test link

c-treeACE V9.1.0

Update Guide

Audience

Developers

Subject

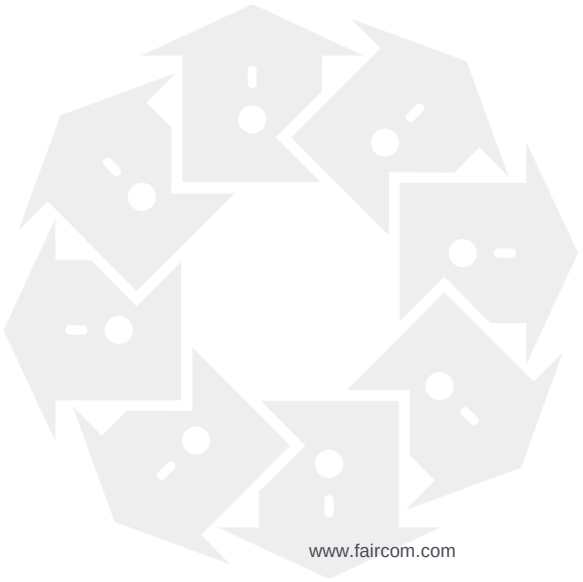
FairCom's high-performance NAV and SQL database technology.

FairCom®

Contents

1. Version V9.1.1 Updates and Changes

Since the release of Version 9.0, there have been additional enhancements, critical production updates, and modifications to c-treeACE that are included in this latest delivery. Please review the information in this document regarding these latest changes.

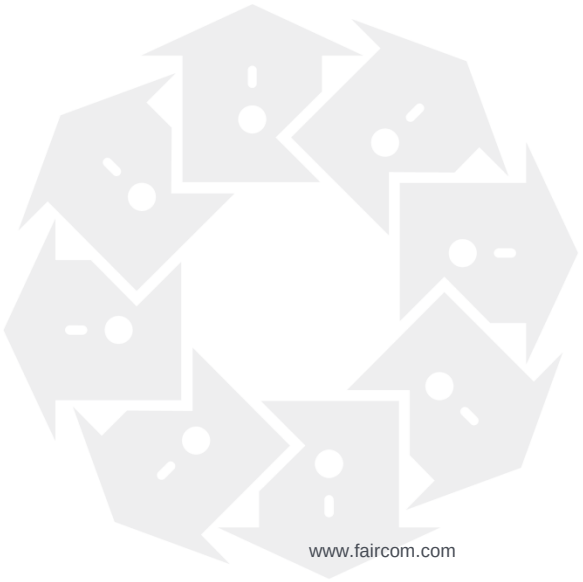


1.1 FairCom Typographical Conventions

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

The following formatted items identify special information.

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



2. Critical Production Updates

The following notifications have been found to be important considerations in many production systems. Please Contact your nearest Faircom office if you are concerned if any of these may impact your application or have any questions.

2.1 Potential Variable Length Data Corruption Prevented During Automatic Recovery

Affected Builds: All c-tree versions with builds prior to 091005
Criteria: Variable Length Files under Transaction Control that have Undergone Automatic Recovery After a Failed Server Process
Indications: Error 37 during recovery and/or evidence of data or index corruption

Sometime after automatic recovery on a system where the c-treeACE Server had previously gone through the process of automatic recovery, data and index files were found to contain unexpected 10-byte headers beginning with 0xfbbf marks. The headers appeared at unexpected locations, sometimes overwriting data records or index nodes, and sometimes written at the physical end of the file, preceded by a region of 0x00 bytes.

A second observed symptom was automatic recovery failing with error 37 due to an invalid file descriptor, as indicated by the following entries in *CTSTATUS.FCS*:

```
- User# 00002 trandat: scanning log 815
Wed Sep 23 15:44:25 2009
- User# 00002 WRITE_ERR: ???? at 0:1457a4ex sysiocod=6 bufsiz=10 bytes written=0[0] ioLoc=0: 37
```

Automatic recovery invalidates the space management index for variable-length data files, and it queues entries for the space reclamation thread to process. These entries trigger the space reclamation thread to reconstruct the space management index by physically scanning the variable-length data files for deleted records and adding keys to the space management index for each deleted region found in the file.

Changes to the space management index can place entries - with associated file numbers - into the transaction log. One such entry was written to the server's preImage space, however, not immediately written to the transaction logs. A later transaction can then commit this entry with a transaction file number not associated with the original file, as the original file could have been physically closed and the file number reassigned to another physical file.

Should the server then experience an abnormal failure and automatic recovery takes place, the entry is processed from the transaction log with a transaction file number associated with the wrong physical file. An attempt is then made to write this entry to the incorrect file causing either data to be overwritten, or a failed write due to an invalid file descriptor.

To prevent this behavior, the transaction log entries are now written immediately and directly to the physical transaction log ensuring a correct associated transaction file number upon recovery.

2.2 Prevent Termination of c-treeACE from LRU Cache Miss Limitations

Affected Builds: All V9 lines with build dates prior to 090602
Criteria: c-treeACE Server with `DATA_LRU_LISTS` or `INDEX_LRU_LISTS` configurations set greater than 1
Indications: Server crash after large number of cache misses

When a c-treeACE Server is running with the configuration options `DATA_LRU_LISTS` or `INDEX_LRU_LISTS` set greater than 1, after a large number (2^32, over 2 billion) data or index cache misses occur, the c-tree Server terminates with an unhandled exception. These options default to 4 in c-treeACE Version 9. Thus all c-treeACE Version 9 servers are susceptible to this situation. You can verify these options in the server startup information found in the server status log file, *CTSTATUS.FCS*.

When a cache page is required to hold a page that is not already in cache and multiple data or index LRU lists are in use, the data and index cache logic increments a variable used to calculate which data or index LRU list is to be used . However, the variable was declared as a signed long integer, and after 2^32 increments, became negative, resulting in a negative index offset for an array of data/index cache LRU list mutexes. Redeclaring the variables as unsigned long integers resolves this issue.

An immediate fix for anyone with potential to be affected by this Version 9.0 only issue is to directly specify the following in your c-treeACE configuration file (*ctsrvr.cfg*) to avoid this unhandled exception condition:

```
DATA_LRU_LISTS 1
INDEX_LRU_LISTS 1
```

You will need to restart your server to enable this configuration change.

2.3 Potential c-tree Server Automatic Recovery Failures with the LOGIDX Feature

Affected Builds: All c-tree versions that enabled this feature through the use of the *LOGIDX* file mode withing the application or through the server configuration keyword `FORCE_LOGIDX YES`.
This feature was enabled by default for c-treeACE V9 Servers with build dates 080111 through 090126.
Criteria: `FORCE_LOGIDX YES` enabled in the c-treeACE Server configuration
Indications: Failed recovery

c-treeACE provides a feature called *LOGIDX* which can allow for quicker automatic recovery time on server startup should you experience an unexpected failure (for example, a power outage). This feature can be enabled with a file mode of *LOGIDX,* or using the `FORCE_LOGIDX` server configuration keyword.

Extensive testing identified an isolated possibility of failure during automatic recovery when the *LOGIDX* logic is active. The net effect is a possibility for failed recovery in the event the c-tree Server was not properly shut down. This has been corrected as of V9.1 build 090522.

A simple solution to ensure you will not be impacted by this default setting is to disable the *LOGIDX* feature via your server configuration file. Follow these easy steps to perform this change:

1. Cleanly shut down your c-tree Server. (For example, use the c-tree Server Administrator utility, **ctadmn**, or the c-tree Server stop utility, **ctstop**.)
2. Add or change the following server configuration keyword in your server configuration file, *ctsrvr.cfg*:

   ```
   FORCE_LOGIDX  OFF
   ```

3. Restart your c-treeACE per your normal operating procedures.

2.4 Correct Handling of Segmented Files During Automatic Recovery

Affected Builds: All c-tree versions with builds prior to 090610
Criteria: Segmented file, Automatic recovery after a failed c-treeACE process
Indications: Renamed file segments

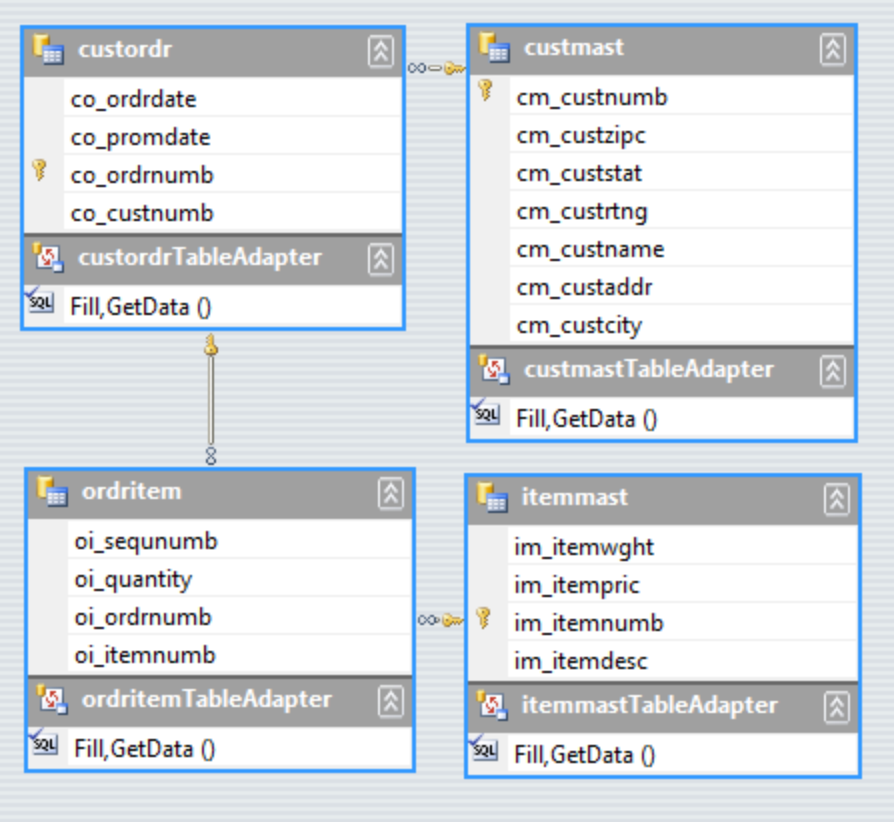
During automatic recovery of a failed c-treeACE process, existing file segments were renamed and thus subsequently not available to the application resulting in apparent missing data. During recovery, an attempt is made to open all segments of a segmented file found in the transaction logs. To open the segments, the segment definition resource must be read from the primary segment. However, an internal file map attribute had not been initialized resulting in a failed call. This failed call caused the recovery process to believe the additional segments did not exist. When later attempting to create the file segment, and that segment already existed, c-treeACE then renamed the segment. The uninitialized attributes are now properly assigned avoiding this potential incorrect renaming of segmented files.

3. Fresh Additions

c-treeACE V9.1 has several significant new features that will be of interest to a wide array of developers and deployed applications.

3.1 ADO.NET Entity Framework

The Entity data model (EDM) specifies a conceptual model of data via an Entity-Relationship data model. Using entity-relationship diagrams (ERDs) a top-down method of modeling can be applied to relational data.



ADO.NET Entity Framework is an object-relational mapping (ORM) framework available for the .NET Framework. The c-treeACE ADO.NET Data Provider now supports the .NET Entity Framework model with a seamless integration into Visual Studio 2008.

Requirements

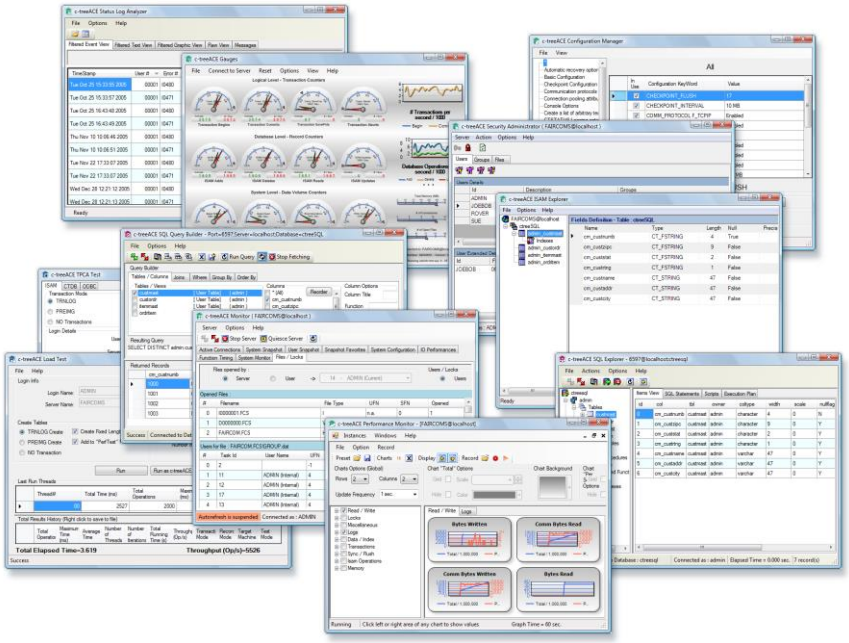
- .NET Framework 3.5 with Service Pack 1

- Visual Studio 2008 with Service Pack 1
- c-treeACE ADO.NET Data Provider V9.1 (Build 090804 or later)

3.2 Expanded Feature Additions to GUI Based Tools

One of the most popular features of c-treeACE V9 was the introduction of the suite of GUI based tools for the development and management of c-tree database technology. V9.1 includes updates and numerous new features for these productivity enhancing tools. The ability to actively view record data is the most visible of these features. Multiple options are now available to scan through your data using Dr.Ctree, ISAM Explorer or SQL Explorer. For developers, exporting file definitions has been expanded and is available in all of these tools. SQL schemas, *IFIL* and *DODA* definitions and complete header files for your applications are easily generated with a click of the mouse.

- Enhanced abilities to view records from files
- Generation of file headers for developers
- Improved presentation and navigation
- History capture for files
- Additional statistics for performance monitoring
- Optional operation modes for advanced maintenance



3.3 Added Platforms and Environments

- AIX 6.0
- Mac OS X 10.6 (Snow Leopard)

- Windows 7
- CodeGear 2009 (Including VCL and c-treeACE SQL dbX Components)
- Sun Studio 12
- c-treeACE SQL for QNX

3.4 Core c-treeACE SQL Performance Enhancements

Increased Query Performance

- Several queries involving JOINS much improved
- Faster queries through new distinct key counts
- Continual improvements to query optimization

Decreased Memory Usage

- Reduced usage in communications buffers
- Reduced usage in fetch ahead buffers
- More efficient use of internal arrays
- Minor memory Leaks in ODBC Drivers repaired

Improved Stability

- Assorted PANIC and unhandled conditions resolved
- Stabilized viewing of execution plans
- Resolved all reported unexpected server terminations

3.5 Distinct Key Count for Improved c-treeACE SQL Query Efficiency

The c-treeACE SQL query optimizer now takes advantage of distinct key counts in an index for improved efficiencies. In some cases, speedups of over 200 fold for selected queries have been reported with this additional index level information. c-treeACE SQL now creates indexes with distinct key count support by default.

3.6 Dynamic Dump Options For Enhanced Performance

When a dynamic dump runs, the disk read and write operations of the backup process can slow the performance of normal database operations. c-treeACE now supports an option that allows an administrator to reduce the performance impact of a dynamic dump.

The c-treeACE configuration option:

- `DYNAMIC_DUMP_DEFER <milliseconds>`

sets a time in milliseconds that the dynamic dump thread will sleep after each write of a 64KB block of data to the dump backup file.

An application developer can also use the c-tree **ctSETCFG()** API function to set the *DYNAMIC_DUMP_DEFER* value. For example, the following call specifies a 10-millisecond DYNAMIC_DUMP_DEFER time:

- **ctSETCFG(** *setcfgDYNAMIC_DUMP_DEFER, "10"* **);**

The *DYNAMIC_DUMP_DEFER* value set by a call to **ctSETCFG()** takes effect immediately, so this API call can be used by administrators to adjust the speed of a running dynamic dump depending on the amount of other database activity.

Note: The maximum allowed *DYNAMIC_DUMP_DEFER* time is 5000 milliseconds, set at compile-time. If a value is specified that exceeds this limit, the *DYNAMIC_DUMP_DEFER* time is set to *DYNAMIC_DUMP_DEFER_MAX*.

The c-treeACE Administrator utility, **ctadmn**, was also updated to support the dump sleep time option to change this value at run time. The "Change Server Settings" menu is available from the main menu of the **ctadmn** utility and this menu supports all of the following options:

- **Configure function monitor**
- **Configure checkpoint monitor**
- **Configure memory monitor**
- **Configure request time monitor**
- **Change dynamic dump sleep time**

3.7 Efficient Transaction Log Template Copies

The log template feature is a fast and efficient means of creating transaction logs in high volume systems. An initial transaction log template is created and copied when a new transaction log is required.

The original implementation used an operating system file copy command (for example., cp *L0000002.FCT L0000002.FCS*) to initiate the copy of the template to the newly named file. This approach required the full contents of the template file to be read. For systems experiencing high volume transaction loads where a template is frequently copied, this method placed unnecessary demand on system resources.

An improved efficient method for copying transaction log template file is available.

Template Copy Options

The following configuration options can be used to modify the speed of copying a log template such that log template disk write performance impact is reduced.

Copy Sleep Time

```
LOG_TEMPLATE_COPY_SLEEP_TIME <milliseconds>
```

This keyword results in the copying of the log template to be paused for the specified number of milliseconds each time it has written the percentage of data specified by the `LOG_TEMPLATE_COPY_SLEEP_PCT` option to the target transaction log file.

- Default value: 0 (disabled)
- Minimum value: 1
- Maximum value: 1000 (1 second sleep)

Copy Sleep Percentage

```
LOG_TEMPLATE_COPY_SLEEP_PCT <percent>
```

This keyword specifies the percentage of data that is written to the target transaction log file after which the copy operation sleeps for the number of milliseconds specified for the `LOG_TEMPLATE_COPY_SLEEP_TIME` option.

- Default value: 15
- Minimum value: 1
- Maximum value: 99

Example

The following example demonstrates the options that cause the copying of the log template file to sleep for 5 milliseconds after every 20% of the transaction log template file has been copied:

```
LOG_TEMPLATE_COPY_SLEEP_TIME 5
LOG_TEMPLATE_COPY_SLEEP_PCT 20
```

Note: If an error occurs using this method an error message is output to *CTSTATUS.FCS* (identified with the "LOG_TEMPLATE_COPY: ..." prefix) and c-treeACE then attempts the log template system copy method.

3.8 CPU Affinity Check for IBM AIX Operating Systems

c-treeACE now performs a CPU affinity check for IBM AIX operating systems. While c-treeACE can now detect the number of CPUs in an AIX system, the server configuration option, `CPU_AFFINITY`, used to set the CPU affinity for the c-treeACE process, has not been implemented for this operating system at this time. Should you need to configure c-treeACE to a particular AIX CPU resource, please consult your local system administrator for options on restricting processes to specific CPU resources.

3.9 Improved File Compaction Behavior for 6-Byte Transaction Enabled Files

Introduction

Prior to V9.1, the compact IFIL and rebuild IFIL API functions (**CMPIFILX()** and **RBLDIFILX()**) did not make assumptions about 6-byte transaction numbers for indexes associated with 6-byte transaction enabled files. As a result, regardless of if the data file was 6-byte transaction number enabled, when new indexes were recreated they lost the *ct6BTRAN* attribute. This caused errors when the server configuration specified 6-byte transaction enabled files only (for example, error **R6BT_ERR**, 745). It could also lead to transaction number overflow errors (**OTRN_ERR**, 534) on high volume systems when 4-byte transaction numbers quickly become exhausted.

When the underlying data file contains an extended header, it is more appropriate for the extended rebuild and compact functions to use 6-byte transaction numbers for the associated indices by default. As a result, any data file containing an extended header that is compacted or rebuilt, will now have have the 6-byte transaction number attribute enabled by default.

New CMPIFILX8 API

The **CMPIFILX()** extended API call does not facilitate an array of *XCREblk*s to be passed in. When the indices associated with the data file to be compacted exist, then compact generally does not need these *XCREblk*'s used when the indices were created. However, if one or more of the indices does not exist, then **CMPIFILX()** recreated a *XCREblk* from the data file. But this does not account for differences between the original data file *XCREblk* and the original index file *XCREblk*.

An extended header version of **CMPIFILX()** has been introduced as a step to alleviate this situation.

This new API call declaration is:

```
ctCONV COUNT ctDECL CMPIFILX8(pIFIL ifilptr, pTEXT dataextn, pTEXT indxextn,LONG permmask, pTEXT groupid,
pTEXT fileword, pXCREblk pxcreblk)
```

Secondly, when either **CMPIFILX()** or **CMPIFILX8()** are run on the server (as opposed to stand-alone) and the server configuration option `COMPATIBILITY EXTENDED_TRAN_ONLY` is specified, then the *ct6BTRAN* mode bit is turned on in the recreated *XCREblk* or the explicit *XCREblk*, respectively, to ensure 6-byte transaction number support.

It should be noted that since **CMPIFILX()** reconstructs an *XCREblk* for the index from the data file, there may be attributes of the data file that were not part of the original index file definition. For example, if the data file supported a large extent size, then the index inherits this large extent size regardless of how it was originally created.

If the configuration option, `COMPATIBILITY 6BTRAN_NOT_DEFAULT`, is specified, the *XCREblk* array passed into **RBLIFILX8()** and **CMPIFILX8()** can still explicitly set the *ct6BTRAN* attribute bit. Likewise, if *ctNO6BTRAN* is passed in explicitly in calls to **CMPIFILX8()** and **RBLIFILX8()** then the default behavior is overridden, and *ct6BTRAN* will not be turned on.

Compact File Utility ctcmpcif

`#define USEXCREBLK` and `#define LOCALGETXCREBLK` have been added to the compact *IFIL* utility source code (*cmpifil.c*, as in *ctrbldif.c,*) to control definition of the data and index *XCREblks* using the new **CMPIFILX8()** API call. This option is off by default. Uncomment this #define should you desire this functionality in the compact *IFIL* utility.

4. Notable Resolved Issues

c-treeACE V9.1 contains minor modifications that address items found in continued testing as well as several reported issues since the V9 release.

4.1 Resolved 2GB c-tree File Limitations on Linux

It was found that c-treeACE failed to read or write a c-tree file at an offset greater than 2 GB on Linux systems. An internal c-tree disk I/O function had been previously modified on Unix systems to natively support 64-bit systems. Adding the proper compiler flags when building on Linux kernels 2.4 and later resolves this 2GB file size limitation.

Note: This 2GB file limitation affected only c-treeACE 32-bit Linux Servers.

4.2 Improved Performance of Delete Node Queue Management

A very long server shutdown was noticed in a particular environment due to delete node processing. Two areas of server operation were enhanced to minimize this time spent in delete node processing.

Delete Node Queue Entries

Previously, the delete node queue exercised the following pseudo code for each queue entry:

- read queue
- open file
- begin transaction (for *TRANPROC* files)
- prune tree
- commit transaction
- close file

This basic loop was modified in two ways:

1. An attempt is made to peek at the next queue entry to determine if the next tree pruning will be for the same index file (the host index or one of its members).
2. If the files match, skip the commit/begin operations and skip the close/open operations. More precisely, the number of skipped transaction commits and skipped file closures are actually tracked and an artificially imposed limit is set on the number of skips to avoid very long transactions and keeping the file open in shared mode for an extensive period.

By default, *NODEQ_TRNLEN* is set to ten (10), and *NODEQ_OPNLEN* is set to twenty (20). Once shutdown begins, these limits are increased by a factor of ten.

Limited testing shows that shutdown processing of the delete queue runs at least twice as fast with this new behavior.

Note: These changes have little effect if the queue entries are widely varied across different files.

c-tree Server Shutdown Processing

The c-tree Server now supports a configurable limit on the number of delete node queue entries that the c-tree Server processes when it is shutting down. If the option `DNODEQ_SHUTDOWN_LIMIT` is specified in the c-tree Server configuration file, then when the c-tree Server shuts down, if there are more than the specified number of entries in the delete node queue, the delete node thread writes all the unique queue entries to a disk stream file named *DNODEQUE.FCS*. A memory-based index file is used to eliminate duplicate queue entries, and only the unique entries are written to disk. If the number of unique entries is less than `DNODEQ_SHUTDOWN_LIMIT`, those entries are returned to the delete node queue and are processed by the delete node thread before the c-tree Server completes the shut down.

`DNODEQ_SHUTDOWN_LIMIT 0` causes the c-tree Server to process all entries in the delete node queue when shutting down.

The c-tree Server always attempts to open and read all entries from the file *DNODEQUE.FCS* into the delete node queue at startup, regardless of the `DNODEQ_SHUTDOWN_LIMIT` setting. The c-tree Server deletes the file *DNODEQUE.FCS* after populating the delete node queue with its contents. An administrator can delete the file *DNODEQUE.FCS* before starting the c-tree Server to avoid processing these persistent delete node queue entries.

4.3 Improved Thread Safety of System Time Calls

A review of **localtime()** behavior has determined this call was not thread safe on all operating systems. Use of this function, and related functions, has been modified. In the case of **localtime()**, **localtime_r()** is now used, and similarly with other functions included from the system time libraries.

Note: This system call was also included in the default SQL Types SDK custom callback libraries found in the *ctsqlcbk.c* module utilized by some customers. This module has been modified as well when available in our code repositories.

4.4 Windows Resource Error (1450) Configurable Retry Logic

When the Windows kernel has allocated all of its paged-pool memory, it will not be able to perform many tasks and instead returns a STATUS_INSUFFICIENT_RESOURCES (0xC000009A) message. This is a restriction of 32-bit addressing (only 2GB addressable within the kernel), regardless of the amount of memory available in the system.

When the FairCom Server configuration option `IO_ERROR_BLOCK_SIZE` option is specified in the FairCom Server configuration file, a read or write operation that fails with Windows system error 1450 (ERROR_NO_SYSTEM_RESOURCES) is retried in blocks of the specified size. If any one of those read or write operations fails, the FairCom Server fails the read or write operation.

The FairCom Server supports two additional configuration options that permit additional disk read/write retries and a sleep interval between retries.

`IO_ERROR_BLOCK_RETRY <retries>` specifies the maximum number of failed IO_ERROR_BLOCK_SIZE-sized I/O operations that must occur before the I/O operation is considered to have failed. If the IO_ERROR_BLOCK_SIZE-sized I/O operations that are being attempted for a particular I/O operation fail more than *<retries>* times, the FairCom Server writes a **READ_ERR** (36) or **WRITE_ERR** (37) message to *CTSTATUS.FCS* and considers the I/O operation to have failed.

A value of -1 signifies infinite retries. The default is 0, which means that the I/O operation is tried only once in IO_ERROR_BLOCK_SIZE-sized blocks, and if any of these I/O operations fails, the entire I/O operation is considered to have failed. As another example, if IO_ERROR_BLOCK_RETRY is set to 20 and IO_ERROR_BLOCK_SIZE is set to 65536, if a 327680-byte write is retried as 5 65536-byte write operations, then the I/O operation fails if there are 20 failures to perform those 5 write operations.

`IO_ERROR_BLOCK_SLEEP <time>` specifies a time in milliseconds between retry attempts. The default is zero, which means that retries are attempted immediately.

SNAPSHOT Monitoring of Failed Retires

To permit monitoring the number of I/O error 1450 retries that have occurred, a counter has been added to the system snapshot structure. The *sctioblkretry* field of the *ctGSMS* structure is defined as an unsigned long integer that stores the total number of I/O error 1450 retries that have occurred since the FairCom Server started. The snapshot log file *SNAPSHOT.FCS* displays the I/O error 1450 retry counter value with a description of "I/O ERR(1450) automatic retries:". The system snapshot structure version has been changed from 9 to 10 to note the presence of this new field in the structure and the statistics monitoring utility, **ctstat**, and **ctsnpr** utilities have been updated to properly handle the presence of this field in the system snapshot structure and snapshot log.

4.5 Allow c-treeACE V9 Conditional Expression Parser Field Names to Match Data Type Names

It was found that c-treeACE V9 failed to parse some conditional expressions that were allowed in the c-tree Plus V8 SDK. In V9, the expression parser for conditional indexes has been moved into the FairCom DB API layer and that logic includes support for a *CAST* function:

```
F_CAST '(' Expression F_AS ATYPE ')'
F_CAST '(' Expression ',' ATYPE ')'
```

ATYPE is a data type which can be any of the type names from the *symtab* list defined in *ctdbcrun.c*. This new logic could cause some prior expressions to fail. For example, one data type is named *NUMBER* and when used in an expression such as:

```
strlen(Number) > 0
```

c-treeACE V9 considers "Number" as a data type and not as a field name, and fails to parse the expression.

To support previous expressions, the grammar rules for the cast function have been modified to use the symbolic *IDENTIFIER* instead of *ATYPE*. Now, in the rules for the cast function, after finding an *IDENTIFIER*, a check is made if the *IDENTIFIER* is the name of a data type. If not, an error is returned. Otherwise, the data type is added to the parse tree and expression parsing continues.

4.6 Correct Conditional Index Results with Zero-Length String Literals

An index specified the conditional index expression:

```
 !strnicmp(myField, "", 6)
```

and the data file contains records having values of *myField* that match this condition (that is, *myField* is ""). However, for c-treeACE V9, the expression did not evaluate as true for these field values. The same conditional expression produced the expected results in c-tree Plus V8. The V9 conditional expression logic parses zero-length (empty) string literals differently than V8. A NULL pointer was placed into the expression tree rather than allocating space to hold an empty string and adding that value into the expression tree. The string comparison function returns a non-zero value when only one of its parameters (*Source* or *Dest*) is NULL, and caused the expression to return unexpected results in this situation.

The function that adds a string literal value to the expression tree was modified to allocate a one-byte buffer when the string length is zero. Similar logic was also reviewed for other cases where an empty string literal value is not properly handled and these other functions were modified accordingly.

4.7 PreImage Memory Files are no Longer Promoted to TRANLOG files During a Dynamic Dump

The c-treeACE `PREIMAGE_DUMP YES` configuration option promotes *PREIMG* files to *TRANLOG* files during a dynamic dump. An unintended side effect of this option was that *PREIMG* memory files were also promoted to *TRANLOG* files. As a result, memory index nodes could be placed onto the list of updated buffers for transaction controlled files. As these buffers are never flushed to disk (because memory files are never written to disk) these entries resulted in the number of active transaction logs to increase, as c-treeACE keeps all transaction logs beginning with the transaction log in which the transaction on the memory file promoted to the *TRANLOG* mode occurred.

When a dynamic dump is performed with the `PREIMAGE_DUMP YES` configuration option specified in *ctsrvr.cfg*, and *PREIMG* memory files are open, c-treeACE could exhibit any of the following symptoms:

- The number of transaction logs may increase, with *CTSTATUS.FCS* showing a reason of "Cache/Buffer Pending Flush"
- Creating a memory data file during a dynamic dump fails with error **ITIM_ERR** (160)
- c-treeACE may terminate during a checkpoint
- Automatic recovery fails with error 12 attempting to open a non-existent memory data file

Dynamic dump logic has been modified such that *PREIMG* memory files are no longer promoted to *TRANLOG* files to prevent this situation.

4.8 LOG_ENCRYPT Option Now Correctly Supported with Advanced Encryption

c-treeACE failed to start with a log incompatibility error **LFRM_ERR** (666) when both the log encryption option `LOG_ENCRYPT` and advanced encryption (`ADVANCED_ENCRYPTION YES`) options were in effect. c-treeACE would even fail to start in a directory with no existing transaction log files.

Comprehensive checks during c-treeACE startup are now done to ensure that log encryption settings are appropriate for the type of encryption for existing transaction logs. When an incompatibility is found, c-treeACE startup fails with an error message in *CTSTATUS.FCS* indicating which options must be changed to access the transaction log files.

The table below shows the expected results for the possible combinations of Log Encryption and Advanced Encryption for each possible transaction log encryption type used by an existing log file.

Log Encryption Option	**Advanced Encryption Option**	**Log Encryption Type**	**Expected result**
N	N	None	OK
N	Y	None	OK
Y	N	None	OK, new logs are scrambled with *CAMO** technology
Y	Y	None	OK, new logs encrypted with advanced encryption
N	N	*CAMO**	Error: Enable log encryption to proceed
N	Y	*CAMO**	Error: Enable log encryption and disable advanced encryption to proceed
Y	N	*CAMO**	OK
Y	Y	*CAMO**	Error: Disable advanced encryption to proceed
N	N	Advanced	Error: Enable log encryption and advanced encryption to proceed
N	Y	Advanced	Error: Enable log encryption to proceed

Log Encryption Option	Advanced Encryption Option	Log Encryption Type	Expected result
Y	N	Advanced	Error: Enable advanced encryption to proceed
Y	Y	Advanced	OK

* 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.

Copyright Notice

Trademarks

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

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

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

Open Source Components

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

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

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

2/5/2025

5. Index