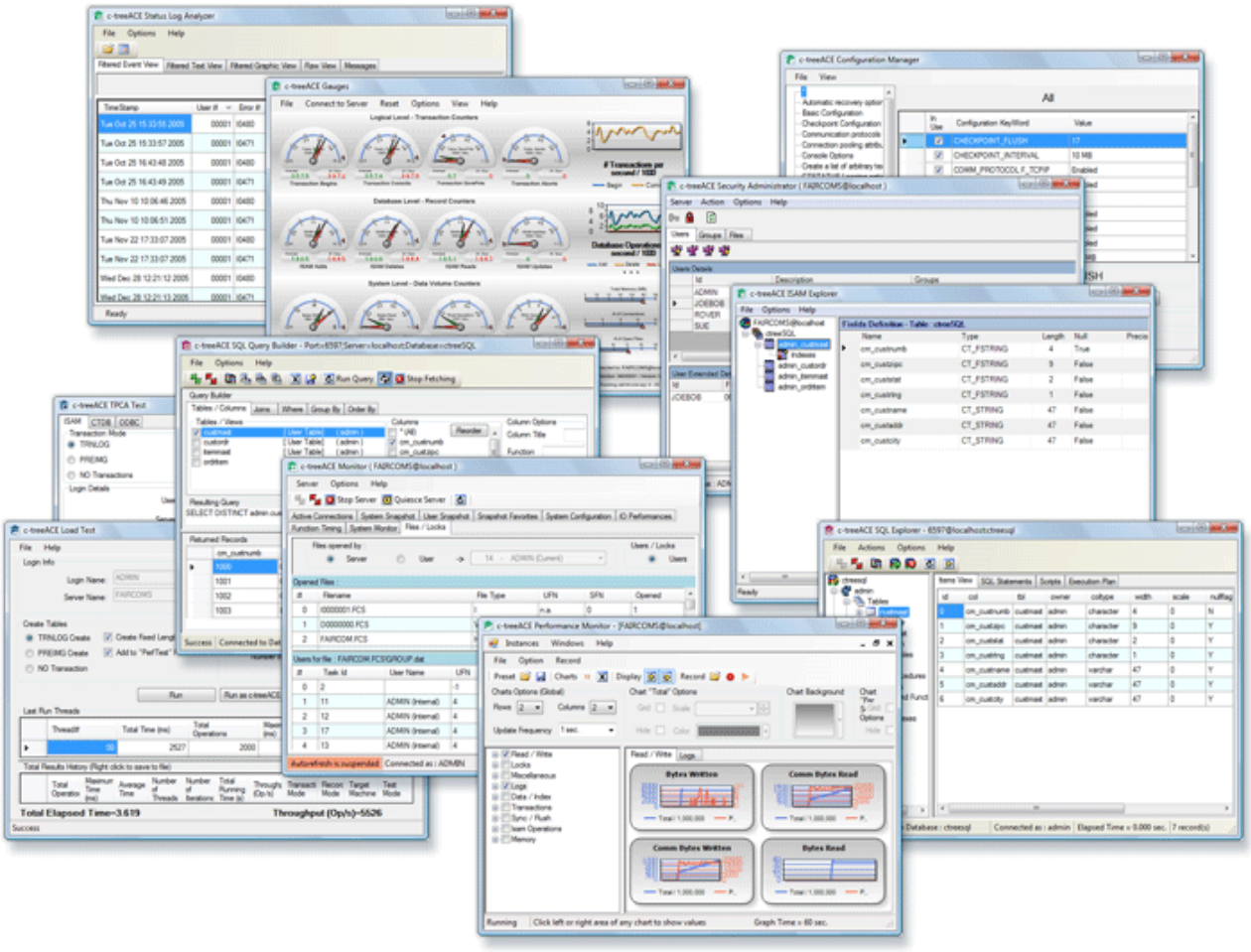


Error Viewer - An updated version of the error viewer includes an up-to-date list of c-treeACE errors



7.2 .NET GUI tools

For our customers who are already familiar with the .NET-based Windows GUI tools introduced in V9.0, we are pleased to provide the following enhancements. We plan to continue supporting both the Windows .NET GUI tools and the newer Java-based GUI tools for the foreseeable future, so you can choose the tools that best suit your needs.



c-treeACE SQL Explorer - Avoid UPDATE on unchanged columns

The logic in c-treeACE SQL Explorer has been changed to improve performance by eliminating unnecessary updates. This change also eliminates problems with COBOL "read-only" columns.

c-treeACE SQL Explorer - Export schema enhancement

The Export Schema function in the .NET version of the c-treeACE SQL Explorer has been improved over the older version of the tool. The prior tools considered an index with the file name starting with "sys_" as a system index and may not correctly export it in the schema script. The logic in the new version of the tools checks to see if an index has already been scripted as one of the PK, UK or FK constraints indexes.

c-treeACE SQL Explorer - Statements page shows column types

The Statements page in c-treeACE SQL Explorer now shows the column's data type below the column's name.

c-treeACE SQL Explorer - Support for breakpoints on scripts

The **Scripts** panel in the c-treeACE SQL Explorer now supports set/disable/clear breakpoints allowing scripts to pause if a breakpoint is set on a specific line.

c-treeACE SQL Explorer and c-treeACE Monitor Servers Manager

The new **Servers Manager** feature available in c-treeACE SQL Explorer and c-treeACE Monitor provides a way to store the login information for several servers in one place and lets the user easily select the server to which they will connect.

c-treeACE ISAM Explorer - RawMode enabled on all tables

In c-treeACE ISAM Explorer, "Raw" mode has been enabled for all tables; records are displayed as a byte array. The default file open mode can be changed using the "Open Mode" submenu list under the "Options" menu.

A spinner control is available in Table Records when viewing in Raw mode. It allows users to set an offset from the start of the record buffer from which to begin displaying bytes. This can be especially useful for viewing COBOL data.

A ruler is displayed above the records grid in Raw mode.

c-treeACE ISAM Explorer - Option for max characters in columns

To reduce memory usage, the maximum number of characters displayed in columns in the Records tab was set to 256. This value is now configurable through an option in the Options dialog.

c-treeACE ISAM Explorer - Replication tab removed

The Replication tab has been removed from the c-treeACE ISAM Explorer. The contents of this tab have been added to the new c-treeACE Replication Monitor tool (**c-treeReplMonitor.exe**), which is included with the Replication Agent optional add-on package and in the c-treeACE Professional package.

c-treeACE Gauges - AutoLogin and RememberPassword

Enhancements to c-treeACE Gauges include AutoLogin and RememberPassword. These features simplify the login process to make it more like the other c-treeACE GUI tools.

c-treeACE Gauges - Expanded range

The c-treeACE Gauges tool has been enhanced to handle a wider range of values. When memory usage is > 1GB, the value is scaled by a factor of one thousand and the label is changed to show GB instead of MB.

c-treeACE Status Log Analyzer - Added drag & drop support

Drag & drop support has been added to the c-treeACE Status Log Analyzer. Additionally, the log file path and name are now displayed on title bar.

c-treeACE ErrorViewer - Updated error file

Updates were made to the *FairComErrors.xml* error file to bring it up-to-date with the current list of errors in *cterrc.h*.

7.3 Improved Windows ompliance

cAll .NET based tools have been updated for improved compliance with Windows conventions. The directory where files used by the tools are saved/loaded has been changed from the executing directory to:
<user name>\AppData\Roaming\FairCom\Tools.

The following tools are updated in this manner:

- c-treeACE SQL Explorer
- c-treeACE Error Viewer
- c-treeACE Replication Monitor
- c-treeACE ISAM Explorer
- c-treeACE Query Builder
- c-treeACE Monitor

8. Command-Line Utilities

Improvements in the command-line utilities give you expanded control of the c-treeACE data management environment.

8.1 ctstat - c-treeACE Statistics Utility

ctstat: List file and user lock information

Added functionality to **ctstat** to retrieve lock information by file or by user. The **ctstat** *-filelocks datafile* and **ctstat** *-userlocks user* reports have been added.

For the *-filelocks* report:

- *-filelocks file [N]* lists all locks on a data file. Displays the *N*th key. Keys are displayed in hexadecimal format following each lock.
- See **-filelocks Notes** below.

For the *-userlocks* report:

- If *UserID* is a number, it is interpreted as a task ID.
- If *UserID* is a string, it is interpreted as a name, and information on locks held by each task ID with a matching name is returned.

Dumping large quantities of locks in a very active system could affect performance. Because the *-userlocks* report may generate a large number of server calls (for each task ID and file), the *-userlocks* report interval may be increased up to 60 seconds, depending on the number of matching users and files involved.

-filelocks Notes

The *-filelocks* option lists all locks on a data file and, optionally, displays the *N*th key. The lock offset and the associated keys are not read at the same time. Since we are reading records locked by other users to generate the key, there is no guaranteed relationship between the lock and the displayed key. The following are possible scenarios:

1. The displayed key is from before or after any changes made by the lock holder.
2. The locked offset no longer holds a valid record (it has been deleted, or updated and moved).

3. The locked offset could have been locked/modified/unlocked more than once between the time the lock offset was acquired and the time the record is read, so the offset could hold an entirely different record than what was originally locked.

ctstat -filelocks file [key] - Wildcards displaying record locks by file

The **ctstat** *-filelocks file [key]* command has been extended to support c-tree's standard wildcard filename matching for the specified file, allowing locks from multiple files to be displayed.

The standard c-tree wildcards (used by *ctsrvr.cfg* keywords such as MEMORY_FILE and REPLICATE, etc) are:

* - Multi-character match

? - Single-character match

^ - Negation (must be first character)

8.2 ctquiet - Unix option to avoid disconnect

The potential for unintended user interference was greatly reduced in V10 by use of the USERPRF_ADMSPCL bit. This modification makes further improvements by providing an alternative approach to avoid disconnecting the **ctquiet** on Unix systems. When the optional -w COMMAND switch is used, the behavior of **ctquiet** is modified to perform as follows:

1. After successfully quieting the server, it makes a system call to execute **COMMAND**.
2. It remains connected and waits for **SIGINT** to unquiet the server. If **ctquiet** is killed before receiving **SIGINT**, the server will remain in a quiet state until a new connection unquiets the server.

When connecting to a quieted server with the intent to remove the quiet state when the original caller of **ctquiet** has disconnected you must now set the USERPRF_ADMSPCL bit.

Syntax:

```
ctquiet [-s server][-f][-u][-w command] {-p password|-a authfile}
```

- *-s* - Server (default: FAIRCOMS@localhost)
- *-f* - Full consistency (default: crash consistency)
- *-u* - Unquiet server
- *-w* - Execute command on successful quiet. Waits for SIGINT to unquiet server
- *-a* - Authentication file name
- *-p* - Admin Password

8.3 ctsqlimp - c-treeACE SQL Import Utility

ctsqlimp -B switch grants public read-only access to linked tables

A new switch has been added to **ctsqlimp** to grant public read-only access to linked tables.

The existing *-b* (lowercase) switch grants *all* permissions on the table to the public.

The new *-B* (uppercase) switch grants *read-only* permissions on the table to the public.

If both *-B* and *-b* are specified, the read-only setting takes precedence.

Note: This introduces a change in behavior for existing applications because this switch is now-case sensitive.

Notice that the owner of the table, as well as the DBA, have all the permissions.

A new mode, CTSQLCB_GRANT_PUBLIC_RO was added to the SQLcallback, and a new member, `grntro`, was added to the CTSQLIMPOPTS structure.

ctsqlimp Check for Max Number of Indexes/Segments

In V10.3 and later, logic in **ctsqlimp** checks for the maximum number of indexes or segments. When importing tables, this logic checks if the number of indexes defined is less than `MAX_DAT_KEY` value and the number of segments per key is less than `MAX_KEY_SEG` value. The new logic prevents the import unless you are using interactive **ctsqlimp**, which asks if you want to continue with the import because it may be useful to import the table and adjust the server setting afterwards.

8.4 ctadmn - c-treeACE Administration Utility

ctadmn utility checks for active transactions before quiescing FairCom Server

When the administrator selects the **ctadmn** utility's option to quiesce a FairCom Server, **ctadmn** checks for active transactions. If any transactions are active, **ctadmn** prompts the user for a maximum time to wait for transactions to complete. That value is passed to the **ctQUIET()** function, which waits up to the specified number of seconds before aborting active transactions and quiescing FairCom Server.

ctadmn user listing for rtexecute thread running report launched by RTSCRIPT

The Communications section of the user listing shows **rtexecute** for threads launched by RTSCRIPT calls. Below is an example listing:

```
UserID: ---                              NodeName:
Task 15                                  Communications: rtexecute
        Memory: 1512K     Open Files: 8     Logon Time:   --
        Tran Time:   --   Rqst Time:  0:30  InProcess  Rqst# 0   -unknown-
```

This is analogous to how the launched dynamic dump thread, **idyndmp**, is listed. The listing states which thread to kill if one is using **ctadmn** to stop a report. Killing the thread that shows RTSCRIPT as the last function called will not interrupt the report being compiled by **rtexecute**.

8.5 Changes to ctrbldif, ctcmpcif, and ctinfo

The following changes have been made to **ctrbldif**, **ctcmpcif**, and **ctinfo**:

- **ctrbldif** now assigns data file owner/group/permissions to the index file.
- **ctcmpcif** now assigns original data file owner/group/permissions to compacted data/index files.
- **ctinfo** displays the file's security attributes.

8.6 ctrbldif - c-treeACE Rebuild Utility

Option to set index's automatic segment attributes

If a segmented index file was deleted and rebuilt using the **ctrbldif** utility or the rebuild API function, the new index file was not segmented. The index file's segment attributes, which are stored in the index file, were lost when the index file was deleted.

Beginning with V10.3, an option is provided with the rebuild and compact utilities that causes index files that are created by the utilities to be created using automatic segments of the specified size and maximum. The option is:

```
-idxseg=M@S
```

where *M* is the maximum number of segments and *S* is the segment size. *S* is interpreted as megabytes by default. Two suffixes, *MB* and *GB*, are also supported. *MB* indicates that the value is in megabytes and *GB* indicates that the value is in gigabytes.

Examples:

50 segments of 10 MB each:

```
ctrbldif test.dat -idxseg=50@10
```

100 segments of 300 MB each:

```
ctrbldif test.dat -idxseg=100@300MB
```

20 segments of 3 GB each:

```
ctrbldif test.dat -idxseg=20@3GB
```

Note: If the index files exist, the *-idxseg* option has no effect. It is only when the rebuild or compact utility creates new index files because the old index files do not exist that the *-idxseg* option affects the automatic segment properties of the index files.

Updates in ctrbldif, ctcmpcif, and ctinfo handling of security attributes

Updates have been made to the way in which these utilities handle security attributes:

- The **ctrbldif** utility now assigns data file owner/group/permissions to index files.
- The **ctcmpcif** utility now assigns original data file owner/group/permissions to compacted data/index files.
- The **ctinfo** utility displays the file's security attributes.

Error 456 (group access denied) had been seen when opening a c-tree data file at the ISAM level in client/server mode after rebuilding or compacting the file. This was because of the way the rebuild and compact utilities set the security attributes

It is desirable for the newly rebuilt or compacted files to preserve the original security attributes as much as possible and the index file security attributes should match the data file security attributes. The rebuild and compact utilities now read the permission mask, owner, and group settings from the original data file and after the rebuild or compact is completed, the utilities assign these same attributes to the new files.

The **ctinfo** utility now displays the owner, group, and permissions for the specified file. Example output:

```
    File owner         = ADMIN
    File group         = ADMIN
    Permission mask    = 0x85e = {
                             owner: read write def delete
                             group: read
                             world: read
                           }
```

A command-line option has been added to the utilities to restore the previous security attribute behavior. Use the *-oldsec* option to cause the rebuild and compact utilities to set the security attributes as they did before this revision. For example:

```
ctrbldif mark.dat -oldsec ADMIN ADMIN FAIRCOMS
```

The *-oldsec* option can be used if rebuild and compact are failing with error 455 (user does not belong to group) if you delete an index then run the rebuild or compact utility on the data file using a user account that does not belong to the group that is assigned to the data file.

Note: These changes only apply to the client and server versions of these utilities. Setting the security attributes is not supported in standalone mode, so the standalone rebuild and compact utilities behave as follows:

1) When the standalone mode rebuild utility creates new index files instead of reusing the new index files (for example if the original index files have been deleted before running the rebuild utility), the new index files are assigned a permission mask of zero (no restrictions on permissions), and the owner and group are unassigned (empty).

2) The standalone mode compact utility always preserves the security attributes of the data file. If the index files do not exist, the newly-created index files are assigned the same values as the standalone rebuild does when it creates new index files.

8.7 ctinfo - c-treeACE Information Utility

ctinfo -isam option added

The new **ctinfo** *-isam* option causes the utility to open the specified data file in ISAM mode. This option is useful for checking if a REPLICATE keyword enables replication for the file.

8.8 ctdmpidx - c-treeACE Dump Index Utility

ctdmpidx option to list all key values in index

The **ctdmpidx** utility now supports an option, *-listkeys*, to list all the key values and their associated record offsets in an index. The new usage for **ctdmpidx** is:

```
ctdmpidx [-<page size>] [-listkeys] <file name> <member #> [<rflg>]
```

8.9 ctldmp - c-treeACE Dump Utility

ctldmp option to create transaction start files from checkpoints in transaction log files

The **ctldmp** utility allows you to create transaction log start files for the transaction logs that it scans. Specify the *csf* ("create start files") option to use this feature.

The start files are named *S<lognumber>_<sequencenumber>.FCA*, where *lognumber* is the transaction log number and *sequencenumber* is the checkpoint number in that log (starting from 1).

These start files can be renamed to *S0000000.FCS* and *S0000001.FCS* so that FairCom Server can use the checkpoint positions as starting points for automatic recovery.

Example:

```
# ctldmp log 1254 csf

getlogfil
LOGPOS:1254-002473fax #0:690034176 P0051a057f3-00x  F0000-000 T26 U03 A0000x L139672 042 CHKPNT
0000e0007fa0a000a000920092007300000000007100120 0e000000000051000000000000000000
300054001cb10000200002004 2002c00000030008a20f000340000004000 4b20100000000000000
........q............""..."".. r<..........x.... ..............T.................
Created start file S0001254_0001.FCA for log position 0x002473fa

        Sat May 25 01:19:31 2013

LOGPOS:1254-0026a624x #0:690034177 P0051a057f3-00x  F0000-000 T26 U17 A0000x L10684 042 CHKPNT
ffffe000f720a000a000920092009200000000009200120 0e000000000072000000000000000000
bfff6400a340000020000200420082000000300 0c200f000340000004000830010000000000000
.........s$..........""..."".....""........... ""... ..............x#.............
Created start file S0001254_0002.FCA for log position 0x0026a624

        Sat May 25 01:19:31 2013
```

8.10 ctmtap - c-treeACE Utility

ctmtap - Command-line options for user name and password

The **ctmtap** test program now supports command-line options to specify the user name and password. If these options are not specified, **ctmtap** connects as guest. The options are:

- *uUSERID*
- *pPASSWORD*

The following example connects as user ADMIN to server FAIRCOMS, creates new files using the ctTRNLOG mode and runs 8 threads, each adding 100000 records:

```
ctmtap uADMIN pADMIN sFAIRCOMS fc mT t8 aA n100000
```

8.11 cttctx - c-treeACE Utility

cttctx locking options for record read

In V10.3 and later, the **cttctx** test program supports locking options for record reads. This test program adds a record, reads it with a write lock, and deletes it. It can be run in a mode that only reads the records. Prior to this revision, the test program always acquired a write lock when reading the record.

This revision adds an option to perform the reads with no lock, with a read lock, or with a write lock. The following command line options are available:

- *-or* - Read records with no lock.
- *-orl* - Read with a read lock.
- *-orL* - Read with a write lock (this was the behavior prior to this update).

When these options are used, the test program displays the type of locking that is used:

- Read with no lock
- Read with read lock
- Read with write lock

8.12 dbdump - c-treeACE Dump Utility

dbdump speed enhanced using batch operations

The performance of **dbdump** has been improved by modifying the logic to allow batch reads.

The number of records to fetch per call is controlled by the *-z* switch. To be consistent with the switch of **dbload**, it defaults to 1.

The new *-l* switch controls the progress output frequency in terms of record reads (the output is generated whenever the number of record reads since the last output becomes greater than or equal to the specified value).

dbdump -p query passthru support

By default, **dbdump** interprets the query in the command file, converts it to lowercase, and wraps everything that is considered an identifier with double-quotes. If the command file contained functions, **dbdump** was wrapping the functions in double-quotes resulting in a syntax error.

The **dbdump** *-p* command-line switch activates the new query passthru mechanism, which does not perform any major change in the query. When this switch is in use, anything that is not in double-quotes gets converted to lowercase without adding double-quotes.

For example, consider the following statement:

```
SELECT RTRIM(cm_custnumb),...
```

The default behavior of **dbdump** resulted in the following statement, which would cause a syntax error:

```
select "rtrim" ( "cm_custnumb" ) ,...
```

Using the new **dbdump** *-p* switch results in this statement, which does not cause an error:

```
select rtrim ( cm_custnumb ) ,...
```

9. Notable Compatibility Changes

This section lists c-treeACE changes that affect compatibility. It is important to review these issues to ensure that your product functions correctly after upgrading to this version.

9.1 File rebuild memory limit - File memory usage keyword for rebuild memory usage

Support was added for `SORT_MEMORY` to specify the maximum memory to use in a rebuild. The `SORT_MEMORY` keyword specifies a memory size in bytes and can use the MB and GB keywords (unlike `MAX_K_TO_USE`).

The maximum `SORT_MEMORY` value is:

- 4 TB - 1 for 64-bit c-treeACE
- 4 GB - 1 for 32-bit c-treeACE

As the `SORT_MEMORY` option is more intuitive, its use is recommended over MAX_K_TO_USE. (MAX_K_TO_USE remains available for backward compatibility). If both SORT_MEMORY and MAX_K_TO_USE are specified in *ctsrvr.cfg*, only the one that is specified last in the configuration file takes effect.

9.2 Suppress logging File Delete Error for I0000000.FCS during startup

c-treeACE Server no longer logs the following error message to *CTSTATUS.FCS* when it starts, as this is considered a normal condition:

```
Wed Jan  9 10:44:05 2013
- User# 00001  Scanning transaction logs
Wed Jan  9 10:44:05 2013
- User# 00001  idltfil: File <I0000000.FCS> unlink failed with error={2}
```

9.3 ODBC SQLColAttributes renamed

SQLColAttributes() is a deprecated ODBC 2.0 function replaced by **SQLColAttribute()** in ODBC 3.0. Internally, our ODBC driver used a function named **SQLColAttributes** to implement **SQLColAttribute()**. On Unix, this function (and all global functions) were exported by our *libctodbc.so*. When an ODBC 2.0 application called **SQLColAttributes()**, the driver manager was supposed to convert this to **SQLColAttribute()** for an ODBC 3.0 driver, but unixODBC used **SQLColAttributes()** if it existed.

This revision renames our **SQLColAttributes()** to **iSQLColAttributes()** to avoid this confusion.

Note: On Windows, **SQLColAttributes** was not exported by the ODBC driver.

9.4 ctOpenSequence() returns NO_ERROR when specified sequence does not exist

A call to the c-treeACE ISAM API **ctOpenSequence()** specifying the name of a sequence that does not exist returns **NO_ERROR**. The logic has been corrected so that such a call now returns **SEQNAM_ERR** (as documented).

9.5 ctSETENCRYPT - Passing a NULL to disable encryption

Passing a NULL value for the *key* parameter to ctSETENCRYPT was inadvertently resetting *keylen* to 0, which was disabling encryption. In V10.3 and later, the code in the client library interface to **ctSETENCRYPT()** has been changed so that it is consistent with the following definition:

In the **ctSETENCRYPT()** function, *keylen* > 0 enables encryption for all files created after that point, and *keylen* <= 0 disables encryption. This is true regardless of whether *key* is NULL or not.

Note: A non-NULL *key* is *required* when a call to **ctSETENCRYPT()** is made that enables basic encryption; *key* is *ignored* when using Advanced Encryption.

9.6 Rebuild Fails with Error 484 (Could Not Open Sort Work File)

When `COMPRESS_FILE` was specified in *ctsrvr.cfg* with a filename specification that matched rebuild sort work file names (for example `COMPRESS_FILE *.*` to match all filenames), a rebuild failed with **error 484** (could not open sort work file).

NOTE: **Error 401** could occur when opening any data file (not just a rebuild sort work file) that is created with resource support disabled and whose name matches the `COMPRESS_FILE` filename specification.

The logic has been changed so that the `COMPRESS_FILE` keyword does not enable data compression for a data file that is created with resource support disabled.

9.7 Treat fixed-length compressed data files consistently across batch record returns, inserts, and updates

Prior to c-treeACE V10.3, a fixed-length data file that had record compression turned on was implemented internally as a variable-length file with the **ctAugmentedFxd** file attribute. The **ctAugmentedFxd** attribute enforced the constraint that all the records are of the specified fixed length. After compression, the records may be of different lengths. To determine if your file is behaving in this manner, you can execute the **ctinfo** utility over the data file and look for the presence of the **ctAugmentedFxd** file mode in the c-tree extended header block.

ctBEHAV_BAT_FXDCMP changes the behavior of the batch retrieval and insertion code so that compressed, fixed-length data file records are now treated as fixed-length records, not variable-length. This new behavior is consistent with other c-tree API calls and the BAT_UPD option. **ctBEHAV_BAT_FXDCMP** is on by default, but `COMPATIBILITY NO_BAT_FXDCMP` placed in the *ctsrvr.cfg* file will turn off the new behavior at runtime.

The new behavior affects BAT_RET_REC and BAT_RET_BLK retrieval operations by removing the record size field that precedes each record image.

BAT_RET_FIX is mutually exclusive with BAT_RET_REC, and designed to be used with variable-length files, and not used in conjunction with BAT_RET_BLK.

Before the new behavior, BAT_RET_FIX would have resulted in compressed, fixed-length data file records being returned like variable-length records. Now they will be returned like fixed-length records.

The new behavior affects the BAT_INS option by expecting the input buffer to be in fixed-record length format for compressed fixed-length data file records.

9.8 Unix Shared Memory Protocol Not Freeing Shared Memory Segments (different client and server user accounts)

When clients used the shared memory protocol on Unix systems and the client and server processes were run under different user accounts, shared memory segments could be left behind after the connections were closed. The **ipcs** *-m* listing showed shared memory segments with no processes attached. In V10.3 and later, the logic has been modified to correct this.

Note: These changes add a field to the client and server logon data structures to pass the user ID between the client and server processes, resulting in the following compatibility considerations:

An old client can connect to a new server and it will behave as it did before these changes.

A new client cannot connect to an old server using shared memory. It will receive **error 133** if the server is configured to only use shared memory. It will connect with TCP/IP if the server is using both shared memory and TCP/IP. The old server will log the following message to *CTSTATUS.FCS*:

```
FSHAREMM: The client's shared memory version (2) is not compatible with the
server's shared memory version (1)
```

At startup, the FairCom Server now logs messages to *CTSTATUS.FCS* to indicate the shared memory directory used for logon purposes and the shared memory protocol version that it is using:

```
FSHAREMM: SHMEM_DIRECTORY=/tmp/ctreedbs/
FSHAREMM: Protocol version=2
```

9.9 Error codes adjusted for uniqueness

We now enforce the convention that all error codes have unique absolute values. A small number of positive error code values were shared with a negative value. For example, **FNOP_COD(-10)** and **SPAC_ERR(10)** share the absolute value of 10.

Changing to unique error codes caused the following negative sysiocod values to change:

Symbolic constant	Old value	New value
FNOP_COD	-10	-975
FNOR_COD	-11	-976
PRTL_COD	-970	-972
PRTL_FF	-971	-973
LWRT_COD	-972	-974

None of the positive error codes were changed.

Applications checking sysiocod using symbolic constants need to be re-complied to map to the appropriate value. Applications checking sysiocod using error values need to be updated, where necessary, to the new error codes.

9.10 c-treeACE SQL option to force previous $DELFLD$ size

In c-treeACE SQL V9.5 the FairCom DB API $DELFLD$ size changed from 4 bytes (or 5 bytes in case of non-HUGE files) to 9 bytes. A configuration keyword has been added allowing c-treeACE SQL CREATE TABLE and ALTER TABLE commands to use the prior $DELFLD$ size to avoid re-coding any C structures that describe the record.

See **c-treeACE SQL option to force previous $DELFLD$ size** (page 36).

9.11 c-treeACE SQL shutdown when Callback Library loading fails

When `SQL_OPTION LOAD_CALLBACK_LIB` was specified, and callback loading failed, the server no longer continued regular operations without any notice.

See **c-treeACE SQL shutdown when Callback Library loading fails (page 36) for more details.**

9.12 ctrbldif, ctcmpcif, and ctinfo - Updates to handling of security attributes

Updates have been made to the way in which the **ctrbldif**, **ctcmpcif**, and **ctinfo** utilities handle security attributes.

See **Updates in ctrbldif, ctcmpcif, and ctinfo handling of security attributes** (page 62).

Copyright Notice

Trademarks

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

10. Index

R

S

T

U