test link

c-treeACE

V11.5 Update Guide

Audience

Developers

Subject

Updates to the FairCom DB Database Technology

FairCom®

Contents

1. Introduction to c-treeACE V11.5

We are excited about the new features we are introducing in the latest release of c-treeACE: Version 11.5.

The FairCom team is constantly evaluating ways to advance the state of the art of the industry's original multimodel database, c-treeACE. This release is the result of extensive research and development of a useful set of new features.

The rich set of powerful new features making their debut in this release includes advances in such areas as replication, schema management, search, and more:

- **Performance Gains** - As a company founded by—and run by—developers, FairCom is keenly aware of the need to speed up your applications. This release demonstrates performance gains throughout the product.
- **Hot Alter Table** - Allows you to make changes to a table's schema on-the-fly for minimum downtime rolling out application upgrades.
- **Full-Text Search** - This powerful new tool for indexing and searching textual data provides plain-text search for applications that process large volumes of textual data.
- **Replication User-Defined Extensions** - Extend Replication Agent flexibility and customization by calling custom functions so you can automatically perform operations such as extract, transform, & load (ETL), data aggregation, and conflict resolution.
- **Replication Resync** - Resynchronize a target "replica file" based on the current "source file" with minimum impact in runtime. Can be utilized to address occasional situations when a loss of connection causes a target file to become out-of-sync with the source.

- **Transport Layer Security** - TLS is now available through ADO.NET and JDBC, allowing applications to secure data in transit between network clients and c-treeACE servers.
- **Index Range Support for Segment Groups** - This new feature provides a new range operator that can perform retrievals on key ranges defined over a set of contiguous key segments called a "segment group."
- **c-treeACE PHP 7 PDO** - c-treeACE now includes native support for the PHP 7 PDO (PHP Data Object) extension by including a database-specific PDO driver.
- **Data Integrity** - A substantial effort is always aimed at maintaining the utmost standards in data integrity. This release consolidates these efforts throughout the product.
- **More New Features in V11.5** - c-treeACE V11.5 is packed with new features throughout the product: Security, Enhancements to the c-treeACE Server and the Core Engine, SQL and NoSQL APIs, Extended Features, Configuration, and Administrative Tools.

Notable Compatibility Changes - This section lists important c-treeACE known compatibility changes from prior releases. It is important to review these issues to ensure your application continues to function correctly after upgrading.

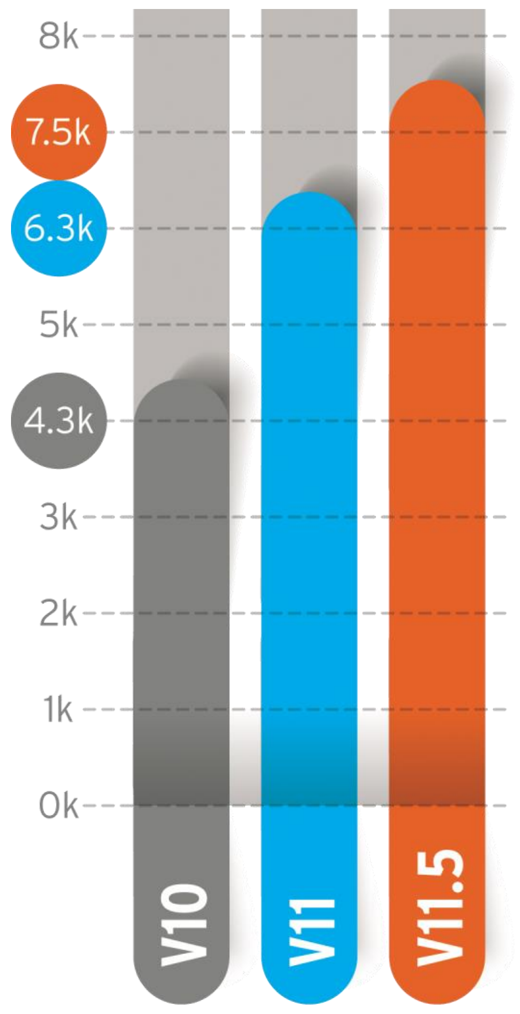
Release Notes - Please also review the *V11.5 Release Notes* document, which has information on fixes included in this release, including critical fixes.

We hope you are as excited about these new features as we are.

2. c-treeACE V11.5 at a Glance

3. Performance Gains

Accelerate your applications with FairCom's continued performance improvements. New gains are especially apparent when scaling across multiple users and large data stores.



Performs nearly 20% better than V11.0!

Performs nearly 75% better than V10.0!

Transaction Processing Add, Read, Delete Test: Transactions per second

This graph shows performance results with full transaction processing (ctTRNLOG file mode) on all data files with the default c-treeACE Server Configuration File (ctsrvr.cfg) settings for each release.

Performance testing was conducted using FairCom's cttctx load test utility to simulate a record add, read, delete sequence on 23 files with indexes. All tests were executed on a Dell PowerEdge R710 with 2 Xeon 3.6 GHz Processors with a total of 32 logical cores, 32 GB RAM, Seagate ST3600057SS SCSI 15K RPM hard drive with Windows Server 2012.

3.1 Instant Improvements

The developments listed below and in the c-treeACE SQL *(page 8) &* FairCom DB API *(page 9) sections allow you to enhance performance without rewriting your c-treeACE application. You won't need to change anything: simply upgrade and enjoy seamless performance improvements.*

Operating System cache flush suppressed for non-TRANLOG files

By default, c-treeACE now suppresses calls to flush updated file system cache pages to disk for all non-TRNLOG data and index files, except during a call to **CTFLUSH()**. This is a performance enhancement for non-transaction and PREIMG files.

Note that this change may introduce a vulnerability to possible data loss in the event of a system crash if updates to a c-treeACE data or index file have been written to the file system cache but not yet written to disk. This can be mitigated by installing a battery backup (UPS) on the system so the file-system cache can still be flushed in case of a power failure. Note that this does not introduce any vulnerability involving abnormal termination of the c-treeACE Server process, as it only affects file system caching, not c-tree caching behavior.

c-treeACE Server supports a configuration option, `NONTRAN_FILESYS_FLUSH_OFF`, that accepts two values, `YES` and `NO`. `YES` suppresses the flush calls, and `NO` enables the flush calls (the default behavior prior to c-treeACE V11.2.3). This option can also be changed at runtime. To disable this option in standalone mode, recompile the standalone library with `#define NO_ctBEHAV_NONTRAN_FILESYS_FLUSH_OFF`.

Note: This is a Compatibility Change for V11 users.

Background file flush threads optimized

Performance has been enhanced by optimizing checkpoint and c-treeACE transaction log flushing when the background flush threads fire, as controlled by the following keywords:

```
NONTRAN_DATA_FLUSH_SEC
NONTRAN_INDEX_FLUSH_SEC
TRAN_DATA_FLUSH_SEC
TRAN_INDEX_FLUSH_SEC
```

There are no side effects to this change—only performance gains. Additional details about this change are available in *Features You Can Use to Boost Performance* (page 10).

Improved Auto Restore Point performance with checkpoints

An Automatic Restore Point that is configured to log a checkpoint sometimes suspends server operation for an unexpectedly long time. When `CHECKPOINT_MONITOR DETAILS` is enabled in *ctsrvr.cfg*, the *CTSTATUS.FCS* log entries show that the time is spent in the initial phase of the checkpoint that includes flushing updated buffers. The logic has been revised to greatly improve the efficiency of this operation.

Performance Improvement - Transaction commit of many files

A transaction commit required more time when many files were open regardless of how many of those files were updated (10 microseconds vs. 3 microseconds in one of our test cases, comparing 1000 extra file opens to no extra file opens). The performance of the transaction commit has been improved in this situation by enhancing the logic that determines which files require processing.

Update list synchronization object management improvement

The handling of mutexes, checkpoints, and calls to **CTFLUSH(-1)** has been improved for increased performance. The CPU is now released before reacquiring the mutex to help give priority to other threads that want to acquire the mutex. This change was applied to all such loops, both over updated index buffers and updated data pages.

Reduced overhead in password file decryption

The function that decrypts file passwords sent by the client has been optimized to reduce overhead.

Rebuild performance increased

Rebuild performance has been increased by enhancing the logic that controls the flushing of files. We are now able to skip certain file system flush calls for sort work files to improve rebuild performance.

V11.5 includes a variety of new features you can use to fine-tune rebuild performance. See *Numerous index rebuild options enhance performance* (page 12).

Rebuild optimized when an index already exists

When a rebuild is performed on an index that already exists, the index is scanned to check for resources. This scan has been changed from reading the entire index file to following the resource chain and saving the resources into a temporary file, then recreating the index file and adding the resources to the new index file (as is done by the file compact function). This speeds up the index scan and it can reduce the size of the index file if many unused nodes exist in the original index file.

Speed improvement in forward roll

The logic in a forward roll has been analyzed and optimized. A forward roll involving over a thousand transaction logs previously took several hours to complete. The optimized logic yields performance improvements of several hundred percent for one set of transaction logs tested by our QA team.

3.2 c-treeACE SQL Instant Performance Enhancements

SQL performance enhancement on Insert

Research by our development team indicated that Insert operations could be slower when the current database contained multiple tables with IDENTITY fields and a new IDENTITY field was added in unrelated tables. The logic has been modified to improve performance in these situations.

Performance improvement in parameterized query

Performance has been improved for a parameterized query that required a very long time to execute compared to the same query with literal values. The optimization logic has been improved to execute the parameterized query as efficiently as the non-parameterized version.

3.3 FairCom DB API Instant Performance Enhancements

FairCom DB API performance enhanced

The performance of FairCom DB API has been enhanced by streamlining logic in certain operations, such as Open, Create (temp tables), and Alter. Some of the checking that FairCom DB API performs has been optimized to remove checks that were already performed by other functions. After a thorough analysis of the logic, the redundant checking has been removed.

Performance enhancement using ctdbFindRecord on variable-length records

The **ctdbFindRecord** call is a mainstay of programing using the c-treeACE database. Our engineering team has been able to achieve a performance enhancement in this function by optimizing the logic when it is called on variable-length records.

Performance improvements in ctdbOpenTable

While opening a Unicode table, the logic looks for KSEG definitions at the table level, index level, and segment level. For a table with multiple indexes and multiple segments per index, this results in a large number of client/server calls. The logic has been refined to reduce the number of calls required.

The *kseg** element names in the CTDBTABLE and CTDBINDX structures have been changed to `unicode_kseg*` to clarify their meaning. It is not expected for applications to be accessing these member elements directly, so this should be a benign change.

Note: This is a Compatibility Change when accessing these members directly using FairCom DB API and Unicode.

3.4 Performance-Boosting Features

FairCom understands developers are obsessed with application performance. In c-treeACE V11.5, we introduce the following enhancements you can use to boost the performance of your application.

*_FLUSH_SEC background flush rate reduces contention

In c-treeACE V11.0 and later, the following background flush thread configuration options remove the dirty cache page flushing overhead from the checkpoint logic, and place this important requirement on these new dedicated background threads:

```
NONTRAN_DATA_FLUSH_SEC
NONTRAN_INDEX_FLUSH_SEC
TRAN_DATA_FLUSH_SEC
TRAN_INDEX_FLUSH_SEC
```

This change reduces the time that other threads are blocked waiting to acquire the update list mutex and has smoothed out performance curves for high-speed systems. If you would like a more thorough refresher on this change, please review this topic in the V11.0 Update Guide: *Controls for Performance AND Safety of Non-Transaction Updates* (https://docs.faircom.com/doc/*v11ace/64961.htm*).

Configuring background flush at a given rate

Starting with c-treeACE V11.5, it's now possible to configure a fixed rate of background flushing of updated data cache pages and index buffers, which provides the ability to further smooth out performance curves, reducing periodic drops in performance. The two parameters that can be specified for the background flushing are:

- *msec* - The time in milliseconds to defer after flushing, and
- *nfls* - The number of flushes after which the defer occurs.

With this new change, the keywords now support specifying these new parameters using the syntax *rate=<msec>/<nfls>*.

For example, to defer for 1 millisecond after every two flushes of updated TRNLOG data cache pages, use this option:

```
TRAN_DATA_FLUSH_SEC rate=1/2
```

The rate value sets a maximum speed at which the background flush threads will write updated buffers to disk. The goal is to flush updated pages quickly enough that the total number of updated buffers does not become extremely large (which could cause the number of transaction logs to increase and could cause checkpoints to take more time to build the list of updated pages and their corresponding transaction numbers and write them to disk), without flushing at too fast a rate, which would impact performance. The *msec* and *nfls* values can be used to set the maximum rate of flushing. FairCom suggests starting with *rate=1/1*, which means delay for one millisecond after every buffer flush. In our testing, we have found 1/1, or 1/2 to be the optimal settings.

DELSTK_COMMIT_SEC improves performance of TRNLOG operations

TRNLOG fixed-length record deletes when using COMPATIBILITY TDATA_WRITETHRU exhibited delays for record Adds and Deletes and file Opens and Closes. The delay was attributable to operations waiting for the file header mutex.

The new DELSTK_COMMIT_SEC server configuration can be used to improve performance of committing record deletes in a fixed-length data file that uses the TDATA_WRITETHRU option. To use this option, specify the following in *ctsrvr.cfg*:

```
DELSTK_COMMIT_SEC nsec
```

When *nsec* is greater than zero, this option causes commits of fixed-length record deletes in files that use the TDATA_WRITETHRU option to write the delete stack value to the file's header at most every *nsec* seconds. This means that, in the event of a server crash, the delete stack value in the header can be out-of-date, not accounting for the last *nsec* seconds of deleted records. The only impact is that some space in the file will not be available for reuse after automatic recovery.

Default: 0 (disabled)

The maximum value that can be specified is 30.

Changing at Runtime

This option can be changed at runtime using either of these approaches:

Using the **ctadmn** utility:

Select menu option 10. Change Server Settings, then select 10. Change the specified configuration option, and then enter the configuration option and its value:

```
delstk_commit_sec 1
```

Calling the **ctSETCFG()** function, for example:

```
ctSETCFG(setcfgCONFIG_OPTION, "DELSTK_COMMIT_SEC 1");
```

Index rebuild options enhance performance

A number of changes were made to the index rebuild logic to improve performance. In addition to the *rebuild performance increase* (page 6) mentioned earlier, several new features allow you to fine-tune rebuild performance:

Option to skip initial data file scan

Normally the rebuild functions **RBLIFIL()**, **RBLIFILX()**, and **RBLIFILX8()** start by scanning the data file, checking for valid record marks if it is a variable-length data file, and checking that the logical and physical end of file values are correct. If the data file is known to be in a good state, it can be beneficial to skip this scan.

The rebuild functions now support an option to skip the initial scan of the data file. To use this option, run the **ctrbldif** utility with the *-skipdatascan* command-line option. If you are calling the rebuild functions, OR the `skipdatascanIFILoption` bit into the `tfilno` field of the IFIL structure that you pass to the rebuild function. When using this and other options, remember to negate the `tfilno` value after you OR in the options. For example:

```
myifil.tfilno = -(redosrlIFILoption | skipdatascanIFILoption);
RBLIFIL(&myifil);
```

ctrbldif progress notifications

The **ctrbldif** utility now supports an option so that it can receive progress notifications from FairCom Server during the rebuild. Use the *-callback* command-line option to enable this feature.

For example, here is a typical rebuild, without the *-callback* parameter:

```
D:\FairCom\v11.2.0\winX64\tools\cmdline\admin\client>ctrbldif ord.dat admin ADMIN FAIRCOMS

c-treeACE(tm) Version 11.2.0.7875(Build-160501) Index rebuild utility

Copyright (C) 1992 - 2016 FairCom Corporation
ALL RIGHTS RESERVED.
Successfully retrieved IFIL.
Scanning data file and rebuilding indexes...
Successfully rebuilt file.
```

Here is the same rebuild with the *-callback* parameter:

```
D:\FairCom\v11.2.0\winX64\tools\cmdline\admin\client>ctrbldif ord.dat -callback admin ADMIN FAIRCOMS

c-treeACE(tm) Version 11.2.0.7875(Build-160501) Index rebuild utility

Copyright (C) 1992 - 2016 FairCom Corporation
ALL RIGHTS RESERVED.

Successfully retrieved IFIL.

Scanning data file and rebuilding indexes...
Rebuilding data file...
Rebuilding index file.
Writing index file...
Rebuilding additional index #1.
Writing index file...
Rebuilding additional index #2.
Writing index file...
Rebuilding additional index #3.
Writing index file...
Rebuilding additional index #4.
Writing index file...
      180000

Successfully rebuilt file.
```

Rebuild queue optimization

FairCom Server supports an option to use a worker thread to sort the keys read from the data file. The rebuild thread reads records from the data file and constructs key values, which it passes to the sort thread using an inter-thread queue. After all the key values have been passed to the sort thread, the sort thread sorts the key values and passes them back to the rebuild thread in a second queue.

This option is enabled by specifying `MAX_REBUILD_QUEUE <max_queue_size>` in *ctsrvr.cfg* with a positive value. The value is a number in bytes indicating the maximum size of the queue used to pass the keys to the worker thread. `MAX_REBUILD_QUEUE` defaults to 100 MB. A 64-bit server has a limit of 4 TB - 1 for this setting, and a 32-bit server has a limit of 4 GB - 1 for this setting.

Rebuild performance

The performance of the original and optimized rebuild have been compared using a 17 GB copy of the *mark.dat* and *mark.idx* files created by the **ctmtap** test program. Using `MAX_K_TO_USE 100` and `MAX_REBUILD_QUEUE 100 MB`, the original rebuild completed in an average of 1816 seconds, and the optimized rebuild completed in an average of 623 seconds, which is 66% faster (about 1/3 the time of the original rebuild).

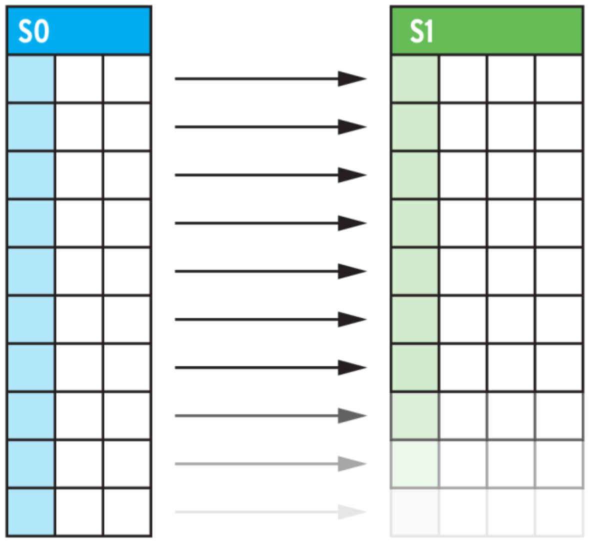
RECOVER_MEMLOG improves ctrdmp recovery performance

If you specify `!RECOVER_MEMLOG N` in your dump restore script (where *N* is a number of transaction logs), **ctrdmp** will read up to that many transaction logs into a memory buffer and will use the memory buffer to read the transaction log entries. This can speed up dynamic dump restore recovery in cases in which the dump backup includes many transaction logs.

4. c-treeACE Hot Alter Table

Application downtime during schema changes almost disappears with FairCom's new Hot Alter Table. This feature minimizes the impact to your customer when you roll out changes to your database schema.

FairCom's Hot Alter Table feature can dramatically increase efficiency and minimize downtime when rolling out changes to your application data schema. Known in the industry by various names such as "Dynamic Alter Table," "Real Time Alter," and "Online Alter," Hot Alter Table allows you, the developer, to provide "rolling upgrades" to your product deployed in the field. This means that, when your customer installs upgrades to your schema, they will not have to go through a lengthy file migration process, which previously could require many hours of downtime. Instead, they can use the new Hot Alter Table feature to upgrade records upon read or write.



The conventional Alter Table statement converts all records in the affected table at the time the statement is issued, working its way through the table processing one record at-a-time. It must create a new table, convert every record in the file and copy it into the new file, re-index, etc. This process can be time consuming if the table contains a large number of records. The table must remain locked, and therefore inaccessible to the application, during this process, resulting in a lengthy downtime when users cannot be connected to the database.

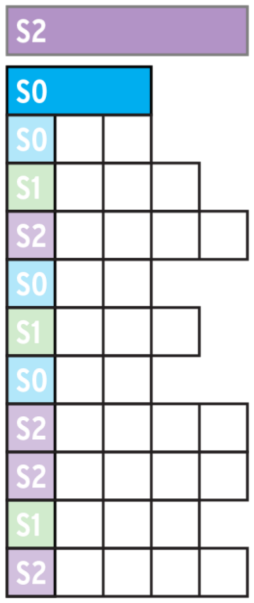
Hot Alter Table Performs Updates On-the-Fly

Hot Alter Table allows records to be automatically converted to a new format as they are read so you can effectively roll out schema changes on-the-fly. Your customers can upgrade their database without the need to make all of their users disconnect from the database during a time consuming change. **Application and database updates that previously required hours to accomplish can now be completed in seconds.**

By relaxing the requirement to alter a table's schema while the users are locked out of that table, application downtime is minimized. **Now you can roll out application updates that require database changes without worrying about significant downtime.**

Multiple Schemas

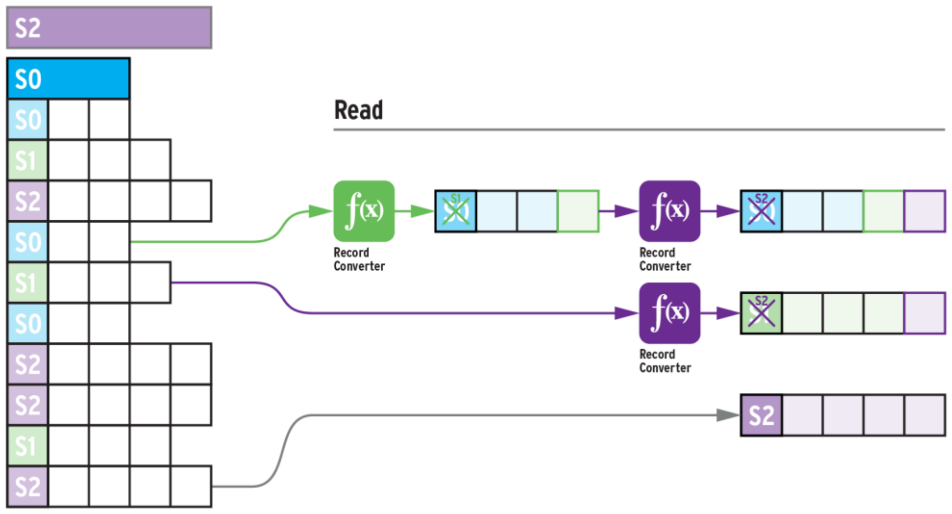
When using Hot Alter Table, the table is allowed to contain records with different schemas: the original schema, a new schema requested by an Alter Table, and potentially more schemas requested by additional Alter Table statements.



Each record contains one extra piece of information in the “header” of the record (the record’s metadata): a schema ID that indicates which schema applies to the record (S0=the original schema, S1=the first altered schema, S2=the next altered schema, etc.).

Read: Conversion “On-the-Fly”

If someone is reading a record that does not use the current schema, it is converted on-the-fly to make it appear current. The table's metadata contains the record converters necessary to convert records to each schema. The current schema version is always presented back to the reader, regardless of the version stored on disk.



Write: Uses the Current Schema

The write uses the current format when it writes a record to disk. Write operations remain efficient because there is very little special work to be done: records are always written in the current format.

Reduced Time of Exclusive Access

Although exclusive access to the file is required when Alter Table is requested (dynamic or otherwise), the Hot Alter Table feature will dramatically reduce the time spent in that state. The new functionality simply updates the record converter information in a resource. It does not have to update any records.

FLEXREC – Flexible Record Support

A new record format is now introduced that includes provisions for supporting Hot Alter Table. This is a permanent attribute of a file at create time. Files cannot undergo a Hot Alter Table until they have been converted to FLEXREC support.

Limitations

Hot Alter Table cannot presently modify or add indexed values.

4.1 Support for dynamically altering a table's schema on-the-fly

To use the Hot Alter Table feature, use one of the following procedures, based on the API in use:

FairCom DB API API

The FairCom DB API API is the recommended way to use Hot Alter Table.

To perform Hot Alter Table using the FairCom DB API API:

1. Create a file that uses the flexible record (FLEXREC) support by including `CTCREATE_FLEXREC` in the create mode. For example:
   ```
   NINT rc = ctdbCreateTable(hTable, "mytable", CTCREATE_FLEXREC);
   ```
2. To perform a Hot Alter Table, follow one of these procedures:
   a. A call to the **ctdbAlterTable()** function with mode of `CTDB_ALTER_NORMAL` determines if the Alter Table can be performed using Hot Alter Table. If so, it does. If not, it uses regular Alter Table.
   b. If the mode is `CTDB_ALTER_HOT`, a Hot Alter Table will be performed if possible (which requires the file to use the FLEXREC feature and for the changes to the schema to be possible to accomplish using Hot Alter Table). When using this mode, if a Hot Alter Table cannot be performed, **ctdbAlterTable()** returns error **CTDBRET_NOTSUPPORTED**.

ISAM API

To perform Hot Alter Table using the ISAM API, follow these steps:

1. Create a file that uses the flexible record (FLEXREC) support by including the `ctFLEXREC` bit in the *splval* field of the extended create block structure for the data file. For example:
   ```
   IFIL    myifil = { ... };
   XCREBLK xcreblk[2];
   NINT    rc;

   memset(xcreblk,0,sizeof(xcreblk));
   xcreblk[0].splval = ctFLEXREC;
   rc = CREIFILX8(&myifil, NULL, NULL, 0, NULL, NULL, xcreblk);
   ```
2. Use the **ctAlterSchema() https://docs.faircom.com/doc/ctreeplus/87643.htm** function to begin updating records on the fly as they are rewritten to the database. Its prototype is as follows:
   ```
   extern ctCONV NINT ctDECL ctAlterSchema(FILNO datno,pVOID precdsc);
   ```
   - *datno* is the user file number of the data file, which must be open in exclusive mode.
   - *precdsc* is a pointer to a record descriptor, which indicates what changes are being made to the schema.

[redacted]

See also:

- For using Hot Alter Table in SQL, see `ALTER TABLE` (https://docs.faircom.com/doc/`sqlref/altertable.htm`) in the *SQL Reference Guide*.

4.2 Adding Hot Alter Table support to existing files

CompactIFile() and the **ctcmpcif** utility support options to create compacted files with FLEXREC on or off. Note the flexible record schema (FLEXREC) is the internal flag that is required for Hot Alter Table support.

By default, compacting a FLEXREC file creates a FLEXREC file with all records updated to the most current schema.

When using the **ctcmpcif** utility, if the original data file does not use FLEXREC and you want the new compacted data file to use FLEXREC, specify the *-flexrec* option. If the original data file uses FLEXREC and you want the new compacted data file to not use FLEXREC, specify the *-noflexrec* option.

When using **CompactIFile()**, specify `chgflexrecIFILoption` to create a non-FLEXREC data file from a FLEXREC format data file, and vice-versa. Use the **setIFILoptions()** function to set the compact options in the *tfilno* field of the IFIL structure that you pass to **CompactIFile()**. For example:

```
pifil->tfilno = setIFILoptions(chgflexrecIFILoption);
```

4.3 Rebuilding data files with Hot Alter Table support

Support has been implemented within the normal c-tree index rebuild support for rebuilding files that use the flexible record schema (FLEXREC) feature, which is an internal part of the new Hot Alter Table. The changes for index rebuild are internal and therefore no extra provisions need to be considered when performing an index rebuild.

4.4 Hot Alter Table supports changing lengths of string fields

The Hot Alter Table feature supports changing the length of a string field. If the field length is increased, the field is padded using the field padding byte from the DODA. If the field length is decreased, the field values are truncated to the new shorter length.

4.5 Support for older versions

If an older FairCom Server attempts to open a data file that uses the flexible record support feature, it will fail to open the file with error 744 (**FREL_ERR**, file requires unavailable feature).

4.6 Replication of Hot Alter Table

The c-treeACE V11.5 Replication Agent includes support for files that have Hot Alter Table enabled.

4.7 Considerations

Items to be aware of when using Hot Alter Table:

1. It is not possible to perform a hot alter on a keyed field. If a field is part of an index, then it cannot be manipulated through Hot Alter Table.
2. It is presently only possible to change the data type for a string that has a length count, and it can be changed to another string data type that supports a larger length count.

   DETAILS: Hot Alter Table supports changing a string field type that uses a 1-byte or 2-byte length count to the same string field type with a larger length count or with no length count. That is:

   CT_FPSTRING can be changed to CT_F2STRING, CT_F4STRING, or CT_FSTRING.

   CT_F2STRING can be changed to CT_F4STRING or CT_FSTRING.

   CT_PSTRING can be changed to CT_2STRING, CT_4STRING, or CT_STRING.

   CT_2STRING can be changed to CT_4STRING or CT_STRING.

   CT_F2UNICODE can be changed to CT_FUNICODE.

FairCom R&D is researching options to relax these two dependencies in the future. If you are impacted by these limitations, please notify the FairCom Support Team.

5. Full-Text Search

Fast, efficient access to textual data.

This chapter presents an overview of the c-treeACE Full-Test Search (FTS) feature. More information is available online. For details about each of the FTS functions, see the **Full-Text Search Guide** (https://docs.faircom.com/doc/**fts/**) in the *Documentation* (*https://learn.faircomcorp.com/developers/documentation_directory*) section of the FairCom website.

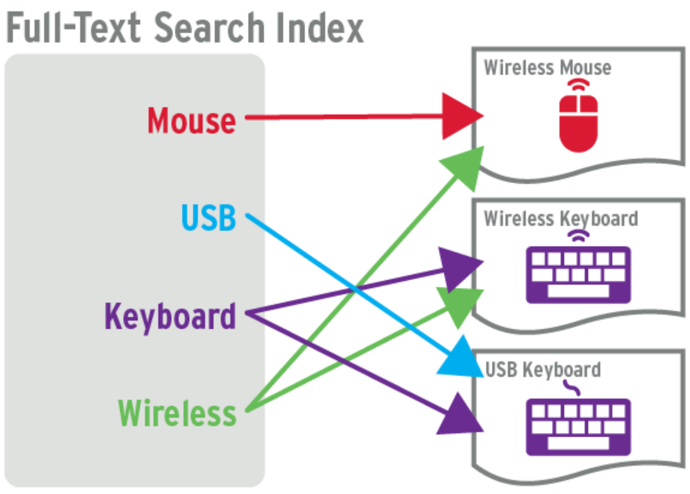
Today's applications process vast amounts of textual data. Full-Text Search (FTS) is a great mechanism for fast, efficient access to character-type data elements. c-tree applications with large volumes of text can now be complemented with high-performing text search capabilities.

Similar to a traditional b-tree index over a c-tree data file, you may now define a Full-Text Index (FTI) by specifying which character-type fields to include in this search index. An additional set of FTI files will be maintained on disk.

This support is on a file-by-file basis (the same as a typical c-tree index).

Once an FTI is defined for a file, it is maintained in “real-time” along with any other b-tree type c-tree indexes, including *deferred index* support.

Accessing data records through a full-text search index is simple. Using new API search functions, you provide a word or phrase (text) for which you are searching. All records whose FTI-indexed fields contain this text are returned. It is left up to the developer to utilize this result as needed for the application.



The first step in using Full-Text Search is creating an FTS Index (FTI). Functions are available through several different APIs for creating the index, adding a field to the index, setting the default index, and managing the FTI handle. Functions also allow options to be set and other parameters to be controlled.

The FTS indexes are maintained as records are added to the table. Internally, a *tokenizer* divides the text into "tokens" (which are roughly equivalent to a list of categorized words).

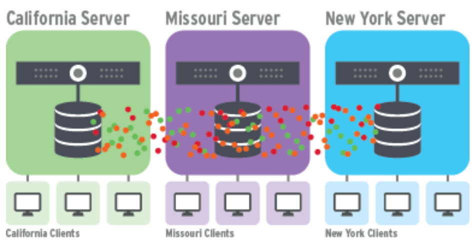
When a full-text search is performed, the function is passed one or more words. Each word is compared to the tokens in the FTS Index. The function returns the records that contain those words in their indexed fields.

Additional information can be returned, such as the proximity of multiple tokens. For example, when searching for "FairCom" and "database", they are in closer proximity in the text "FairCom database" than in "FairCom announces new enhancements to its highly acclaimed database." Statistics about relevance can also be returned.

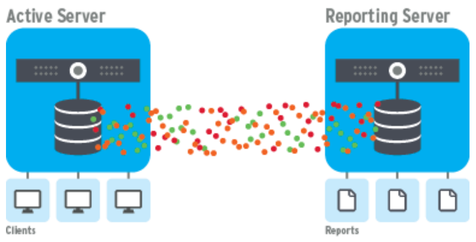
Additional details on this powerful new feature are available in the **Full-Text Search Guide** (https://docs.faircom.com/doc**/fts/**) in the *Documentation* (*https://learn.faircomcorp.com/developers/documentation_directory*) section of the FairCom website.

6. c-treeACE Replication Solutions

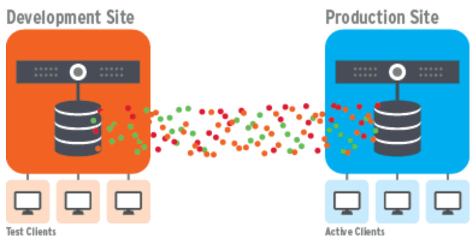
FairCom replication solutions take a giant step forward with this release. This section highlights new features such as resync, conflict resolution extensions, and other additions to the FairCom Replication Agent.



Distributed Data for Horizontal Scaling



Backend Reporting & Data Warehousing



Replica Environments for Testing

Failover Solutions for Disaster Recovery

6.1 Replication User-Defined Extensions

The Replication Agent can now perform custom operations to handle special situations, such as data aggregation and conflict resolution.

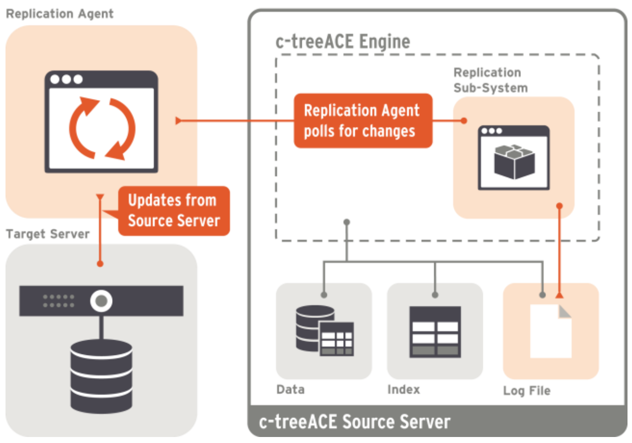
FairCom's goal is to allow customers the greatest control over their database environment. Now, to provide a new level of flexibility and customization, we are adding Replication Agent support for User Defined Extensions.

By loading a user-defined external library, the Replication Agent can perform customized actions when processing operations read from the source server. This feature allows a software developer to customize the Replication Agent's behavior to support capabilities such as:

- Conflict detection and resolution during replication
- Extract, Transform, and Load
- Filtering or redirecting data updates
- Replication between non-similar databases (even third-party databases)

For example, if you need the ability to transform data during replication, a DLL could be written to perform that transformation during replication. Suppose you take a replicated server offline, perform a schema upgrade on it, and then resume replication. Because it now has a newer schema than the other replicated servers, the DLL could transform the data structure from the original server to the upgraded server.

Replication extensions can be used in conjunction with the new **Hot Alter Table** *(page 15) support for implementing application upgrades and schema changes with little or no downtime.*



These extensions are provided in an external library that the Replication Agent loads on startup. When a user-defined extension library is used, the Replication Agent calls a function in the external library for each operation that it reads from the source server. The function can modify the operation values, or take custom actions, and it can indicate to the agent what action to take for that operation.

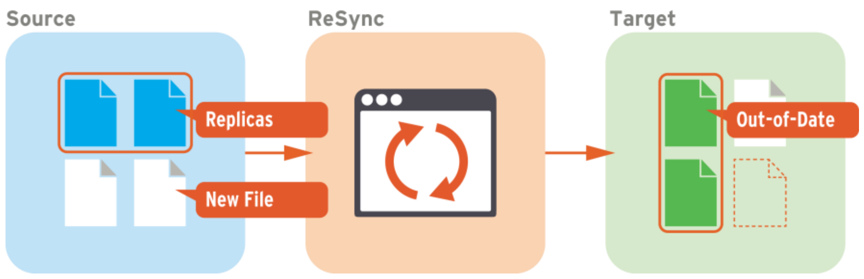
Initially, your extensions will be implemented within an SDK framework using a new set of user-defined functions. You will create a dynamic plug-in library containing your user-defined callback functions. The Replication Agent can then load this plug-in and execute the functions when appropriate.

With these replication callback routines, c-treeACE expands the possibilities of our replication solution to meet your exact needs. If you have a requirement that is not met by FairCom replication out-of-the-box, you may be able to write a custom DLL or shared object to meet your needs.

To use this feature, the developer needs access to c-treeACE Professional and to a development environment that supports writing and compiling C code into a DLL or shared library.   Details about the user-defined functions are available separately in the **Documentation section of the FairCom website** (**https://learn.faircomcorp.com/developers/documentation_directory**).

6.2 Replication ReSync

A fast fix for occasional situations when a loss of connection causes a target file becomes out-of-sync with the source.



The new Replication ReSync feature allows you to resynchronize a target "replica file" based on the current "source file" with minimum impact in runtime. This feature also allows you to add a new file to a running replication environment.

Synchronize with repadm resync

Ideally, the target file in a replication environment would never be out-of-sync with the source. In some situations, the target server or the Replication Agent goes down or loses its connection. When all parts of the system are up again and the connections are reestablished, the Replication Agent is able to catch-up with all the updates that must be applied from the moment that it stopped replicating. Although it may take some time to apply the missing updates from the backlog, it should not miss any. Unfortunately, some problems could happen in this process, making it possible that the replica file might get out-of-sync. Some examples:

- A lost transaction log file from the source side that was not applied to the target, making it impossible to bring the target copy back in sync.
- A conflict while applying the modification to the target (perhaps an inadvertent update to the target while replication was inactive).

The Replication ReSync feature provides a way to stop replication on one or more files in a replicated environment and resynchronize them. It copies "current files" from the source server to the target server and then restarts the replication on these files.

The administrator can use **repadm** *-c resync* and **repadm** *-c resyncclean* to manage the resync operation. The new modes allow the user to indicate the name of the source file to be synchronized. A text file with a list of source file names can be provided in case of multiple files to be synchronized. Based on the source file name, the Replication Agent identifies the target file based on the existing redirect rules.

Add a new file to a running replication environment

This new feature allows you to add a new file to a running replication. The Replication Agent will create the replica file based on an initial version of the source file.

Now you can execute this synchronization with the Replication Agent running, with minimum impact on performance. The process is performed as transparently as possible for the end-user.

6.3 More Advances in the FairCom Replication Solutions

Continual replication refinements for your most challenging business continuity needs.

TLS Support for Replication

In V11.5 and later, the Replication Agent supports configuration options that can be specified in *ctreplagent.cfg* to enable and configure TLS support for the connections to the source and/or the target server.

To enable a TLS connection for the source server, use:

```
source_use_tls yes
```

To enable a TLS connection for the target server, use:

```
target_use_tls yes
```

To set the certificate file name for the source server, use:

```
source_tls_cert_file filename
```

To set the certificate file name for the target server, use:

```
target_tls_cert_file filename
```

where filename is the name of the file containing the server certificate. The default value is *ctsrvr.pem*. To use no certificate, specify an empty string. For example:

```
target_tls_cert_file ""
```

Note 1:

The `source_use_tls` and `target_use_tls` options force the Replication Agent to use the TCP/IP communication protocol with TLS enabled. This means that even if the specified source or target server is using the shared memory communication protocol and is running on the same machine as the Replication Agent, when this option is used the Replication Agent will only attempt to connect using TCP/IP with TLS enabled. If you wish to use the Shared Memory communication protocol to connect to a server on the same machine without falling back to TCP/IP if the Shared Memory connection fails, you can specify the source or target server name with the *fsharemm* protocol. For example:

```
source_server FAIRCOMS^fsharemm
```

Note 2:

It is not permitted to specify the communication protocol in the `source_server` or `target_server` configuration option (for example, `FAIRCOMS^fsharemm`) when you use the `source_use_tls` or `target_use_tls` option. In this situation, the Replication Agent logs one of the following errors and terminates:

```
Error in replication agent configuration file ctreplagent.cfg: When using the source_use_tls option, the
source_server option cannot specify a communication protocol.
Error in replication agent configuration file ctreplagent.cfg: When using the target_use_tls option, the
target_server option cannot specify a communication protocol.
```

Replication of Hot Alter Table operations

New Hot Alter Table operations are immediately supported by replication for data files supporting the flexible record schema (FLEXREC) feature.

Replication Agent Monitor default server name changed

Note: This is a Compatibility Change in V11.5.

The default server name for the **ctreplagent.exe** process is `FAIRAGENT`. However, the default server in **c-treeReplMonitor.exe** was `FAIRCOMR`. This utility's default server name has been changed from `FAIRCOMR` to `FAIRAGENT` so it will connect by default out-of-the-box.

Note: After the first successful login, the Replication Agent server name is stored in the utility user registry, so users that are already logged in may still see the old server name.

Improved Replication Agent logging on Unix systems

On Unix systems, the Replication Agent now redirects its database engine's standard output and standard error logging to the file *ctreplagentstd.log* so that it is not mixed in with the normal logging of messages to the file *ctreplagent.log*. This allows a more standardized logging for automated reporting tools.

Performance enhancement of Replication Agent log read

The performance of the Replication Agent log read has been improved by reducing transaction log mutex acquisition on the source server. This change is particularly beneficial when the source server is under heavy transaction update activity by many clients.

Passing Replication Agent Command-Line Options to Its Internal Server

In V11.5 and later, the Replication Agent supports passing FairCom Server configuration options as command-line options, similar to support provided by the FairCom Server. For example:

```
ctreplagent SERVER_NAME REPAGENT LOCAL_DIRECTORY data2/
```

repadm option to list IDs for a Replication Agent process

The **repadm** utility now supports a *-getagentids* option to retrieve Replication Agent IDs that are currently registered in the Replication Agent. For example:

```
C:\> repadm -c getagentids -s FAIRAGENT

c-treeACE(tm) Version 11.4.0.27261(Build-170324) Replication Agent Management Utility

Copyright (C) 1992 - 2017 FairCom Corporation
ALL RIGHTS RESERVED.

Registered agent IDs:
AGENT2
AGENT1
```

Support c-treeACE quiesce state waiting for replication readers to process all committed transactions

c-treeACE quiesce feature now supports an option to wait for all replication readers that have registered with the server to finish processing all committed transactions.   This allows a known clean replication state to be established during a critical quiesce phase.

Enhancement to the ctquiet Utility

A new option has been added to the **ctquiet** utility to enable this state. Use the *-c* option to wait for replication readers to signal their completion. The *-c* option requires the *-f* (full consistency) option. Example:

```
ctquiet -c –f –p ADMIN –s FAIRCOMS
```

Note: The *-c* option requires ALL new client utilities, replication readers, and servers to be installed from c-treeACE V11.3 or later. It is not compatible as a drop-in to current running environments.

Monitoring ctQUIET() Progress

The server logs messages to CTSTATUS.FCS indicating its progress in waiting for replication readers to signal their completion:

```
Wed Mar  8 14:42:31 2017
 - User# 00019  ctQUIET: Waiting for replication readers to signal completion.
Wed Mar  8 14:42:31 2017
 - User# 00019  ctQUIET: Replication reader AGENT1 has signaled completion.
Wed Mar  8 14:42:31 2017
 - User# 00019  ctQUIET: All replication readers have signaled completion.
```

Note: If a replication reader does not have the support for responding to a **ctQUIET()** operation, the **ctQUIET()** call fails and the server logs this message to *CTSTATUS.FCS*:

```
Wed Mar  8 15:59:40 2017
 - User# 00018  ctQUIET: Replication reader AGENT1 does not support notification
of a ctQUIET() operation.
```

New ctQUIET API Options

For more detailed control, the **ctQUIET()** API supports several new options to allow replication readers to complete. To enable this support, OR in the *ctQTwaitForReplReaders* option bit into the mode value passed to **ctQUIET()**. If a replication reader is not connected to the server, or if it is connected but has not set its log position, **ctQUIET()** will wait for that replication reader to connect and process all committed transactions, unless the *ctQTignoreInactiveReplReaders* option is also specified, in which case **ctQUIET()** will ignore such inactive replication readers.

The **ctQUIET()** *timeoutSEC* option can be used to set a time limit on how long **ctQUIET()** waits for replication readers. The *timeoutSEC* value is also used as the time period for waiting for active transactions to finish. The amount of the *timeoutSEC* time that remains after active transactions have finished is applied to the wait for replication readers. For example, if *timeoutSEC* is 60 and active transactions take 40 seconds to finish, then **ctQUIET()** will wait for up to 20 seconds for replication readers to signal that they have processed all committed transactions.

Replication Agent redirection rules consider slash & backslash to be equivalent

The redirect rule applied by the Replication Agent now considers the slash character ("/") to be equivalent to the back slash character ("\"). This change addresses problems arising when the path separator used when opening a file is different from the one used in the redirections rules. For example, a file may be opened using a back slash ("\") in the path while the redirect rule may contain forward slash ("/").

Default Replication Agent conflict detection

By default, the Replication Agent now enables basic conflict detection strategies for record updates (add/deletes/rewrites). If a current record image on a target server differs from a record image for a delete operation from a source server, the operation is skipped and the Replication Agent logs the operation as failing with error **REPCNF_ERR** (1105) in the exception log.

The entry that is logged to the exception log in case of error **1105** includes both the information read from the source server (old and new keys and record images) as well as the current record image read from the target server.

The `check_update` option defaults to `yes`. To disable this conflict detection strategy, add the following option to *ctreplagent.cfg*:

```
check_update no
```

Note: This is a Compatibility Change.

Replication file filter list

Replication now supports file filtering that can specifically assign or exclude files to a specific Replication Agent. This allows parallel replication streams for improved performance by partitioning of data between agents. It also simplifies configuration when excluding only a few files.

Replication file filters are defined and deployed via an XML representation. As this is an advanced replication feature, implementation details of this can be obtained from FairCom upon request.

Replication Agent function profiling

To analyze where it is spending its time, the Replication Agent now supports tracking its function call times. This feature involved modifications to the Replication Agent and the **repadm** utility.

The repadm utility

The **repadm** utility's *getfuncstats* option reads and displays the function timings. By default, the cumulative figures are displayed, in order of highest to lowest total call time. Example:

```
repadm -c getfuncstats -u ADMIN -p ADMIN -s FAIRAGENT

Wed Jan 06 10:45:12 2016

         function name  %tot     time(usec)      calls    avg(usec)
                ADDREC    65       92147487      71237         1293
               TRANBEG    20       28112621      71277          394
               TRANEND     3        5067465      61559           82
   ctReplGetNextChange     2        4183921       3638         1150
               RWTVREC     2        4043569      61559           65
              Internal     2        3279513          1      3279513
               TRANRST     1        1782020       9718          183
              TRANABTX     0        1029898       9718          105
   ctReplPersistMinLog     0          64001         39         1641
                  Idle     0          18704          1        18704
```

When the *-d* option is used, the differences in stats between the previous and current time interval are displayed. For example:

```
repadm -c getfuncstats -d -u ADMIN -p ADMIN -s FAIRAGENT

Wed Jan 06 10:46:17 2016

         function name  %tot     time(usec)      calls    avg(usec)
                ADDREC    83        4244397       2799         1516
               TRANEND     4         212929       2491           85
   ctReplGetNextChange     3         169176        143         1183
               RWTVREC     3         167174       2492           67
              Internal     2         137212          1       137212
               TRANBEG     1          83824       2800           29
               TRANRST     1          57137        309          184
              TRANABTX     0          34180        309          110
   ctReplPersistMinLog     0           2741          2         1370

-c resetfuncstats resets the function timings to zero. Example:

repadm -c resetfuncstats -u ADMIN -p ADMIN -s FAIRAGENT

Successfully reset function statistics.
```

Notes to Aid Analyzing Function Timings

Generally, the **ctRepl*** function calls are made on the source server, and other function calls, such as **TRANBEG**, **ADDREC**, **RWTVREC**, and **TRANEND** are made on the target server.

The "Internal" function indicates time spent in the Replication Agent's internal operations, rather than in calls to the source and target servers. For example, the internal time includes writes to the Replication Agent's exception log.

The "Idle" function is a calculated value: When reading cumulative function times, "Idle" is the time since the statistics were last reset minus the total time used by all functions. When reading delta function times, "Idle" is the time since the last time this **repadm** connection read the statistics minus the time used by all functions in that time interval.

The `%tot` value is calculated by the **repadm** utility using the total time and individual function call times returned by the replication agent.

Calls to **ctReplGetNextChange** that time out are not included in the timings. So when the Replication Agent is caught up and waiting for additional changes from the source server, the **ctReplGetNextChange** function times are expected to be low, and the time is counted as idle time. Here is an example showing a Replication Agent that is caught up and waiting for more changes from the source server (using the *-d* option):

```
Wed Jan 06 10:51:17 2016
         function name  %tot     time(usec)      calls    avg(usec)
                  Idle    99        5002207          1      5002207
              Internal     0             37          1           37
   ctReplGetNextChange     0              0          4            0
```

Replication Agent Configuration Options

Note: By default, the Replication Agent does not collect function timings. This can be changed at Replication Agent startup by adding the option `function_timing on` in *ctreplagent.cfg*. It can be changed at runtime by using the **repadm** utility's *fnctim=on* command. Likewise, the **repadm** *fnctim=off* command turns off function timing collection. The **repadm** utility automatically enables function timing collection when the *getfuncstats* option is used.

The Replication Agent now also supports an option to disable exception logging. Use the option `exception_logging off` in *ctreplagent.cfg*, or at runtime use the **repadm** utility's *exceptionlog=off* option.

Replication administration utility (repadm) upgraded to latest statistics version

Two fields have been added to the RASTAT structure, which holds the Replication Agent statistics:

```
LONG8   nsalt; /* successful alter schema operations */
LONG8   esalt; /* failed alter schema operations */
```

The structure version has been changed from 1 to 2. The **repadm** utility now checks that the version used by the Replication Agent matches its structure version. If the versions differ, **repadm** outputs the following error message and terminates:

```
Error: Replication agent stat version (<agent_stat_version>) does not match repadm utility stat version
(<utility_stat_version>)
```

Older versions of the **repadm** utility do not check this version. Because these fields were added at the end of the structure, the old utility can still read the fields that are present in the version 1 structure. The Replication Agent detects an agent that supplies a structure that is the size of the original structure and returns only those fields.

Local/master replication model diagnostic logging of errors during two-phase transaction commit

When using the local/master replication model, it was possible to experience out-of-sync data between the master and local servers. In this model, we use a two-phase transaction commit on the master and local servers. To help us understand the sequence of events that precede the out-of-sync data situation, we added the server configuration option `DIAGNOSTICS TR_TRAN_ERR` to enable logging of errors during this two-phase commit sequence. The log entries are written to the file *TR_TRAN.FCS* in the server's *LOCAL_DIRECTORY* directory.

Sample log messages:

```
Wed Mar  1 09:35:46 2017 Transaction operation TRANABT failed: loc=1 master_err=127 local_err=0
Wed Mar  1 09:37:44 2017 Transaction operation TRANEND failed: loc=5 master_err=128 local_err=71
Wed Mar  1 09:43:25 2017 Transaction operation TRANEND failed: loc=5 master_err=128 local_err=0
Wed Mar  1 10:30:45 2017 Transaction operation TRANEND failed: loc=6 master_err=0 local_err=768
```

The log message includes function that failed, location code where the error occurred and master and local server error codes.

This feature can be enabled/disabled at runtime using **ctSETCFG()** or the **ctadmn** option to change a DIAGNOSTICS option.

Note: This is a Compatibility Change.

We also changed the following behavior based on our testing:

1) When aborting a transaction on the master server failed, we previously returned immediately rather than aborting the transaction on the local server. Now, in this situation, we mark the master server transaction as no longer active and then we abort the local transaction.

2) When phase 1 of the master server commit fails, we did not reset the active transaction indicator for the master server. This caused us to believe that the transaction on the master server was still active, even though it was not. Now we properly reset the active transaction indicator in this situation.

7. Interface Technology

Refine your development frameworks with c-treeACE interface advances.

7.1 Hibernate support adds a persistence layer to Java applications

Programming languages such as Java organize their data as objects. The underlying database typically uses a relational model that can be at odds with object-oriented data. The Hibernate object-relational mapping library ("ORM") was developed to address this issue. This library has become the de facto standard for mapping Java objects to a relational database.

To facilitate using the c-treeACE database with Hibernate, FairCom has developed a c-treeACE dialect for Hibernate. This allows c-treeACE to provide persistence for Java applications by simply using the same techniques documented on the *Hibernate website* (*http://hibernate.org/*).

If you have a Java application currently using Hibernate, simply follow the instructions in the ReadMe located in *\FairCom\V*\<your platform>\sdk\sql.hibernate\* to set up the c-treeACE Hibernate dialect and you are ready to begin using the c-treeACE database.

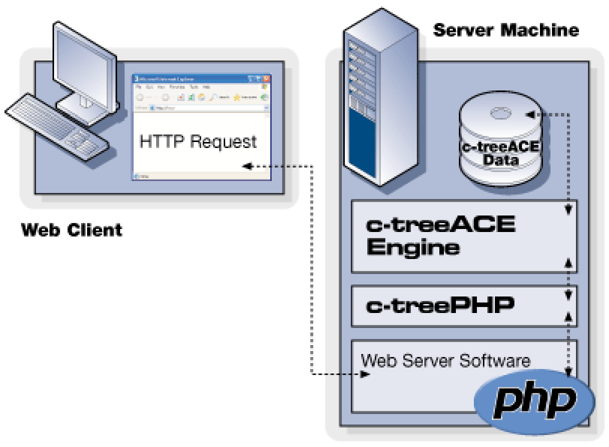
The configuration files, entity classes, DAO classes, the *hbm.xml* mapping file, etc. are expected to work exactly the same as any other generic Hibernate application.

7.2 c-treeACE PHP 7 Support

FairCom introduces support for the PHP Data Objects extension to the scripting language that drives the Internet.

PHP (Hypertext PreProcessor) is a widely-used scripting language that is especially suited for creating data-driven websites.

Originally, PHP used conventional SQL to access a database to retrieve data. Developers have long recognized one drawback: many databases use variations of "standard" SQL, making it difficult to migrate between different database providers.



PHP Data Objects (PDO) Extension

To address this issue, the community that develops PHP introduced the PHP Data Objects (PDO) extension for accessing a database. PDO gives you an abstraction layer so that your applications use the same functions to issue queries regardless of the underlying database. PDO allows an application to be independent of its database, so a developer can change databases without changing the application logic.

c-treeACE PHP offers native support for this extension by including a database-specific PDO driver to access the FairCom Server.

- Developers who are interested in taking advantage of this interface for accessing the c-treeACE database using the latest PHP standard will want to use this interface.
- Developers who have an existing PHP application that uses PDO are able to upgrade their application to use c-treeACE by simply installing the driver—no re-coding should be required.

c-treeACE provides two versions of the c-treeACE PHP drivers:

- **sql.php.pdo -** Support for PHP 7 and the PDO extension. **This is the preferred way to access c-treeACE from PHP.** An **example** showing the usage of this new support is included in the *\FairCom\V*\<your platform>\sdk\sql.php.pdo* directory. See the **ReadMe** located in that directory.
- **sql.php** - Support for PHP 4, PHP 5, and PHP 7.

7.3 c-treeACE Python

SQL support for Python V3.x

Notes:

1. The FairCom *pyctree* interface (the Python DB API 2.0-compliant driver) is compatible with Python versions 2.6 and 3.*x*.
2. Support for the following data types has not been implemented: BINARY, VARBINARY, LONGVARBINARY, LONGVARCHAR and UNICODE fields.
3. The following features do not conform to the PEP 249 standard:
   - **Auto Commit** - The autocommit read/write property of the Connection class allows the programmer to turn on/off the autocommit capability.
   - **Query Timeout** - The querytimeout read/write attribute of the Cursor class can be set to 0 (default) to indicate no timeout or to a positive value to specify a query timeout in seconds.
4. The provided c-treeACE SQL libraries are linked with OpenSSL support. Due to the link order of the libraries, some platforms may require specific run-time library pre-load instructions. For example, on some Linux distros:

```
export LD_LIBRARY_PATH=../../../lib/sql.python/
LD_PRELOAD=/usr/lib/libssl.so python PY_tutorial2.py
```

Best practice dictates LD_PRELOAD should only specify full paths to your *libssl.so* library, which may reside in different locations on different Unix/Linux distros.

Python support expands beyond Windows and Linux

c-treeACE Python support has been expanded beyond the Windows and Linux platforms to also include Solaris and AIX. If you desire Python support for other platforms, please contact your nearest FairCom Office.

Cursor.rowcount now returns the number of returned rows from "select" before fetching

The Python driver's *Cursor.rowcount* property has been enhanced to be compliant with the definition and return the appropriate number of rows for the recordset before starting to fetch.

7.4 c-treeACE JDBC

JDBC connection URL new format

Note: This is a Compatibility Change.

In releases prior to V11, the JDBC driver expected a connection URL in a non-standard format that application generation tools may not support:

```
jdbc:ctree:[<portnumber>@]<host>:<database>[?param=value[&param=value]...]
```

The JDBC URL format has been changed to be compliant with standard JDBC spec:

```
jdbc:ctree://<host>[:portnumber]/<dbname>[?param=value[&param=value]...]
```

where

- *<host>* can be a host name, an IPv4 address, or an IPv6 address enclosed with square brackets as per RFC 2732.
- The valid parameters are:

  characterEncoding

  password

  user

Note: For complete backward compatibility, the c-treeACE SQL JDBC URL supports both the old format as well as the new format. The new format is the preferred format.

JDBC Support for TLS

In c-treeACE V11.2 and later, c-treeACE SQL JDBC supports TLS connections per the JDBC standard. Enable TLS in a JDBC connection URL using the *ssl=value* parameter string.

TLS connections are enabled in the JDBC connection URL using the new format (it is not supported on the old URL format) and a new parameter *ssl*.

The new URL format is:

```
jdbc:ctree://<host>[:portnumber]/<dbname>[?param=value[&param=value]...]
```

The valid *param* values are:

- *characterEncoding* - Replace encoding with a valid Java encoding name (e.g., US-ASCII, ISO-8859-1, UTF-8, etc.).
- *password*
- *user*
- *ssl* - The valid values for *ssl* are:

  basic

  peerAuthentication
- *trustStore* - The name of a local trust store that includes the server's CA certificate.
- *trustStorePassword* - The password for the specified trust store.
- *keyStore* - The name of a local key store that has the user certificate and private key.
- *keyStorePassword* - The password for the specified keyStore.

NOTE: For backward compatibility, the older format (`"jdbc:ctree:6597@localhost:ctreeSQL", "ADMIN", "ADMIN"`) is still supported but should be considered deprecated.

Basic TLS with JDBC clients

Traffic to the server is encrypted, but there is no assurance of the server's identity.

Basic SSL encryption on the client is enabled by the URL parameter *ssl*, for example:

```
Connection c = getConnection("jdbc:ctree://localhost:6597/ctreeSQL?ssl=basic");
```

Peer Authenticated TLS with JDBC clients using System properties

If the client wants to authenticate the server, then the client's trust store must contain the server's CA certificate (or a self-signed server certificate).   A Java keystore must first be created that contains this CA certificate.

For example, the following adds the trusted CA certificate `ctsrvr.pem` to a keystore named `server.store`

```
keytool -importcert -file ctsrvr.pem -keystore server.store
```

`keytool` is part of the Java distribution, and will prompt you for a password used to encrypt the trust store.   In this example we used `mypassword` as the password.

Client SSL with server authentication is enabled by the URL parameter *ssl* set to *peerAuthentication*.

You must set the system properties *javax.net.ssl.trustStore* and *javax.net.ssl.trustStorePassword* to reference the trust store for the desired database server.

Example:

```
System.setProperty("javax.net.ssl.trustStore","server.store");
System.setProperty("javax.net.ssl.trustStorePassword","mypassword");
Connection c = getConnection("jdbc:ctree://localhost:6597/ctreeSQL?ssl=peerAuthentication");
```

Full TLS authentication with JDBC clients

The FairCom database server may be configured to allow or require client TLS authentication in place of password based authentication. The client needs to both authenticate the server (with the same requirements as for peerAuthentication), and provide proof of identity to the server by providing a client certificate that is trusted by the server. This may be set in the connection string by specifying a `trustStore` for the server and `keyStore` for the user.

Example:

Create the trust store with the trusted CA certificate `ctsrvr.pem` to a keystore named `server.store`. Keytool is part of the Java distribution, and will prompt you for a password used to encrypt the trust store. In this example we use `mypassword` as the password.

```
keytool -importcert -file ctsrvr.pem -keystore server.store
```

The client must possess a keystore containing their certificate and private key. Here we assume a keystore exists named `johndoe.pkcs12` protected with password `secret`.

NOTE: The default keystore format is controlled by the `java.security` configuration file under the property `keystore.type`.   The PKCS12 keystore format is supported by all Java implementations, but prior to Java9 JKS was the default format.   You may need to adjust this configuration for your keystore to be accessed by Java.

The following connection string will attempt to connect using these certificate sets for TLS authentication.

```
Connection c =
getConnection("jdbc:ctree://localhost:6597/ctreeSQL?trustStore=server.store&trustStorePassword=mypassw
ord&keyStore=johndoe.pkcs12&keyStorePassword=secret");
```

See Also:

Java keytool https://docs.oracle.com/javase/8/docs/technotes/tools/windows/keytool.html documentation provides examples of how to generate a key pair, request a certificate from a CA, or generate certificates for a server.

JDBC tutorials updated to use the current URL format

The JDBC tutorial has been updated to use the new URL format described in *JDBC connection URL new format* (page 38). In V11.1 and later, the previous URL format is now considered as legacy (although it is still supported).

7.5 c-treeACE SQL ADO.NET

ADO.NET Support for TLS

In V11.5 and later, the c-treeACE ADO.NET provider supports TLS/SSL connections per Microsoft specifications.

The ADO.NET provider requires *ctsrvr.pem* to be added to the trusted root certificate store on the client machine for the .NET framework's certificate authentication to succeed:

```
CertMgr.exe /add ctsrvr.pem /c /s /r localMachine root
```

Note that the *Common Name* specified in the server certificate is the name that the application must specify in the ADO.NET connection string for the TLS option.

For this certificate, we used *support.faircom.com* as the *Common Name*, and so the ADO.NET connection string must specify *sslcert=support.faircom.com* for the TLS authentication to succeed.

Connection String

The ADO.NET connection string is similar to the JDBC string. The connection string accepts a new property:

```
ssl=<value>
```

which can have two values:

- *basic* - Basic SSL setting, no peer certificate authentication, only communication encryption as requested by server certificate
- *peerAuthentication* - Server certificate authentication.

In the case of *peerAuthentication* the server certificate Common Name must be provided by the new property:

```
sslcert=<value>
```

If this property is not specified, the value of the *Server* setting is used to match the certificate.

Examples:

```
"UID=ADMIN;PWD=ADMIN;Database=CtreeSQL;Server=localhost;Service=6597;ssl=basic";
"UID=ADMIN;PWD=ADMIN;Database=CtreeSQL;Server=localhost;Service=6597;ssl=peerAuthentication;sslcert=su
pport.faircom.com";
```

connectionidletimeout Connection Option for ADO.NET Provider

In V11.2.3 and later, a `connectionidletimeout` connection-string option is available to indicate how long the connection may stay idle in the pool before getting closed and removed from the pool. This option is separate from the existing connection timeout option, which has a different meaning in ADO connection pool.

connectionidletimeout connection option for ADO.NET provider

In V11.2.3 and later, a `connectionidletimeout` connection-string option is available to indicate how long the connection may stay idle in the pool before getting closed and removed from the pool. This option is separate from the existing connection timeout option, which has a different meaning in ADO connection pool.

ADO.NET Provider - Updated list of reserved words

A few missing Reserved words (e.g., CASE, WHEN, THEN etc.) have been added to the ADO.NET provider list.

ADO.NET tutorial projects upgraded to Framework 4.5

The ADO.NET tutorial projects were referencing .NET Framework 4.0. They have been upgraded to reference version 4.5. All the relevant project files have been updated.

7.6 Direct SQL

DSQL no longer allows using ctsqlGetChar() to retrieve long variable columns

In V11.5 and later, the DSQL logic has been modified to return an error if you attempt to use **ctsqlGetChar()** on long columns because the value they return is not correct. Using the following functions on a long column now returns the **CTSQLRET_INTYPES** (**SQL_ERR_INTYPES -20008**) error:

ctsqlGetChar	**ctsqlGetInteger**	**ctsqlGetMoney**	**ctsqlGetBinary**

ctsqlGetNChar	**ctsqlGetReal**	**ctsqlGetTime**	**ctsqlGetBit**
ctsqlGetNumeric	**ctsqlGetFloat**	**ctsqlGetTimeStamp**	**ctsqlGetBigInt**
ctsqlGetSmallInt	**ctsqlGetDate**	**ctsqlGetTinyInt**	**ctsqlGetNumericAsString**

Note: This constitutes a Compatibility Change.

DSQL - New ctsqlGetParameterIndexByName function

In c-treeACE V11.5 and later, **ctsqlGetParameterIndexByName**, provides a way to identify the parameter index number from its name when using named parameters.

ctsqlGetParameterIndexByName

Declaration

```
CTSQLRET ctsqlGetParameterIndexByName(pCTSQLCMD hCmd CTSQLCHAR* name)
```

Description

Retrieve the parameter index given its name when using named parameter *:name*.

- *hCmd* - the command handle
- *name* - parameter name (without '`:`').

Return

Return the parameter index or -1 if not found.

DSQL partial support for scrollable cursors

New calls have been implemented in the DSQL API to set/get the scrollable cursor flag. When the scrollable cursor flag is turned on in the command handle, the cursor returned by the Execute function will be scrollable and the number of rows properly set to the rows in the resultset. This support has been implemented in the DSQL API to obtain the number of rows in the resultset.

ctsqlGetScrollableCursor

Retrieves the state of the scrollable cursor flag.

ctsqlSetScrollableCursor requires a valid command handle and needs to be called before the "prepare" call.

Note: Currently, the only reason to set scrollable cursors is to obtain the number of rows in the resultset immediately. At this time, there is no support to scroll through the resultset other than "next" (forward only).

Parameters:

- [hCmd] - Must point to a valid command handle.

Return:

Returns CTSQL_TRUE if scrollable cursor flag is on.

ctsqlSetScrollableCursor

Added in c-treeACE V11.5, **ctsqlSetScrollableCursor** sets or clears the auto-commit flag. When the scrollable cursor flag is turned on in the command handle, the cursor returned by the Execute function will be scrollable and the number of rows properly set to the rows in the resultset. This support has been implemented to obtain the number of rows in the resultset.

Note: Currently, the only reason to set scrollable cursors is to obtain the number of rows in the resultset immediately. At this time, there is no support to scroll through the resultset other than "next" (forward only).

Parameters:

- [hCmd] - Must point to a valid command handle
- *flag* [IN] - Indicate if the scrollable cursor flag is to be turned on or off.
  CTSQL_TRUE indicates that the scrollable cursor flag is to be turned on, while CTSQL_FALSE indicates that the scrollable cursor flag is to be turned off.

Return:

Returns CTSQLRET_OK on success.

7.7 ODBC

c-treeACE setup error 126 during ODBC installation phase

Issue:

When you are installing your c-treeACE package you get an error **126** during the ODBC Installation Phase.

Note: This issue may happen on Windows Vista, Windows 7 and Windows 8.

Cause:

The needed Microsoft Visual C++ Redistributable package was not installed properly because of missing Windows dependencies.

Solutions:

Solution 1: Make sure your Windows operating system is up-to-date. Please use Windows Update to get all the latest updates.

Solution 2: Manually download and install the Visual C++ 2017 Redistributable Package from the Microsoft website.

8. Security

Protect your data at rest and now in-flight with the latest c-treeACE security features. Advanced Encryption, TLS, and multiple authentication control levels guard valuable application data access.

Existing customers, please be sure to also review the addendum to this security chapter included with your software distribution called **c-treeACE.V11.5.0.SecurityAddendum.pdf**.

8.1 Transport Layer Security Secures Data in Transit between Network c-treeACE Clients and Servers

Note: c-treeACE File and User Security are available only when using the client/server operational model.

FairCom applications can now secure data in transit between c-tree network clients and FairCom servers. Transport Layer Security (TLS, also commonly referred to as its predecessor SSL, Secure Sockets Layer) is a cryptographic protocol designed for secure network communications using public key cryptography for authentication of the communicating party. Symmetric cryptography is used to encrypt transmitted data over the wire. c-treeACE TLS relies on OpenSSL toolkit support and implements TLS protocol V1.3. Earlier versions of TLS (and predecessor SSL) protocols contain known and exploited vulnerabilities. c-treeACE only supports TLS via TCP/IP communications protocols (IPv4 and IPv6). (For more about TLS, see https://en.wikipedia.org/wiki/Transport_Layer_Security (https://en.wikipedia.org/wiki/Transport_Layer_Security).)

Two modes of TLS connections are available: basic and peer authenticated. Basic TLS connections are encrypted using only a server-side certificate; there is no local certificate requirement for a client. This makes deployment and management of secured connections easy.

TLS Certificates

It is the server administrator's responsibility to ensure a correct and valid certificate pair, as well as proper configuration of allowed TLS connections.

Creation and management of TLS certificates, as well as use of a Certification Authority (CA), is beyond the scope of this document. Consult OpenSSL and other TLS supporting documentation and be sure you firmly grasp all details regarding use of TLS for network security before deploying.

Server certificates may be created and provided as two separate files: a certificate and a key. They can also be combined into a single file. There is no required file naming convention. Certificate files are usually created and/or provided as Base64 encoded X.509 certificate files and denoted with a *.pem* extension (Privacy Enhanced Mail). They can be identified with this set of surrounding identifiers within the file:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

The private key is likewise identified:

```
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

The private key is often included in the same certificate file or as a separate *.key* file.

A certificate key can be optionally passphrase protected. However, this then requires the passphrase to be presented at server startup. FairCom key store files provide this ability.

Always securely maintain key files. Store key files only in permissions protected server areas and never distribute.

Server-Side Configuration

To enable TLS (SSL), see keywords for TLS (https://docs.faircom.com/doc/ctserver/87457.htm).

Standard c-tree TCP/IP ports are used for connections regardless of TLS configuration. That is, a single ISAM port will handle both TLS encrypted and non-encrypted connections, and likewise for the SQL port. There is no need for separate port configurations.

For the HTTPD plugin you can verify ciphers in use using the system nmap command (Linux) against the HTTPD port when no ciphers are specified in *cthttpd.json*. This can be done with the linux command:

```
nmap --script ssl-enum-ciphers -Pn -p 8443
```

Client-Side Configuration

Both FairCom ISAM and SQL clients support TLS connections.

Peer Authentication - TLS connection with server certificate validation:

By default, a c-tree client requires a PEM file containing the server public certificate—**ONLY the public certificate, not the server private key**.

Important: The server private key should be securely maintained only at the FairCom Server location at all times.

By default, ISAM and SQL client libraries use the file *ctsrvr.pem* in the client process' working directory when connecting. An ISAM client can change the file name that the client library searches for by calling the **ctSetCommProtocolOption()** function with the option *ctCOMMOPT_FSSLTCP_SERVER_CERTIFICATE*:

```
ctSetCommProtocolOption(ctCOMMOPT_FSSLTCP_SERVER_CERTIFICATE, "myservercert.pem");
```

Basic - TLS connection without server certificate validation (ISAM only):

It is also possible to establish a TLS connection from an ISAM client without validating the FairCom certificate. Such a connection is encrypted but there is no guarantee of the server's identity. To use this option, call the **ctSetCommProtocolOption()** function with the option *ctCOMMOPT_FSSLTCP_SERVER_CERTIFICATE* with an empty certificate name before connecting:

```
ctSetCommProtocolOption(ctCOMMOPT_FSSLTCP_SERVER_CERTIFICATE, "");
```

When the client does not use a server certificate, the connection is encrypted, but there is no guarantee that the client is connected to that specific server. This implies that a "man in the middle" attack could be possible.

If an error occurs when connecting using SSL, the connection attempt returns error 1104 (**SSLCONN_ERR**). To get more detailed information enable SSL logging by either:

- calling **ctSetCommProtocolOption()** with the *ctCOMMOPT_FSSLTCP_DEBUG_LOG* option:

```
ctSetCommProtocolOption(ctCOMMOPT_FSSLTCP_DEBUG_LOG, "ssldebug.log");
```

  or

- setting the environment variable `CTSSL_DEBUG_LOG` to the name of the SSL debug log file.

To request a TLS-enabled ISAM connection, append `^fssltcp` to the server name. For example:

```
ctadmn ADMIN ADMIN "" "FAIRCOMS@localhost^fssltcp"
```

Note: When you append `^fssltcp` to the server name you must quote the *entire* connection string, for example:

`"FAIRCOMS@localhost^f_tcpipv6"`

Windows normally interprets the carat as an escape character or line continuation, so the entire server connection string must be quoted when it includes a caret.

Other examples showing how to specify a desired communication protocol when connecting:

To connect using shared memory (without falling back to TCP/IP if the shared memory connection fails):

```
FAIRCOMS@localhost^fsharemm
```

To connect using TCP/IP v4 without SSL:

```
FAIRCOMS@localhost^f_tcpip
```

To connect using TCP/IP v6 without SSL:

```
FAIRCOMS@localhost^f_tcpipv6
```

To connect using TCP/IP v6 with SSL:

```
FAIRCOMS@localhost^fssltcpv6
```

To request a TLS-enabled connection for a SQL connection, prepend `ssl:` to the connection string. For example:

```
isql -u admin -p ADMIN ssl:6597@localhost:ctreesql
```

Standard c-treeACE SQL interfaces (JDBC, ADO.NET, ODBC, PHP, Python) each have independent standards for connection strings. JDBC and ADO.NET are specifically described later in this chapter.

ctadmn and c-treeACE Monitor show the communication protocol that is in use, including whether or not the connection is using TLS (SSL):

```
F_TCPIP     indicates an unencrypted ISAM TCP/IP connection.
FSSLTCP     indicates an SSL-enabled ISAM TCP/IP connection.
SQL_TCPIP   indicates an unencrypted SQL TCP/IP connection.
SQL_SSLTCP  indicates an SSL-enabled ISAM TCP/IP connection.
```

X.509 Support

Support has been added for X.509 client/server authentication. For now, only X.509 authentication when using TLS-encrypted TCP/IP for ctree is supported. Future revisions will extend this to SQL and shared memory protocols.

X.509 authentication is not supported in combination with LDAP authentication.

To use X.509 authentication, the client must provide a PEM formatted X.509 certificate with a complete certificate chain using the same root CA as the server certificate. If the client's certificate is successfully validated by the server, the subject field on the client X.509 will be parsed by the server to extract a username. This username is used to determine the database permissions to be granted. A complete certificate chain consists of a PEM formatted file with the certificate for the user, followed by the certificate for the issuer, followed by the certificate for that issuer, and so on, ending in the root CA certificate.

When X.509 authentication is enabled and a certificate is provided by the client, the username and password arguments passed to InitISAMX() are ignored.

New Security keywords have been added to enable X.509. See Client Communications Keywords for TLS (https://docs.faircom.com/doc/ctserver/87457.htm).

The following functions have been enhanced for configuring X.509 authentication

- `ctSetCommProtocolOption`
- `InitISAMXtd`

Expected TLS Performance

The resource-intensive portion of a TLS connection is the initial creation. A secure communications channel is established via asymmetric encryption requiring a session key exchange. This initial key exchange and encryption process is the slowest computational epoch. Once connected, ongoing connection overhead is negligible. Further, most modern hardware contains dedicated CPU instructions for enhanced encryption performance. Thus, it is important to avoid repeated connections and maintain an established TLS connection when at all possible.

Compatibility

COMPATIBILITY TCPIP_CHECK_DEAD_CLIENTS (https://docs.faircom.com/doc/ctserver/compatibility-tcpip-check-dead-clients-config.htm)

Advanced SSL Certificate Options

The `ssl_certificate` keyword in the *config/cthttpd.json* web server plug-in configuration supports the *fccert.pem*, which is a self-signed certificate we supplied. To use your own certificate, use the following keywords to *config/cthttpd.json*:

- *ssl_key: SSL private key* - This can be embedded in the same file provided the in *ssl_certificate*. For example, our default *fccert.pem* certificate file has both the certificate and the private key, so, *ssl_key* is not required.
- *ssl_ca: SSL Certificate Authority* - External authority that issues and validates the certificate.
- *ssl_cipher_suites* - Colon-delimited list of SSL cipher suites.

Troubleshooting

There is a client-side environment variable for debugging. To enable SSL logging for the client, set the `CTSSL_DEBUG_LOG` environment variable to the log file name.

For server side SSL debugging, add `DEBUG_LOG <logfile>` to the SSL subsection.

Testing with Default c-treeACE Certificates

For testing and evaluation purposes only, a self-signed X.509 certificate is included in your default c-treeACE package. OpenSSL was used to create this certificate.

Security Note: It is critical this included certificate is never used in a production setting.

In the example below, a certificate and private key are both included in the file *ctree_ssl.pem*. Unencrypted TCP/IP connections are allowed, and the specified ciphers are the ones that are allowed to be used in encrypting the SSL connection:

```
SUBSYSTEM COMM_PROTOCOL SSL {
 SERVER_CERTIFICATE_FILE ctree_ssl.pem
 SSL_CONNECTIONS_ONLY NO
 SSL_CIPHERS ALL:!aNULL:!eNULL:!SSLv2:!LOW:!EXP:!RC4:!MD5:@STRENGTH
}
```

To enable only TLS-encrypted communications for all connections, change `SSL_CONNECTIONS_ONLY` to `YES`, and, optionally, comment Shared Memory communications support to prevent local unencrypted shared memory connections. (c-treeACE Shared Memory connections are not supported for TLS encryption.)

```
;COMM_PROTOCOL  FSHAREMM
```

Finally, for peer authentication, an additional cross-check validation against the server certificate, copy *ctsrvr.pem* to your client working directory. c-treeACE management and administration tools already include this local client certificate file in their working folder:

<faircom>\server

All GUI tool connection dialogs contain specific parameter options for enabling TLS connections. *support.faircom.com* was specified as the Common Name in provided c-treeACE default certificates. Specify *support.faircom.com* when testing c-treeACE SQL Explorer for TLS authentication using ADO.NET to succeed using these included certs.

ADO.NET Support for TLS

In V11.5 and later, the c-treeACE ADO.NET provider supports TLS/SSL connections per Microsoft specifications.

The ADO.NET provider requires *ctsrvr.pem* to be added to the trusted root certificate store on the client machine for the .NET framework's certificate authentication to succeed:

```
CertMgr.exe /add ctsrvr.pem /c /s /r localMachine root
```

Note that the *Common Name* specified in the server certificate is the name that the application must specify in the ADO.NET connection string for the TLS option.

For this certificate, we used *support.faircom.com* as the *Common Name*, and so the ADO.NET connection string must specify *sslcert=support.faircom.com* for the TLS authentication to succeed.

Connection String

The ADO.NET connection string is similar to the JDBC string. The connection string accepts a new property:

```
ssl=<value>
```

which can have two values:

- *basic* - Basic SSL setting, no peer certificate authentication, only communication encryption as requested by server certificate
- *peerAuthentication* - Server certificate authentication.

In the case of *peerAuthentication* the server certificate Common Name must be provided by the new property:

```
sslcert=<value>
```

If this property is not specified, the value of the *Server* setting is used to match the certificate.

Examples:

```
"UID=ADMIN;PWD=ADMIN;Database=CtreeSQL;Server=localhost;Service=6597;ssl=basic";
"UID=ADMIN;PWD=ADMIN;Database=CtreeSQL;Server=localhost;Service=6597;ssl=peerAuthentication;sslcert=su
pport.faircom.com";
```

connectionidletimeout Connection Option for ADO.NET Provider

In V11.2.3 and later, a `connectionidletimeout` connection-string option is available to indicate how long the connection may stay idle in the pool before getting closed and removed from the pool. This option is separate from the existing connection timeout option, which has a different meaning in ADO connection pool.

JDBC Support for TLS

In c-treeACE V11.2 and later, c-treeACE SQL JDBC supports TLS connections per the JDBC standard. Enable TLS in a JDBC connection URL using the *ssl=value* parameter string.

TLS connections are enabled in the JDBC connection URL using the new format (it is not supported on the old URL format) and a new parameter *ssl*.

The new URL format is:

```
jdbc:ctree://<host>[:portnumber]/<dbname>[?param=value[&param=value]...]
```

The valid *param* values are:

- *characterEncoding* - Replace encoding with a valid Java encoding name (e.g., US-ASCII, ISO-8859-1, UTF-8, etc.).
- *password*
- *user*
- *ssl* - The valid values for *ssl* are:

  basic

  peerAuthentication
- *trustStore* - The name of a local trust store that includes the server's CA certificate.
- *trustStorePassword* - The password for the specified trust store.
- *keyStore* - The name of a local key store that has the user certificate and private key.
- *keyStorePassword* - The password for the specified keyStore.

NOTE: For backward compatibility, the older format (`"jdbc:ctree:6597@localhost:ctreeSQL", "ADMIN", "ADMIN"`) is still supported but should be considered deprecated.

Basic TLS with JDBC clients

Traffic to the server is encrypted, but there is no assurance of the server's identity.

Basic SSL encryption on the client is enabled by the URL parameter *ssl*, for example:

```
Connection c = getConnection("jdbc:ctree://localhost:6597/ctreeSQL?ssl=basic");
```

Peer Authenticated TLS with JDBC clients using System properties

If the client wants to authenticate the server, then the client's trust store must contain the server's CA certificate (or a self-signed server certificate).   A Java keystore must first be created that contains this CA certificate.

For example, the following adds the trusted CA certificate `ctsrvr.pem` to a keystore named `server.store`

```
keytool -importcert -file ctsrvr.pem -keystore server.store
```

`keytool` is part of the Java distribution, and will prompt you for a password used to encrypt the trust store.   In this example we used `mypassword` as the password.

Client SSL with server authentication is enabled by the URL parameter *ssl* set to *peerAuthentication*.

You must set the system properties *javax.net.ssl.trustStore* and *javax.net.ssl.trustStorePassword* to reference the trust store for the desired database server.

Example:

```
System.setProperty("javax.net.ssl.trustStore","server.store");
System.setProperty("javax.net.ssl.trustStorePassword","mypassword");
Connection c = getConnection("jdbc:ctree://localhost:6597/ctreeSQL?ssl=peerAuthentication");
```

Full TLS authentication with JDBC clients

The FairCom database server may be configured to allow or require client TLS authentication in place of password based authentication. The client needs to both authenticate the server (with the same requirements as for peerAuthentication), and provide proof of identity to the server by providing a client certificate that is trusted by the server. This may be set in the connection string by specifying a `trustStore` for the server and `keyStore` for the user.

Example:

Create the trust store with the trusted CA certificate `ctsrvr.pem` to a keystore named `server.store`. Keytool is part of the Java distribution, and will prompt you for a password used to encrypt the trust store. In this example we use `mypassword` as the password.

```
keytool -importcert -file ctsrvr.pem -keystore server.store
```

The client must possess a keystore containing their certificate and private key. Here we assume a keystore exists named `johndoe.pkcs12` protected with password `secret`.

NOTE: The default keystore format is controlled by the `java.security` configuration file under the property `keystore.type`. The PKCS12 keystore format is supported by all Java implementations, but prior to Java9 JKS was the default format. You may need to adjust this configuration for your keystore to be accessed by Java.

The following connection string will attempt to connect using these certificate sets for TLS authentication.

```
Connection c =
getConnection("jdbc:ctree://localhost:6597/ctreeSQL?trustStore=server.store&trustStorePassword=mypassw
ord&keyStore=johndoe.pkcs12&keyStorePassword=secret");
```

See Also:

Java keytool https://docs.oracle.com/javase/8/docs/technotes/tools/windows/keytool.html documentation provides examples of how to generate a key pair, request a certificate from a CA, or generate certificates for a server.

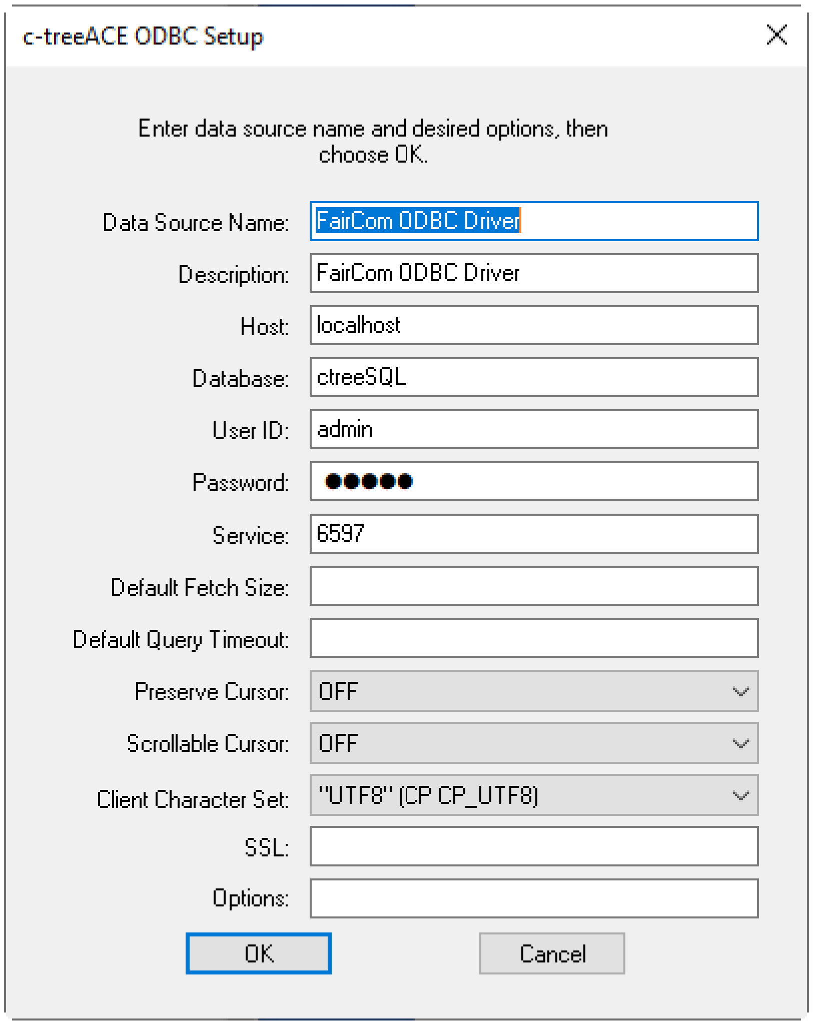
ODBC Support for SSL

You can use SSL with an ODBC data source. To configure SSL, open the ODBC Data Source Administrator.

Once you invoke the ODBC Data Source Administrator:

1. In the dialog box for the type of data source you choose, choose the **Add** button. The Add Data Source dialog box appears.
2. Select c-treeACE SQL from the list of installed drivers and choose **Finish**. The c-treeACE ODBC Setup dialog box appears.

3. Fill in the dialog box fields as shown in the following figure and choose **OK**. The ODBC Data Source Administrator writes the values you supply to *ODBC.INI* or to the DSN file you indicated.



Enter data source name and desired options, then choose OK.

Data Source Name:	FairCom ODBC Driver
Description:	FairCom ODBC Driver
Host:	localhost
Database:	ctreeSQL
User ID:	admin
Password:	●●●●●
Service:	6597
Default Fetch Size:	
Default Query Timeout:	
Preserve Cursor:	OFF
Scrollable Cursor:	OFF
Client Character Set:	"UTF8" (CP CP_UTF8)
SSL:	
Options:	

Data Source Name - A local name for the c-treeACE SQL data source for use in connect calls and by the ODBC Administrator.

Description - Optional descriptive text.

Host - Specify the machine name on which the c-treeACE SQL Server is running.

Database - The name of the database where the c-treeACE SQL data source resides.

User ID / **Password** - User name and password for connecting to the database. The driver uses those values if the application does not supply them in the call. You can leave these fields blank if you want the driver to use the defaults on the server. If no defaults are defined and you leave these fields blank, the user will be prompted when the application connects.

Service - The name of the Service c-treeACE SQL listens to. If empty, `sqlnw` is used.

Default Fetch Size - This value is the size (in bytes) used by the driver to fetch multiple rows from the server. It reduces network requests resulting in performance gains. If not set, the internal buffer size is 5000 bytes.

- In your connection string, set the attribute `"FETCH_SIZE=[number of bytes]"`
- In your ODBC.INI file, set the attribute `"Default Fetch Size=[number of bytes]"`

Connection string settings take precedence over DSN and *ODBC.INI* settings.

Default Query Timeout - It is possible to set the Default Query Timeout (in seconds) in the DSN and in the connection string.

- In the connection string, the attribute is: `QUERY_TIMEOUT=[number of seconds]`
- In the *ODBC.INI* file, the attribute is: `Default Query Timeout=[number of seconds]`

Settings in the connection string take precedence over the setting in the DSN or *ODBC.INI*.)

Preserve Cursor - This behavior is configurable from within the ODBC administrator.

Scrollable Cursor - This allows you to enable support for ODBC applications that require a scrollable cursor.

Client Character Set - Use the drop-down list to select the character set used by the client.

SSL - Enter optional parameters to configure the SSL. The following entries can be used to configure the SSL connection:
1) *empty* - Do not use SSL.
2) *BASIC* - Use SSL without certificate checking.
3) *<certificate file name>* - Use SSL with certificate checking using the certificate file specified. If no path is entered, the file must be in the current working directory.

Options - Enter any optional parameters to be included in the connect string.

4. The Data Source Dialog box reappears, and now includes the newly-added data source.

Configuring SSL in Your Connection String

In the ODBC connection string, it is possible to add "*SSL=xyz*" where "*xyz*" is one of the options 2 or 3 from the SSL parameters listed above in the c-treeACE ODBC Setup dialog box.

You can add one of the following to your existing ODBC connection strings to enable TLS/SSL:
"SSL=BASIC" - Encryption with default server certificate.
"SSL=ctsrvr.pem" - Use Peer authentication with explicitly named cert located in the local directory.
"SSL=C*\certs\ctsrvr.pem" - Use Peer authentication with an explicitly named cert with a full path.

OpenSSL Headers for Linking c-treeACE Client Applications

The OpenSSL headers have been included in this package for your convenience. These headers must be compiled and linked into your project. Examine your *opensslv.h* header in the *ctree.drivers/include/openssl* folder for the current version included in your build.

The OpenSSL include files are located under */ctree/include*.

Libraries are provided for multiple build platforms and are located in */ctree.drivers/lib/License.Lib/openssl*, where you'll find:

- *libeay32.lib*
- *ssleay32.lib*

Choose the appropriate build platform from the provided versions.

TLS/SSL Tutorials

As an example of linking with OpenSSL libraries, you can view the Tutorial makefiles and IDE Projects in the SDK directories. For example:

```
/linux.x64.64bit/sdk/ctree.ctdb/tutorials/cmdline/Makefile
```

9. Data Integrity - Dynamic Dump & Restore

Fast, safe, efficient hot data backups ensure you are prepared for nearly any unexpected circumstance.

9.1 Improvements in Dynamic Dump immediate restore

Improvements in dynamic dump provide greater control over immediate restore. Additionally, support has been added for sending immediate restore progress notifications to the client.

On Windows, an immediate restore of a dynamic dump by FairCom Server creates a command prompt window. To avoid this and to provide better control over the dump restore process, the launching of the dump restore process now uses the operating system's process creation function (rather than opening a shell or command prompt window).

!IMMEDIATE_RESTORE_WITH_PROGRESS

FairCom Server reads the output of the dump restore utility during immediate restore and sends progress notifications to the client that invoked the dump (if requested). To use the notification feature during an immediate restore of a dynamic dump, follow these steps:

1. Install a version of the **ctrdmp** utility that supports the `!IMMEDIATE_RESTORE_WITH_PROGRESS` option into a directory that is in FairCom Server's path.
2. Use the new dump script option `!IMMEDIATE_RESTORE_WITH_PROGRESS` in the dump script rather than the `!IMMEDIATE_RESTORE` dump script option.
3. Use the *-n* option when running the **ctdump** utility to request FairCom Server to send dynamic dump progress notifications to the **ctdump** utility.

!IMMEDIATE_RESTORE_LOGFILE

The `!IMMEDIATE_RESTORE_LOGFILE` option can be specified in a dynamic dump script to cause FairCom Server to write the output of the **ctrdmp** utility during the immediate restore to the specified file. This can be useful for troubleshooting failed immediate restores. For example, to write immediate restore output to the file *ctrdmp.log* in FairCom Server's working directory (in the LOCAL_DIRECTORY directory if that option is used), use this dump script option:

```
!IMMEDIATE_RESTORE_LOGFILE ctrdmp.log
```

Note that the `!IMMEDIATE_RESTORE_LOGFILE` option is only used by FairCom Server during the backup phase. It is ignored by the **ctrdmp** utility.

Developer Options

For developers who are invoking a dynamic dump and receiving notifications, the dynamic dump callback function will now receive event notifications during the immediate restore. When the user-defined dynamic dump callback function is invoked with the opcode parameter set to *ctDDCBKstatus*, the *pdata* parameter is of type pDDST. The DDST structure's phase field indicates which event occurred. The new event values for immediate restore are:

```
1. DDP_BEGINRESTORE: start of dump restore
2. DDP_BEGINFILE, DDP_PROGFILE, and DDP_ENDFILE events occur for each file restored from the backup.
3. DDP_ENDRESTORE: end of dump restore; pddst->errcod is zero if it succeeded or non-zero if it failed.
4. DDP_BEGINROLBAK: start of rollback to dump start time
5. DDP_ENDROLBAK: end of rollback to dump start time
6. DDP_BEGINCLNIDX: start cleaning index nodes
7. DDP_ENDCLNIDX: end of cleaning index nodes
8. DDP_BEGINUPDIFL: start of IFIL update
9. DDP_ENDUPDIFL: end of IFIL update
```

See **ctdumpCallback()** in *ctree\source\ctdump.c* for an example of a callback function that handles these events.

Stop when client callback function returns an error

If an error occurs in the client-side dynamic dump callback function, the server will not continue to perform the dump. In V11.2.3 and later, the logic has been modified so that any error returned by the client's dynamic dump callback function terminates the dynamic dump.

Note: This is a Compatibility Change.

9.2 Configurable file sync interval on dynamic dump

To reduce the amount of cache the file system devotes to the dynamic dump file, a server configuration keyword allows specifying a periodic file sync operation on the backup generated by dynamic dump approximately every `<size>` bytes written.

```
DYNAMIC_DUMP_SYNC_INTERVAL <size> [<scale>]
```

- `<size>` must be less than 4GB.   Default size is 0, which results in no explicit sync calls on the backup file. The minimum `<size>` is 1 MB. Non-zero values less than 1MB are increased to 1MB.

This value can be changed at runtime using menu option 10 in the **ctadmn** command-line utility.

9.3 Dynamic Dump now stops when client callback function returns an error

If an error occurs in the client-side dynamic dump callback function, the server will not continue to perform the dump. In V11.2.3 and later, the logic has been modified so that any error returned by the client's dynamic dump callback function terminates the dynamic dump.

Note: This is a Compatibility Change.

9.4 Dynamic Dump status and error messages

The dynamic dump client notification option sends a status message to the client when a file listed in the dump script fails to be opened (e.g., error **13** or **14**).

The message for error **442** for the **ctdump** utility prints a message when the dump completes but some files can't be included in the dump (error **442**).

An option (*-m*) was added to the **ctdump** utility to show minimal output (suppressing the % complete output).

Sample output:

```
# ctdump -u ADMIN -p ADMIN -s FAIRCOMS -t dump.scr -n -m

c-treeACE(tm) Version 11.3.0.7876(Build-160501) Dynamic Backup Utility

Copyright (C) 1992 - 2016 FairCom Corporation
ALL RIGHTS RESERVED.

Reading dump stream from server with buffer size of 100000


Error: Failed to back up file atomrd.idx: error code 13.

 Start dump. Estimated dump size:     94769152

Successfully backed up file atomrd.dat. File size: 65536
Successfully backed up file mark.dat. File size: 56868864
Successfully backed up file mark.idx. File size: 37814272
Successfully backed up file S0000000.FCS. File size: 128
Successfully backed up file S0000001.FCS. File size: 128
Successfully backed up file L0000529.FCS. File size: 2040

   End dump.    Actual dump size:     94779392

Dynamic dump completed, but some files could not be backed up.

Dynamic Dump completed, but some files could not be backed up (..\dump.scr). See Server Operation Log:
CTSTATUS.FCS (442)
```

9.5 Restore recovery options

The **ctrdmp** utility now supports the `RECOVER_DETAILS` and `RECOVER_MEMLOG` transaction recovery options (the same options that FairCom Server supports).

- If you specify `!RECOVER_DETAILS YES` in your dump restore script, **ctrdmp** will log progress messages to the file *CTSTATUS.FCS* as it performs its automatic recovery.
- If you specify `!RECOVER_MEMLOG N` in your dump restore script (where *N* is a number of transaction logs), **ctrdmp** will read up to that many transaction logs into a memory buffer and will use the memory buffer to read the transaction log entries. This can speed up dynamic dump restore recovery in cases in which the dump backup includes many transaction logs.

9.6 TRAN_RECOVERY - Restore recovery diagnostic logging

In V11.5 and later, the Dynamic Dump restore utility supports enabling diagnostic logging of its recovery process. The logging is the same transaction recovery logging that FairCom Server supports and is written to the file *RECOVERY.FCS*. This logging can be useful for analyzing the dynamic dump restore recovery process. Note that the logging can slow down the recovery time.

To enable this logging during dynamic dump restore, add the option `!DIAGNOSTICS TRAN_RECOVERY` to your dump restore script.

9.7 Forward Roll - ctfdmp now creates directory

The forward roll utility, **ctfdmp**, now enables automatic directory creation if the file create fails due to a missing directory. Prior to this revision a forward roll failed with error **17** regardless of the `!SKIP` setting. You no longer need to use the workaround, which was to manually create the directory structure required by the forward roll.

10. Advances in the Core Engine

Stability, performance, and efficiency with administrative ease: c-treeACE brings peace of mind to your most advanced installations.

10.1 FairCom Server

DISK_FULL_ACTION (SUBSYSTEM)

FairCom Server supports a configuration option to call an external script when a specified `DISK_FULL_LIMIT` or `DISK_FULL_VOLUME` limit is reached. This allows server administrator to configure an external process to run when a disk-full limit is reached, for example, the process could write information to a log or notify the administrator.

To configure an external process to run when the space on the specified volume is below the specified limit, set the `DISK_FULL_ACTION` configuration option in *ctsrvr.cfg* with the *run* action.

The supported syntax is:

```
SUBSYSTEM EVENT DISK_FULL_ACTION {
    volume      VOLUME
    limit       LIMIT
    run         EXE [OPTIONS]
    freq        FREQUENCY
    maxruntime  MAXRUNTIME
}
```

The *volume*, *limit*, and *run* parameters are required. The *freq* and *maxruntime* are optional.

- *VOLUME* is the name of the disk volume to check: a drive name (for example, *C:\*) on Windows systems, or a directory name (for example, */users/faircom/data*) on Unix/Linux systems.
- *LIMIT* is the threshold of available space that triggers the running of the script. The KB, MB, GB, TB, and PB suffixes can be used for this value (for example: "limit 100 GB" means 100 gigabytes).
- *EXE* is the name of an executable, batch file, or Unix shell script to run, and OPTIONS are optional command-line options to pass to the external program.
- *FREQUENCY* indicates the frequency at which this condition is checked. The default is to check the condition every 60 seconds. Minimum frequency is 1; maximum frequency is 32,000,000.
- *MAXRUNTIME* indicates the maximum amount of time, in seconds, that the external script is permitted to run. If the external script has not terminated after the specified amount of time, the server terminates the script. The default is 60 seconds. A setting of 0 means the script will never be forced to terminate.

Note: When setting a `MAXRUNTIME` value, consider that no other `DISK_FULL_ACTION` checks will take place until the script completes, regardless of the `FREQ` setting. This is of particular importance if more than one `DISK_FULL_ACTION` is created.

In V12 and later, disk full monitoring keywords have been added to the default *ctsrvr.cfg*. These keywords are present in this config file, but they have been commented out for so they are there if you need them, but they are NOT enabled.

```
; SUBSYSTEM EVENT DISK_FULL_ACTION {
    volume      VOLUME
    limit       LIMIT
    run         EXE [OPTIONS]
    freq        FREQUENCY
    maxruntime  MAXRUNTIME
}
```

Example DISK_FULL_ACTION with external process action

The following configuration option causes the batch file *diskfull.bat* to be run when writing to a c-tree data or index file detects that the space on *C:\* has dropped below 1 GB. This disk-full condition is checked once every 60 seconds.

```
SUBSYSTEM EVENT DISK_FULL_ACTION {
    volume     C:\
    limit      1 GB
    run        diskfull.bat
    freq       60
    maxruntime 120
}
```

Reading parameters sent from the server to the external process

FairCom Server passes a set of ASCII key/value pairs to the external process using standard input, where the input format is *key=value*, one per line. The keys that FairCom Server passes to the external process are:

- *volume* - The *VOLUME* setting from the `DISK_FULL_ACTION` configuration option.
- *limit* - The disk-full limit in bytes that was specified for that volume.
- *available* - The amount of disk space in bytes that is currently available on that volume.

Writing parameters from the external process to the server

The external process can optionally pass ASCII key/value pairs back to the server by writing them to standard output in the format key=value, one entry per line. FairCom Server recognizes the following keys:

- *action=continue* - (indicating to continue FairCom Server operation; this is the default)
  or
- *action=shutdown* - (indicating to shut down FairCom Server)
- *comments=MESSAGE* - A user-defined message that will be written to *CTSTATUS.FCS*.

Note: FairCom Server reads only the first 1024 bytes of the output data returned by the script. All data after the first 1024 bytes of output data is ignored.

If the external script's output is not in the supported format, a message is written to *CTSTATUS.FCS*. Example:

```
DISK_FULL_ACTION: Command diskfull.bat output does not match format var=value: line 3 column 8
```

Unlimited MAXRUNTIME script

An unlimited script runtime can be set with `MAXRUNTIME 0`. When configured as unlimited and a script is run due to the LIMIT and RUN options, the script will never be forced to terminate.

Note: When setting a `MAXRUNTIME` value, consider that no other `DISK_FULL_ACTION` checks will take place until the script completes, regardless of the `FREQ` setting. This is of particular importance if more than one `DISK_FULL_ACTION` is created.

Transaction log limit for replication and deferred indexing

Note: This is a Compatibility Change for c-treeACE V11.5 and later.

The Replication Agent and deferred index processing thread for *TRNLOG* files read transaction logs and inform the server of their minimum transaction log requirements respectively. In early V11 releases, c-treeACE retained all required logs indicated by these clients. In some cases, this resulted in the number of transaction logs to increase without limit, filling all drive space. A typical scenario is a Replication Agent taken offline without removing its minimum transaction log requirement from the server.

c-treeACE Server now limits by default the number of active transaction logs that are kept for Replication Agent and deferred index processing to 50. This default configuration is a deliberate trade off to avoid accumulating an unlimited number of active transaction logs and potentially running out of critical drive space.

The worst case scenario with this change is your Replication Agent or internal thread that is reading transaction logs can fail with error **96** when reconnecting and searching for a log that it expects present is no longer there. In this case, it will likely be necessary to resync your files through the Replication ReSync feature, a backup and restore, or manual file copy.

Two configuration options were introduced controlling these limits independently for these two features:

- `MAX_REPL_LOGS <max_logs>` - Sets the maximum number of logs to be held specifically for replication.
- `MAX_DFRIDX_LOGS <max_logs>` - Sets the maximum number of logs to be held specifically for the deferred indexing thread.

Use of either of these configuration settings does not impact your c-treeACE Servers' ability to retain any necessary logs required for Automatic Recovery.

The more comprehensive `KEEP_LOGS` configuration keyword always overrides both `MAX_DFRIDX_LOGS` and `MAX_REPL_LOGS` configuration settings in either case.

When a new transaction log is created and one of these limits is in effect and the required number of logs exceeds the limit, c-treeACE logs one of the following messages to *CTSTATUS.FCS* and sets the minimum log requirement of the deferred index thread or Replication Agent to a larger value equal to the log limit:

```
- User# 00039 Deferred index log requirement (current=42, required=32) exceeds MAX_DFRIDX_LOGS limit.
Setting lowlog to 33.
- User# 00038 Replication reader AGENT1 log requirement (current=43, required=32) exceeds MAX_REPL_LOGS
limit. Setting lowlog to 33.
```

These keywords can be set at runtime by using the command-line tool **ctadmn**, option 10 (Change the specified configuration option) and setting the new value, like this:

```
10. Change the specified configuration option
Enter your choice (1-10), or 'q' to return to previous menu>> 10
Enter the configuration option and its value >> MAX_REPL_LOGS 75
Successfully changed the configuration option.
```

They can be set programmatically by calling the **ctSETCFG()** function. Examples:

```
/* Set maximum logs for deferred index thread to 100. */
ctSETCFG(setcfgCONFIG_OPTION, "MAX_DFRIDX_LOGS", "100");
/* Set maximum logs for replication agents to 0 (no limit). */
ctSETCFG(setcfgCONFIG_OPTION, "MAX_REPL_LOGS", "0");
```

LOPN_ERR (96) - allow Automatic Recovery to continue

In unusual situations, index reconstruction during automatic recovery could fail with error **LOPN_ERR** (96) due to a missing transaction log (e.g., *L*.FCS*). Starting with c-treeACE V11.5, it is possible to use the configuration option `TRANIDX_LOPN_ERR_CONTINUE` to allow automatic recovery to continue by adding this keyword to *ctsrvr.cfg* and restarting the c-treeACE Server process.

If recovery then completes successfully, the only issues caused by the missing transaction log was an inability for the c-treeACE Server to properly update an index node on disk.

In this case, the data files under transaction logging control (e.g., with file mode *ctTRNLOG*) are fully intact, and only the indexes need to be rebuilt to ensure the data and index files are fully in sync. **Note, this is expected to be an unusual situation and FairCom recommends using extreme caution with this keyword. FairCom does not recommend enabling this keyword by default. It should only be enabled when required to recover from a catastrophic situation.**

To use this option, add `TRANIDX_LOPN_ERR_CONTINUE` to *ctsrvr.cfg*. When this option forces the automatic recovery to continue, the following message is logged to *CTSTATUS.FCS*:

```
WARNING: tranidx could not open previous log file: 96
```

Upon completion of automatic recovery in this situation, the following message is logged to *CTSTATUS.FCS*, and c-treeACE Server terminates with error **571** to indicate to the administrator that all indexes should be rebuilt:

```
NOTE: tranidx continued without required logs. Recovery completed but index rebuild is required.
Automatic recovery terminated with error: 571
```

If the automatic recovery does NOT complete successfully after adding this keyword, then the missing transaction log is preventing c-treeACE from being able to successfully recover from the catastrophic situation and a backup should be used to restore the data to a known state.

Reduced minimal index cache size requirements

Historically FairCom Server has automatically set the index cache size so that the number of index buffers is at least 3 times the maximum number of allowed connections. This ensures that sufficient index buffers are available to handle index operations (key inserts, retrievals, and deletes) if all supported connections are performing index operations at the same time.

This minimum requirement means the memory used by FairCom Server could be higher, perhaps much higher, than the amount requested in the server configuration file. For example, using the default settings of `PAGE_SIZE 8192` and `IDX_MEMORY 100 MB` and the number of allowed connection is larger than 4267, `IDX_MEMORY` becomes larger than requested.

The list below shows examples for `PAGE_SIZE 8192`, showing the original index cache memory requirements:

CONNECTIONS	IDX_MEMORY
8	0.2 MB
16	0.4 MB
32	0.8 MB
64	1.5 MB
128	3.0 MB
256	6.0 MB
512	12.0 MB
1024	24.0 MB
2048	48.0 MB
4096	96.0 MB
8192	192.0 MB
16384	384.0 MB
32768	768.0 MB

In V11.5 and later, instead of increasing the number of index buffers to the required amount, if the `IDX_MEMORY` value is smaller than the required amount of index buffers, c-tree Server limits the number of concurrent threads that can be performing index operations so that concurrent requests for index buffers do not exceed the allocated number of index buffers.

Note: This is a Compatibility Change.

FairCom Server supports a configuration option to restore the old behavior in case it is needed. Specifying `COMPATIBILITY MINIMUM_INDEX_BUFFERS` in *ctsrvr.cfg* restores the old behavior. When this option is used, at startup, FairCom Server logs the following message when it increases `IDX_MEMORY` due to minimum index buffer requirements based on the number of supported connections:

```
- User# 00001  Setting IDX_MEMORY to 53805056 due to concurrent connection requirements.
```

CRITICAL_SECTION_SPIN - Configurable mutex spin for Unix/Linux

In an earlier modification, support was added for a configurable spin count on critical sections for FairCom Server on Windows systems. This was done because it is more efficient to have threads spin trying to acquire the synchronization object (using atomic operations) rather than immediately calling a kernel function to block until the synchronization object can be acquired in cases in which a synchronization object such as a critical section or a mutex is held for a short period of time.

The configuration option `CRITICAL_SECTION_SPIN`, which controls this behavior, has been extended to Unix/Linux systems in V11.3.0 and later. The default value is 1000, meaning that we try up to 1000 calls to **pthread_mutex_trylock()** before making the blocking **pthread_mutex_lock()** call.

In a file open/close mutex contention test on a Linux system, a significant speedup was observed when using this option.

FairCom Server physical CPU counting method on AIX

Note: This is a Compatibility Change.

The physical CPU counting method used by FairCom Server on AIX has been changed to account for LPAR CPU assignment. The physical CPU count returns the total number of CPUs on the system. However, on AIX, an LPAR can be defined that has access to a subset of the CPUs. To accommodate this, we now base our count on the "maximum virtual CPUs" assigned to the LPAR.

Improved log message when PAGE_SIZE does not meet encryption block size requirements

The message logged to *CTSTATUS.FCS* when the `PAGE_SIZE` setting does not meet encryption block size requirements has been improved.   The `PAGE_SIZE` in *ctsrvr.cfg* must meet encryption block size requirements. Now, instead of terminating the server with an internal error code, a more descriptive error message is logged to *CTSTATUS.FCS*. Example messages:

```
PAGE_SIZE 768:
 - User# 00001  Configuration error: ctsrvr.cfg, line 127: The encryption block size of 2048 is required
to be a multiple of a PAGE_SIZE that is smaller than the encryption block size.

PAGE_SIZE 3072:
 - User# 00001  Configuration error: ctsrvr.cfg, line 126: A PAGE_SIZE that is larger than the encryption
block size of 2048 is required to be a multiple of the encryption block size.
```

Log additional detail on server shutdown

When a server shutdown is triggered by a client API request, we now log to *CTSTATUS.FCS* the c-tree username and additional details about the calling client depending on their communication protocol.

For shared memory, we log the client PID. Unix systems include TID and effective UID.

For TCP connections, we log the client IP address.

Example:

```
Fri May 12 10:52:31 2017
 - User# 00021  Shutdown request from user ADMIN. PID: 21419 TID: 21419 UID: 1000
Fri May 12 10:52:34 2017
 - User# 00001  Server shutdown initiated
Fri May 12 10:52:35 2017
```

FairCom Server lock wait-for graph now uses less memory

FairCom Server uses a two-dimensional array to track the wait-for relationships of locks, so that deadlocks can be detected and avoided. This array uses four bytes times the maximum number of connections squared. This means that a 32000 connection server uses 4 * 32000 * 32000 bytes for this array, which is over 4GB.

A redesign of the logic has substantially reduced memory usage. Here are some example memory use figures:

```
Connections    Original   Optimized
             Memory Use  Memory Use
   1024          577 MB      573 MB
   8192          839 MB      640 MB
  32000         4595 MB     1585 MB
```

Environment variable to enable c-tree advanced encryption

In V11.5 and later, c-tree supports enabling advanced encryption at run time using an environment variable. Set the environment variable `CTREE_ADVANCED_ENCRYPTION` to `YES` to enable advanced encryption if it is supported. This environment variable can be used to allow c-tree utilities to enable advanced encryption even if they haven't been updated yet to automatically enable advanced encryption when needed. Examples include the rebuild and compact utilities, **ctrbldif** and **ctcmpcif**.

Note: If c-tree does not support advanced encryption and this environment variable is set, the c-tree initialization will fail.

Support for commenting out an entire SUBSYSTEM block in ctsrvr.cfg

In V11.5 and later, FairCom Server supports commenting out a SUBSYSTEM block in *ctsrvr.cfg* by commenting out just the line containing the SUBSYSTEM keyword. Prior to this enhancement, each line of the SUBSYSTEM block had to be commented out. Now an entire SUBSYSTEM block can be commented as shown below:

```
;SUBSYSTEM EVENT DISK_FULL_ACTION {
    volume      VOLUME
    limit       LIMIT
    run         EXE [OPTIONS]
    freq        FREQUENCY
    maxruntime  MAXRUNTIME
}
```

Check for out-of-range value of FILES configuration option

The FairCom Server `FILES` configuration option has a limit of 32767. If a larger value was specified, it overflowed and produced an unexpected smaller effective value. For example, `FILES 70000` produced an effective `FILES` setting of `4475`. The range is now checked and an appropriate error message is logged for out-of-range values, either indicating that it is negative or that it exceeds the maximum value of 32767.

Server keyword to enable UNOD_ERR (51) diagnostic stack dump

In V11.5 and later, a server keyword can be used to enable **UNOD_ERR** (51) diagnostic stack dump. The keyword is `DIAGNOSTICS UNOD_ERR`. It can be turned on/off at runtime using the **ctSETCFG()** function or option 10 of the **ctadmn** utility.

10.2 Communications Layer

Shared Memory Protocol Change

A change has been made in V11.5 and newer to the c-treeACE shared memory client-side code to return the c-treeACE *sysiocod* value if the shared memory connection fails due to incompatibility. In this case, a message similar to the following will be written to the log:

```
The client's shared memory version (3) is not compatible with the server's shared memory version (4)
```

The values shown in parentheses indicate the shared memory protocol versions. The message above means that the client is using version 3 of the client protocol and the server requires version 4.

The following messages may appear as warning symptoms:

```
Mon Jul  2 10:51:05 2018
 - User# 00001      FSHAREMM: Protocol version: server=3 client=4
Mon Jul  2 10:51:29 2018
 - User# 00020      FSHAREMM: The client's shared memory version (3) is not compatible with the server's
shared memory version (4)
```

Note: SQL communication is NOT included in this change.

Two situations may be encountered that will not fall back to TCP/IP:

1. On Windows, it is possible for a shared memory connection attempt to return error **SHMP_ERR** (978) due to insufficient permission.
2. On Unix (and some very old Windows clients), it is possible for a shared memory connection attempt to return error **SHMC_ERR** (841) due to client/server using incompatible shared memory protocol versions.

Both of the above situations are different from the standard error **133** ("server not listening"), which would be retried using TCP/IP.

Compatibility Note: This change is considered a compatibility change as the shared memory communication protocol may not automatically fail over to TCP/IP support if the cause of the shared memory connection failure is permission related.

Server broadcast feature enabled on all platforms

The server broadcast feature is now enabled on all platforms. Previously, it was enabled only on Windows, Linux, and Mac OS X systems.

10.3 Indexing

Index Range Support for Virtual Segment Groups

This new range operator can retrieve key ranges defined over a group of contiguous key segments.

The c-treeACE range retrieval function (e.g., **AllocateRange**) previously operated at the key segment level. In V11.5 and later, it has been enhanced to support a range operator that can be used to perform retrievals on key ranges that are defined over a set of contiguous key segments (referred to here as a “virtual segment group”).

This feature allows you to search for all entries at the key level between two targets: from a low-key to a high-key and from a lower-bound to an upper-bound. By introducing support for grouping two or more contiguous segments into one virtual segment group, it is now possible to perform a new type of range retrieval.

As an example, consider a database that contains two fields of one character each:

```
"First",  CT_FSTRING, 1
"Second", CT_FSTRING, 1
```

The index would consist of two segments, one for each field.

For this example, assume the data in these fields is as follows:

```
(A, 1)
(A, 7)
(A, 9)
(A, 11)
(B, 1)
(B, 2)
(B, 5)
(C, 1)
(C, 2)
(C, 5)
```

It has always been possible to specify a key range that would get us all of the records where the first field is `A` or an even more complex range where the first field is between `A` and `B` and the second field is between `1` and `3`. That would get us the following records:

```
(A, 1)
(B, 1)
(B, 2)
(C, 1)
(C, 2)
```

A sequential range of records between `(A, 7)` and `(C, 2)` is a more challenging case. It would require maintaining a second index of a single segment that includes the first field and the second field. This solution was not possible unless the two fields were adjacent to each other and in the correct order in the database.

With virtual segment group support, we can specify a key range involving two or more index segments, which are adjacent in the index, and treat them as a single segment. The fields need not be adjacent in the database. In our example, that would allow us to specify a key range from `(A, 7)` to `(C, 2)`, which would get us the following records:

```
(A, 7)
(A, 9)
(A, 11)
(B, 1)
(B, 2)
(B, 5)
(C, 1)
(C, 2)
```

New Range Operator Modifiers

This revision introduces two range operator modifiers to be OR-ed in to the operator parameter values passed to **AllocateRange()**:

- CTIX_SSG marks the beginning of a segment group
- CTIX_ESG marks the end of a segment group

Example:

```
NINT oper[2],rc;

oper[1] = CTIX_BET | CTIX_SSG; /* first segment: start segment group */
oper[2] = CTIX_BET | CTIX_ESG; /* second segment: end segment group */
rc = AllocateRange(keyno, 2, ""A4"", ""C6"", &oper);
```

Temporary Index RVHD_ERR (123)

Note: This is a Compatibility Change.

c-treeACE temporary indexes are created on a per-user basis. This means that, if user A creates a temporary index over a shared data file and the shared data file is updated by user B, user A’s temporary index will not see user B’s update.

In this situation, it’s possible for **ITIM_ERR** (160) and **RVHD_ERR** (123) to occur. The c-treeACE logic has always internally dealt properly with the **ITIM_ERR** (160), by ignoring the **ITIM_ERR** (160) for temporary indexes and setting and *sysiocod* to **ITMP_COD** (-832).

RVHD_ERR (123) is ignored for temporary indexes and *sysiocod* is set to **ITMP_COD** (-832).

It is possible to disable this new behavior for **RVHD_ERR** (123) by adding `COMPATIBILITY TEMP_INDEX_ERROR` to *ctsrvr.cfg*.

Standalone FPUTFGET, index file open in exclusive mode

FPUTFGET rebuild performance has been improved. For an index file open in exclusive mode when the code is compiled with exclusive file open support, it was determined that an index flush to disk is unnecessary. The node will be written to disk later, either due to LRU reuse of the index buffer or when closing the file.

Note: Using a larger index cache size could improve the effect of this exclusive mode optimization because having more index buffers can mean less frequent need to reuse a buffer, which could require an updated buffer to be flushed.

10.4 File Management

Enable ctCHECKLOCK for specified data files at runtime

The FairCom Server now provides the ability to turn on the ctCHECKLOCK file mode at runtime, without having to change the application to specify this option when creating or opening files. A configuration file option and runtime API call can be used to turn ctCHECKLOCK on or off for the specified files.

Configuration Option

```
CHECKLOCK_FILE
```

Enables or disables *ctCHECKLOCK* mode for specified data files. This file mode can be invoked a number of ways. A file can be created with this set or it can be opened with this set. In V11.5 and later, you can use this server keyword to enable it dynamically.

When enabled, the server will block all updates to the file without a write lock, returning an error **57**. This allows you to debug situations where two processes might attempt to update the same record in an overlapping manner. This is primarily used in non-transaction files as transaction-controlled files always require a write lock for updates. You can use the wildcard character "*" to specify multiple files.

The following FairCom Server configuration option enables *ctCHECKLOCK* mode for data files whose name match *filespec*:

```
CHECKLOCK_FILE +filespec
```

- where *filespec* is the name of a file, which can include wildcards, as described in *c-treeACE Standard Wildcards*. *filespec* needs to match the actual file name including extension.

This option takes effect immediately: all currently open and subsequently opened or created data files whose name match *filespec* have *ctCHECKLOCK* enabled.

The following FairCom Server configuration option is used to undo the effect of the `CHECKLOCK_FILE` *+filespec*:

```
CHECKLOCK_FILE -filespec
```

This option disables the *ctCHECKLOCK* mode for data files whose name match *filespec* (which can include wildcards) and whose *ctCHECKLOCK* option was turned on by the *ctCHECKLOCK +filespec*.

This option takes effect immediately: all currently open data files whose name match *filespec* have *ctCHECKLOCK* disabled, and subsequently opened or created data files whose name match *filespec* will not have *ctCHECKLOCK* enabled (unless another CHECKLOCK option has been specified that matches the data file name, or the file mode at file open or create time specifies the *ctCHECKLOCK* mode).

This feature is also available at runtime via the **ctSETCFG()** API call.

Default: Off

Runtime Option

The **ctSETCFG()** API function can be used to turn on or turn off the ctCHECKLOCK file mode at runtime.

For example, to turn on ctCHECKLOCK for the file *test.dat* (immediately if it is already open, or the next time a file by this name is created or opened), call:

```
ctSETCFG(setcfgCONFIG_OPTION, "CHECKLOCK_FILE +test.dat");
```

To turn off ctCHECKLOCK for the file *test.dat* and prevent ctCHECKLOCK from automatically being turned on for this file the next time it is created or opened, call:

```
ctSETCFG(setcfgCONFIG_OPTION, "CHECKLOCK_FILE -test.dat");
```

As with the configuration option, this call to **ctSETCFG()** can specify a filename with wildcard characters.

Note: A call to **ctSETCFG(***setcfgCONFIG_OPTION, "CHECKLOCK_FILE -filespec"***)** will return **INOT_ERR** error 101 if no matching *filespec* is currently in effect.

Diagnostic Logging of DADV_ERR (error 57)

```
DIAGNOSTICS DADV_ERR
```

The configuration option `DIAGNOSTICS DADV_ERR` can be used to enable logging of a diagnostic message to *CTSTATUS.FCS* when a **DADV_ERR** error occurs. The message includes the name of the data file, the record offset at which the update was attempted without holding a lock, the c-tree function number and subfunction number, the caller's user ID, and the caller's node ID. Sample message:

```
Fri Mar 31 10:17:28 2017
 - User# 00022  DADV_ERR: file=qaopnfil.dat offset=18863 func=121(0) userid=ADMIN nodeid=qaopnfil043: 57
```

This diagnostic logging can be enabled and disabled at runtime.

To enable diagnostic logging:

```
ctSETCFG(setcfgDIAGNOSTICS, "CHECKLOCK");
```

To disable diagnostic logging:

```
ctSETCFG(setcfgDIAGNOSTICS, "~CHECKLOCK");
```

Using the **ctadmn** utility to enable or disable diagnostic logging:

```
10. Change Server Settings
 9. Change a DIAGNOSTICS option

Enter the DIAGNOSTICS option to enable or disable >> CHECKLOCK

Successfully changed the DIAGNOSTICS option.
```

NO_FIXED_OFFSET_CHECK - Prevent reads/writes to arbitrary offsets for fixed-length files

In V11.5 and later, a feature prevents corruption and errors when an attempt is made to write to arbitrary offsets in a fixed-length file. Checks were added to ensure that a read/write/lock call on a fixed-length file is properly aligned. If an incorrect alignment is detected, the call will fail with **FALG_ERR** (767).

This feature can be disabled by keyword `COMPATIBILITY NO_FIXED_OFFSET_CHECK`.

Note: This is a Compatibility Change.

10.5 Preventing Possible Data Loss with Compact & Rebuild Operations

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

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

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

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

Note: This is a compatibility Change.

Full List of Functions Changed

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

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

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

Default Behavior Changes the Original File

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

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

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

Compact and Rebuild Best Practice

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

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

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

For example:

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

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

10.6 ADDREC DELFLG_ERR (31) following recovery

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

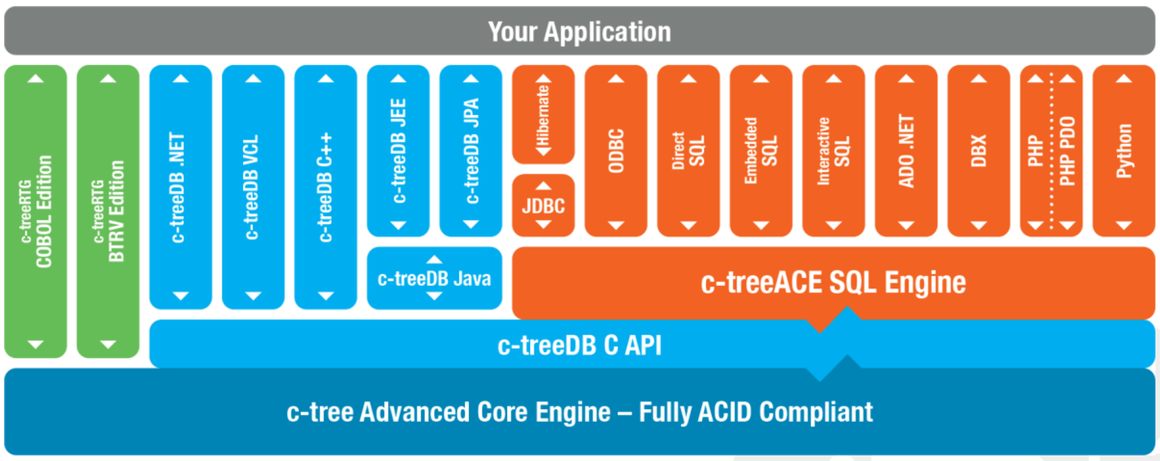
```
WARNING: an invalid delete stack has been reset to zero for fixed length data file
```

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

Note: This is a Compatibility Change.

11. Relational c-treeACE SQL

Multiple industry-standard c-treeACE SQL interfaces open your data to its fullest. Stay abreast of the latest c-treeACE SQL developments and additions in this section.



11.1 SQL Sequences

SQL sequence support

c-treeACE SQL Sequence support allows a sequence of numeric values to be created. The sequence can be used as a source of values, which can be used wherever numeric values in ascending or descending order are required.

CREATE SEQUENCE

Syntax:

```
CREATE SEQUENCE [owner_name.]sequence_name
                [ START WITH    start_value ]
                [ INCREMENT BY  increment_value ]
                [ MAXVALUE      max_value | NOMAXVALUE ]
                [ MINVALUE      min_value | NOMINVALUE ]
                [ CYCLE  |  NOCYCLE ]
```

Description:

The CREATE SEQUENCE command creates a new sequence in the current database.

- *sequence_name* is the name that is to be assigned to the sequence. The *sequence_name* in a database must be unique.
- *start_value* specifies a starting value for the sequence. The value of *start_value* must be between -9223372036854775808 and 9223372036854775807. The default is 1.
- *increment_value* specifies the value by which the sequence is incremented when sequence.NEXTVAL is called. Increment by can be positive or negative. The default is 1.
- *max_value* specifies the maximum value that can be returned by the sequence. If the sequence is a cycling sequence, then the next value after the *max_value* will be *min_value* for an ascending sequence and the next value after *min_value* will be *max_value* for an ascending sequence. Note that the value of *increment_value* may mean that the exact *max_value* or *min_value* will never be returned. The default is 9223372036854775807.
- *min_value* specifies the minimum value that can be returned by the sequence.   The default min_value is -9223372036854775808.

When *CYCLE* is specified, a sequence will wrap from the *max_value* to the *min_value* for an ascending sequence or from the *min_value* to a *max_value* for a descending sequence.   When *NOCYCLE* (the default) is specified, an error is returned when NEXTVAL (see *Sequence Values*) would exceed the *max_value* for an ascending sequence or the *min_value* for a descending sequence.

Example:

The following command creates a sequence called *myseq*. It starts with a value of 5, increments by 5, and returns a maximum value of 50.

```
CREATE SEQUENCE myseq START WITH 5 INCREMENT BY 5 MAXVALUE 50;
```

DROP SEQUENCE

Syntax:

```
DROP SEQUENCE sequence_name
```

Description:

The DROP SEQUENCE command deletes the specified sequence.

- *sequence_name* is the name of the sequence that is to be deleted.

Example:

To delete the sequence *myseq*, enter:

```
DROP SEQUENCE myseq ;
```

ALTER SEQUENCE

Syntax:

```
ALTER SEQUENCE sequence_name
    [ INCREMENT BY  increment_value ]
    [ MAXVALUE      max_value | NOMAXVALUE ]
    [ MINVALUE      min_value | NOMINVALUE ]
    [ CYCLE  |  NOCYCLE ]
```

Description:

ALTER SEQUENCE is a SQL command used to alter one or more characteristics of a sequence.

- *sequence_name* is the name of the sequence that is to be modified.
- *increment_value* specifies a new increment by value to be associated with the sequence.
- *max_value* specifies a new maximum value to be associated with the sequence.
- *min_value* specifies a new minimum value to be associated with the sequence.
- *cycle* specifies if the sequence should cycle or not.

Note that it is possible to put the sequence into a state from which no further values can be returned through the use of the alter command. For example, if the new *max_value* is less than the current value of the sequence for an ascending sequence, or if the new *increment_value* would make the next value be outside of the sequence bounds.

Sequence Values

Sequence values in SQL statements can be referred as follows:

CURRVAL: Returns the current value of a sequence

NEXTVAL: Increments the sequence and returns the next value

You must qualify CURRVAL and NEXTVAL with the name of the sequence:

- sequence.CURRVAL
- sequence.NEXTVAL

Example:

```
select mysequence.nextval;
insert into mytable values (mysequence.nextval, 'aaa');
```

Error Codes

-20265 "Sequence with the same name already exists"

-20266 "Sequence cannot be used here"

-20267 "Sequence not found"

-20268 "START-WITH/CURRENT-VALUE cannot be greater than MAXVALUE"

-20269 "START-WITH/CURRENT-VALUE cannot be less than MINVALUE"

-20270 "Invalid sequence MINVALUE specified"

-20271 "Invalid sequence INCREMENT specified"

-20272 "START-WITH cannot be altered in sequence"

-20273 "No options specified for ALTER SEQUENCE"

-20274 "Sequence increment has exceeded MAXVALUE"

-20275 "Sequence decrement has exceeded MINVALUE"

-20276 "Only SELECT and ALTER privileges are valid for sequences"

Improvements to Sequence API operation involving active transactions

In V11.5 and later, FairCom sequence support has been improved as follows:

1. By default, sequence creates and deletes within an active transaction use Immediate Independent Commit Transactions (IICT) to commit immediately. If the sequence attribute *ctSEQTRNDEP* is specified when creating the sequence, the creation and deletion of the sequence is committed only when the transaction commits. This option makes it possible to rely upon a transaction abort to undo the sequence create or delete.
2. The functions that update the sequence, **ctSetSequenceAttrs()**, **ctSetCurrentSequenceValue()**, and **ctGetNextSequenceValue()**, commit their changes immediately if possible. If the changes cannot be immediately committed, rather than failing with error 935 (**IICT_FIL**), the changes commit when the transaction commits. The case where these functions cannot immediately commit is when one of these functions is called in the same transaction that created the sequence, and the sequence is using the *ctSEQTRNDEP* option. In that situation, an IICT cannot be used because the file has been updated in the transaction, and so the sequence record remains locked until the transaction commits or aborts.
3. **ctSetSequenceAttrs()**, **ctSetCurrentSequenceValue()**, and **ctGetNextSequenceValue()** left the sequence record locked until the transaction committed. In V11.5 and later, these

functions ensure that the record is unlocked before they return (except for the case mentioned in point 2 above, where a transaction is active and IICT cannot be used).

4. **ctCreateSequence()** and **ctDeleteSequence()** failed with error 588 (**CPND_ERR**) if called within an active transaction and *OPS_DEFER_CLOSE* was not in effect. In V11.5 and later, we temporarily enable *OPS_DEFER_CLOSE* in this situation to avoid the error.

11.2 SQL Language

SQL - Create [if not exists] and Drop [if exists]

The CREATE TABLE syntax has been expanded to avoid failure in case the table exists (during create) or does not exist (during drop). The feature is similar to syntaxes found in the MySql and Postgres dialects.

We now implement this feature for tables, indexes, procedures, triggers, and functions:

```
CREATE TABLE [IF NOT EXISTS]...
DROP TABLE [IF EXISTS]...

CREATE INDEX [IF NOT EXISTS]...
DROP INDEX [IF EXISTS]...

CREATE PROCEDURE [IF NOT EXISTS]...
DROP PROCEDURE [IF EXISTS]...

CREATE TRIGGER [IF NOT EXISTS]...
DROP TRIGGER [IF EXISTS]...

CREATE FUNCTION [IF NOT EXISTS]...
DROP FUNCTION [IF EXISTS]...
```

SQL - SET IDENTITY_INSERT support

In V11.5 and later, support for `SET IDENTITY_INSERT...` syntax has been added.

Description

Applies or removes the IDENTITY_INSERT property to the specified table.

Syntax

```
SET IDENTITY_INSERT tablename { ON | OFF }
```

Arguments

- *tablename* - The name of a table with an identity column.

Notes:

Only one table in a session can have the IDENTITY_INSERT property set to ON. If a table already has this property set to ON, and a SET IDENTITY_INSERT ON statement is issued for another table, an error will be returned.

If the value inserted is larger than the current identity value for the table, the server uses the new inserted value as the current identity value.

The setting of SET IDENTITY_INSERT is applied at execute time, not at parse time.

Permissions

User must own the table or have ALTER permission on the table.

SQL - COUNT fix and new COUNT_BIG support

Running **select count(*)** returned the wrong result when the number of rows was greater than 2^31-1 (the maximum value that can be stored in a SQL INTEGER type). The COUNT aggregate function returns an INTEGER and, if the returned number overflowed the maximum allowed value, it returned a bad value instead of returning an error. This behavior has been fixed and an overflow error is properly returned.

To resolve the problem of counting rows when they are more the 2^31-1, a new aggregate function **COUNT_BIG** has been added in V11.5 and later. This function is like **COUNT** except that it returns a BIGINT instead of an INTEGER.

11.3 c-treeACE SQL Import

CTSQLIMPOPTS structure prior to V11.2.3

If you are using a version of c-treeACE *prior to V11.2.3*, the CTSQLIMPOPTS structure is as follows. (If you are using the current version of c-treeACE, see the **ctSQLImportTable** (https://docs.faircom.com/doc**/ctreeplus/ctsqlimporttable.htm**) documentation):

```
typedef struct tagCTSQLIMPOPTS {
     pTEXT   tblnam; /* name of table to import                          */
     pTEXT   symnam; /* (-n) symbolic table name                         */
     pTEXT   prefix; /* (-q) table name prefix                           */
    pTEXT   dbsnam; /* (-d) name of c-treeSQL database (default: ctreeSQL)   */
    pTEXT   srvnam; /* (-s) c-treeACE SQL Server name (default: FAIRCOMS)    */
    pTEXT   usrnam; /* (-u) userid for connecting to c-treeACE SQL Server    */
    pTEXT   usrpwd; /* (-a) password for authentication                      */
     pTEXT   tblown; /* (-o) assign table owner                          */
     SQLCBF  clbkfn; /* callback function                                */
    NINT    chkfld; /* (-k) skip fields that don't comply with
                        conventional identifier rules                    */
     NINT    skpidx; /* (-x) skip indexes                                */
     NINT    rmlink; /* (-r) unlink table from database                  */
    NINT    frctbl; /* (-c) allow table names that don't comply with         */
    NINT    grntpb; /* (-b) grant public access permissions                  */
    NINT    nonint; /* (-i) non-interactive mode: ignore errors and continue */
} CTSQLIMPOPTS, *pCTSQLIMPOPTS;
```

COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD

A new compatibility keyword has been added to instruct the Sqlize process in FairCom RTG (and **ctSQLImportTable** and the **ctsqlimp** utility) to verify the admin password (if any):

```
COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD
```

At this time this keyword affects only FairCom RTG or Server SDK compiles calling **ctSQLImportTable** on the server-side reusing the current session.

This server configuration option can be changed at runtime using the **ctadmn** utility or the **ctSETCFG()** API function.

- **To enable this option using ctadmn**:
  Choose option 10, then option 10, then enter:
  `COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD`
- **To disable this option using ctadmn**:
  Choose option 10, then option 10, then enter:
  `COMPATIBILITY ~SQLIMPORT_ADMIN_PASSWORD`

See also:

- *ctSQLImportTable function enhancements* (page 85)

ctsqlimp removed synonyms and grants on table removal

The following items have changed with c-treeACE V11.5 with respect to import tables and granting permissions. Prior to c-treeACE V11.5, after using **ctsqlimp** (or **ctutil** *-sqlize* for FairCom RTG) to import a table and grant permissions on it and then re-import the table, the permissions remained unmodified.

After unlinking a table using or **ctsqlimp** *-r* (or **ctutil** *-sqlunlink*), the system table contained permissions and synonyms on the removed table, which then took effect on a table link or creation with the same name as the removed table.

Other symptoms included:

1. Increasing duplicate entries in the internal c-treeACE SQL Server dictionary *systabauth* table for tables relinked multiple times.
2. Internal errors when running statements using synonyms for unlinked tables.
3. Queries entering infinite loops while resolving synonyms.

The logic has been modified to correct these symptoms.

Note: This results in a **behavior change** - After relinking a table, permissions and synonyms get reset (earlier they were left untouched).

Ability to promote INT8U columns to NUMERIC

The ability to promote INT8U columns to NUMERIC during the SQL import process has been added in V11.5 and later. This modification allows unsigned 64-bit integers to be promoted to NUMERIC(20) when the *-p* option is presented to **ctsqlimp**.

Improved scrollable cursor support

Scrollable cursor support was introduced in c-treeACE SQL V11. In V11.5, server-side logic and JDBC driver-side logic have been enhanced to include the following support:

- The methods **isBeforeFirst()**, **isFirst()**, **isLast()**, and **isAfterLast()** now return appropriate values for an empty resultset.
- Using a negative value with **absolute()** is now supported.
- Wrong results from **isBeforeFirst()**/**isAfterLast()** after calls to various positioning functions (next, prev, absolute, relative...) now provide proper results.

These modifications should greatly enhance scrollable cursor usage.

11.4 Escape field name matching reserved words in c-tree expressions

In V11.5 and later, it is possible to escape the field names by wrapping them in square brackets, thus identifying them directly as fields so they do not conflict with function names.

This is useful if you have a field name that is the same as a function name. For example, if you have a field named "year" and you want to set a filter on year being 2000, so the filter string would be "year = 2000". The parser would fail because it identified "year" as a function with the wrong syntax.

The new syntax would be "[year] = 2000", which the parser unambiguously interprets as a field named "year".

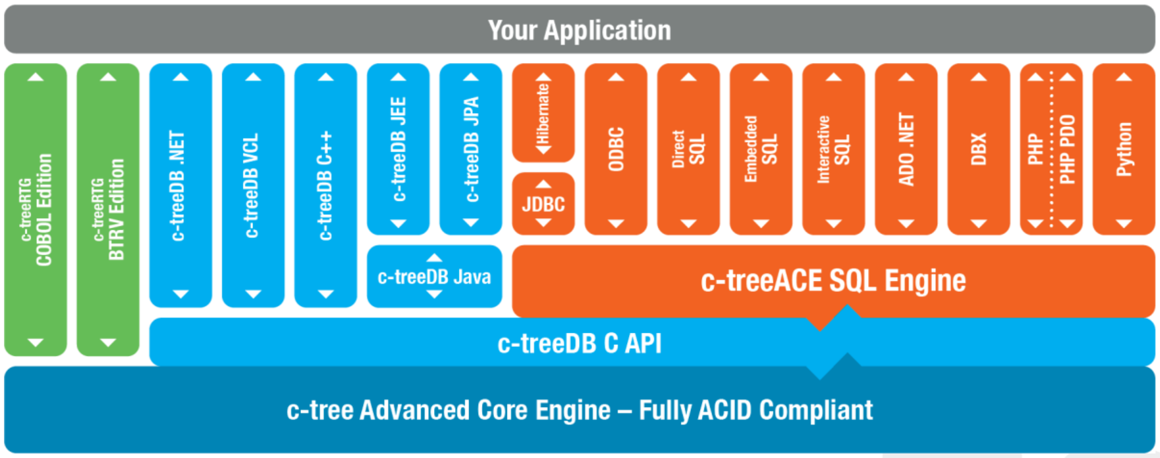
The parser allows the content of "`[]`" to be composed by any character excluding "`]`", space, tabs, newline, and linefeed.

11.5 Stored procedure to retrieve SQL database version

A new stored procedure, **fc_db_version()**, has been added in V11.5. This procedure returns the version of the SQL database.

12. High-Velocity c-treeACE APIs

The interfaces are your application's way of accessing the many features and benefits of c-treeACE. The c-treeACE High-Velocity, "NoSQL," APIs provide developers with extreme control over their data storage solution. This section lists changes in those APIs.



12.1 FairCom DB API NAV API

FairCom DB API - New ctdbAddRowIdSegment function

This function provides a way to add a new segment to the index referenced by the Handle parameter based on the $ROWID$ field:

```
ctdbAddRowIdSegment(CTHANDLE Handle);
```

This allows a part of the index to include a segment containing the internal ROWID field value.

Once such a segment exists, the table creation may fail with error **CTDBRET_ROWIDSEG** indicating an attempt to create an index with ROWID reference on a table with no ROWID support.

ctdbTimeMsecToString and ctdbStringToTimeMsec

New FairCom DB API functions allow you to convert a string to a *timems* value (time with milliseconds).

The string format does not support milliseconds: the milliseconds will either be lost (toString) or set to 0 (toTimeMsec). In the future the string format will be extended to support milliseconds.

ctdbTimeMsecToString

Convert a packed CTTIMEMS into a string

Prototype:

```
CTDBRET ctdbDECL ctdbTimeMsecToString(CTTIMEMS Time, CTTIME_TYPE TimeType, pTEXT pStr, VRLEN size)
```

Parameters:

- *Time* [IN] - The Time value to convert
- *TimeType* [IN] - One of the time types
- *pStr* [OUT] - Time in string format
- *size* [IN] - Size of the Time parameter

Returns:

Return CTDBRET_OK on success.

ctdbStringToTimeMsec

Convert a time in string format to CTTIMEMS packed format.

Prototype:

```
CTDBRET ctdbDECL ctdbStringToTimeMsec(pTEXT pStr, CTTIME_TYPE TimeType, pCTTIMEMS pTime)
```

Parameters:

- *pStr* [IN] - Time in string format
- *TimeType* [IN] - One of the time types
- *pTime* [OUT] - Packed time (with milliseconds)

Returns:

Return CTDBRET_OK on success.

FairCom DB API new flags for CTDB_BINARY_FLAG

A new set of Binary Flags has been introduced for the CTDB_BINARY_FLAG enum in *CTDBSDK.H*:

```
CTDB_FIELDFLAG_USER1  = 8,  /* field used to store USER type */
CTDB_FIELDFLAG_USER2  = 9,  /* field used to store USER type */
CTDB_FIELDFLAG_USER3  = 10, /* field used to store USER type */
CTDB_FIELDFLAG_USER4  = 11, /* field used to store USER type */
CTDB_FIELDFLAG_USER5  = 12, /* field used to store USER type */
CTDB_FIELDFLAG_USER6  = 13, /* field used to store USER type */
CTDB_FIELDFLAG_USER7  = 14, /* field used to store USER type */
CTDB_FIELDFLAG_USER8  = 15, /* field used to store USER type */
CTDB_FIELDFLAG_USER9  = 16, /* field used to store USER type */
CTDB_FIELDFLAG_USER10 = 17, /* field used to store USER type */
```

These values can be used to indicate particular characteristics of a field definable by the programmer bindable to the field definition by calling **ctdbSetFieldBinaryFlag**. A natural use of these flags is to aid FairCom DB API data-type conversion callbacks to identify which fields need conversions and which type of conversion they need.

12.2 c-treeACE ISAM

OPNRFILX() modification

In V11.5 and later, the **OPNRFILX()** function has been modified so that, when it is called with a negative data file number, if the open fails, **isam_fil** is returned as a relative file number:

isam_fil = 0 indicates the data file open failed.
isam_fil = 1 indicates the first index open failed, etc.

Note: This feature is a *Compatibility Change*.

Prior to this change, **OPNRFILX()** set **isam_fil** to the file number that was not possible to open. However, it did not return useful information when it was called with a negative data file number, such as **OPNRFIL(***-1***)**.

ctquiet options to support incremental backup restore points

The **ctquiet** utility will put the FairCom Server into a quiet state using the c-tree ISAM **ctQUIET()** function.

Several changes have been made to **ctquiet** to support restore points necessary to carry out an incremental backup strategy:

- The *-r* feature can be used to generate the restore point files necessary to carry out an incremental backup strategy.
- The *ctQTfailAfterTimeout* mode has been extended to apply to restore point checkpoint creation for incremental forward rolls.
- The *-x* option applies the *ctQTfailAfterTimeout* mode to the already existing options.
- The *-t* timeout output has an effect only if *-f* or *-r* is specified.
- The default timeout has been changed to match the server's maximum.

The new syntax is:

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

- *-s* - Server (default: FAIRCOMS@localhost).
- *-f* - Full consistency (default: crash consistency).
- *-r* - Create an incremental backup restore point. Automatically unquiets server.
- *-u* - Unquiet server.
- *-x* - Abandon Quiet request if timeout exceeded. (Default: abort long transactions).
- *-x* - Apply *ctQTfailAfterTimeout* mode. This mode has been extended to apply to restore point checkpoint creation for incremental forward rolls.
- *-t* - Seconds to wait before aborting transactions (the default timeout matches the server's maximum). This option has an effect only if *-f* or *-r* is specified.
- *-w* - Execute command on successful quiet. Waits for SIGINT to unquiet server.
- *-a* - Authentication file name.
- *-p* - Admin Password.

FairCom.Isam.dll - Added ctVerifyFile API to isam.net.cs DLL

The **ctVerifyFile()** API has been added to the FairCom.Isam.dll, *isam.net.cs DLL*.

Checking if ctThrdInit() has been called

(In V11.5 and later) In the server, multi-threaded client, and multi-threaded standalone c-tree operational models, it is possible to determine if **ctThrdInit()** has already been called by calling **ctThrdInit()** as follows:

```
NINT rc;
rc = ctThrdInit(0,ctThrdCHECK_INIT,NULL);
```

When a mode of `ctThrdCHECK_INIT` is specified, **ctThrdInit()** returns a value of YES if a successful **ctThrdInit()** call has already been made and returns NO otherwise. In both cases it does not perform any initialization.

12.3 FairCom Low-Level

SYSCFG() cfgCFGCHK bit to indicate if advanced encryption is enabled

The **SYSCFG()** function's *cfgCFGCHK* element is a 32-bit value that contains bits to indicate if particular features are enabled or disabled at runtime.

In V11.5 and later, a bit has been added to indicate if the advanced encryption feature is enabled:

```
#define cfgCFGCHKadvenc ((LONG)0x00000008) /* advanced encryption enabled */
```

13. Extended c-treeACE Features

c-treeACE maintains its reputation for innovative features that extend its usefulness well beyond traditional databases. This section presents our latest advanced data-management features.

13.1 Multi-Record Type (MRT Tables) Updates

ctdbCreateMRTTable behavior change

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

- CTCREATE_NODELFLD
- CTCREATE_NOROWID
- CTCREATE_NONULFLD

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

Note: This constitutes a Compatibility Change.

SQL Drop Table on MRT Tables now removes physical file

After importing in SQL using **ctsqlimp** (or **ctutil** *-sqlize* for FairCom RTG), a SQL Drop Table should remove the table from disk, which is the expected behavior.

A Multi-Schema table (also called a Multi-Record Table, or MRT) is linked to multiple SQL tables. When removing them, it is expected that the physical file does not go away, otherwise all other defined tables would be "dangling" with no physical file. When the last table belonging to an MRT table is removed, it can be desirable to remove the physical file for consistency with non-MRT tables.

A new configuration option `SQL_OPTION DEL_MRT_HOST YES` in *ctsrvr.cfg* enables the removal of the physical file after all MRT tables have been dropped. When this option is omitted or set to `NO`, the physical table is not deleted by SQL Drop Table statements.

ctdbGetRecordCount behavior on MRT Table modified

In V11.5, the behavior of the FairCom DB API function **ctdbGetRecordCount** has been modified when used on Multi-Schema Tables (MRT Virtual Tables). When used on MRT Tables, **ctdbGetRecordCount()** used to fail with error code **CTDBRET_NOTYET**.

The behavior on MRT Tables has been changed as follows:

1. return the error **CTDBRET_ISVTABLE** (4155)
2. set the *pCount* to the number of total entries in the physical table.

The effect of this change is to return an error as before, although more specific, and also return the number of records in the physical table.

Note: This is a Compatibility Change.

MRT Table support added to Java interfaces

As part of the effort to add methods for MRT Tables to C++, Java, and .NET API stacks, the following methods have been added to the .NET low-level Java class:

- public static CTDBRET CreateMRTTable(IntPtr Table, String Name, String ParentName, CREATE_MODE CreateMode, String Filter)
- public static CTDBRET AddMRTTable(IntPtr Database, String Name, String ParentName, ULONG Info)
- public static ULONG GetVTableNumber(IntPtr Table)
- public static bool IsVTable(IntPtr Table)
- public static CTDBRET RemoveVTableResource(IntPtr Table, NINT Number)
- public static CTDBRET SetMRTTableFilter(IntPtr Table, String Condition)

The following methods have been added to the Java high-level *CTTable* class:

- public virtual void CreateMRTTable(String Name, String ParentName, CREATE_MODE CreateMode, String Filter)
- public virtual void AddMRTTable(IntPtr Database, String Name, String ParentName, ULONG Info)
- public virtual ULONG GetVTableNumber()
- public virtual bool IsVTable()
- public virtual void RemoveVTableResource(IntPtr Parent, NINT Number)
- public virtual void SetMRTTableFilter(String Condition)

13.2 Record update callback

An application that uses the record update callback function feature needs to know what operations are associated with a particular transaction and what are the first and last operations for that transaction. In V11.5 and later FairCom Server provides this information to the record update callback function:

- The first operation in a transaction has the ctDFR_TRANFRS bit set in the status1 field of the *pdfrky* parameter that is passed to the record update callback function.

  The last operation in a transaction has the ctDFR_TRANLST bit set in the status1 field of the *pdfrky* parameter that is passed to the record update callback function. So, if a transaction has only one DFRKEY entry, that entry will have both the ctDFR_TRANFRS bit and the ctDFR_TRANLST bit set. If a transaction has more than one DFRKEY entry, the first entry will have the ctDFR_TRANFRS bit set and the last entry will have the ctDFR_TRANLST bit set.
- An optional parameter of type pRUCBSTT is passed to the record update callback function. This structure has the following definition:

  ```
  typedef struct rucbstt_t {
      LONG    verson; /* structure version   */
      LONG    avail;  /* padding- available for use  */
      LONG8   tranno; /* transaction number for the operation */
  } RUCBSTT, *pRUCBSTT;
  ```

Note: For FairCom Server to pass this third parameter to your record update callback function, your record update callback DLL or shared library must export the function **rucbCheckVersionCallback()**, which has the following function prototype:

```
NINT rucbCheckVersionCallback(pRUCBACB prucbacb,pNINT pversion);
```

prucbacb is the record update callback definition in the format of the record update callback add operation structure, RUCBACB. The function can choose to examine the record update callback definition to decide which version of the RUCB API it supports. For example, it can choose based on the callback name (`prucbacb->cbname`) or the name of the record update callback function (`prucbacb->fncnames[2]`).

The function should set *pversion* to the version of the record update callback API your DLL uses and return zero to indicate success. Supported versions are:

- a) Version 1 of record update callback API. Your record update callback function must conform to the following prototype:
  `NINT rucbRecordUpdateCallback (pRUCBF prucbf,pRUCBO prucbo);`
- b) Version 2 of record update callback API. Your record update callback function must conform to the following prototype:
  `NINT rucbRecordUpdateCallback (pRUCBF prucbf,pRUCBO prucbo,pRUCBSTT prucbstt);`

If your record update callback DLL does not export a function named **rucbCheckVersionCallback()**, FairCom Server uses version 1 of the record callback API.

ctRecordUpdateCallbackControl - Option to update all existing records

The default when adding a callback is to not call it for existing records. An option can be specified to choose a new behavior of calling the callback for all existing records. This option, *RUCBcallexist*, is specified by OR'ing it into the *calltm* field of the RUCBACB structure when calling **ctRecordUpdateCallbackControl()**.

13.3 Automatic restore point logging

FairCom Server supports a configuration option to automatically perform restore points. This option can be set in the configuration file using the `SUBSYSTEM TRNLOG AUTO_RESTORE_POINT` block. The supported sub-options are:

- `LOG_INTERVAL` *N*, where is the number of logs between each automatic restore point. Specify 1 to write an automatic restore point to every log. Specify 0 to turn off automatic restore points. `LOG_INTERVAL` defaults to zero (no automatic restore points).
- `TIMEOUT` *T*, where *T* is a time in seconds that the automatic restore point waits for active to transactions to commit. `TIMEOUT` defaults to 2 seconds.
- `FAIL_AFTER_TIMEOUT` *F*, can be *YES* or *NO*: *YES* indicates that if transactions remain active after the restore point timeout period, the restore point call fails; *NO* indicates that if transactions remain active after the restore point period, those transactions are aborted and the restore point is logged. `FAIL_AFTER_TIMEOUT` defaults to *NO*.
- `CHECKPOINT` *C*, where *C* is *YES* or *NO*. *YES* indicates that a checkpoint is to be logged with the restore point; *NO* indicates that no checkpoint is to be logged with the restore point. `CHECKPOINT` defaults to *NO*.

Example:

```
SUBSYSTEM TRNLOG AUTO_RESTORE_POINT {
    ; write an automatic restore point every 2 logs
    LOG_INTERVAL        2

    ; wait for up to 3 seconds for transactions to finish
    TIMEOUT             3

    ; fail if transactions remain active after timeout
    FAIL_AFTER_TIMEOUT  YES

    ; write a checkpoint
    CHECKPOINT          YES
}
```

Changing automatic restore point settings at runtime

Automatic restore point settings can be changed at runtime by using the **ctSETCFG()** API function. For example:

```
rc = ctSETCFG(setcfgCONFIG_OPTION, "subsystem trnlog auto_restore_point {\n\
 log_interval 2\n\
 timeout 3\n\
 fail_after_checkpoint yes\n\
 checkpoint yes\n}");
```

ctadmn use of ctSETCFG()

The **ctadmn** utility's "Change the specified configuration option" option uses this function, and it now detects when a SUBSYSTEM configuration option is specified. In that situation, it prompts for the SUBSYSTEM sub-options. Here is an example demonstrating how to use **ctadmn** to turn off automatic restore points at runtime:

```
10. Change Server Settings
10. Change the specified configuration option
Enter the configuration option and its value >> subsystem trnlog auto_restore_point {
Enter the SUBSYSTEM options you wish to change, one per line.
Finish with a line containing only a closing curly brace: }
To cancel, enter a line containing only an asterisk: *
Enter next line >> log_interval 0
Enter next line >> }
Successfully changed the configuration option.
```

If an error occurs when changing a SUBSYSTEM option using the **ctSETCFG()** function, the function returns an error code and may log a descriptive error message to *CTSTATUS.FCS*. Here is an example, showing a typo in which `log_interval0` is specified instead of `log_interval 0`:

```
Enter the configuration option and its value >> subsystem trnlog auto_restore_point {

Enter the SUBSYSTEM options you wish to change, one per line.
Finish with a line containing only a closing curly brace: }
To cancel, enter a line containing only an asterisk: *

Enter next line >> log_interval0

Enter next line >> }

Error: Failed to change the configuration option: 749
```

The following error message will be logged to *CTSTATUS.FCS*:

```
Wed Jan 18 14:42:53 2017
 - User# 00018  Configuration error: ctSETCFG(), line 2: The option LOG_INTERVAL0 is not supported in the
TRNLOG AUTO_RESTORE_POINT subsystem block.
```

Monitoring automatic restore points

The system snapshot structure, `ctGSMS`, now includes fields for the automatic restore point settings. The system snapshot version has been updated from 19 to 20:

```
 LONG sarplogint; /* auto restore point log interval */
 LONG sarptimout; /* auto restore point tran timeout */
 LONG sarplogint; /* auto restore point log interval */
 LONG sarptimout; /* auto restore point tran timeout */

/* Auto restore point options bits (ctGSMS sarpoptions field): */
#define ctARP_FAIL_AFTER_TIMEOUT 0x00000001 /* fail if trans active after timeout */
#define ctARP_WRITE_CHECKPOINT   0x00000002 /* write a checkpoint */
```

A text snapshot now also includes the automatic restore point settings:

```
        automatic restore point interval: 2
           active transaction time limit: 3
      fail if trans active after timeout: yes
                        write checkpoint: no
     log of last automatic restore point: 81
```

The **ctsnpr** utility (included in c-treeACE PRO packages in the source directory) has been updated to support the latest version of the text snapshot output format.

13.4 Restore point file name now includes log number

A change has been made in V11.2.3 to simplify determining which transaction logs are required for a restore to use this restore point. The restore point file naming has been updated to include the log number of the checkpoint that the restore point references.

- This affects names of lightweight and checkpoint restore points.

The new formatting is as follows:

```
RSTPNT_CHKPNT.L#######.YYYYMMDD.HHMMSS.FCS
RSTPNT_NO_CHK.L#######.YYYYMMDD.HHMMSS.FCS
```

Where `#######` is the transaction log number of the associated checkpoint.

Note: This is a **Compatibility Change** due to the new naming of restore point files.

13.5 Option to specify external library name in platform-independent format

Features such as the record update callback function use an external shared library or DLL that c-tree loads to support user-defined callback functions. The library name is stored in a resource in the file. Because the name was platform-dependent (starting with lib and ending in `.so` on Unix and ending with `.dll` on Windows for example), a data file containing a record callback library reference could not be easily used on both Unix and Windows systems.

In V11.5 and later, c-tree supports specifying the DLL or shared library name in a way that causes c-tree to automatically convert the library name to the standard format for the platform on which it is running.

To use this feature, specify `^` as the first character of the library name. For example, specifying the name `^mycallback` causes c-tree to convert the name to `libmycallback.so` on Unix systems (`libmycallback.sl` on HP/UX and `libmycallback.dylib` on MacOSX), and to `mycallback.dll` on Windows systems.

When the library name starts with `^`, the conversion is ALWAYS applied. For example on Linux, the library name `^liberty` is converted to `libliberty.so`.

This new feature can also be used in the following situations:

1. when using a conditional index external library
2. when using a data record filter external library
3. when using a deferred index external library

13.6 Conditional expression functions to convert Unix time_t fields to c-tree Date and Time types

Conditional expression functions have been added to convert Unix time_t fields to c-tree Date and Time types. Customers using native Unix time_t types for data and time storage will need these functions to convert the Unix types to c-tree data types for use in partitioned file and conditional index expressions. Three new conditional expression functions have been added for this conversion:

- **TIMET2CTDATE** - Convert integer time_t value to a c-tree Date type (CTDATE).
- **TIMET2CTTIME** - Convert integer time_t value to a c-tree Time type (CTTIME).
- **TIMET2CTTIMES** - Convert integer time_t value to a c-tree Timestamp type (CTTIMES).

Note: These functions use ctrt_gmtime_r as the time conversion routine. This corresponds to the initial Unix date partitioned file support in *ctpart.c*.

Example

Usage in a partitioned file expression to partition into months since Jan, 2010:

```
CREATE TABLE unixtest (name CHAR(10), u_date INTEGER)
    or
CREATE TABLE unixtest (name CHAR(10), u_date BIGINT)

CREATE INDEX unixtest_date_idx ON unixtest (u_date) STORAGE_ATTRIBUTES 'partition=( ( YEAR( TIMET2CTDATE(
u_date) ) -2010) * 12) + MONTH ( TIMET2CTDATE(u_date))'

    Date String Unix Date Partition created
    Mon, 23 Feb 2015 11:01:32 GMT 1424689292 62
    Sat, 23 Jan 2016 11:01:32 GMT 1453546892 73
    Tue, 23 Feb 2016 11:01:32 GMT 1456225292 74
    Wed, 23 Mar 2016 11:01:32 GMT 1458730892 75
```

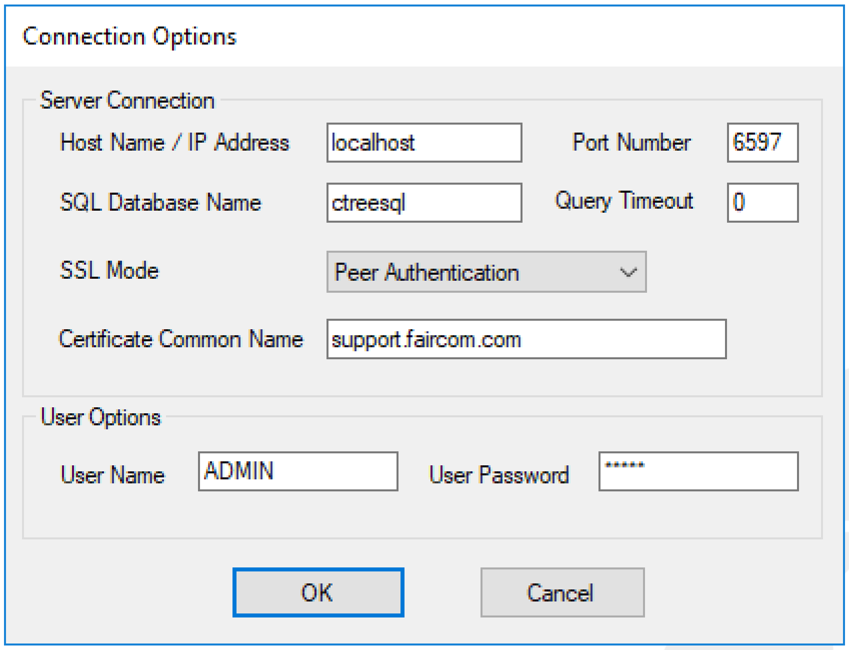
14. Graphical Administrative Tools

Enhanced ease-of-use and the latest advanced features bring elevated levels of administrative control and monitoring to your c-treeACE environments.

14.1 GUI Tools now support SSL

Controls have been added to the GUI tools to allow selecting the type of SSL connection (None, Basic, Peer Authentication) and, in case of Peer Authentication, the name of the certificate. This support has been added to both the Windows and Java versions of these GUI tools:

- c-treeACE SQL Explorer
- c-treeACE Query Builder



Connection Options

Server Connection

Host Name / IP Address: localhost
Port Number: 6597
SQL Database Name: ctreesql
Query Timeout: 0
SSL Mode: Peer Authentication
Certificate Common Name: support.faircom.com

User Options

User Name: ADMIN
User Password: *****

14.2 GUI Tools now support IPv6

Internal functions have been modified to support IPv6 addresses. This new feature now makes the graphical tools, both the Windows and Java versions, compatible with the expanded addressing offered by IPv6.

14.3 GUI tools now set the "Node Name" identification attribute

The Windows and Java versions of the Monitor GUI tools now set the node name in the **Node ID Info** column of the **Active Connections** tab for easy identification of the connection.

14.4 c-treeACE SQL Explorer (Windows Version)

List all users for non-DBA users

The Windows version of the c-treeACE SQL Explorer can now list all the users/schemas in the left-side tree even if the logged in user does not have DBA privileges.

A new option has been added in the **Options** dialog to allow a user to select the old or new behavior. If the option is set to **Yes**, the left-side tree will show all the users/schemas.

Scripting feature: PK retrieval Stored Procedure

The Windows version of the c-treeACE SQL Explorer has a scripting feature: a new PK retrieval Stored Procedure is now used for retrieving PK information in V11.5 and later. This feature accelerates execution time for these queries.

Added scripting feature to Convert/Import

Migrating (importing) a large Microsoft SQL database may be complex because the more proprietary SQL options, clauses, etc. may not be supported by c-treeACE.

A new option has been added to the **Convert / Import** tab in the Windows version of the c-treeACE SQL Explorer. This option allows creating one or more scripts, with the needed statements, instead of trying to directly import the objects. This allows the scripts to be analyzed so that unsupported statements can be fixed. The scripts can then be run in the **Scripts** panel or passed to the command-line ISQL utility.

Dump rows when multi-line control has focus

If a query was executed from the multi-line input, the **Dump Rows** button was attempting to execute the query in the single-line input, possibly returning an error. The **Dump Rows** button now executes the query in the multi-line input query as expected.

Support for CT_TIME_MS and CT_TIMES_MS

c-treeACE SQL Explorer support for CT_TIME_MS and CT_TIMES_MS millisecond types has been added.

Added "Max Rows" control to Scripts and Statements panels

A new **Max Rows** control was added to the Scripts and Statements panels of the Java and Windows versions of the c-treeACE SQL Explorer. This control allows users to select the maximum number of rows to fetch.

Additional Improvements

Several improvements have been made to the Windows version of the c-treeACE SQL Explorer:

- The "Execute SQL Statements" logic has been modified to execute in a background thread.
- The button panel now allows resizing of the upper panel.
- Indexes are correctly displayed when multiple users have tables and indexes with the same name.

14.5 c-treeACE ISAM Explorer Encoding Option

The c-treeACE ISAM Explorer Encoding option provides a way to choose the character encoding. There is a new **Encoding** tab in the **Options > Preferences** menu. This addition is new to the Windows version of the c-treeACE ISAM Explorer. The Java version already has this support.

14.6 c-treeACE Explorer (Java Version)

The Java version of c-treeACE Explorer has been enhanced to match recent improvements in the Windows version of the tool:

- The thousands separator can be turned off so that import programs don't interpret them as separate fields.
- The quoting of numeric fields can be turned off so numbers import into a spreadsheet properly.
- Fields with embedded quotes are properly handled.
- Field separators and test quoting characters are now selectable.
- Improved handling of large exports.
- Field names in the left panel remain visible while writing a query to avoid continually bouncing between tabs to find the field names and data types.
- A new **Max Rows** control was added to the **Scripts** and **Statements** panels. This control allows users to select the maximum number of rows to fetch.
- A fix has been implemented to resolve displaying too many indices when multiple users had tables and indices with the same name. The logic has been improved to display the correct indices in this situation.

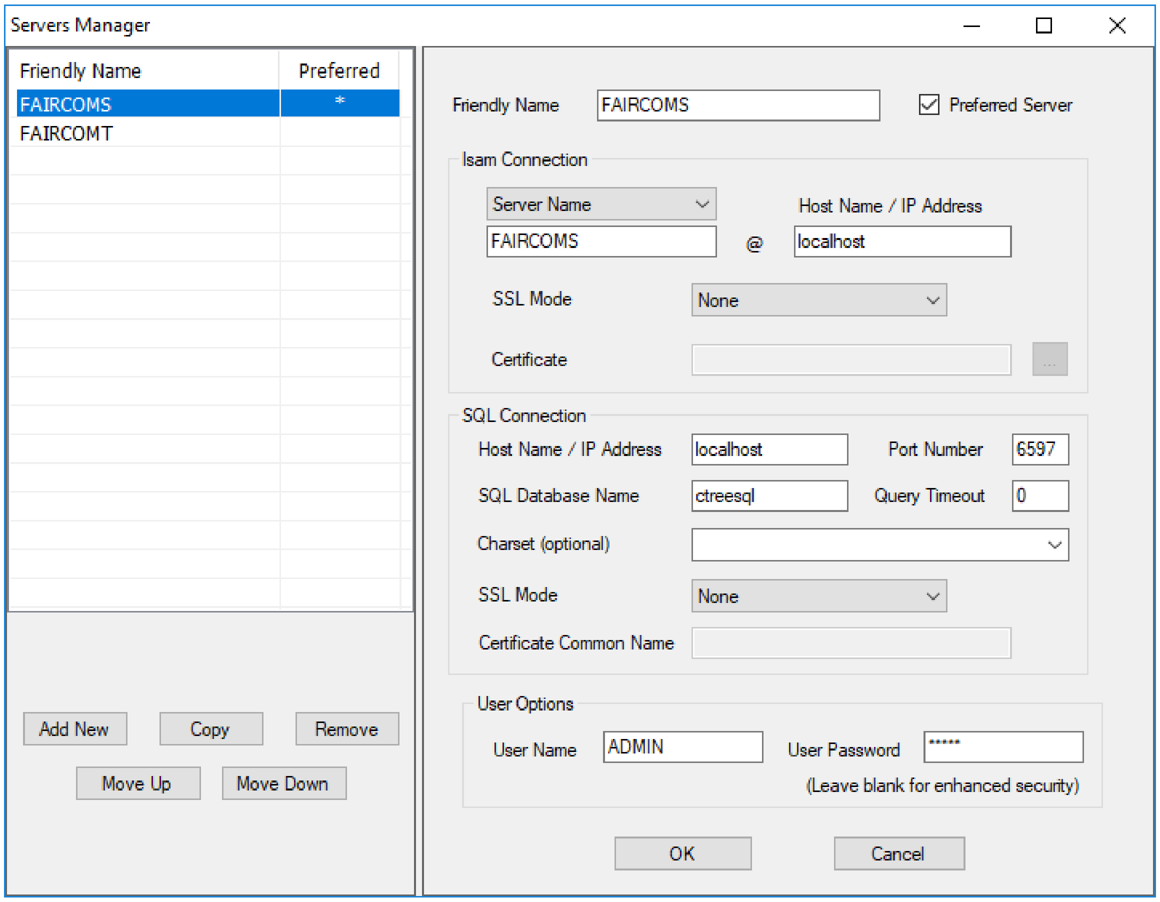
14.7 c-treeACE Gauges (Windows Version)

The following changes have been made:

The server name is now displayed on the Advanced Layers tab. When several c-treeACE Gauges windows are open, this feature makes it easy to tell which server each window is connected monitoring.

The node name is now set for easy identification of the connection.

The utility now includes the **Server Manager** feature to handle multiple server credentials for ease of use.



Friendly Name	Preferred
FAIRCOMS	*
FAIRCOMT	

Friendly Name: FAIRCOMS ☑ Preferred Server

Isam Connection
Server Name: FAIRCOMS @ Host Name / IP Address: localhost
SSL Mode: None
Certificate:

SQL Connection
Host Name / IP Address: localhost Port Number: 6597
SQL Database Name: ctreesql Query Timeout: 0
Charset (optional):
SSL Mode: None
Certificate Common Name:

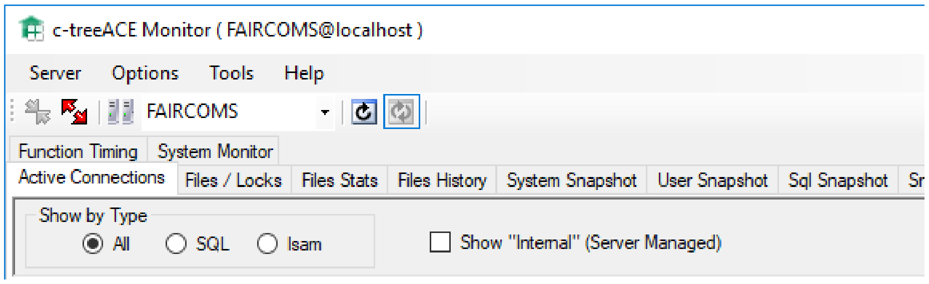
User Options
User Name: ADMIN User Password: *****
(Leave blank for enhanced security)

14.8 c-treeACE Monitor

These changes have been made to the c-treeACE Monitor:

- In the Windows and Java versions of the c-treeACE Monitor, two options were removed: **Stop Server** and **Quiesce Server**. These options could be considered "dangerous" by some customers who want to prevent tampering with server operation.

- A check box has been added to the Windows version of the c-treeACE Monitor to allow writing Date and Time counters as unformatted numbers when exporting to a *.csv* file.
- Controls have been added to the Active Connections panel of the Windows version of the c-treeACE Monitor to allow filtering the list of connections. The user can now select if the **Internal** (managed by the server) connections will be listed or not and also select to list the following connections: **All**, **SQL only**, or **ISAM only**.
- A filter has been added to the **File/Locks** grid in the Java version of the c-treeACE Monitor to allow filtering rows.



14.9 c-treeACE Query Builder

A **Refresh** button has been added to the Windows version of the c-treeACE Query Builder. The AutoRun Query button has a new icon to better depict its function.

14.10 Dr. c-tree

The Windows version of Dr. c-tree crashed if the *pfilnam* member of the *ifil* structure was null. This is caused by user input when editing the IFIL. This has been corrected by adding a validation check to IfilEditor.

14.11 c-treeACE Performance Monitor DLL changed

In c-treeACE V11.2.2 and later, the application name of the c-treeACE Performance Monitor DLL has been changed to make it clearer that a Windows error event originated from it. Prior to this change, the event was reported as originating from "c-treeACE." The logged name has been changed to "c-treeACE PerfMon."

The messages DLL name has been changed from *MessagesDll.dll* to *fcPerfMonMessagesDll.dll* to avoid conflict and better reflect the DLL scope.

In addition, the path to the SSL include directory has been changed to a relative path.

Note: This is a Compatibility Change.

14.12 Server Manager store file is now interchangeable between Windows and Java

The GUI tools requiring server access, both Windows and Java versions, save the servers connection options and credentials in a file called *ACEServers.xml*. The storage format has been modified so files used by the Windows and Java versions of the tool are now compatible. The tools accept both the old and the new file format. Only the new format is interchangeable between the Windows and Java tools. The Windows tools automatically convert the file from the old to the new format.

Note: The *ACEServers.xml* file must be converted by a Windows tool before it can be copied to the Java directory. The conversion can be done by simply starting one of the updated FairCom Windows GUI tools that requires server access (e.g., c-treeACE Monitor) and then closing it. The file will be located in the tool's working directory.

15. Administrative Utilities

Command-line utility improvements bolster expanded control of your c-treeACE data management environment.

15.1 ctstat Statistics Monitoring Utility

-I option

Some **ctstat** options, such as **ctstat** *-isam -i 60 1*, print a line of statistics which are all zeros. This is because **ctstat** may not have accumulated significant statistics before the first row is printed.

A new option, *-I* (upper case i) has been added to allow for more meaningful values without changing the current behavior. It behaves the same as the *-i* (lower case) switch, except that it waits *int* seconds to provide time to accumulate significant statistics before returning the first row.

ctstat *-I int [cnt]*

- *-I* - Pause *int* seconds for optional *cnt* times, waiting *int* seconds before returning the first row.

-isam header changed to avoid misunderstanding

The header of the **ctstat** *-isam* command displays a ***/s*** suffix to indicate that the statistics are *per second*. For example, the following syntax will produce the output below:

```
ctstat.exe -s FAIRCOMS -u ADMIN -p ADMIN -h 2 -isam
add/s   del/s   upd/s  read/s  total/s
      0       0       0       0        0
      0       0       0       0        0
   add/s   del/s   upd/s  read/s  total/s
      0       0       0       0        0
```

Note: This constitutes a Compatibility Change.

-text run by non-ADMIN user account reports memory use of -1

When **ctstat** *-text* is run by a user that is not a member of the ADMIN group, the following statistics in *SNAPSHOT.FCS* show values of -1 instead of the proper value:

```
system memory highwater mark: -1
       current aggregate sum: -1
```

The logic has been modified to report the correct values even if the user is not a member of the ADMIN group.

-userinfo to output the user connection communication protocol

The **ctstat** *-userinfo* function has been modified to output additional information. As of V11.5, the output of **ctstat** *-userinfo* is as follows:

```
status   lastrequest   trntime   mem fils   time   uid/tid/nodename   commprotocol
```

Because the column labeled `uid/tid/nodename` is not fixed in size, the space between `nodename` and `commprotocol` is a tab character. Even if the output is not properly aligned, the parsing will be easy by looking for a '\t' (tab).

15.2 cttrnmod - Change Transaction Mode Utility

If replication is enabled for a c-tree data file and then an action is taken that causes the data file to not have a qualifying replication unique index (such as setting REPL_SRLSEG_ALLOW_UNQKEY NO on the server, if the SRLSEG index was used as the replication unique index and no other qualifying unique index exists), ISAM-level read/write opens for the file will fail with error **UNQK_ERR** (775). Attempting to use the **cttrnmod** utility to turn off replication for the file also fails with this error.

The **cttrnmod** utility has been modified to detect this situation and uses a special mode to open the file, turn off replication, and close the file.

15.3 ctinfo - File Information Utility

The **ctinfo** utility now includes details on IDENTITY definitions within a file.

15.4 ctfilblkif - Manual Reopen

The **ctfilblkif** utility now supports an option (*-m*) to block a file with the reopen option. This means that when the file is unblocked, those connections that had the file open when it was blocked must "manually" reopen the file (e.g., the application must explicitly reopen the file).

Example 1:

Block the file with the reopen option:

```
ctfilblkif -f mydatafile.dat -m -p ADMIN -s FAIRCOMS
```

Example 2:

Unblock the file. The connections that originally had the file open must reopen the file:

```
ctfilblkif -f mydatafile.dat -u -p ADMIN -s FAIRCOMS
```

Note: FairCom does not guarantee preserving the state of a transaction, locks, ISAM context when a file is blocked and then unblocked, even if using the auto open option. Because ISAM context information is lost when the file is blocked, an error **590** may be returned after the file has been blocked and unblocked using *ctFBautopen* or when using **ctfilblkif** utility, which uses this mode.

15.5 ctscmp - Superfile Compact Utility

The **ctscmp** utility now returns a non-zero value when it fails. This can be useful for a script that calls the utility to be able to detect when it fails.

15.6 dbdump, dbload Unicode support

The **dbdump** Data Unload and **dbload** Data Load utilities have been updated to properly work in Unicode configuration. These utilities have been enhanced to properly dump and load data out of a Unicode-enabled c-treeACE SQL Server.

The command file must be in ASCII format. The output file generated by **dbdump** is in Unicode format optionally with a Byte Order Mark (BOM) indicating the Unicode encoding form (using the *-B* new command-line switch).

The input file for **dbload** needs to be in Unicode (native "wchar" encoding form) optionally with a BOM, in which case the utilities check for the proper format.

Usage:

```
dbdump -f commands_file [options] database_name
```

Valid options:

- *-n* - Check the command file for errors w/o loading data
- *-p* - select query passthru
- *-l* - progress frequency
- *-z* - maximum multiple reads
- *-B* - use BOM (Unicode Byte Order Mark) in output file
- *-u Username* - identifiable to the DBMS
- *-a Password* - for authentication

The **dbload** utility uses the same parameters documented in the *Command Line Utilities* (https://docs.faircom.com/doc/*cmdline/*) book.

15.7 dbload allows ";" after NEXT RECORD command in command file

The **dbload** command file did not allow a semicolon (";") after the final NEXT RECORD command in the file. The use of a semicolon after the NEXT RECORD is accepted (but not mandatory) so it no longer causes a syntax error.

15.8 ctdmpidx Index Dump Utility

The following changes have been made to this utility:

- The **ctdmpidx** utilitiy now displays the file mode and server ID in hex. This change provides the information in a more meaningful format.
- The **ctdmpidx** utility now defaults to using the largest supported page size. This allows all non-superfile indices to be opened using the defaults. For superfiles, the exact page size must be specified.

```
```

16. Notable Compatibility Changes

This section lists compatibility changes from prior releases. It is important to review these issues to ensure your c-treeACE applications function properly after upgrading.

The following list summarizes compatibility changes in this release. Most of the items are fairly minor in scope, but have been flagged to be thorough. Follow the links (or page numbers) to items that may impact your use of c-treeACE.

Important Changes:

- **Transaction log limit for Replication and Deferred Indexing** (page 66)
- **Operating System cache flush suppressed for non-TRANLOG files** (page 5)
- **Performance improvements in ctdbOpenTable** (page 9)
- **Default Replication Agent conflict detection** (page 32)
- **Local/master replication model diagnostic logging of errors during two-phase transaction commit** (page 35)
- **Dynamic Dump now stops when client callback function returns an error** (page 59)
- **Reduced minimal index cache size requirements** (page 67)
- **FairCom Server physical CPU counting method on AIX** (page 69)
- **Shared Memory protocol change** (page 71)
- **Temporary Index RVHD_ERR (123)** (page 73)
- **NO_FIXED_OFFSET_CHECK - Prevent reads/writes to arbitrary offsets for fixed-length files** (page 76)
- **Preventing Possible Data Loss with Compact & Rebuild Operations**
- **ADDREC DELFLG_ERR (31) following recovery** (page 78)
- **ctSQLImportTable function enhancements** (page 85)
- **JDBC connection URL new format** (page 38)
- **DSQL no longer allows using ctsqlGetChar() to retrieve long variable columns** (page 42)
- **OPNRFILX() modification** (page 91)
- **ctdbGetRecordCount behavior on MRT Table slightly modified** (page 95)
- **ctdbCreateMRTTable behavior change** (page 94)
- **Restore point file name now includes log number** (page 99)
- **ctstat -isam header changed to avoid misunderstanding** (page 107)

Changes to the Tools:

- **c-treeACE Performance Monitor DLL changed application name & messages DLL name** (page 105)
- **Replication Agent Monitor name changed** (page 29)

Configuration Keywords:

- **COMPATIBILITY TDATA_WRITETHRU** (page 11)
- **COMPATIBILITY MINIMUM_INDEX_BUFFERS** (page 67)
- **COMPATIBILITY TEMP_INDEX_ERROR** (page 73)
- **COMPATIBILITY NO_FIXED_OFFSET_CHECK** (page 76)
- **COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD** (page 86)

16.1 Java Stored Procedures and Triggers Require JDK V1.7 or Newer

Note: The c-treeACE SQL Java Stored Procedure and Trigger support requires a Java Development Kit (JDK) be installed on your computer. A Java Runtime Environment (JRE) is *not* sufficient. c-treeACE V11.5 (FairCom RTG and FairCom Edge V2.5) and later require JDK V1.7 or newer.

Note: For the tutorials to work correctly, the three Java-related “SETENV” lines in your *<faircom>/config/ctsrvr.cfg* file need to be active (the lines are *not* commented out with leading semicolons), the lines are all set to paths that are valid on your computer, and there is no whitespace before or after the "=" sign. Here are some typical examples, which will need to be adjusted for your machine:

Microsoft Windows

```
; JDK environment settings - Be sure to set the JDK to your version.
SETENV    CLASSPATH=C:\Program Files\Java\jdk1.7.0_75\jre\lib\rt.jar;.\classes\ctreeSQLSP.jar
SETENV    JVM_LIB=C:\Program Files\Java\jdk1.7.0_75\jre\bin\server\jvm.dll
SETENV    JAVA_COMPILER=C:\Program Files\Java\jdk1.7.0_75\bin\javac.exe
```

Linux

```
; JDK environment settings - Be sure to set the JDK to your version.
SETENV CLASSPATH=/usr/java/jdk1.7.0_75/jre/lib/rt.jar:./classes/ctreeSQLSP.jar
SETENV JAVA_COMPILER=/usr/java/jdk1.7.0_75/bin/javac
SETENV JVM_LIB=/usr/java/jdk1.7.0_75/jre/lib/amd64/server/libjvm.so
```

The lines above all need to correctly point to your JDK installation folder before you start the c-tree server, because changes to *ctsrvr.cfg* take effect only when you launch the c-tree server.

All of these Java-related lines should point at files which are in the same JDK folder (“*jdk1.7.0_75*” in this example), and not from a JRE installation. Violating these rules can result in problems that can be difficult to track down.

In the CLASSPATH line, the path to *ctreeSQLSP.jar* is relative to the “server” folder. **On Linux, the entries in the CLASSPATH line should be separated with colons instead of semicolons.**

The purpose of these three lines is to give the c-tree server the information it needs to compile and run Java source code. This is because the stored procedures demonstrated by this tutorial are written in Java.

If you are using FairCom RTG or FairCom Edge, adjust the path to match your product.

16.2 .NET stored procedure plugin updated for Visual Studio 2015 and 2017

The Microsoft Visual Studio stored procedure plugin, *CtreeSP.vsix*, has been updated to support the latest versions of Visual Studio 2015 and 2017. Users with prior versions should uninstall their existing plugin and install this version for latest compatibility. You will find this plugin in the *\sdk* area of your c-treeACE installation package at this location:

<faircom>\drivers*\dotNet*

Copyright Notice

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

17. Index