

FairCom

DB

v13

DB Made Simple



FairCom

Contents

1.	Highlights	1
2.	Server	3
2.1	Resize Data and Index Cache for Elasticity.....	3
2.2	Enhancements to Prime Cache at Server Startup.....	6
2.3	Enable Default Data Cache Empty Lists	8
2.4	Online Compact and Rebuild	8
2.5	On-line Compact for Background Table Defragmentation	9
2.6	Online Rebuild Support.....	10
2.7	Recycle Bin Preserves Deleted Files and Protects Against Accidental Deletes	11
2.8	File Names Supported up to 4K	14
2.9	PAGE_SIZE Changes Automatically Applied to all FairCom Controlled Files	15
2.10	Create system log files as huge files to remove 4GB file size limit	20
2.11	SYSLOG improved reuse of disk space	20
2.12	SYSLOG - Efficiently Truncate.....	20
2.13	Server Read-only Status Logged to CTSTATUS.FCS.....	21
2.14	Server internal processing files moved from working directory to LOCAL_DIRECTORY	21
2.15	Read Permissions - Server now checks for read permission on function calls that read data or key values	22
2.16	Log message to CTSTATUS.FCS when transaction commit fails	22
3.	Server Plug-Ins	23
3.1	dbNotify - database triggers to send MQTT messages	23
3.2	Interface for Managing Plug-ins on-the-fly.....	24
4.	SQL	26
4.1	Auto timestamp support	26
4.2	SQL Performance Improvements.....	26
4.3	Retrieve MRT Host Table Name via SQL Stored Procedure	27
4.4	Support for Multiple DISTINCT Keywords in a Single Query	27
4.5	SYSLOG SQL_STATEMENT - Exclude users from logging by name	27



4.6	Overcome 128K record size limit	27
4.7	Entity Framework Table Addition Speed Optimized	28
4.8	SQL Import Utility (ctsqlimp) - Option to keep existing permissions and synonym	28
4.9	Sequence Number Entities now Unique Between Databases	29
4.10	Create ISAM Compatible CT_MONEY Column Type from FairCom DB SQL	30
4.11	SQL - support to map money type to c-tree's CT_MONEY	30
4.12	ROWID - CTDB/SQL - Expose 'rowid' as a field in SQL	30
4.13	SQL - Correct Numeric String Conversion	32
4.14	SQL Stored Procedures - Informational Methods Added to DhSQLException	32
4.15	SQL Windowing Functions for Analytics	32
4.16	Tuple Expression Enhancements	33
4.17	.NET Stored Procedures - Implement access to LVAR* fields	33
4.18	.NET Stored Procedure - More information	34
4.19	DSQL - Direct SQL	36
	DSQL - New function ctsqlGetCommandFromCursor	36
	DSQL - New functions to scroll the resultset using a scrollable cursor	36
	r288975 - DSQL - New function ctsqlGetConnectionFromCursor	37
	DSQL - New function ctsqlGetParameterDirection	37
	DSQL - New functions ctsqlGetConnectionFromCommand() and ctsqlGetParameterAddress()	38
5.	Drivers	39
5.1	Drivers Overview	39
	c.isam - C Language ISAM	40
	c.lowlevel - C Language Low-Level	40
	c.mqtt- C Language MQTT Client	40
	c.nav - C Language Record Navigation (CTDB)	41
	c.sql.direct - C Language Direct SQL	41
	cpp.nav - C++ Language Record Navigation (CTPP)	41
	csharp.nav - C# Language Record Navigation	42
	csharp.sql.ado.net - C# Language for ADO.NET	42
	csharp.sql.storedprocs - C# Language Stored Procedures	42
	java.jpna.nav - Java Language Persistence API (JPA)	43
	java.json.db - Java Language over JSON DB	43
	java.mqtt - Java Language MQTT Client	44
	java.nav - Java Language Record Navigation (JTDB)	44
	java.sql.hibernate - Java Language Hibernate Persistence	44
	java.sql.storedprocs - Java Language SQL Stored Procedures	45



	nodejs.json.db - JSON DB Client for Node.js	45
	nodejs.mqtt - MQTT Client for Node.js	45
	php.sql - SQL for PHP.....	46
	php.sql.pdo - SQL for PHP PDO.....	46
	python.json.db - Python Language over JSON DB	46
	python.mqtt - MQTT Client for Python	46
	python.nav - Python Language Record Navigation (CTDB)	47
	python.sql - Python Language SQL.....	47
	python.sql.sqlalchemy - Python Langage for SQLAlchemy.....	47
	sql.cli - SQL Language Interactive SQL Command-Line Interface (iSQL)	48
	sql.jdbc - Java Language Database Connectivity (JDBC)	48
	sql.odbc - SQL Language Open Database Connectivity (ODBC)	48
	thingworx.always-on - ThingWorx Connector	49
	vb.nav - Visual Basic Language Record Navigation.....	49
5.2	JAVA.....	50
	JAVA JSON DB - Add new driver for Java Deveopers.....	50
5.3	JDBC/JTDB	51
	JDBC crash fixed	51
	JDBC Installers now available	51
	FindTarget Method Added to c-treeDB for Java	51
	Improved JDBC efficiency.....	51
	Access c-treeDB Java API with Embedded Server Models.....	51
	JDBC - allow URL without hostname.....	52
	JDBC now Supports Setting Parameter to NULL	52
	Remove JDBC setEscape Processing Flag Exception.....	53
	JTDB - Added CTable.HasLocks()	53
5.4	ODBC	53
	Unexpected 64-bit ODBC Driver Overflow Error	53
	Corrected ODBC Error -5 'wrong number of parameters' When Creating Stored Procedures	53
	Changed Character Validation in ODBC Connection String.....	53
5.5	C++.....	54
	Improved c-treeDB Truncate Table Functionality	54
	C++/CTPP - Added SetBinaryFlag and GetBinaryFlag to the CTField class	54
	C++/ CTPP - CTSession::FindDatabase with char* parameters	55
	CTPP - CException::GetErrorMsg() may return a pointer to released memory	55
	Copy Constructor added to CException Class	56
	C++ CTPP - added overloads for CTSting::Compare and CTString::CompareIC.....	56
	C++/CTPP: Added CTable::HasLocks()	57
5.6	PYTHON.....	58
	Python APIs	58
5.7	.NET	58



.NET Core	58
.NET Stored Procedures	60
.NET Entity Framework	62
.NET Gui Tools.....	62
6. CTDB.....	65
6.1 c-treeDB Default Extent Sizes Supported Beyond 64K	65
6.2 Set First Extent Size in FairCom DB API.....	65
6.3 Improved c-treeDB Truncate Table Functionality	66
6.4 Change SQL Database Path Utility	66
6.5 ctpath - Change Internal (SQL) Database Paths	66
6.6 Set Table Partition Base in FairCom DB API.....	67
6.7 Support for Modifying file Partition Added to c-treeDB	67
6.8 Getter Setter Methods for Blobs as base64 Strings.....	67
6.9 CTPP - CTEException::GetErrorMsg() may return a pointer to released memory.....	68
6.10 Copy Constructor added to CTEException Class	69
6.11 ctdbGetFieldAs* and ctdbSetFieldAs* from RTG column with dbtype set.....	69
6.12 CTDB Filter support for BTRIEVE date and time data types	70
6.13 ctdbNumberOfEntries() Added to CTDB	70
6.14 CTDB - alter table from attribute - alter table support for RTG.....	70
6.15 CTDB - New function ctdbInsNAVField().....	71
6.16 ctdbSetDefaultValueAsBinary Added to CTDB	71
6.17 NULL KEY VALUES Support - CTDB level metadata stored file's unified resource.....	72
6.18 CTREE/CTDB/RTG - support for CT_BT_BIT field type.....	73
6.19 CTDB - improved batch processing.....	73
6.20 CTDB/CTREE/RTG - support for BTRV BLOB and CLOB	74
6.21 CTDB - New functions to retrieve all dictionary information for tables	74
6.22 Added new batch mode "BAT_AUGFIX"	75
6.23 Performance Enhanced During Batches that Filter out Fields	75
6.24 Performance Enhanced for Batch Variable Length Record Read.....	76
6.25 Retrieve the conditional expression function	76
6.26 ctdbGetNAVFieldProperties maps unsigned integer c-tree field to signed larger numeric.....	76
6.27 CTDB - ctsqlimp utility and "sqlimp" functions default to promote unsigned integers.....	77



6.28	c-treeDB API Functions for User-Defined Hot Alter Table Field Conversion Callback	77
6.29	C++/CTPP - CTable Open method to open tables using the table UID	78
6.30	CTDB - New function ctdbDeleteBackgroundLoadStatus.....	78
6.31	CTDB - added support for changelD field	79
6.32	CTDB - Alter Table - ability to create/update conditional indices with table open shared.....	79
6.33	CTDB - Set scanner cache feature on table	80
6.34	CTDB - ability to set a index as primary key index	80
6.35	CTDB - SQL import logic improved to take primary key index information into account	81
6.36	CTDB - alternative fixed string padding.....	81
6.37	Improved Change Batch Performance	82
6.38	FairCom DB Batch Calls now Return Number of Records Rejected from Active Filter	82
6.39	CTDB Function to Retrieve Number of Filtered Records	83
6.40	CTDB - new function ctdbBlobReserveSize	83
6.41	CTDB - Performace - avoid useless SWTCTREE call during ctdbGetFieldAs() and ctdbSetFieldAs()	84
6.42	CTDB - new functions to retrieve primary key candidate index.....	84
6.43	ctSQLImportTable() function and ctsqlimp utility support for TLS	85
6.44	Record buffer optimization - memset in record handling may have significant performance impact.....	86
6.45	CTDB - ctdbClearRecord() performance improvement.....	86
7.	CTREE	87
7.1	Alter Table Compression - modes to copy compression from existing open file and to reset to the default	87
7.2	Prevent Updates and Deletes for Tables.....	87
7.3	Support adding identity field and auto system time field to partition host that has existing members	88
7.4	Support Added to hot Alter Table for non-Schema-Based key Segment Modes.....	88
7.5	Support new types of null key value checks for c-tree indexes.....	88
7.6	Record Update Callback - Support record update callback that is called at the start of an add or update operation, allowing the callback to modify the record image	89
7.7	User-Defined Hot Alter Table Field Conversion Callback	90



7.8	CTREE - Implement change id field to support optimistic locking.....	92
7.9	File Block Enhancement: Support a file block mode that allows active transactions time to complete before blocking the file	93
7.10	Support adding a condition to an index when creating the index with the data file open in shared mode.....	94
7.11	Support changing a file's password or permission mask when the file is open in shared mode	95
7.12	CTREE - New function to generate a sequence name that does not conflict with existing sequences	96
7.13	Partition: support for non-schema key segment modes (REGSEG, INTSEG, etc.)	96
7.14	CTREE/CTDB - support for unixtime stamp data type (time_t).....	97
7.15	Retry opening transaction log when updating deferred index log entry status	99
7.16	When changing security attributes for a partition host file, change the attributes for all existing partition members	99
7.17	Read Permissions - Server now checks for read permission on function calls that read data or key values.....	100
7.18	NULL Key Support - Improve indexing of null values	100
7.19	ctinfo: Retrieve Assigned OS System File IDs.....	101
8.	ISAM.....	103
8.1	Return Server's Read-Only Mode with GETNAM().....	103
8.2	Automatic Temporary Directory Cleanup After Crash.....	103
8.3	Improve ctQuiet Timeout.....	104
9.	StandAlone.....	105
9.1	StandAlone Full Source Code Package - Porting to alternative platforms made easier	105
9.2	FairCom SQL Service - Simpler StandAlone SDK Packages	105
9.3	Performance enhancement: Turn on log writethru for Single-User TRANPROC library	106
9.4	Updated Standalone Rebuild and Compact Utilities for Data Encryption.....	106
9.5	Partitioned file support in multi-user StandAlone operational model.....	106
9.6	Change FPUTFGET model's ctREADFIL and ctSHARED open to a shared read-only open	107
9.7	Fixed index file error 14 in FPUTFGET after index update in exclusive writethru mode	108
9.8	Compact Error RCPT_ERR (1156) Fixed in Multi-user Standalone Models	108



9.9	StandAlone batch calls with locks fail with LEOF_ERR(30).....	108
9.10	ctREADFIL may be ignored by FPUTFGET	109
9.11	File Close Silently Failing.....	109
10.	Utilities.....	110
10.1	Dynamic Dump - Online Backups	110
	Improved Dynamic Dump Performance.....	110
	Hot Backup Without Quiet Checkpoint Requirement.....	110
	Redirect All Restored Files to a Specified Path	111
	ctfdmp and ctldmp no longer fail with 1166 (ENCL_ERR) on non-encrypted logs	111
	Dynamic dump error message improvement.....	111
	Dynamic Dump Hangs	111
	Recovery Script Configurations Ignored from Backup Scripts.....	112
	Update to Dynamic Dump Keys	112
	Improved Hot Backup Quiet Checkpoint Requirement	112
	Dynamic dump missing data or keys for non-ctTRNLOG Files	113
	Prevent Unix Dump Restore Hang During Large Deployment	113
	Correct Dump Restore Redirect Log Messages to Include Filename	113
	Prevent Dynamic Dump Restore Errors 499, 12, and Files Missing after Restore	114
	Server wait on external ctrdmp or disk full action process may hang on Windows	115
	Dynamic dump !REPLICATE	115
	Dynamic dump !REPLICATE option now applies to superfile host and its members.....	116
	!REPLICATE option in dynamic dump script can cause backed up file to fail to open with error 14 or 413	116
	Update to Dynamic Dump Error Message	116
10.2	ctinfo: Retrieve Assigned OS System File IDs.....	116
10.3	ctinfo: File encryption and file name information	117
10.4	ctinfo improved: Show DODA type names; Pause before dumping attributes content	117
10.5	ctinfo: display record update callback information	117
10.6	ctinfo: display distinct key counts for indexes	118
10.7	ctVerifyFile: Improved Index Leaf Nodes Key Mark Checks	118
10.8	ctVerifyFile: Additional Test for keys Referencing Deleted Records	118
10.9	ctVerifyFile: Check for unexpected key marks in index leaf nodes crashed on variable-length data file	118
10.10	ctVerifyFile: Internal Index Node Validation Improvements	119
10.11	ctVerifyFile/ctvfidx - Add distinct key count verification.....	119
10.12	Block Logon when Quiesce is Abandoned	121
10.13	Retrieve MRT Host Table Name via SQL Stored Procedure	121



10.14	ctCreateAuthFile() Creates Encrypted Auth Files	121
10.15	ctpath - Change Internal (SQL) Database Paths	122
10.16	ctscmp - Corrected IAIX_ERR (608) Error While Compacting V6 Superfiles.....	122
10.17	ctrbldif - Toggle Index Null Key Support with Rebuild Utility	123
10.18	Support Added to hot Alter Table for non-Schema-Based key Segment Modes.....	123
10.19	Update to ctsqlcdb Utility.....	123
10.20	Segmented Sort Work Files for Rebuild	124
10.21	Retain Existing Permissions and Synonyms on SQL Import	124
10.22	ctfchk Utility added to packages.....	124
10.23	PTADMIN() option to purge all existing partitions below the new partition base value when increasing the partition base.....	124
10.24	ctpartadmin utility now supports partitioning an existing data file.....	125
10.25	ctldmp utility now decodes DIFIMAGE entries into old and new image values	126
10.26	ctldmp/ctlchg: Support 6-byte transaction numbers.....	126
10.27	ctcompare utility - Support running with a server that has a different path separator	126
10.28	Specialized Superfile Compact Options	126
11.	Diagnostics	127
11.1	Suppress DIAGNOSTICS READ_ERR stack traces when no error is returned	127
11.2	Robust System Log Features Secures SQL Database Auditing	127
11.3	Configuration keyword CTSTATUS_MASK WARNING_PAGESIZE disables mismatch warning	129
11.4	Configuration keyword disables mismatch warning	129
11.5	Mask PAGE_SIZE Warnings in CTSTATUS.FCS	129
11.6	Diagnostic Message for Transaction Log Deletion	130
11.7	Resource Read Errors (RRED_ERR, 407) Diagnostics.....	130
11.8	Failed Client Shared Memory Connection Diagnostics.....	130
11.9	Added Diagnostic Logging for INIX_ERR.....	131
11.10	Server Read-only Status Logged to CTSTATUS.FCS.....	131
11.11	Improved System File id Diagnostic Logging.....	132
11.12	Diagnostic logging for commit delay livelock	132



11.13	Server logs diagnostic messages to help analyze slow or hanging plugin startup and plugin unload.....	132
11.14	Node Name Added to DLOK_ERR and Transaction Abort Messages.....	133
11.15	Log message to CTSTATUS.FCS when transaction commit fails	133
12.	Compatibility	134
12.1	BINARY CAST of Padding Spaces Retained in VARCHAR Fields	134
12.2	SQL - Correct Numeric String Conversion	134
12.3	Background Transaction Data and Index Flush Threads no Longer Default to Immediate	134
12.4	TLS/SSL Correctly Configured from Encrypted Configuration File	135
12.5	Expanded Windows Broadcast Machine Name Length	135
12.6	Mask PAGE_SIZE Warnings in CTSTATUS.FCS	135
12.7	32K Page Size now Always Set as Default	135
12.8	V6 to V7 SDK Mapping Macros Disabled	136
12.9	Recovery Script Configurations Ignored from Backup Scripts	136
12.10	Default COMPATIBILITY BATCH_SIGNAL Behavior.....	136
12.11	Enable Default Data Cache Empty Lists	137
12.12	Default COMPATIBILITY FILE_DESCRIPTOR_LIMIT Configuration	137
12.13	Stricter Transaction Dependent Behavior for all Table Definition Operations	139
13.	Security	140
13.1	Source Code Security - Coverity Scan	140
13.2	Apple Packages now signed - 'Contents' directory introduced.....	140
13.3	Security Improvements for LDAP Authentication.....	140
	LDAP authentication failure no longer incorrectly indicates timeout error on Linux/Unix	141
	LDAP_LOCAL_PREFIX option to filter LDAP authentication by username	142
	LDAP errors generate correct error messages at login	142
	LDAP - Server crash on Linux using LDAP with SSL fixed	142
	LDAP support for Solaris	142
	LDAP Connection and File Open Passwords Memory Leaks Fixed	143
13.4	Transport Layer Security Secures Data in Transit between Network FairCom DB Clients and Servers	144
	OpenSSL Updated to 3.0 for Latest Connection Security	148
	OPENSSL_ENCRYPTION keyword no longer controls internal cipher	148
	Troubleshooting and Testing	148
	Client FIPS Configuration for TLS Connections	149



Compatibility.....	150
FIPS-compliant OpenSSL 3.0 libraries for Windows and Linux	150
Client-side Support Added for FIPS.....	150
SSL_CIPHERS	151
Ephemeral Elliptic Curve Diffie-Hellman Cipher (ECDHE) now Supported for TLS Connections.....	152
Diffie-Hellman parameters can now be read from server certificate file if present	153
FairCom DB Server FIPS Support Details.....	153
AES Data at Rest Encryption	153
TLS Network Encryption	154
AES Data Encryption	155
TLS Communications Encryption.....	156
Compatibility.....	157
13.5 X509 Certificate based authentication.....	158
13.6 HTTPD Service Extra Headers Configuration	160
14. Configuration	162
14.1 Important - PAGE_SIZE Conversions	162
14.2 32K Page Size now Always Set as Default	167
14.3 Mask PAGE_SIZE Warnings in CTSTATUS.FCS	167
14.4 Single-line Status Log Messages Improves External Integration	168
14.5 BINARY CAST of Padding Spaces Retained in VARCHAR Fields	168
14.6 FairComConfig Utility now detects prior .NET configurations	169
14.7 Block Logon when Quiesce is Abandoned	169
14.8 Shared Memory Key Configuration Through Environment Variables.....	169
14.9 Disable Diagnostic pstack Generation From Shared Memory Errors.....	170
14.10 Background Transaction Data and Index Flush Threads no Longer Default to Immediate.....	170
14.11 File Open Automatically Retries on Pending Creation State.....	170
14.12 Avoid OPEN I-O Exclusive Open Errors.....	171
14.13 Create system log files as huge files to remove 4GB file size limit	172
14.14 SYSLOG improved reuse of disk space	172
14.15 SYSLOG - Efficiently Truncate.....	172
14.16 SYSLOG_EXCLUDE_SQL_USER	173
14.17 Server Startup Error 90 Fixed	173
14.18 File Copying/Quiesce Updates.....	173
14.19 Default COMPATIBILITY BATCH_SIGNAL Behavior.....	174
14.20 Support setting server configuration directory with environment variable	175



14.21	Configure DIAGNOSTICS FULL_DUMP at Runtime	175
14.22	Server Configuration - Default FILES 4096. Was FILES 1024.....	175
14.23	services.json replaces cthttpd.json Configuration.....	176
14.24	Set CHECKPOINT_INTERVAL to 12th of total log space in server configuration file.....	176
14.25	Improve ctQuiet Timeout.....	176
14.26	Docker uses NAT - New property "dbengine_behind_nat" in ctagent.json	177
14.27	Server rebuild options may be set at runtime - SORT_MEMORY and MAX_HANDLES	178
14.28	Configuration option MAX_REBUILD_QUEUE is runtime configurable	178
14.29	V12/V11 Encryption Backward Compatibility.....	178
14.30	OPENSSL_ENCRYPTION keyword no longer controls internal cipher	179
14.31	Ephemeral Elliptic Curve Diffie-Hellman Cipher (ECDHE) now Supported for TLS Connections	179
14.32	Recovery Configuration File for Secondary Synchronous Server Coordination	180
14.33	Expanded Windows Broadcast Machine Name Length.....	181
14.34	Return Replication Log Scan Last Log Entry Offset	181
15.	Communication Protocols.....	182
15.1	Shared Memory	182
	Shared Memory protocol connect function on Windows now uses the optional socket timeout value.....	182
	Shared Memory Key Configuration Through Environment Variables.....	182
	Unix Shared Memory connection errors logged by server now include pipe or socket method in use.....	182
	Expanded Windows Broadcast Machine Name Length.....	183
	Named Pipe connection logic removed from Unix shared memory.....	183
	Improved Shared Memory Connections When Permissions Discrepancy Between Server and Client.....	183
	Improved SQL Shared Memory Connections When Permissions Discrepancy Between Server and Client.....	184
	Failed Client Shared Memory Connection Diagnostics	184
	Configurable Shared Memory Connection Timeout.....	185
16.	Replication	186
16.1	Support more than one replication plan between two FairCom DB Servers	186
16.2	Support replication of superfile hosts and members.....	186
16.3	Support Publication "By Database"	186
16.4	Support Publication "By Dictionary".....	187



16.5	Support multiple folder rules per Publication	187
16.6	Non-recursive filename wildcard matching "by folder" replication plans	187
16.7	Support for "recursive" parameter for Publication "by folder"	188
16.8	Support Regular Expression (regex) rules in Publication.....	188
16.9	Replication agent sync shutdown.....	189
16.10	Replication deployment now copies indexes instead of "Rebuild on Deploy"	189
16.11	Return Replication Log Scan Last Log Entry Offset	189
16.12	Replicate Restore Point Operations	190
16.13	Support FileSystem scan of non-ctree files	190
16.14	Automatic Purge and Archive of Replication Logs.....	190
16.15	Log Only Failed Operations to Exception Log	191
16.16	Replication Command Line Utility - ReplUtil	191
16.17	repadm - Replication Administrator Utility Updates	192
	Force All Database Changes Before Synchronous Server Shutdown	192
	Toggle READONLY Target Server Mode on Replication Shutdown	192
	Check for Active Replication	193
	Reset Replication Statistics	193
16.18	repadm utility can be used to reset the replication agent statistics	194
16.19	repadm Utility - option to display key value and record image in exception log record.....	194
16.20	Encrypted Password File Support for Replication Diagnostic Utility ctrepd.....	194
16.21	ctrepd utility: Add option to set node name.....	194
16.22	Display Replication Latency from repadm Utility.....	195
16.23	Replicate File Copy Operations.....	195
16.24	Additional Replication Extension Event Callbacks.....	196
16.25	Replication callback called after replicating a CREIFIL operation.....	196
16.26	Replication Manager files to be created under data directory	197
16.27	Improve performance of dump restore when deploying a publication with many files	197
16.28	Improved Replication Manager Plan Shutdown.....	198
16.29	Reduce Number of Parallel Replication Apply Thread Target Server Connections.....	198
16.30	Reduce Number of Log Read Thread Connections to Target Server when Using Embedded Replication Agent.....	199
16.31	Initialize Replication File Handle Values with ctReplInitFileHandle()	199



16.32	Update Replication Manager Publication Files and Folders When Server Moved to Alternate Environment	200
16.33	Obtain Oldest Uncommitted Transaction Log Position from a Replication Agent	200
16.34	Replication Agent Log File now Created in Local Executing Server Directory	202
16.35	Replication manager logging improvements.....	202
16.36	Change replication agent exception logging default option to log errors only	202
16.37	Add replication agent configuration option to write error 1105 occurrences to ctreplagent.log and apply the transaction anyway	202
16.38	ctAgent - forcing IP address option ignored.....	203
16.39	Docker uses NAT - New property "dbengine_behind_nat" in ctagent.json	203
16.40	Automatically check for a write lock when updating a data file during synchronous replication	204
16.41	Client failOver/Broadcast notifications.....	204
16.42	File Groups	205
	Replication File Groups.....	205
16.43	Configure Embedded Replication Agent	205
16.44	Requirements for Source and Replica FairCom DB Servers	206
16.45	Replication Agent Manager Supported Operations	206
	Replication File Group Fixes and Improvements	210
16.46	Replication Demos.....	211
	Replication Demo - Replication Using JSON RPC	211
	Replication Tutorial - Create test program to demonstrate ctReplAgentOp() opcode usage	211
17.	High Availability	213
17.1	Pacemaker.....	213
18.	Platform/Compiler Support	214
18.1	Visual Studio 2022 Support.....	214
18.2	Apple macOS Support: M1/M2/M3 - Code Signing - Monterey; Ventura; and Sonoma	215
18.3	Android Port - Code Compatible	216
18.4	Windows ARM Native Port.....	217
18.5	Linux MIPS Port.....	218
18.6	Legacy platforms remain supported	218



19.	Docker Support	220
19.1	Docker: Scripts to create Docker image	220
20.	Java JDK Version Considerations	221
20.1	Java Version Support.....	221
20.2	Java Stored Procedures and Triggers Require JDK V1.7 or Newer	221
21.	Miscellaneous	222
21.1	Windows Installers and FairComConfig Improved	222
21.2	Windows Installer Error 1722 Fixed	224
21.3	Install in Program Files.....	224
21.4	VSS Writer: Volume Shadow Copy Service	224
22.	Bug Fixes.....	225
22.1	CTDB Fixes	225
22.2	CTREE Fixes	227
22.3	SQL Fixes	235
22.4	Utility Fixes	237
22.5	Other Fixes	238
22.6	Replication	240
23.	Index	250



1. Highlights

Subsequent to celebrating 40 years in business, the FairCom team “pushes on” and are happy to announce our next Major Database Product Release: FairCom DB (Lucky) V13. Over many years, the extent of FairCom’s breadth and scope of technology offerings has grown considerably. This release is packed with significant progress over the last few years, including: new product offerings; customer requests; innovative features; performance enhancements; and stability improvements.

This document highlights our most significant areas of focus. We hope there is a “special something” for everyone.

[Click here to see our Highlights](#)

We appreciate your business and hope you are pleased with our progress.

2. Server

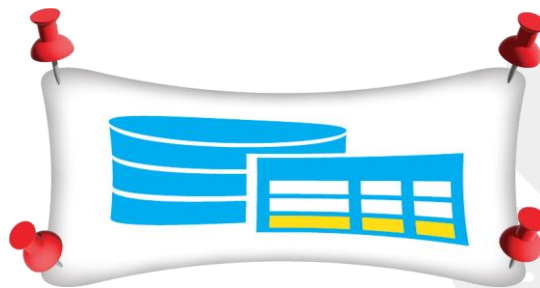
2.1 Resize Data and Index Cache for Elasticity

Memory caching of data is one of the most important subsystems of a database server, including FairCom DB. Caching allows the database process to optimally serve clients with data directly from memory when possible. Separate independent memory caches are used for data and indexes (buffers). Proper cache sizing is a critical performance tuning task of the database administrator. Too little memory, and the database engine becomes bottlenecked with persisted storage I/O. Too much, and not enough memory is made available to the operating system and other applications running on the host machine.

Cache resizing is a powerful feature for performance tuning with increasing database loads. Adding additional client connections, thus increasing user concurrency, frequently benefits from larger caches. Another situation is a very large ad-hoc database load or query where temporarily having a larger memory cache improves overall performance. Changing a cache size requires restarting the FairCom DB server process which is not an easy task in the 24/7 enterprise environment.

FairCom DB "hard" allocates memory for caches at server startup directly from the OS memory heap. Once allocated, the memory is permanently associated with the database for the life of the process. Allocated memory is chunked into page size blocks within the memory cache subsystem, historically making it difficult to resize as needed. An additional complication is how to handle existing data residing in cache. Losing that "hot" data will likely result in performance loss for connected applications until a runtime steady state is again achieved over time.

FairCom has had increasing requests for ability to change cache sizes at runtime and this release introduces dynamic cache resizing. Now, you can fine tune data and index caches as performance demands based on database connections and load avoiding costly database down time.



How to Resize Your Current Cache

The server administrator utility, **ctadmn**, supports a new configuration option `set_cache_options` as a parameter. This is followed with a null-terminated JSON string with desired options.



Supported cache options are as follows:

- **"*timeout*":** `<timeoutInSeconds>` where `<timeoutInSeconds>` is the maximum number of seconds to wait for active transactions to complete when quiescing the server in order to change the cache sizes. The default timeout is 1 second.
- **"*dat_memory*":** `"<new_data_cache_size>"` where `<new_data_cache_size>` is an integer cache size followed by optional suffix such as mb for megabytes or gb for gigabytes. For example, `"dat_memory": "100 mb"`. If not specified, the data cache size is not changed.
- **"*idx_memory*":** `"<new_index_cache_size>"` where `<new_index_cache_size>` is an integer cache size followed by optional suffix such as mb for megabytes or gb for gigabytes. For example, `"idx_memory": "200 mb"`. If not specified, the index cache size is not changed.
- **"*keep_cached_data*":** `"<yes_or_no>"` where `<yes_or_no>` is either yes, which means the currently-cached data is kept in the data cache, or no, which means the currently-cached data is removed from the data cache. The default is yes.
- **"*keep_cached_nodes*":** `"<yes_or_no>"` where `<yes_or_no>` is either yes, which means the currently-cached nodes are kept in the index cache, or no, which means the currently-cached nodes are removed from the index cache. The default is yes.

Administrator Utility Usage

From **ctadm**, choose option 10 "Change the specified configuration option"



```
Change Server Settings:

1. Configure function monitor
2. Configure checkpoint monitor
3. Configure memory monitor
4. Configure request time monitor
5. Change dynamic dump sleep time
6. Change dynamic dump sleep interval
7. Enable or disable a status log mask option
8. Change advanced encryption master password
9. Change a DIAGNOSTICS option
10. Change the specified configuration option
11. Change Logon Block Settings

Enter your choice (1-11), or 'q' to return to previous menu>> 10

Enter the configuration option and its value >> set_cache_options {"dat_memory": "200 mb"}

Successfully changed the configuration option.

Press RETURN to continue...
```

Expected Status Log Messages

The resize request triggers an internally generated server quiesce state with the following common messages.

```
Mon Jun 13 13:47:51 2022 - User# 00030      ctQUIET: attempt quiet transaction state with action:
1100a0x timeout: 1 sec
Mon Jun 13 13:47:51 2022 - User# 00030      ctQUIET: blocking call with action: 641a2x
Mon Jun 13 13:47:54 2022 - User# 00030      Transaction aborted at ctQTaborttran for user# 33: 817
Mon Jun 13 13:47:54 2022 - User# 00030      Transaction aborted at ctQTaborttran for user# 40: 817
Mon Jun 13 13:47:54 2022 - User# 00030      Transaction aborted at ctQTaborttran for user# 47: 817
Mon Jun 13 13:47:54 2022 - User# 00030      Transaction aborted at ctQTaborttran for user# 52: 817
Mon Jun 13 13:47:54 2022 - User# 00030      Transaction aborted at ctQTaborttran for user# 58: 817
Mon Jun 13 13:47:54 2022 - User# 00030      Transaction aborted at ctQTaborttran for user# 65: 817
Mon Jun 13 13:47:54 2022 - User# 00030      ctQUIET: end of blocking call
Mon Jun 13 13:47:54 2022 - User# 00030      Using 3 index node LRU lists with 3962 buffers per list
Mon Jun 13 13:47:54 2022 - User# 00030      ctQUIET: unblocking call with action: 809e00x
Mon Jun 13 13:47:55 2022 - User# 00030      ctQUIET: end of unblocking call
```

Examples

1. Set data cache size to 100 MB and keep existing data in cache (since "keep_cached_data" defaults to "yes"). No change to index cache. In this example, if data cache is already 100 MB, no action is taken:
`set_cache_options {"dat_memory": "100 mb"}`
2. Set data cache size to 100 MB and remove all data from data cache:
`set_cache_options {"dat_memory": "100 mb", "keep_cached_data": "no"}`



3. Keep current data cache size and remove all data from data cache:
`set_cache_options {"keep_cached_data": "no"}`
4. Set index cache size to 200 MB and keep existing nodes in cache:
`set_cache_options {"idx_memory": "200 mb"}`
5. Set index cache size to 200 MB and remove all nodes from index cache:
`set_cache_options {"idx_memory": "200 mb", "keep_cached_nodes": "no"}`
6. Keep current index cache size and remove all nodes from index cache:
`set_cache_options {"keep_cached_nodes": "no"}`

This example shows actions on both data and index cache:

```
set_cache_options {"dat_memory": "500 mb", "idx_memory": "400 mb",  
"keep_cached_nodes": "no", "timeout": 5}
```

Direct SDK API

An application that is logged in as the ADMIN user can change data and/or index caches by calling the **ctSETCFG()** API function with option of *setcfgCONFIG_OPTION* and value set to:

```
set_cache_options {<cache_options>}
```

where <cache_options> is the null-terminated string in JSON format.

This example showing how to change the cache sizes in C or C++ code by calling the **ctSETCFG()** API function:

```
NINT rc;  
rc = ctSETCFG(setcfgCONFIG_OPTION, "set_cache_options {\"dat_memory\": \"100 mb\"}");
```

Diagnostic Logs

The configuration option *DIAGNOSTICS RESIZE_CACHE*, which can be specified in *ctsrvr.cfg* and changed at runtime, causes cache resize diagnostic messages to be logged to *CTSTATUS.FCS*.

2.2 Enhancements to Prime Cache at Server Startup

FairCom Server technology supports configuration options that are used to prime the server's data and index caches when the server starts up. The functionality is similar to cache priming



configuration options PRIME_CACHE, PRIME_CACHE_BY_KEY, and PRIME_INDEX, except that those options take effect only when an application opens the files.

This feature relies on atomic operations. If atomic operation support is off at compile time, the feature will be disabled.

The new configuration options are specified in JSON format, such as in the following example:

```
subsystem cache prime_cache_at_startup {
  "allow_connections": true,
  "thread_count": 8,
  "prime_cache": [
    {"file": "filename1.dat", "size": "1 gb"},
    {"file": "filename2.dat"}
  ],
  "prime_cache_by_key": [
    {"file": "filename1.dat", "size": "1 gb", "index": 1, "reverse": false},
    {"file": "filename2.dat", "size": "1 gb", "index": 1}
  ],
  "prime_index": [
    {"file": "filename1.dat", "size": "1 gb", "index": 1},
    {"file": "filename2.dat", "size": "1 gb", "index": 1}
  ]
}
```

Supported attributes:

- **allow_connections** is a *boolean* value that indicates if connections to the server are allowed while the cache priming is performed at server startup. Defaults to false.
- **thread_count** is an *integer* value that sets the number of threads to use to prime the cache. It defaults to 4. Maximum is 255.
- **prime_cache** is an *array* of names of data files that are to be read into the cache in physical order, and an optional size indicating the maximum number of bytes to read into data cache for the file. The file name is a string. The file name can include wildcard characters (for example, *.dat). The wildcard character does not recurse into subdirectories. The size can include a suffix, such as kb, mb, gb, tb (for example, "1 gb"). If size is not specified, the entire file is read into the data cache.
- **prime_cache_by_key** is an *array* of names of data files that are to be read into the cache in the order of the specified index. The index number is a positive value, where 1 means the first index. If the index number is omitted, the first index is used. Size has the same definition as for prime_cache. Reverse is a boolean indicating whether the key traversal is in ascending or descending key order.
- **prime_index** is an *array* of names of index files that are to be read into the index cache. Size and index have the same meaning as for prime_cache_by_key.

In addition to the specified size limits, the data and index cache sizes limit the cache priming as follows:

prime_cache and **prime_cache_by_key** stop if the data cache becomes full.

prime_index stops if the index cache becomes full.

The SUBSYSTEM CACHE PRIME_CACHE_AT_STARTUP configuration option can be specified in a server settings file. If it is used in a settings file, the option cannot also be used in the



configuration file. In that case, the subsystem is considered blocked and the configuration file options are ignored. Likewise, if this subsystem is specified more than once in the configuration file, only the first occurrence of the subsystem block takes effect. The other occurrences are ignored.

The subsystem can be commented out by placing a semicolon at the start of the SUBSYSTEM line. However, individual lines within the subsystem block cannot be commented out. Attempting to do so will cause a syntax error.

By default, the server logs information about the files that it loads into cache to *CTSTATUS.FCS*. The configuration option `CTSTATUS_MASK PRIME_CACHE_AT_STARTUP` can be used to disable this logging.

2.3 Enable Default Data Cache Empty Lists

Cache improvements introduced in v11 changed the data cache from an LRU list approach, which involved taking pages on and off the LRU lists on each access, to a sequential search over lists of all cache pages. On these lists, the next page that is found is used, regardless of whether it is currently in use or not. As a result, no priority is given to assigning an empty page to a cache request.

Reports described slowdowns as expected data was not retained in cache. This was due to the fact that "hot" cache pages were being reused rather than obtaining pages from the unused pages available.

FairCom DB supported maintaining a list of empty cache pages, however, the feature was not enabled. Prior to v12.5.0 it was possible to add `USE_EMPTY_LISTS YES` to *ctsrvr.cfg* and optionally enable the empty data cache list feature. The empty list feature is now enabled by default as it is much more efficient cache usage. To disable this feature if needed, add `USE_EMPTY_LISTS NO` to *ctsrvr.cfg*.

2.4 Online Compact and Rebuild

Online file compact and rebuild support is available starting with FairCom DB V13.

For background, since FairCom DB reclaims the space consumed by deleted records when new records are inserted and additionally for variable length files when records increase in size, a compact operation is typically only needed in the following scenarios:

1. A compact operation is being used as a convenient way to reformat the file. For example, changing the `PAGE_SIZE` of the file, adding or removing encryption, adding or removing data compression or index compression, and so forth.
2. A Hot Alter Table (<https://docs.faircom.com/doc/ctreeplus/74546.htm>) operation has been called and you want all the records converted to the latest schema version.
3. A large number of records have been deleted and the disk space consumed by the deleted records is needed immediately. You do not want to wait until new records are inserted to reclaim the space.



The FairCom compact and rebuild operations available prior to V13 require files to be opened in exclusive mode. This is problematic for systems that have minimal windows available for downtimes, therefore an online facility for managing these processes is now available.

The online compact and rebuild operations allow for the files to be actively used by other users while these operations take place. These operations are generally slower than the compact and rebuild versions that require the file to be opened exclusively.

Online compression is not allowed for changing encryption or compression settings.

During an online compact operation, if the file is under full transaction control, the indexes are automatically rebuilt using Online Rebuild.

Online Rebuild is only available for indexes under full transaction control (ctTRNLOG file mode).

See also

- CMPIFIL CompactFile function and its extended versions
- ctcmpcif compact utility
- ctdbRebuildTable
- RebuildIfil function and its extended versions
- ctrbldif rebuild utility

2.5 On-line Compact for Background Table Defragmentation

FairCom DB tables can become larger than necessary as small fragments of deleted records accumulate over time. To remove this unused and abandoned space, it was necessary to take the table off-line and compact the file. It was required to be off-line as the compact operation read all valid records from the original table, and wrote into a new copy of the table. Only at the end of the process did the new copy replace the original. This could also take a long time for very large tables.

For high availability an on-line facility for managing this process was needed where a table can be "compacted" while open and concurrently accessed by applications. This support is available from FairCom DB.

On-line Compact

To enable background defragmentation processing, call one of the **CompactIfil()** family of functions using the *onlineIfilOption* bit:

```
myifil.tfilno = setIfilOptions(onlineIfilOption);
```

The user initiating the compact must have the file closed before calling the compact API. When *onlineIfilOption* is enabled a more limited form of table compact is initiated and referred to here as defragmentation. The table remains open and available to other users. Defragmentation attempts to reorganize records within the table to minimize file size. When defragmentation is completed, the size of the file is reduced, if possible.



ctdbRebuildTable() also supports an optional mode *CTREBUILD_ONLINE* when used with the existing *CTREBUILD_COMPACT* mode. This allows for an on-line version of the compact operation, which runs while tables are concurrently open for update by other users.

Operational Details

Defragmentation works similar to a regular user updating records as usual within a table. A record is deleted and then re-added into a position selected to reduce the overall file size. Due to the type and extent of changes during defragmentation, it may highlight artifacts of multiuser interference not frequently observed by applications. Applications using tables without transaction control, or using physical order scans, or using more relaxed locking protocols may see increased rates of these artifacts when defragmentation is in progress. This may appear as unexpected “missing” or even “duplicate” records (which a future scan doesn't encounter).

Limitations

On-line compact is only available in server models. Other compact options specified via **setIFILoptions()** are not available in combination with *onlineIFILoption*. Defragmentation is not supported for memory files, superfiles, partition host files, segmented files, or index files.

2.6 Online Rebuild Support

An online index rebuild has been implemented augmenting the online file compact feature. It requires the target file to have *ctTRNLOG* file mode (full transaction control).

To invoke an online rebuild, use the *onlineIFILoption*:

```
myifil.tfilno = setIFILoptions(onlineIFILoption);
```

```
RBLIFIL(&myifil);
```

The file must be closed by the calling user, as for a normal rebuild, but may be open by other users. The *IFIL* definition provided by the caller is NOT used, but rather the *IFIL* resource embedded in the file. Behavior is undefined if the *IFIL* resource embedded in the file doesn't match definitions used to create the indexes.

Details

Online rebuild uses the dynamic dump and immediate restore, so it requires the **ctrdmp** binary exists in the server working directory. Only one dynamic dump may occur at a time, so the online Rebuild will wait to begin if backups or other online rebuilds are in process. When the new index is ready, access to the file will be suspended while the index is replaced. Any active transactions that have updated this file will be aborted at this point. If an error occurs after this point, the index will need to be rebuilt in exclusive mode.

The rebuild works in several stages:

1. A background thread (rebuildManager) is spawned and uses a new type of log reader identified by the *REPLSESS_REBUILD* replication session type. However, the target file does not need to meet replication requirements. This thread watches for key changes on the target file during the rebuild process.



2. The rebuild thread makes a dynamic dump backup and IMMEDIATE_RESTORE of the target data file to a temporary directory.
3. The rebuildManager begins reading key changes from the transaction log to a memory queue.
4. The rebuild thread does a standard exclusive mode rebuild on the backup of the target data file.
5. The rebuild thread applies key changes from the memory queue to the new copy, bringing it up to date.
6. The rebuildManager does a fileblock (using mode *ctFBSuspend* | *ctFBsysclose*) on the target data file. Any active transactions that have updated this file are aborted. This should be seamless to external users of the file, except their active transactions may be aborted.
7. The rebuild thread deletes the original index and replaces it with the newly rebuild copy.
8. The file is unblocked. The unblock does some sanity checking that the new index headers match the originals, beyond attributes that are expected or anticipated to change. All cached index pages for the target index are invalidated.

LIMITATION: If an error occurs on step 7 or later, the index must be rebuilt in exclusive mode.

2.7 Recycle Bin Preserves Deleted Files and Protects Against Accidental Deletes

FairCom DB is a file-based database technology. All data and indexes reside in independent files on the filesystem. These can be quite numerous in many applications, and span multiple filesystem folders. Applications can easily delete files on-the-fly and in some applications this is under end user control. And mistakes happen! In nearly all filesystems, recovering a deleted file is extremely difficult if not impossible. A backup might not be up to date enough. What has been requested is a temporary holding of deleted files such as a "rubbish bin" that can later be emptied.

FairCom DB now supports optionally deleting a file to a recycle bin. Delete operations can now move a deleted file to a designated folder instead of immediately deleting the file from disk. A configurable background task then runs periodically emptying, that is, permanently deleting, previously deleted files.



Recycle bin support

The recycle bin feature is supported at FairCom DB ISAM and low-level API layers. Extended ISAM and low-level file delete function versions are introduced allowing this new support.

DELRFILX()

DELIFILX()

DELFILX()



SERVER CONFIGURATION OPTIONS

The recycle bin is configured with `SUBSYSTEM CTFILE RECYCLE_BIN`. This subsystem configuration option supports the following options:

- *recycle_folder_name*: directory name where the recycle bin is stored and can be absolute or relative. If relative, the directory is relative to the server's `LOCAL_DIRECTORY` configured directory or current working directory if `LOCAL_DIRECTORY` is not used.
- *purge_frequency_minutes*: time in minutes between checks for files that should be purged from the recycle bin.
- *max_file_age_days*: maximum number of days a file is kept in the recycle bin.
- *max_folder_storage_size*: maximum total byte size of the files in the recycle bin. If deleting a file causes the recycle bin to go over the limit, files are deleted until the size is below the limit.

Example

```
subsystem ctfile recycle_bin {
    recycle_folder_name      deleted_files
    purge_frequency_minutes  60
    max_file_age_days        30
    max_folder_storage_size  1 gb
}
```

MONITORING

The statistic utility, **ctstat** supports a *deleted_files* option listing the total size and detail list of files currently in the recycle bin.

Example

```
ctstat -deleted_files -u ADMIN -p ADMIN -s FAIRCOMS -i 1 -h 1 -t

Thu Dec  9 15:01:48 2021
Total file size: 917500 for 4 file(s)
Deleted file of size 131071 deleted\1639081086.20211209\qatranrep1097.sup.992
Deleted file of size 131071 deleted\1639081086.20211209\qatranrep1097.sup.998
Deleted file of size 327679 deleted\1639081086.20211209\qatranrep1097.sup.1017
Deleted file of size 327679 deleted\1639081086.20211209\qatranrep1097.sup.1026
```

NEW SERVER FILES

Two new files are required for tracking inactive files in the recycle bin. *RECBINDT.FCS* and *RECBINIX.FCS* are transaction-controlled data and index files retaining data about the current recycle bin state. The data file contains a state record for each deleted file, and a state record that holds the total space in use by deleted files. The index enables the background thread to find files that were deleted from the oldest to newest timestamp.

REPLICATION AGENT CHANGES

The replication agent supports a configuration option to enable whether the agent uses the recycle bin any time it deletes a file on the target server. The *ctreplagent.cfg* option is `delete_to_recycle_bin yes | no` and is on by default.

**NEW FUNCTION PROTOTYPES:****CTERR** DELRFILX(FILNO datno, pctDELETE_FILE_XTD_OPTIONS pDeleteFileOptions);**CTERR** DELFILX(FILNO filno, pctDELETE_FILE_XTD_OPTIONS pDeleteFileOptions);**CTERR** DELIFILX(pctDELETE_IFIL_XTD_OPTIONS pDeleteIFILXtdOptions);**NEW DATA TYPE DEFINITIONS:**Parameters for **DELFILX()** and **DELRFILX()** function calls as defined in *ctopt2.h*.*optionBits* bit field values

```
typedef enum ctDeleteFileOptionsBits_t {  
    ctDELETE_TO_RECYCLE_BIN = 0x00000001 /* move files to recycle bin instead of deleting */  
} ctDELETE_FILE_OPTIONS_BITS;
```

```
typedef struct ctDeleteFileOptions_t {  
    UCOUNT structVersion; /* [IN] version of this structure */  
    UCOUNT pad; /* unused padding */  
    ULONG optionBits; /* [IN] option bits */  
} ctDELETE_FILE_XTD_OPTIONS, *pctDELETE_FILE_XTD_OPTIONS;
```

Parameters for **DELIFILX()** function call

```
typedef struct ctDeleteIFILXtdOptions_t {  
    UCOUNT structVersion; /* [IN] version of this structure */  
    UCOUNT pad; /* unused padding */  
    ULONG optionBits; /* [IN] option bits */  
    pIFIL pifil; /* [IN] IFIL pointer */  
} ctDELETE_IFIL_XTD_OPTIONS, *pctDELETE_IFIL_XTD_OPTIONS;
```

COMPATIBILITY NOTES

The new extended delete functions require both a new client library and a new server. Calling a new function on an old server will fail with error **SFUN_ERR** (170) "bad function number".

Likewise, if a new replication agent is using the *delete_to_recycle_bin* yes option in *ctreplagent.cfg* with an old server, its replicated file delete operations will fail with error 170.

AFFECTED COMPONENTS: server, client library, replication agent.



2.8 File Names Supported up to 4K

Prior to this release, c-tree limited file names to 255 bytes. This limit is controlled by a compile-time macro, `MAX_NAME`. Modern operating systems support file names over 255 bytes and in this revision will allow `MAX_NAME` to be increased, supporting file names up to 4096 bytes.

Note: When enabled, these changes break compatibility with existing c-tree binaries and dynamic dump backups.

Windows Notes:

Historically, Windows API functions have limited file names passed to its file and directory functions to `MAX_PATH` (260) bytes. Starting with Windows 10, version 1607, this limit has been removed from common Windows API file and directory functions. For information about Windows file naming conventions, see: *Microsoft Windows 10 - Naming Files, Paths, and Namespaces* (<https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file#enable-long-paths-in-windows-10-version-1607-and-later>).

As noted on that page, you must opt in to the new behavior. To enable this support:

1. Create the registry key `HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\LongPathsEnabled` (Type: `REG_DWORD`) and set it to 1. Then reboot the system.
2. Create a manifest file named `longPath.manifest` containing the following text:


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0">
<application xmlns="urn:schemas-microsoft-com:asm.v3">
  <windowsSettings
xmlns:ws2="http://schemas.microsoft.com/SMI/2016/WindowsSettings">
    <ws2:longPathAware>true</ws2:longPathAware>
  </windowsSettings>
</application>
</assembly>
```

In the Visual Studio project settings for each executable that uses Windows API functions with a file name over 260 bytes (such as `ctsrvr.exe` and `ctrdmp.exe`), specify the name of this manifest file under **Properties | Manifest Tool | Input and Output | Additional Manifest Files**. Note this has already been done in the makefiles created by FairCom's `mtmake` make builder.

mtmake Parameter Adds Manifest for Long Paths

A new `mtmake 4Kfilepaths` parameter has been added to enable the extra manifest for `longPathAware` builds.

```
{"4Kfilepaths", "Add manifest to support 'longPathAware' 4K file paths.", MTOPTN_NORMAL},
```



2.9 PAGE_SIZE Changes Automatically Applied to all FairCom Controlled Files

Operating systems have a default file block size they read and write data in. Most recent Linux and Windows operating systems are 4K. Other operating systems will vary. Setting the index page size to match the OS or multiples of OS file block size can improve I/O performance. FairCom Database Engine supports setting PAGE_SIZE from 512 bytes up to a maximum of 64K (65,536).

The PAGE_SIZE setting used by FairCom DB can have a dramatic effect on database performance. In most applications, a larger page size is a benefit, especially if index keys are relatively large (> 20 bytes or so), or there is a reasonable number of key values (consider 100K key values or greater).

You should set your c-tree server PAGE_SIZE to at least the OS file block size, nothing smaller. For example, if you set c-tree's PAGE_SIZE to 2K, then on Windows you are reading 4K at the OS level and throwing away 2K every time.

For most applications, 32K is a well performing value. If you are looking to fully optimize your application, and if your files are fairly large, you might gain performance by increasing this setting, in multiples of your operating system page size value. The best method for tuning this value is to try a few different sizes with exact application configurations and a copy of your data. For example, increase from 8K to 16K and if your performance increases, then continue and test with 32K. Once your performance starts dropping, then scale back in smaller increments looking for the sweet spot for your application. It's possible your application sweet spot might be less than 32K.

For V13/V12, FairCom has increased the default PAGE_SIZE setting to 32K (V10 and V11 defaulted to 8K). There are two options for addressing this topic:

1. Staying with your current PAGE_SIZE setting is an option. You can set PAGE_SIZE X to your current value (where X is your current PAGE_SIZE). This will prevent the need to rebuild your application indexes.
2. Rebuild your application indexes to utilize either FairCom's new 32K default (or a PAGE_SIZE better suited to your application) by following the testing procedure above.

Changing PAGE_SIZE

Upgrading FairCom Servers to a different page size is a maintenance task that should be done carefully after a full, reliable backup. Failure to perform the procedures (provided later in this section) will cause the system to be halted with a KSIZ_ERR, error 40.

In V12.0.1 and later, the server will automatically convert its internal superfiles to the server's current page size. However, you are responsible for converting your application index files and any superfiles created by your FairCom DB based application.

When the PAGE_SIZE server configuration option is changed, FairCom applies this change to all FairCom server-controlled files. At server startup we check the page size of the following FairCom files:

- *ctdbdict.fsd*
- *Database dictionaries*



- *FAIRCOM.FCS*
- *SEQUENCEDT.FCS*
- *DFRKSTATEDT.FCS*
- *SYSLOGDT.FCS*
- *REPLFFCHGDT.FCS*
- *REPLOGDT.FCS*
- *REPLOGSHIPDT.FCS*
- *REPLSTATEDT.FCS*
- *REPLSYNCDT1.FCS*
- *REPLSYNCDT2.FCS*

If any do not match the newly specified `PAGE_SIZE`, a backup copy of the existing file is made with an *.FCB* extension. The server process then attempts to invoke the FairCom **ctscmp.exe** (**ctscmp** on Linux/Unix) superfile rebuild utility with the new configured `PAGE_SIZE`.

Superfiles require that the `PAGE_SIZE` at open matches the `PAGE_SIZE` at file creation time, or a **KSIZ_ERR** (40) or **SPAG_ERR** (417) error occurs at file open. For *FAIRCOM.FCS*, this prevents the server from starting when operating with a different configured `PAGE_SIZE` setting.

A new configuration keyword has been added to control the automatic page size conversion at startup:

```
PAGE_SIZE_CONVERSION {YES|NO}
```

Default: YES

Limitations:

1. We expect **ctscmp.exe** (**ctscmp** on Linux/Unix) to exist in the process working directory and have permission to be executable by the server process. If this utility does not exist, the server fails to start with the following messages likely logged to *CTSTATUS.FCS*:
 - User# 00001 Wrong PAGESIZE for FAIRCOM.FCS, attempting to convert file from PAGESIZE 8192 to 32768
 - User# 00001 Did not find ctscmp in working directory for Superfile conversion: 2
 - User# 00001 Could not process User/Group Information in FAIRCOM.FCS: 417
 - User# 00001 Could not initialize server. Error: 417
 - User# 00001 O1 M0 L74 F417 P0x (recur #1) (uerr_cod=417
2. The `LOCAL_DIRECTORY` keyword must be set and must be different from the working directory. Otherwise, **ctscmp.exe** will encounter a **TCOL_ERR** (537) and fail.
3. Superfile conversion occurs only after auto-recovery. Recovery is likely to fail if run with a different `PAGE_SIZE` setting. We expect any needed recovery should occur before making a major configuration impacting the physical attributes of critical operational files such as authentication information and SQL database dictionaries (system tables).
4. **NO FURTHER ATTEMPT IS MADE to convert any other existing indexes. All other application created indexes will require manual rebuilding to convert to a new page size. See the steps below for the best practice procedures.**

The following messages may be found logged to *CTSTATUS.FCS* after a successful conversion:



```
- User# 00001      Wrong PAGESIZE for FAIRCOM.FCS, attempting to convert
file from PAGESIZE 8192 to 32768
- User# 00001      Backup created ../data/\FAIRCOM.FCS =>
../data/\FAIRCOM.FCS.1621883226.FCB
- User# 00001      Superfile conversion successful
- User# 00001      Wrong PAGESIZE for ctddbdict.fsd, attempting to
convert file from PAGESIZE 8192 to 32768
- User# 00001      Backup created ../data/\ctddbdict.fsd =>
../data/\ctddbdict.fsd.1621883232.FCB
- User# 00001      Superfile conversion successful
- User# 00001      Wrong PAGESIZE for
.\ctreeSQL.dbs\SQL_SYS\ctreeSQL.fdd, attempting to convert file from
PAGESIZE 8192 to 32768
- User# 00001      Backup created
../data/\.\ctreeSQL.dbs\SQL_SYS\ctreeSQL.fdd =>
../data/\.\ctreeSQL.dbs\SQL_SYS\ctreeSQL.fdd.1621883237.FCB
- User# 00001      Superfile conversion successful
```

Once correct server operations are confirmed after conversion, the ***.FCB** files can be removed and deleted.

Changing Application PAGE_SIZE Manually

Changing the application index page size is a maintenance task that should be done carefully after a full, reliable backup. When making a **PAGE_SIZE** configuration change with existing indexes, those indexes must be rebuilt using the new node size before they can take advantage of the new configured size.

- **ctrbldif.exe** can be used to make this modification for fixed-length files
- **ctcmpcif.exe** must be used for variable-length files.



Existing indexes with nodes smaller than a configured `PAGE_SIZE` will display the following informational messages in `CTSTATUS.FCS` upon file open:

```
- User# 00027      Mismatched PAGE_SIZE is wasting 75% of Index cache used by  
.\ctreeSQL.dbs\admin_custmast.dat
```

This message appears because the smaller node size of the existing index fits into the larger page size value as allocated in the index buffer cache with the remainder of the cache page space wasted. Note that even though in this case the file will open successfully, you are wasting memory. Consider with the message above, if you are using an 8K `PAGE_SIZE` in the FairCom DB Server, and your application index file is set to 2K, then 6K of memory is wasted with each index page loaded into the Index Cache.

FairCom DB V13/V12 defaults to a larger page size configuration recommended for modern client-server applications. However, existing index files must be rebuilt to take advantage of this larger index node size. Opening existing indexes smaller than the configured server value results in a warning message logged to `CTSTATUS.FCS` about wasted memory usage.

For applications that have large numbers of files, or that frequently open and close files, this may result in large numbers of messages to `CTSTATUS.FCS`, and this logging is disabled after the first 20 occurrences.

The following command turns off all `PAGE_SIZE` warnings so *no* warnings are produced:

```
CTSTATUS_MASK WARNING PAGESIZE
```

Setting this keyword disables the logging of warning messages like the one shown above.

For debugging, if it is necessary to see all warnings, you can use the following keyword to log *all* the warnings to `CTSTATUS.FCS`:

```
CTSTATUS_MASK WARNING PAGESIZE_SEEALL
```

Note: Existing indexes with a larger page size than the configured `PAGE_SIZE` continue to result in an error `KSIZ_ERR` (40) on open.

Rebuilding Application Files

Recommendation: If you are considering changing your `PAGE_SIZE` setting, be sure to rebuild ALL of your index files.

If you are reducing your `PAGE_SIZE` setting, and you miss rebuilding an index, and its value is set larger than your current `PAGE_SIZE` setting, then you will get an error 40 (`KSIZ_ERR`, Index node size too large) when this file is opened and it will need to be rebuilt to resolve this error.

If your index file(s) have a `PAGE_SIZE` set less than your current `PAGE_SIZE` setting in `ctsrvr.cfg`, then you will be wasting memory. The file will open fine, because a smaller index `PAGE_SIZE` will fit in the larger `PAGE_SIZE` set in `ctsrvr.cfg`.

Recommendation: Experiment on *copies* of your data and server folder. When you are confident with the results, back up your source data and then change the live production files.



Compatibility

The page (node) size is a permanent attribute of an index when it is created. Index nodes remains that size until rebuilt or compacted with a different size. There are limitations with existing files using a new larger page size.

- c-tree can open existing indexes with a *smaller* page size than currently configured. There is a minor memory use trade off in doing so. As each index node maps to a single server cache page, and the server cache page is allocated as a page size, the unused space in the cache page is lost. This can be significant. If a server configured for a 32K page size index cache is at 100% capacity of 8K index nodes, *up to 75% of cache memory is unused*. For very large caches this is significant.

It will be extremely beneficial to rebuild existing indexes and take full advantage of both increased index page size benefits as well as full cache memory usage.

Opening indexes created with page sizes larger than currently configured results in error 40 (**KSIZ_ERR**).

- *Superfiles can ONLY be opened with the same configured page size as they were originally created.* This has important implications with critical c-tree housekeeping files:
 - *FAIRCOM.FCS* - This file maintains all user, group and password hash information.
 - *ctdbdict.fsd* - This file maintains the catalog of available databases (SQL and c-treeDB).
 - *<database_name>.fdd* - This file maintains the SQL database system tables (catalog or dictionary).

Using these existing superfiles with a different page size will result in a server startup failure. Opening a superfile with a different page size also results in error 40 (**KSIZ_ERR**). You will find this message logged in *CTSTATUS.FCS* should the server make this failed attempt.

Rebuilding Existing Indexes

The following options are available for rebuilding indexes to take advantage of increased page size.

ALWAYS MAKE CLEAN BACKUP COPIES OF THESE FILES BEFORE YOU BEGIN THESE PROCEDURES

The index rebuild utility, **ctrblidf**, is the quickest easiest option for most indexes. Simply pass the new **sect** size.

Remember, each sect = 128 bytes. For 32768 your sect size will be 256.

```
> ctrblidf mytable.idx -256 ADMIN ADMIN FAIRCOMS
```

For superfiles, use the superfile compact utility, **ctscmp**.

```
> ctscmp ctdbdict.fsd Y 256
```

To make superfiles that can be opened by versions earlier than V12, use **ctscmp** with the **-v11** option. This option forces the utility to avoid introducing V12-specific features to the resulting file.
Usage: **ctscmp** filename [-v11] [sectors]

PAGE_SIZE Change Procedures

For procedures explaining how to rebuild files affected by a change of **PAGE_SIZE**, see **Adjusting PAGE_SIZE** (/doc/FairCom-Installation/AdjustingPAGE_SIZE.htm) in the *FairCom Installation Guide*.



2.10 Create system log files as huge files to remove 4GB file size limit

The server now creates system log files as HUGE files so they are not limited to 4GB. HUGE files can be as large as 16 exabytes. At startup, if the server finds existing non-huge system log files, it renames them to *SYSLOGDT.FCA* and *SYSLOGIX.FCA* (first deleting any existing files having those names), and then the system log thread re-creates the system log files as huge files.

To disable the creation of huge system log files, add `SYSLOG NONHUGE` to the server configuration file.

2.11 SYSLOG improved reuse of disk space

After a `SYSLOG PURGE`, messages could fail to be inserted because they exceeded the largest available deleted space and the `SYSLOG` was at its non-huge size limit. This can occur because *ctTRNLOG* files do not coalesce adjacent deleted records by default unless a `RECBYT` index is available.

For efficient space reuse a `RECBYT` index has been added to the `SYSLOG` tables. At startup, existing `SYSLOG` tables are checked for the requested `RECBYT` setting, and recreated if necessary. *SYSLOG*.FCS* files are renamed to *SYSLOG*.FCA* before this conversion and can be removed once successful `SYSLOG` behavior is confirmed.

Prior behavior can be restored by specifying keyword `SYSLOG V11_REUSE`.

2.12 SYSLOG - Efficiently Truncate

`SYSLOG` is a FairCom DB subsystem to secure database events for auditing. Over time this table can become quite large and existing purge functions are not efficient enough. The `SYSLOG()` function supports purging records, optionally filtered by time and by event code. A notable problem is even when all entries are "purged" storage device space is not released leaving potentially very large (up to 4GB) empty files on disk. The solution to this problem is a new `SYSLOG` truncate capability.

With `SYSLOG TRUNCATE`, present in your configuration file if a complete purge is requested (no filtering by time or event) we truncate the file rather than delete individual records.

This approach is much faster and avoids limitations with space reuse. Any users reading from `SYSLOG` will encounter error `FBLK_ERR` if the log is truncated while they have it open. The table must be closed and reopened after receiving this error.



ctadmnn

The **ctadmnn** utility has been enhanced to support **SYSLOG** purging with time-based filtering.

The "Monitor Server Activity" menu now has a new option, "Purge SYSLOG entries", which prompts for the number of days of activity to keep. "0 Days" results in a truncate, while larger values use the regular approach.

2.13 Server Read-only Status Logged to CTSTATUS.FCS

CTSTATUS.FCS logging has been updated to record when a server is configured in read-only mode, or when read-only mode is turned off.

The following message is now printed when read-only mode is set:

```
server set in read-only mode
```

When read-only mode is turned off, the following is printed:

```
server set in read/write mode
```

2.14 Server internal processing files moved from working directory to LOCAL_DIRECTORY

We have moved some internal processing files to the **LOCAL_DIRECTORY** folder.

Before we had a number of files default to being written in the database process working directory. Common security practices mark directories with executable files as non-writable.

Some of these file include:

- The ctagent plugin replication deploy operations intermediate backup and restored files.
- Server generated diagnostic pstacks and dumps.
- For Replication Manager we moved the 'CTSTATUS.FCS' created by the 'ctrdmp' process to the replication folder on completion.



2.15 Read Permissions - Server now checks for read permission on function calls that read data or key values

We have added the ability to prevent clients from reading records from a table. The server allowed read operations on the table even if the file permissions did not grant read permission. We changed the server's behavior to deny read access if the file permission does not grant read access to the caller.

This support was added to support a dbNotify client's ability to add data to a FairCom MQ broker, yet restrict that data from being read.

2.16 Log message to CTSTATUS.FCS when transaction commit fails

In order to help tracking down customer related issues, we added an additional diagnostics message to our server's log file. When a transaction commit fails (which is an unexpected situation), we log an error message in the following format to CTSTATUS.FCS:

```
Fri Feb  2 11:57:11 2024 - User# 00005  WARNING: Commit of transaction  
<transactionNumber> failed: <errorCode>
```

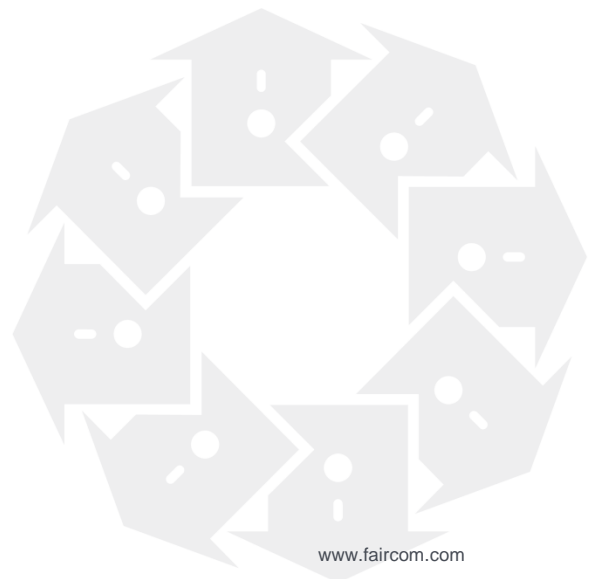
3. Server Plug-Ins

3.1 dbNotify - database triggers to send MQTT messages

A new FairCom DB feature, called DBNotify, allows the user to set up database triggers in the FairCom DB database, which will send MQTT messages to the broker. The broker offers standard topic and subscription management thus bridging traditional database data to the world of Messaging.

Here are a handful of issues addressed in this domain:

- change default quality of server (qos) from 0 to 2.
- add an additional message to payload identifying which field(s) have been changed.
- add an additional metadata to know which fields are “key” fields.
- support TLS in dbNotify.
- support X.509 in dbNotify.
- support metadata: constants to be added in the JSON generated by dbNotify.
- support end-user tamper proof (.set file) dbnotifyconnections.json.





3.2 Interface for Managing Plug-ins on-the-fly

FairCom plug-in support has been enhanced by providing multiple interfaces for starting, stopping, and making calls to FairCom plug-ins on-the-fly. This allows more flexibility in how and when plug-ins are run. Previously, all plug-ins had to be configured and enabled at server launch. Now, run plug-ins on demand as they are needed and stop them when no longer required. Change the plug-in configurations and restart, allowing new settings to immediately take effect --all without taking the server down. A plug-in must be loaded (by uncommenting it in the server configuration file or via the **ctadmn** interactive option) before it will be available to stop or start.

Command-Line Administrator Utility

You can call **ctadmn** as a command-line utility to start and stop a plug-in using the following syntax:

```
ctadmn -s <server name> -u <user name> -p <password> -c "<command>"
```

where the <command> format is:

```
"plugin <plug-in name> start|stop"
```

Note: The command section must be enclosed in quotes.

ctadmn can execute any generic operation from the plug-in using the following syntax:

```
ctadmn -s <server> -u <user> -p <password> -c "plugin <plug-in name> <command> <arguments>"
```

Example 1 - Execute **ctadmn** command-line utility with direct arguments and start the web services plug-in:

```
ctadmn -s FAIRCOMS -u ADMIN -p ADMIN -c "plugin cthttpd start"
```

Example 2 - Execute **ctadmn** command-line utility interactively via option 10 and load and start the web services plug-in:

1. Execute the **ctadmn** utility.
2. Select option 10 - "Change Server Settings."
3. Again select option 10 - "Change the specified configuration option."
4. Enter the configuration option and its value:


```
>> PLUGIN cthttpd; ./web/cthhttpd.dll
```

Successfully changed the configuration option.

Example 3 – Using the **ctadmn** command-line utility to restart the OPC/UA plug-in after changing the settings:

1. Edit the *ctopc.json* file to add a device or change the connection details to a device.
2. Stop the OPC plug-in via **ctadmn**: `ctadmn -s FAIRCOMS -u ADMIN -p ADMIN -c "plugin ctopc stop"`
3. Restart the plug-in via **ctadmn**: `>ctadmn -s FAIRCOMS -u ADMIN -p ADMIN -c "plugin ctopc start"`
4. The OPC/UA plug-in will now load the modified configuration file and use the new settings.

Controlling plug-ins via C API

Control plug-ins via 2 C API functions, **ctSETCFG()** and **ctPlugin()**. Use **ctSETCFG()** to load a plug-in, **ctPlugin()** to start or stop a loaded plug-in.

ctSETCFG Function Call



```
NINT ctSETCFG(NINT option, pTEXT value);
```

Where:

- *option*: Set this to *setcfgCONFIG_OPTION*
- *value*: Set this to a string containing the same plug-in name and path as shown in your *ctsrvr.cfg* server configuration file.

Example 4 - Load the web services plug-in:

```
ctSETCFG(setcfgCONFIG_OPTION, "PLUGIN cthttpd;./web/cthttpd.dll");
```

ctPlugin Function Call

```
NINT ctPlugin(ctPLUGIN_COMMAND command, pTEXT inputBuffer, pTEXT outputBuffer, pVRLEN pOutputBufferSize);
```

where:

- *command*: currently, the options are *ctPLUGIN_START* or *ctPLUGIN_STOP* (defined by the *ctPLUGIN_COMMAND* enum type)
- *inputBuffer* is flexible; currently, it expects only the plug-in name (loaded in the server by the *PLUGIN* keyword in *ctsrvr.cfg*)
- *outputBuffer*: Starting and stopping plug-ins doesn't have any output, so this is unused. Pass a pointer to a TEXT string.
- *pOutputBufferSize*: Starting and stopping plug-ins doesn't have any output, so this is unused. Pass a pointer to a VRLEN number set to the length of the string passed to *outputBuffer*.

Example 5 -Stopping the web services plug-in:

```
ctPlugin(ctPLUGIN_STOP, " cthttpd ",outBuff,&outBuffLen);
```

Error Code

A new c-tree error code has been added: **INV_CMD_ERR 1167** - Invalid plug-in command.

4. SQL

4.1 Auto timestamp support

The SQL syntax has been extended to support specifying the default value of a column of type date, time, timestamp to be one of the following:

```
autotimestamp_insert  
autotimestamp_update  
autotimestamp
```

When the default values is set in such way, the field content is always generated by the engine at row insert time (autotimestamp_insert or

autotimestamp) or at row update time (autotimestamp_update or autotimestamp). The automatically generated value causes the field to be not editable, so specifying it or its value, in an update or insert statement, results in an error (similar to an identity field).

Example:

```
create table mytest (creation_time timestamp default autotimestamp_insert, last_update timestamp default  
autotimestamp, name char(40));  
insert into mytest values ('Mark');  
select * from mytest;
```

4.2 SQL Performance Improvements

64K string performance in V13/V12 has been improved since its release in two areas:

- First, better optimized internal memory buffer usage for large strings.
- Second, selected scalar string functions were examined and modified for areas of improved string buffer usage.

Performance should now match or better V11 performance in most cases. Testing with specific queries and tables demonstrated up to a 20% performance improvement over the V12 prior build with 64K string handling increase.



4.3 Retrieve MRT Host Table Name via SQL Stored Procedure

A new stored procedure, **fc_get_hosttablename**, retrieves the host table name of a Multi-Record Type (MRT) table. As an input parameter, it receives an integer that is the table ID for which it will retrieve the MRT host name and UID. It returns a result set with exactly 1 row containing four columns:

1. The table UID or the host UID in case of ,ulti-record type (MRT) table
2. The c-treeDB table name
3. The filesystem path where the table is located
4. The filesystem name of the table

4.4 Support for Multiple DISTINCT Keywords in a Single Query

Prior to this adjustments, the SQL engine failed when more that one DISTINCT keyword was placed in a single query.

```
Error -20042 "Distinct specified more than once in query" (SQL_ERR_MULTIDIST).
```

Our fix removes the specific checks in the grammar and implement the support by simply ensuring internals are properly initialized for the count scalar function when distinct is in use.

4.5 SYSLOG SQL_STATEMENT - Exclude users from logging by name

We added a new configuration keyword **SYSLOG_EXCLUDE_SQL_USER** <name>.

When SQL query logging is enabled by **SYSLOG SQL_STATEMENTS**, by default all SQL statements are written to the log. **SYSLOG_EXCLUDE_SQL_USER** allows specifying a user name to exclude from this logging. Multiple users may be excluded by specifying the keyword multiple times. No validation is made that the name specified matches an existing user name.

4.6 Overcome 128K record size limit

When the table(s) in the query have multiple fields with large sizes, and temporary tables/indexes need to be created to execute the query, the size of the record for these temporary objects may been too large to handle it and this results in error. Previously, we had an internal limitation of



128Kb on the record size and variable length string needed to be considered to their maximum size. We have adjusted the internal code to alleviate this limitation.

4.7 Entity Framework Table Addition Speed Optimized

Previously, adding a table to an existing c-tree based entity data model could have taken up to 4 hours. This was due to an internal recursive call that caused the stack to grow increasingly large, and has since been fixed by modifying the code to avoid recursive calls and frequent memory allocations.

AFFECTED COMPONENTS: Entity framework 6 driver

4.8 SQL Import Utility (ctsqlimp) - Option to keep existing permissions and synonym

We identified that the existing SQL import option keeps the existing permission and synonyms but this option is not exposed in 'ctsqlimp' utility. Therefore we now have a new 'ctsqlimp' utility switch "-y".

Description:

ctsqlimp imports an existing ISAM c-tree data file into SQL database by updating c-treeACE SQL Server's internal dictionary.

Usage:

```
ctsqlimp <filename> [-d database] [-s srvname] [-u userid] [-a password]
[-n symbolic] [-o userid] [-i] [-r] [-k] [-x] [-p|-P] [-z] [-g] [-b]
[-w script] [-m idxname] [-f s|z|sz] [-l size] [-h]
<filename>      data file name/path (relative paths to c-treeACE Server directory)
[-d database]   database name (default: ctreeSQL)
[-s srvname]    c-treeACE SQL Server name (default: FAIRCOMS)
[-u userid]     userid for logging to c-treeACE SQL Server
[-a password]   user password for authentication
[-n symbolic]   set SQL table name to 'symbolic'
[-o userid]     set ownership of table to 'userid'
[-i]            non-interactive mode: ignore errors and continue
[-r]            remove table from dictionaries (don't delete files)
[-k]            skip fields not complying with conventional identifiers rules
[-c]            allow table names not complying with conventional identifiers rules
```



[-x]	skip indices
[-p]	promote unsigned types to greater signed type
[-P]	promote unsigned types to greater signed type and set check for fitting value
[-w script]	build script file with CREATE statements. (don't import table)
[-m idxname]	set 'idxname' as primary key
[-z]	allow indices with missing string terminator in key segments
[-f s z sz]	force string padding to (s)paces (z)eroes or (sz)spaces zero terminated
[-l size]	specify LONGVAR* field size threshold
[-g]	ignore existing index name in IFIL resource
[-b]	grant public access permissions to imported table
[-B]	grant readonly public access permissions to imported table
[-j]	non-interactive relink of existing table
[-q prefix]	prefix SQL table name with 'prefix'
[-y]	keep existing permissions and synonym
[-h]	display usage help

4.9 Sequence Number Entities now Unique Between Databases

FairCom DB supports sequence numbers that can be used for application specific numbering purposes. Multiple sequence pools can be created by different users.

There was a longstanding limitation that on a server with more than one database it was not possible to create in two different database two independent sequences with the same name by the same user. SQL requires a sequence name to be unique both per database as well as per user name. However, at a core ISAM API layer it was unique server-wide, breaking SQL conventions.

FairCom DB SQL now supports per database named sequence number support.

Notes:

This enhanced sequence number support requires a SQL system table update. This will be automatically applied to existing databases on new server startup.

Also, the underlying ISAM API layer sequence name and the SQL name are unrelated to maintain expected backward compatibility with existing sequence number usage.



4.10 Create ISAM Compatible CT_MONEY Column Type from FairCom DB SQL

FairCom DB SQL is generally compatible with nearly all FairCom DB application data, including data created with FairCom DB ISAM API layers. One data type that is not fully compatible is CT_MONEY. The ISAM type is a signed 32-bit integer value having a maximum size of 10 digits (and not even all 10 digits are arithmetically used.) SQL MONEY is either a CT_NUMERIC (with precision of 2) or CT_CURRENCY (when precision is 4). Both are 32 digit values capable of holding much more larger data values.

It was requested, as a transitional measure, to allow SQL to create fully ISAM compatible tables with CT_MONEY types. A new storage attribute 'ctmoney' has been added to enable a traditional CT_MONEY ISAM data type. When this attribute is specified all money fields in the table gets mapped to CT_MONEY. If a precision of 4 is specified this result in an invalid argument error.

Example

```
CREATE TABLE account (name VARCHAR(128), age SMALLINT, balance MONEY(2) ) storage_attributes 'ctmoney' ;
```

It is important for the application maintainer to ensure SQL values exceeding the 32-bit signed integer value are not used in this field type or unexpected results will be observed. That is, there is no prevention in the SQL handling to check what the underlying storage type is used and expects 32 digit values to be allowed. To prevent this, an SQL CHECK constraint should be applied to those affected ISAM CT_MONEY columns.

4.11 SQL - support to map money type to c-tree's CT_MONEY

From SQL, a customer would like to be able to create tables having the SQL money type mapped to CT_MONEY instead of our default. Our default it to use either CT_CURRENCY or CT_NUMBER based on the precision and scale in order to guarantee the most efficient type depending on the SQL definition. Mapping always to CT_MONEY implies that it is not possible to use a precision of 4 and despite the precision definition in SQL the maximum value that can be stored is what's fits in a LONG.

SOLUTION: Implemented a new storage attribute 'ctmoney'. When this attribute is specified all money fields in the table gets mapped to CT_MONEY. If a precision of 4 is specified this result in an invalid argument error.

4.12 ROWID - CTDB/SQL - Expose 'rowid' as a field in SQL

When using FairCom's CTDB API, some customer take advantage of the hidden 'rowid' (DODA name \$ROWID\$) field and it's unique index. This unique index can be used as a primary unique id key. The 'rowid' is hidden in CTDB and the user can retrieve it with ctdbGetRowid(). By default, SQL and CTDB create the hidden 'rowid' field and a unique index over this field. The 'rowid' field



is hidden to SQL and therefore SQL usage is handicapped because joins can't use this as the unique id.

In this modification, we added the ability to optionally make the 'rowid' appear as a regular field to SQL. We did not change the CTDB's \$ROWID\$ behavior.

In order to make the 'rowid' visible to SQL, you must use the SQL import ('ctsqlimp') Utility. This utility will perform the necessary maintenance on the SQL dictionaries in order to expose the 'rowid' field.

If the file was created with SQL or with a CTDB CTSESSION_SQL session, this file is already registered (with the 'rowid' field hidden) in the SQL dictionary. When executed, the 'ctsqlimp' utility will prompt you to override the existing metadata in the dictionary with your updated options.

There are two new switches for the 'ctsqlimp' utility:

- [--rowid_fld fldname] - expose the ROWID value as field with name (table must have \$ROWID\$ field)
- [--rowid_idx idxname] - expose index on the ROWID value with name (--rowid_fld option must be specified)

Existing switch:

- [-m idxname] - set 'idxname' as primary key

It is also possible to make the rowid index a primary key by specifying the same value of the -m and --rowid_idx options (in which case that's the primary key constraint name). Worth noting that if the table already has a primary key the import logic returns an error.

Example:

```
ctsqlimp myfilename.dat --rowid_fld myrowid --rowid_idx myrowidIndexName -m myrowidIndexName
```

Details:

We enhanced the SQL engine to handle field mapping into the \$ROWID\$ CTDB's hidden field. We can now expose it in SQL as a regular field which name is defined at import time.

This field is "read only" in SQL, attempt to edit it during inserts or updates fails with error (like an identity field). The field can be identified in 'syscolumns' since its 'fldid' is negative (-3).

During import it is possible (and actually suggested) to import also the index on the \$ROWID\$ column. In such case the index id in 'sysindices' matches the table id since there is no good way to give a unique index id that it is guaranteed to not be used.

CTDB code has not been touched. In CTDB the rowid field is still not exposed as a regular field.

CTDB's wrapper for SQL has been enhanced to access and use the \$ROWID\$ exposed column in record retrieval, index searches, pushdown logic.



4.13 SQL - Correct Numeric String Conversion

Converting this SQL string to a numeric value resulted in a loss of 2 significant digits.

```
'-0.9999999999999999999999999999999999999999999'
```

The number of 9s after the decimal point was 30 instead of 32. The conversion logic from string to numeric in both c-treeDB and SQL counted leading '0' as significant digits, which caused the loss of precision. Proper string parsing prevents this situation.

4.14 SQL Stored Procedures - Informational Methods Added to DhSqlException

Previously, the SQL stored procedure `DhSQLException` class did not have methods to directly access error codes and messages. `DhSQLException.getSqlState()` and `DhSQLException.getMessageText()`, are now formally available allowing users to access diagnostic exception information.

4.15 SQL Windowing Functions for Analytics

A window function operates on a group of rows from a result set. The rows are grouped together based on the `OVER` clause. Unlike an aggregate function, window functions do not group rows into a single output, the rows remain separate rows however they include an window function value that is determined by values from all the rows in the group.

The PARTITION BY clause splits the result set into partitions over which the window function operates. When moving to a new partition the window function resets, and values from other partitions are not considered when calculating window function values. The ORDER BY clause orders the rows within a particular partition. The ordering also resets when moving to a new partition. Ranking functions are nondeterministic as they are computed for each generated result set.

Only a single ranking function may be used in a statement.

Available Functions

- AVG (https://docs.faircom.com/doc/sqlref/avg_window_function.htm) - returns the average of the values in a group.
- COUNT (https://docs.faircom.com/doc/sqlref/count_window_function.htm) - returns the number of values in a group.
- CUME_DIST (https://docs.faircom.com/doc/sqlref/cume_dist_window_function.htm) - calculates the cumulative distribution of a value within a group of values.
- DENSE_RANK (https://docs.faircom.com/doc/sqlref/dense_rank_window_function.htm) - returns the rank of a row within a partition of a result set. The rank is one plus the number of ranks that come before the row in question with no gaps in the ranking values.
- FIRST_VALUE (https://docs.faircom.com/doc/sqlref/first_value_window_function.htm) - returns the first value of an ordered set of values.



- **LAST_VALUE** (https://docs.faircom.com/doc/sqlref/last_value_window_function.htm) - returns the last value of an ordered set of values.
- **MAX** (https://docs.faircom.com/doc/sqlref/max_window_function.htm) - returns the maximum of the values in a group.
- **MIN** (https://docs.faircom.com/doc/sqlref/min_windowing_function.htm) - returns the minimum of the values in a group.
- **NTILE** (https://docs.faircom.com/doc/sqlref/ntile_window_function.htm) - distributes the rows in an ordered partition into a specified number of numbered groups. The group numbering starts at one.
- **PERCENT_RANK** (https://docs.faircom.com/doc/sqlref/percent_rank_windowing_function.htm) - calculates the relative rank of a value within a result set.
- **RANK** (https://docs.faircom.com/doc/sqlref/rank_window_function.htm) - returns the rank of a row within a partition of a result set. The rank is one plus the number of ranks that come before the row in question.
- **ROW_NUMBER** (https://docs.faircom.com/doc/sqlref/row_number_window_function.htm) - returns the sequential number of a row within a partition of a result set.
- **SUM** (https://docs.faircom.com/doc/sqlref/sum_window_function.htm) - returns the sum of the values in a group

4.16 Tuple Expression Enhancements

FairCom DB V12 introduced row value constructors for SQL INSERT statements. In this release their use has been expanded.

- Multiple tuple expressions with differing number of fields are now supported in the same query.
- Evaluating tuple expressions when there is not an appropriate index available is now supported.
- Enhanced support for using indexes to evaluate tuple expressions. When there is no index with an exact match for the expression, the optimizer looks for partial matches, prioritizing the index with the longest prefix of the tuple expression. The index is sent a truncated tuple set and the full expression is enforced in the engine. The index prefix chosen may be as short as a single field.
- The optimizer has been enhanced to remove redundant tuple expressions. Expressions that are subsets of other expressions will be removed as well.

4.17 .NET Stored Procedures - Implement access to LVAR* fields

ctsqlapi() now implements 'long var' fields support for server side stored procedures.



- .Net stored procedures make calls to 'ctsqlapi()' inside the server in order to interact with the SQL engine. In 'ctsqlapi()', we had not implemented the server side support for the calls to get/put LVAR data. Therefore (until now) it was not possible to read/write LVAR* data from the stored procedure. Now, we have implemented the support and adjusted the code to satisfy this need.
- We have completed the implementation in the core server layer and in the C# layer to provide easy access to the LVAR* fields with stored procedures
- The access to a LVAR* field in a result set is possible through the `SqlSpCommand.CurrentRow[].Value` is always returned as `Byte[]`.
- The insertion is always made through parameterized prepared statements and assigned using `SqlSpCommand.Parameter[].Value`. The value must be passed as `Byte[]`.
- We manage all LVAR* (LVARCHAR included) fields through byte arrays so the end user has enough flexibility to encode/decode the contents in their preferred encoding without the need to force a default encoding in the SQL Stored Procedure `SqlSP*` functions.

4.18 .NET Stored Procedure - More information

Support for LVAR* fields has been added to the .NET Stored Procedure layer, and the existing `SqlSpCommand` has been updated to make `LVARCHAR` and `LVARBINARY` fields accessible through `SqlSpCommand.CurrentRow[].Value`.

Note: The value is always returned as `Byte[]`.

`SqlSpCommand` has also been updated to support the insertion and update of `LVARCHAR` and `LVARBINARY` fields as well. The insertion is made through parameterized queries, and the value must always be passed as `Byte[]` through `SqlSpCommand.Parameter.CurrentRow[].Value`. `LVARCHAR` and `LVARBINARY` are handled as parameters, so only prepared statements can be used to assign values. Due to this handling, `ExecuteNonQuery()` can't be used to assign `LVARCHAR` and `LVARBINARY`. `Execute()` is the only supported method for assigning these types of fields.

Below are examples of this new behavior.

Read Example:

```
SqlSpCommand cmdSelect = NewSqlSpCommand();
String strCommandSelect = "SELECT lvarbinfield, lvarcharfield, pk FROM lvartbl
where pk = 2";
cmdSelect.Prepare(strCommandSelect);
cmdSelect.Execute();
while (cmdSelect.MoveNext())
{
    //Reading LVARBINARY field as object. It may be casted to Array
    object xml = cmdSelect.CurrentRow[0].Value;
    //Reading LVARCHAR field and casting as Byte[]
    byte[] vs = (byte[])cmdSelect.CurrentRow[1].Value;
```



```
    if (vs != null)
    {
        //If the byte[] is not null convert the byte array using
        //the preferred encoding
        string text = Encoding.Default.GetString(vs);
    }
}
```

Write Example :

```
SqlSpCommand cmdInsert = NewSqlSpCommand();
String strCommandInsert = "INSERT INTO lvartbl(lvarcharfield,lvarbinfield)
VALUES (?,?)";
cmdInsert.Prepare(strCommandInsert);
cmdInsert.Parameter[0].Value = Encoding.Default.GetBytes("This will be a
LVARCHAR");
cmdInsert.Parameter[1].Value = Encoding.Default.GetBytes("We should insert
binary data here");
int rv = cmdInsert.Execute();
if (rv != 0)
    throw new Exception();
```

4.19 DSQL - Direct SQL

DSQL - New function `ctsqliGetCommandFromCursor`

It is useful to be able from a cursor handle to be able to find the command handle it was based on so that I do not have to carry it as well. Therefore we implemented a new DSQL function to achieve that.

```
/******\
* Function:      ctsqliGetCommandFromCursor
*
*               return the command associated with a cursor
* Parameters:    hCursor [IN]*   cursor handle
* Return:        the command associated with the cursor
*****/
pCTSQLCMD ctsqliDECL ctsqliGetCommandFromCursor(pCTSQLCURSOR hCursor)
```

DSQL - New functions to scroll the resultset using a scrollable cursor

DSQL has the capability to set the cursor to scrollable, but the only benefit was to have the count of the rows in the result set. Ideally scrollable cursors are used to be able to move back and forth in the result set but DSQL did not expose this functionality.

We now have added the following functions:

```
CTSQLRET ctsqliDECL ctsqliPrev(pCTSQLCURSOR hCursor);
CTSQLRET ctsqliDECL ctsqliFirst(pCTSQLCURSOR hCursor);
CTSQLRET ctsqliDECL ctsqliLast(pCTSQLCURSOR hCursor);
CTSQLRET ctsqliDECL ctsqliJumpTo(pCTSQLCURSOR hCursor, INTEGER nth);
CTSQLRET ctsqliDECL ctsqliGetRow(pCTSQLCURSOR hCursor, INTEGER nth);
```

`ctsqliJumpTo`: jumps and returns the nth rows from the current position. it jump backward when using negative nth or forward for positive nth). `ctsqliJumpTo(hCursor, 1)` is like calling `ctsqliNext`. `ctsqliJumpTo(hCursor, -1)` is like calling `ctsqliPrev`.

`ctsqliGetRow`: retrieves (and position to) the nth row of the resultset being 1 the first row. nth need to be positive.

If the movement goes beyond the result set first or last row, the operation returns `CTSQLRET_NOTFOUND` (100) and the current position is set the either before (if moving backward) or after the last row (if moving forward).

`ctsqliGetConnectionFromCursor()` was also added enabling users to find the connection handle the cursor handle was based on.



r288975 - DSQL - New function `ctsqlGetConnectionFromCursor`

We find it very useful to, from a cursor handle to be able to find the connection handle it was based on. Therefore we implemented a new DSQL function to achieve that named `'ctsqlGetConnectionFromCursor()'`

```

/*****
* Function:      ctsqlGetConnectionFromCursor
*
*                return the connection associated with a cursor
* Parameters:    hCursor [IN] - cursor handle
* Return:        the connection associated with the cursor
*****/
pCTSQLCONN ctsqlDECL ctsqlfn(GetConnectionFromCursor)(pCTSQLCURSOR hCursor)

```

DSQL - New function `ctsqlGetParameterDirection`

Stored procedure parameters can be input parameters, output parameters or input/output parameters. When calling a stored procedure while input parameters can be specified as literals, output and input/output stored procedure parameters need to specify as SQL statement parameters otherwise the engine returns error -20127

Bad parameter specification for the statement. In DSQL we had no way to distinguish the direction of statement parameters used in a stored procedure call which is an important information to have in developing a generic SQL interface (like `runSQLStatement` action in JSON DB) therefore we added the `GetParameterDirection()` function:

```

/*****
* Function:      GetParameterDirection
*
*                Retrieve the parameter direction (IN/INOUT/OUT).
* Parameters:    hCmd [IN] - command handle
*                index [IN] - parameter number.
*
*                Must be a number greater or equal to zero
*                but less than parameter count.
* Return:        return the direction of parameter. Returns 0 on error.
*****/
SQL_PARAMDIR ctsqlGetParameterDirection)(pCTSQLCMD hCmd, INTEGER index)

```

and

```

typedef enum { /* see sql_paramtype_t in sql_tree.hxx*/
SQL_INV_PARAM = -1,
SQL_IN_PARAM = 1,
SQL_IN_OUT_PARAM = 2,
SQL_OUT_PARAM = 4,

```



```
SQL_SFN_SUBS_PARAM = 5
} SQL_PARAMDIR;
```

DSQL - New functions `ctsqliGetConnectionFromCommand()` and `ctsqliGetParameterAddress()`

While adding support for SQL in the JSON DB interface, we added two new functions:

































```
/******\
* Function:      ctsqliGetConnectionFromCommand
*                return the connection associated with a command
* Parameters:    hCmd [IN] - command handle
* Return:        the connection associated with the cursor
*****/
pCTSQLCONN ctsqliDECL ctsqliGetConnectionFromCommand(pCTSQLCMD hCmd)
```

```
*****\
* Function:      ctsqliGetParameterAddress
*                Retrieve the address of a parameter
* Parameters:    hCmd [IN] - command handle
*                index [IN] - parameter number.
*                Must be a number greater or equal to zero
*                but less than parameter count.
* Return:        return the address of a cursor variable
*****/
TINYINT* ctsqliDECL ctsqliGetParameterAddress(pCTSQLCMD hCmd, INTEGER index)
```

5. Drivers

5.1 Drivers Overview

We would like to remind you about our extensive set of Drivers/APIs supported in our V13 release. These are listed below.

 c.isam	 java.sql.storedprocs
 c.lowlevel	 node-red.json.db
 c.mqtt	 nodejs.json.db
 c.nav	 nodejs.mqtt
 c.sql.direct	 php.sql
 cpp.nav	 php.sql.pdo
 csharp.nav	 python.json.db
 csharp.sql.ado.net	 python.mqtt
 csharp.sql.storedprocs	 python.nav
 ctrees.c.replication	 python.sql
 java.jpna.nav	 python.sql.sqlalchemy
 java.json.db	 sql.cli
 java.json.nav	 sql.jdbc
 java.mqtt	 sql.odbc
 java.nav	 thingworx.always-on
 java.sql.hibernate	 vb.nav



c.isam - C Language ISAM

The FairCom Database ISAM C API provides an advanced set of NoSQL functions for manipulating data and index files using FairCom's ISAM (Indexed Sequential Access Method). Each function performs multiple file and index access operations resulting in improved performance over the FairCom Low-Level API.

FairCom Database ISAM functions assume a hierarchical relationship among the data and index files, in the sense that each data file may have many indexes, while each index relates to only one data file. For example, an accounts payable system may define a vendor data file indexed by vendor name and number, and an invoice data file indexed by vendor number and invoice number. In this case, each data file has two indexes. Although each data file is indexed by vendor number, there is a separate vendor number index for each data file.

This API is included for developers with specialized needs and can provide extensive additional control over particular data handling needs, for example, real-time data capture and processing. If you are just starting with FairCom Database, we recommend the easy-to-use c-treeDB APIs as they reduce the amount of code you need to write and provide the same great performance for which c-tree is famous.

See package folder: `.\drivers\c.isam`

c.lowlevel - C Language Low-Level

The FairCom Low-Level C API Interface Technology is a detailed API intended for very specific application requirements. This is the lowest level API FairCom offers for the FairCom DB database engine. This API allows the programmer to individually control the data and index files. If an application needs a fast, flexible indexing engine, this is an excellent option.

This is the core layer providing the functionality of FairCom's high-performance advanced APIs, which are much easier to use. As such, the FairCom Low-Level API requires an advanced level of understanding for optimal implementation.

See package folder: `.\drivers\c.lowlevel`

c.mqtt- C Language MQTT Client

The FairCom MQTT Broker engine is included in the FairCom Edge and FairCom MQ products. Programs use an MQTT client library to publish and subscribe messages to topics on the MQTT broker. The MQTT broker guarantees receipt of messages from publishers and delivery of messages to subscribers according to requested quality of service (QoS) guarantees.

We offer a quick start guide which includes two tutorials. Both are command-line programs that you can use to publish messages and monitor published messages. Thus, the tutorials perform double duty. They show you how to use MQTT and they are command-line utilities that you can use to learn and troubleshoot MQTT.

See package folder: `.\drivers\c.mqtt`



c.nav - C Language Record Navigation (CTDB)

The c-tree Database API for C is a record-oriented database API. Its short name is c-treeDB or simply CTDB. It is in the "NAV" family of APIs because it makes it easy for application developers to navigate data — no matter how complex.

c-treeDB provides precise control over every aspect of data processing. It leverages the familiar concepts of databases, tables, records and fields. For example, it can connect to a database, open a table, find a record by key, and walk records in key order.

The c-treeDB API makes it easy to navigate and process data very quickly across multiple databases no matter where they are located. Its unique ability is to simultaneously open cursors into multiple databases and remote control them. Applications use cursors to control every detail of record lookup and traversal. There are multiple types of cursors: index, table, filtered, range, metadata, batch query, batch insert, batch update, and batch delete. Each is optimized to make different use cases easy and fast. For example, your application can retrieve a record from one database and use its data to quickly position a cursor in another database to batch retrieve additional records.

See package folder: `.\drivers\c.nav`

c.sql.direct - C Language Direct SQL

The FairCom DB Direct SQL interface (DSQL) is an inline SQL application programming interface (API) designed for C/C++ developers who wish to embed SQL statements directly into their programs.

The FairCom DB DSQL API is designed as a foundation for building your own SQL APIs. It simplifies your code. In fact, many of the SQL GUI tools authored by FairCom utilize this API.

FairCom DB DSQL gives application developers control and ease of deployment over other embedded APIs, such as ODBC, and DSQL requires no time-consuming pre-processing steps or the presence of an ODBC manager.

Consider using FairCom DB DSQL for your next embedded application solution.

See package folder: `.\drivers\c.sql.direct`

cpp.nav - C++ Language Record Navigation (CTPP)

The c-tree Database API for C++ is a record-oriented database API. Its short name is c-treeDB for C++ or simply CTPP. It is in the "NAV" family of APIs because it makes it easy for application developers to navigate data — no matter how complex.

c-treeDB for C++ is an object-oriented API that provides precise control over every aspect of data processing. It leverages the familiar concepts of databases, tables, records and fields. For example, it can connect to a database, open a table, find a record by key, and walk records in key order.

This API makes it easy to navigate and process data very quickly across multiple databases no matter where they are located. Its unique ability is to simultaneously open cursors into multiple databases and remote control them. Applications use cursors to control every detail of record lookup and traversal. There are multiple types of cursors: index, table, filtered, range, metadata,



batch query, batch insert, batch update, and batch delete. Each is optimized to make different use cases easy and fast. For example, your application can retrieve a record from one database and use its data to quickly position a cursor in another database to batch retrieve additional records.

See package folder: `.\drivers\cpp.nav`

csharp.nav - C# Language Record Navigation

The c-tree Database API for C# (i.e. "C Sharp") is a record-oriented database API for Microsoft .NET. It is in the "NAV" family of APIs because it makes it easy for application developers to navigate data — no matter how complex.

This C# API is an object-oriented API that provides precise control over every aspect of data processing. It leverages the familiar concepts of databases, tables, records and fields. For example, it can connect to a database, open a table, find a record by key, and walk records in key order.

This API makes it easy to navigate and process data very quickly across multiple databases no matter where they are located. Its unique ability is to simultaneously open cursors into multiple databases and remote control them. Applications use cursors to control every detail of record lookup and traversal. There are multiple types of cursors: index, table, filtered, range, metadata, batch query, batch insert, batch update, and batch delete. Each is optimized to make different use cases easy and fast. For example, your application can retrieve a record from one database and use its data to quickly position a cursor in another database to batch retrieve additional records.

See package folder: `.\drivers\csharp.nav`

csharp.sql.ado.net - C# Language for ADO.NET

ADO.NET provides relational data access to systems such as FairCom's DB SQL. An ADO.NET Data Provider is a bridge used to connect ADO.NET applications to a database, execute commands, and retrieve results. The FairCom DB ADO.NET Data Provider gives you access to your FairCom DB SQL data from a .NET application. The .NET Data Provider is lightweight, creating a thin layer between the data source and your code, thus increasing performance without sacrificing functionality. A .NET Data Provider consists of a set of classes implementing interfaces specified in Microsoft's specification for .NET Data Providers. This

See package folder: `.\drivers\csharp.sql.ado.net`

csharp.sql.storedprocs - C# Language Stored Procedures

SQL stored procedures and triggers are an easy and powerful way to move advanced logic into your server core, protecting investments in performance-sensitive processing while maintaining a centrally managed approach for long-term maintenance. Stored procedures fully encapsulate business logic for business rule enforcement. Triggers maintain database integrity directly at the database server level.



You can program your procedure in C# or any other Microsoft Windows .NET language. Seamlessly remain in your .NET environment while developing advanced server-side processing logic. With Visual Studio integration, it is easy to create, debug, and deploy your C# stored procedure code remotely. Your assemblies are deployed alongside FairCom DB SQL with a click of a mouse.

See package folder: `.\drivers\csharp.sql.storedprocs`

java.jpa.nav - Java Language Persistence API (JPA)

The FairCom Java Persistence API, commonly referred to as the JPA, provides Java developers a unique record-oriented, non-SQL, Java framework to persist data using the FairCom Database Engine.

Most JPA drivers are built on SQL APIs, but this driver is different in that it avoids the SQL API layers and uses the c-treeDB API record oriented API for increased performance.

FairCom also provides a JPA driver that uses the SQL API; see the FairCom DB Java Hibernate (`java.sql.hibernate`).

Different open source projects exist that implement the JPA specification. Popular projects include Hibernate and EclipseLink. This driver was developed using EclipseLink.

See package folder: `.\drivers\java.jpa.nav`

java.json.db - Java Language over JSON DB

FairCom's Java driver for the JSON DB API is a lightweight abstraction over FairCom's JSON Action APIs. It makes it easy to post JSON directly to a FairCom server and process the JSON results. The `DbApi` and `AdminApi` classes provide a method for each JSON Action to get you up and running quickly. These methods automatically manage authentication, sessions, cursors, and transactions.

FairCom's Java driver uses the Jackson JSON library, which makes it easy for you to serialize your POJOs into and out of JSON. It also makes it fun to programmatically modify and generate JSON commands to send to the server. Modifying the library to use other JSON and HTTP libraries is straightforward.

Spend less time fighting code and more time delivering features by using FairCom DB's new Java driver built on top of FairCom's new JSON DB API, which uses JSON commands to manage the database. You write all queries in JSON, and the server returns all data as JSON.

Reduce development time with copy-paste development. Use our built-in API Explorer to quickly learn, create, modify, and test JSON actions. API Explorer provides over a hundred JSON examples illustrating everything you need to know. Start with a JSON example, tweak it, test it, copy it out of API Explorer, and embed it in your Java program.

See package folder: `.\drivers\java.json.db`



java.mqtt - Java Language MQTT Client

The FairCom MQTT Broker engine is included in the FairCom Edge and FairCom MQ products. Programs use an MQTT client library to publish and subscribe messages to topics on the MQTT broker. The MQTT broker guarantees receipt of messages from publishers and delivery of messages to subscribers according to requested quality of service (QoS) guarantees.

We offer a quick start guide which includes two tutorials. Both tutorials are command-line programs that you can use to publish messages and monitor published messages. Thus, the tutorials perform double duty. They show you how to use MQTT and they are command-line utilities that you can use to learn and troubleshoot MQTT.

See package folder: .\drivers\java.mqtt

java.nav - Java Language Record Navigation (JTDB)

The FairCom DB Java Language DataBase Interface Technology, commonly referred to as the c-treeDB Java API (or "JTDB"), provides Java developers a unique record-oriented, non-SQL, navigational ("NAV") Java framework to manage data with the FairCom Database Engine. This interface offers alternatives to the Java developer requiring direct access to data records for performance advantages while still allowing full JDBC SQL access to the same data.

The c-treeDB Java API makes the c-treeDB navigational, "NAV", database architecture available to Java programmers.

The c-treeDB Java API provides libraries that allow Java programs to access the c-tree database core. It provides a simple interface into c-tree that includes the advanced functionality that distinguishes c-tree from other database solutions. c-treeDB Java offers a method of database creation and maintenance that is easier to use than the FairCom DB ISAM and FairCom Low-Level APIs.

See package folder: .\drivers\java.nav

java.sql.hibernate - Java Language Hibernate Persistence

The Hibernate driver for FairCom DB allows Java developers to use Hibernate to access the FairCom DB SQL Relational Database Engine.

The FairCom DB SQL dialect of the Hibernate Java Persistence API provides Java developers a way to persist data using the FairCom Database Engine by simply replacing the Hibernate layer with the FairCom Hibernate dialect.

The FairCom DB dialect for Hibernate allows FairCom DB to be used in Java applications by simply using the same techniques documented on the Hibernate website. If you have a Java application that is currently using Hibernate, simply follow the instructions in this document to set up the FairCom DB Hibernate dialect and you are ready to begin using the FairCom DB 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.

See package folder: .\drivers\java.sql.hibernate



java.sql.storedprocs - Java Language SQL Stored Procedures

SQL stored procedures and triggers are an easy and powerful way to move advanced logic into your server core, protecting investments in performance-sensitive processing while maintaining a centrally managed approach for long-term maintenance. Stored procedures fully encapsulate business logic for business rule enforcement. Triggers maintain database integrity directly at the database server level.

FairCom DB SQL has long supported stored procedures, triggers, and user defined functions with Java for cross-platform development ease. In this release, stored procedures are available for your Microsoft Windows .NET development environment as well.

You can program your procedure in Java or any .NET supported language.

FairCom stored procedure support also includes command-line utilities for deploying stored procedures, user-defined functions, and triggers. These utilities are described in the FairCom DB SQL Stored Procedures and Triggers manual.

See package folder: `.\drivers\java.sql.storedprocs`

nodejs.json.db - JSON DB Client for Node.js

FairCom has a JSON DB API for performing database functions. With this release of FairCom DB, Node.js clients can now use JSON in JavaScript code to interact with FairCom DB servers.

The JSON DB API is an asynchronous, protocol-agnostic, data definition, manipulation, and query language that uses JSON to convey directions to and receive data back from, the server. The current implementation of the JSON DB API uses the WebSocket TCP/IP protocol standard.

The JSON DB API can be used directly from any language that supports websockets and JSON. The FairCom Java client driver library for the JSON DB API makes using the JSON DB API easy. The client driver uses a single dependency on the org.java-websocket Java library.

See package folder: `.\drivers\nodejs.json.db`

nodejs.mqtt - MQTT Client for Node.js

We include MQTT Client developer tutorials for Node.js. FairCom MQ is a powerful MQTT Broker. Programs use the MQTT client library to publish and subscribe to messages on the MQTT Broker.

You can subscribe to a topic and receives all messages published to that topic by any client.

We offer two tutorials:

- The first is a program that publishes MQTT messages to the broker.
- The second is a program that subscribes to messages.

Both are command line programs that you can use to publish messages and monitor published messages. Thus, the tutorials perform double duty. They show you how to use MQTT and they create command line utilities that you can use to learn and troubleshoot MQTT.

See package folder: `.\drivers\nodejs.mqtt`



php.sql - SQL for PHP

FairCom provides several ways for accessing the FairCom Database Engine from PHP:

- A native PHP API for interfacing with PHP 5.x.x. It does not support later versions of PHP.
- FairCom DB ODBC for interfacing with any version of PHP.
- A PHP PDO (PHP Data Object) driver for supporting PHP 7 and later with the PDO extension. This is the preferred way to access PHP. See `. \drivers\php.sql.pdo`.

We offer PHP developer tutorials for the FairCom DB PHP 5 API. It also includes a section on accessing PHP using ODBC.

In a few simple steps and about 15 minutes, you'll be navigating your data.

See package folder: `. \drivers\php.sql`

php.sql.pdo - SQL for PHP PDO

PHP (Hypertext PreProcessor) is a widely-used scripting language that is especially suited for web development. FairCom's PHP PDO module allows you to access the powerful FairCom DB SQL database from your PHP version 7 (and later) projects. We offer tutorials for the PHP PDO API. In these simple steps and about 15 minutes, you'll be navigating your data.

See package folder: `. \drivers\php.sql.pdo`

python.json.db - Python Language over JSON DB

Manage your data using Python and the JSON DB APIs. The JSON DB API allows users to access rich database functionality using JSON messages over HTTP(S) without any local binary dependencies. The API is simple to use, and writing a driver from scratch is very easy.

We offer information that covers the Python driver for the JSON DB API and how to use it. The Python driver for the JSON DB API makes requests using an HTTP client library as its only direct dependency.

See package folder: `. \drivers\python.json.db`

python.mqtt - MQTT Client for Python

We offer MQTT Client developer tutorials for Python.

FairCom EdgeMQ is an MQTT Broker. Programs use an MQTT client library to publish and subscribe to messages on the MQTT Broker.

An MQTT client does the following:

- Connects to an MQTT broker.
- Publishes a message to a topic.
- Subscribes to a topic and receives all messages published to that topic by any client.

We offer two tutorials:



- The first is a program that publishes MQTT messages to the broker.
- The second is a program that subscribes to messages.

Both are command line programs that you can use to publish messages and monitor published messages. Thus, the tutorials perform double duty. They show you how to use MQTT and they create command line utilities that you can use to learn and troubleshoot MQTT.

See package folder: .\drivers\python.mqtt

python.nav - Python Language Record Navigation (CTDB)

c-treeDB for Python is an object-oriented API that provides precise control over every aspect of data processing. It leverages the familiar concepts of databases, tables, records and fields. For example, it can connect to a database, open a table, find a record by key, and walk records in key order.

The c-treeDB API makes it easy to navigate and process data very quickly across multiple databases no matter where they are located. Its unique ability is to simultaneously open cursors into multiple databases and remote control them. Applications use cursors to control every detail of record lookup and traversal. There are multiple types of cursors: index, table, filtered, range, metadata, batch query, batch insert, batch update, and batch delete. Each is optimized to make different use cases easy and fast. For example, your application can retrieve a record from one database and use its data to quickly position a cursor in another database to batch retrieve additional records.

See package folder: .\drivers\python.nav

python.sql - Python Language SQL

FairCom has implemented a pure Python module called "pyctree" that interfaces with FairCom DB SQL Server through the well-known ctypes module. This architecture has been chosen because it makes the module immediately available for multiple Python implementations, such as PyPy and Jython, and it does not require a C compiler to install the extension.

Pyctree consists of two parts both completely written in Python:

- pyctsqlapi.py - A wrapper library over the DSQL API
- pyctree.py - A DB-API 2.0 implementation conforming to Python PEP 249 standards.

See package folder: .\drivers\python.sql

python.sql.sqlalchemy - Python Langage for SQLAlchemy

SQLAlchemy is one of the most widely used object-relational mapping tools in the Python community. This Object Relational Mapper gives application developers the full power and flexibility of SQL. The FairCom DB SQL Python interface provides support for this technology.

If you have an existing application that uses SQLAlchemy, simply install the FairCom DB SQL dialect for SQLAlchemy to begin taking advantage of the FairCom DB database.

If you are planning a new SQLAlchemy application, the FairCom SQLAlchemy support allows you to take advantage of this powerful style of programming.



See package folder: `.\drivers\python.sqlalchemy`

sql.cli - SQL Language Interactive SQL Command-Line Interface (iSQL)

The FairCom DB Interactive SQL utility (iSQL) provides an industry-standard "command processing" interface to the FairCom DB Database Engine.

FairCom offers two separate implementations of Interactive SQL:

- A command-line version (isql.exe or isql). This utility is designed as a command-line tool and can be placed in job streams.
- A GUI tool version within the Windows program FairCom DB SQL Explorer.

Either implementation allows you to issue SQL statements directly from a command prompt for immediate displayed results. You can use Interactive SQL to test and prototype SQL statements to be embedded in programs; modify an existing database; perform ad-hoc queries; and run existing SQL scripts.

With few exceptions, you can issue any SQL statement in iSQL that can be embedded in a program, including CREATE, SELECT, and GRANT statements.

See package folder: `.\drivers\sql.cli`

sql.jdbc - Java Language Database Connectivity (JDBC)

The Java Database Connectivity (JDBC) API is the industry standard for database-independent connectivity between the Java programming language and the FairCom DB SQL Database Engine. The JDBC API provides a call-level API for FairCom DB SQL-based database access, which then allows you to use the Java programming language to exploit "Write Once, Run Anywhere" capabilities for applications that require access to enterprise data. An application written in the Java programming language alleviates the challenge of writing different applications to run on different platforms.

See package folder: `.\drivers\sql.jdbc`

sql.odbc - SQL Language Open Database Connectivity (ODBC)

ODBC (Open Database Connectivity from Microsoft) is a standard for client applications accessing data from a variety of data sources through a single interface. Users of applications supporting ODBC merely select a new database from a point-and-click menu to connect transparently to that data source.

ODBC is the standard mechanism for client applications to access data from a variety of different sources through a single interface. Users of applications supporting ODBC merely select a new database from a point-and-click menu to connect transparently to that data source.

To become accessible from ODBC client applications, database environments must provide software, called a driver, on the client system where the application resides. The driver translates the standard ODBC function calls into calls the data source can process and returns data to the



application. Each data source provides a driver on the client system for applications to use to access data from that source.

The FairCom ODBC Driver extends this plug-and-play interoperability to the FairCom Database Engine. It allows any Microsoft Windows tool or application that supports ODBC to easily use c-tree as a data source. With it, applications based on tools such as Visual Basic can include FairCom DB SQL as a data source.

See package folder: .\drivers\sql.odbc

thingworx.always-on - ThingWorx Connector

The ThingWorx AlwaysOn Starting Kit for FairCom Edge connects ThingWorx to the FairCom Edge database and hub. The FairCom Edge Connector for ThingWorx uses the ThingWorx AlwaysOn protocol to automatically deliver data from FairCom Edge to ThingWorx. It allows you to gather data into FairCom Edge servers running on the edge next to devices, equipment, and users.

FairCom Edge makes it easy to get data from devices into ThingWorx. It gathers data from edge devices using a variety of protocols, including MQTT, OPC UA, PLCs, REST, etc. It stores this data in its internal, high-speed database, and automatically forwards the data to ThingWorx. It supports all types of data including images, large binary objects, JSON, numbers, strings, etc.

FairCom Edge automatically maps and transforms data so that ThingWorx can consume it. It easily converts complex JSON data into tabular data for easy integration into any that ThingWorx process.

FairCom Edge can store data at high velocity and forward it to ThingWorx at a steady rate.

FairCom Edge provides plug-ins that integrate with many types of devices and equipment. If you need capabilities that are not currently implemented, you can build additional connectors or use FairCom professional services to build them.

Lastly, FairCom Edge connectors can use machine learning, custom logic, and custom code to provide real time feedback to devices on the edge — while simultaneously delivering data to ThingWorx.

See Edge package folder: .\drivers\thingworx.always-on

vb.nav - Visual Basic Language Record Navigation

The c-tree Database API for Visual Basic is a record-oriented database API for Microsoft .NET. Its short name is c-treeDB .NET for VB. It is in the "NAV" family of APIs because it makes it easy for application developers to navigate data — no matter how complex.

c-treeDB for VB is an object-oriented API that provides precise control over every aspect of data processing. It leverages the familiar concepts of databases, tables, records and fields. For example, it can connect to a database, open a table, find a record by key, and walk records in key order. c-treeDB is ideal for achieving extreme, predictable performance because it gives developers complete control over:

See package folder: .\drivers\vb.nav



5.2 JAVA

JAVA JSON DB - Add new driver for Java Developers

FairCom DB Java Driver for the JSON DB API

The Java driver makes it easy to use FairCom's JSON APIs . It combines the Jackson JSON library with a few Java classes to send JSON over HTTPS to FairCom's JSON APIs , such as the JSON DB API.

- The DbApi class provides methods to manage FairCom DB. It makes it easy to learn and manage FairCom DB from within a Java application.
- The AdminApi class provides methods to administer a FairCom server.

The driver's source code is part of the downloadable Faircom server package. It is located in `<faircomInstallationFolder>/drivers/java.json.db/`. The source code is well-documented and supports code completion.

The JSON DB API manages the FairCom database. It provides actions to manage databases, tables, indexes, and records. It is based on the jsonAction protocol, which sends JSON requests to a server and returns JSON responses. This API is based on the jsonAction protocol, which POSTs JSON requests to one endpoint on a server and returns JSON responses. This section defines the JSON payloads that can be POSTed to the FairCom server over HTTP, HTTPS, WS, and WSS using the endpoint `/api`, such as `https://localhost:8443/api`.

There are properties that are universally used within every action of each jsonAction request and response message. For a comprehensive explanation and multiple examples of these properties, see jsonAction protocol.

Features:

- Connect: connect to a FairCom server
- Create sessions: create one or more sessions with the server
- Manage sessions and their settings: create, alter (change), delete, list, and ping sessions
- Manage databases: create, delete, and list databases
- Manage tables: create, alter (change), delete, list, describe, and rebuild tables
- Manage indexes: create, alter (change), delete, list, and rebuild indexes
- Manage data: insert, update, and delete records
- Query data: get records using IDs, keys, partial keys, tables, indexes, key ranges, and SQL
- Use cursors: efficiently paginate, walk, and skip forward and backward through query results
- Run SQL: run statements and queries
- Manage ACID transactions: create, commit, rollback, and list transactions and savepoints



5.3 JDBC/JTDB

JDBC crash fixed

A JDBC crash was seen due to a rare situation of a null variable being passed to **ctree.jdbc.SQLXApi.identifier_case**. The logic has been modified to always initialize the offending variable and others initialized in the same manner.

Affected Components: JDBC driver

JDBC Installers now available

Separate JDBC installers are now available for Windows platforms. The installers include a feature to append the JAR to the CLASSPATH environment variable. If CLASSPATH doesn't exist, it will be created.

FindTarget Method Added to c-treeDB for Java

The **CTRecord::FindTarget()** method has been implemented in c-treeDB Java.

This method automatically handles the **TransformKey** call if required. The target parameter *must* have the same or greater size as the current default index key length.

Improved JDBC efficiency

Internal improvements were made to increase the speed of the JDBC driver. The JDBC internal logic that resets parameters has been made smarter so it reallocates the *bound_flags* and *need_data* arrays only if necessary.

Affected Components: JDBC driver

Access c-treeDB Java API with Embedded Server Models

The usability of FairCom DB API Java (JTDB) has been improved to support for multiple c-tree operational models. New functionality has been added to the JTDB API to allow specifying which operational model to use.

CTSession.SetDBModel(int model)

The very first call for the JTDB API needs to be **CTSession.SetDBModel(int model)** where *model* can be one of the following:

- CTDB_MODEL.CLIENT: multi-threaded client



- CTDB_MODEL.STANDALONE: multi-threaded multi-user standalone
- CTDB_MODEL.SERVER: embedded server dll

If not called, by default the model is CTDB_MODEL.CLIENT. An attempt to call this method when already called or the c-tree library has been already loaded results in an exception (CTDBRET.ISACTIVE).

CTSession.GetDBModel()

A new function has been added to get the library type in use: **CTSession.GetDBModel()**

```
static CTSession::GetDBModel()
```

Java JNI DLLs

Three special Java JNI DLLs have been created to be used with the JTDB interface:

- *ctdbsjni.dll* - Server DLL model
- *ctstdjni.dll* - Single User Tranproc StandAlone DLL
- *mtcljni.dll* - MT Client DLL

LIB Names

The LIB names have been changed from:

ctreeedbs and *cttreestd*

to:

ctdbsjni and *ctstdjni*

The mtree build system has been updated to support these models.

JDBC - allow URL without hostname

SYMPTOM: JDBC driver does not accept a URL with no hostname (thus indicating a local connection).

CAUSE: URL parsing have check to specifically disallow that. Which is not really a good idea, since for local connection it may be a good idea to not specify the hostname and therefore avoid DNS queries (which may be slow).

SOLUTION: changed URL parsing logic to accept URL without the hostname, and in such case set the hostname to "" which in turns avoid DNS queries.

JDBC now Supports Setting Parameter to NULL

When setting up a PreparedStatement object and its parameters, a CtreeSQLException may have been thrown by first calling **SetString(1, "")** and then **SetString(1, null)**.



This was due to an internal logic error involving NULL handling in very isolated cases. In this case, the improper NULL handling prevented the parameter type from being set, though this is now corrected.

Remove JDBC setEscape Processing Flag Exception

The FairCom DB JDBC driver throws an exception when setEscapeProcessing was called from a JDBC application setting the flag to "false". As information is escape processed on the server side the flag was considered an exception. However, this state can be safely ignored. Behavior is now to ignore a "false" flag with no exception.

JTDB - Added CTTable.HasLocks()

5.4 ODBC

Unexpected 64-bit ODBC Driver Overflow Error

Recent 64-bit SQL ODBC drivers could unexpectedly return arithmetic overflow errors. A prior revision introduced code that improperly cast an internal pointer to an incorrect numeric type thus truncating to an invalid value and then later referenced causing the overflow error. An additional check is added with a correct value interpretation preventing the ODBC.

Corrected ODBC Error -5 'wrong number of parameters' When Creating Stored Procedures

The ODBC driver failed a **SQLExecDirect()** call when the statement was a create procedure and the procedure text contained a "?". The logic has been corrected to avoid counting the "?" in the creation statement.

Changed Character Validation in ODBC Connection String

Previously, FairCom DB ODBC failed with **ERROR [01S00][FairCom][ODBC FairCom Driver 11.5.51930(Build-180302_191219)] -4 Invalid connection string attribute** when any string, including passwords contained unsupported characters such as ASCII symbols, for example, () ! / [].

FairCom DB password handling fully supports all characters so it is necessary to provide them via the ODBC connection string. This has since been corrected and all characters except semi-colons are now allowed in ODBC connection strings



5.5 C++

Improved c-treeDB Truncate Table Functionality

The c-treeDB truncate table function has been improved to intelligently handle specific situations that previously caused it to fail. These include:

1. **ctdbTruncateTable** is unsupported (because it is turned off internally in the code via #define's)
2. There was a pending update to a transaction controlled table.

ctdbTruncateTable() has been updated such that if it is not supported or fails with **TRNU_ERR**, **ctdbAlterTable(..CTDB_ALTER_TRUNCATE)** is instead called attempting to carry on the truncate function. Truncating a table using **ctdbAlterTable(..CTDB_ALTER_TRUNCATE)** is less efficient and cannot be "rolled back" by aborting the transaction; however, it works in the cases listed above.

A new method with the same behavior, **CTTable::TruncateTable**, is added to the c-treeDB C++ API.

C++/CTPP - Added SetBinaryFlag and GetBinaryFlag to the CTField class

The C++ CTPP API was missing a method the Set/Get the field binary flag

```

/*^*****\
 *   CTField
 *   CTDB C++ API field class
 *^*****/
class ctdbEXPORT CTField : public CTBase
{
....
    virtual void SetBinaryFlag(const CTDB_BINARY_FLAG binary_flag);
    virtual CTDB_BINARY_FLAG GetBinaryFlag() const;
};
/*^*****\

/*^*****\
 *   Function:   SetBinaryFlag - Set the field binary flag, which many times is actually needed to
 *               indicate that the field is not binary.
 *   Parameters: binary_flag [IN]
 *               Here are the most important ones:
 *               CTDB_BINARY_VARCHAR = 4,  VARCHAR/LVARCHAR field created by c-treeSQL in
 *               v10
 *               CTDB_BINARY_CLOB = 5,    CLOB field created by c-treeSQL in v10
 *               CTDB_BINARY_BLOB = 6,    BLOB field created by c-treeSQL in v10

```



```

*      CTDB_NUMBER_MONEY = 7,      NUMBER field used to store MONEY type (instead of
using  CT_CURRENCY)
\*****/
void CTField::SetBinaryFlag(CTDB_BINARY_FLAG binary_flag)

/*^*****\
*      Function:   GetBinaryFlag - Get the field binary flag.
*      Returns:   binary_flag
*      Here are the most important ones:
*      CTDB_BINARY_VARCHAR = 4,  VARCHAR/LVARCHAR field created by c-treeSQL in
v10
*      CTDB_BINARY_CLOB = 5,     CLOB field created by c-treeSQL in v10
*      CTDB_BINARY_BLOB = 6,     BLOB field created by c-treeSQL in v10
*      CTDB_NUMBER_MONEY = 7,    NUMBER field used to store MONEY type (instead of
using  CT_CURRENCY)
*
\*****/
CTDB_BINARY_FLAG CTField::GetBinaryFlag() const

```

C++/ CTPP - CTSession::FindDatabase with char* parameters

Added new method CTBOOL CTSession::FindDatabase(const char* Name, char*Path, size_t Pathsiz, ULONG& uid)

```

/*^*****\
*      Method:     FindDatabase - Locate a database in a session.
*      Parameters: *
*      Name [IN] - Name of the database to look for.
*      Path [OUT] - If the database name is located, receives the path.
*      Returns:    Returns true if the the database name was found. Returns
*                  false otherwise.
\*****/
CTBOOL CTSession::FindDatabase(const char* Name, char* Path, size_t Pathsiz, ULONG& uid)

```

CTPP - CTEException::GetErrorMsg() may return a pointer to released memory

The following notes are a compilation of a series of updates regarding **CTException::GetErrorMsg()**.

Internal testing revealed that when the stack variable was passed to the **CTException()** constructor (which stores the pointer value for later use), it went out of scope before the **CTException** object referenced its pointer value, but after the exception was thrown. To correct



this, **CTException** implementation has been updated to duplicate the error message and/or the file name, and to release the memory in the object distructor.

CTException::GetSourceFile() and **CTException::GetErrorMsg()** have also been updated to ensure they do not return NULL. If the message or the filename are missing, they now return an empty string.

Additional testing revealed that in some cases, the allocator and de-allocator were different depending on the context where the **CTException** was allocated or released, causing crashes or memory corruption. The **CTException** class was updated to have the message and file strings as predefined arrays, and code was added to copy in the message and file name (when appropriate) with truncation, if necessary.

Lastly, the following updates have been made to the **CTException** class destructor:

- Pre-allocated string sizes are now handled appropriately
- when CTEXCEPTION_ALLOC is turned on, it uses free instead of _ctdb_free.

Copy Constructor added to CTException Class

A crash was detected in CTException destructor when the code caught a CTException and then re-threw the exception, and allocated memory that was freed was dereferenced.

This crash was due to the CTException class not having a copy constructor, which has been added with this update.

C++ CTPP - added overloads for CTString::Compare and CTString::CompareIC

The CTString class exposes two methods to compare against another CTString but there is no overload that take a straight "char ". Thus in case we need to compare with a "char *" (or a literal), the code needs to allocate a CTString and copy the string in to pass to the two members, which then use the passed in parameter to get to the inner char and pass it to the comparison function.

This is highly inefficient due to memory allocation and string copy. Passing the char * directly is significantly quicker.

New methods added:

```
/\*****\
```

Method: Compare - Compares this CTString to str, with case sensitivity.



This Compares this CString to str, with case sensitivity.

This method is NULL safe.

Parameters: str [IN] - char pointer to be compared to.

Returns: The return value is shown below:

Condition	Return Value
S1 > S2	positive number greater than zero
S1 < S2	negative number
S1 = S2	zero

*****/

```
NINT CString::Compare(cpTEXT str) const{    return ctdbCompare(this->m_text, (pTEXT) str, NO);}
```

/^*****\

Method: CompareIC - Compares this CString to str, case insensitivity.

This method is NULL safe.

Parameters: str [IN]

char pointer to be compared to.*

Returns: The return value is shown below:*

Condition	Return Value
S1 > S2	positive number greater than zero
S1 < S2	negative number
S1 = S2	zero

*****/

```
NINT CString::CompareIC(cpTEXT str) const{    return ctdbCompare(this->m_text, (pTEXT)str, YES);}
```

C++/CTPP: Added CString::HasLocks()

SYMPTOM: The CTDB function ctdbTableHasLocks() was never implemented in CTPP

CAUSE: Not yet implemented

SOLUTION: Implemented as BOOL method to return YES in case locks are present false otherwise



5.6 PYTHON

Python APIs

We continue to support the following drivers for Python developers:

- JSON NAV API for Python (python.json.nav) - The JSON NAV API is an asynchronous, protocol agnostic, data definition, manipulation and query language that uses JSON to convey directions to, and receive data back from, the server. The current implementation of the JSON NAV API uses the WebSocket TCP/IP protocol standard. While it is perfectly fine to use the JSON NAV API directly from any language that supports websockets and JSON, this Java client driver library for the JSON NAV API is a convenience library to make using the JSON NAV API easy. The client driver uses a single dependency on the popular, well established Java library `org.java-websocket`.
- NAV API for Python (python.nav) - The c-tree NAV API for Python is a record-oriented database API. Its short name is c-treeDB for Python or simply CTDB. It is in the "NAV" family of APIs because it makes it easy for application developers to navigate data — no matter how complex. c-treeDB for Python is an object-oriented API that provides precise control over every aspect of data processing. It leverages the familiar concepts of databases, tables, records and fields. For example, it can connect to a database, open a table, find a record by key, and walk records in key order.
- SQL for Python (python.sql) - FairCom has implemented a pure Python module called "pyctree" that interfaces with FairCom DB SQL Server through the well-known ctypes module. This architecture has been chosen because it makes the module immediately available for multiple Python implementations, such as PyPy and Jython, and it does not require a C compiler to install the extension. Pyctree consists of two parts both completely written in Python:
 - `pyctsqlapi.py` - A wrapper library over the DSQL API
 - `pyctree.py` - A DB-API 2.0 implementation conforming to Python PEP 249 standards.
- SQLAlchemy Support for Python (python.sql.sqlalchemy) - SQLAlchemy is one of the most widely used object-relational mapping tools in the Python community. This Object Relational Mapper gives application developers the full power and flexibility of SQL. The FairCom DB SQL Python interface provides support for this technology.

5.7 .NET

.NET Core

.NET Core - Microsoft's free, open-source, general-purpose development platform

.NET Core is a version of .NET Framework, which is a free, open-source, general-purpose development platform maintained by Microsoft. It is a cross-platform framework that runs on Windows, macOS, and Linux operating systems. This happens to be one of the major contributions by Microsoft. Developers can now build Android, iOS, Linux, Mac, and Windows applications with .NET, all in Open Source.



FairCom made a minimal adjustments to be portable and compatible with the .NET Core platforms, including:

- See these folders in your 'drivers' folder for makefiles and IDE Projects for .NET Core:
 - `csharp.nav\tutorials`
 - `vb.nav\tutorials`
 - `csharp.sql.ado.net\tutorials`
- Removed all the references to `Windows.Forms`: Use of forms or message box results in an exception with error code 4000 which causes the termination of the .NET application. In order to avoid using message boxes we now return the strong signing errors as `CTExceptions` with a 4000 error code (CTDB base error) and the same error message that used to be on the message boxes.
- Removed extension from native DLL: On Linux/macOS/Unix the end user was forced to rename the FairCom native client- side library (`libmtclient.dll`) to be loaded from FairCom.CtreesDB to an un- natural ".dll" extension name instead of the platform specific one (.so or .dyn). Before, the FairCom.CtreesDB code explicitly requested the native library names with the ".dll" extension.
- With .NET Version 4.0 forward, it is possible to declare the native library name without the extension. Both the 4.0 or the Core runtime will take care to append and/or prepend the proper strings for the platform. Tested the solution on Linux, Linux ARM32 with .NET Core 5.0, Windows 10 With Framework 4 (updated) and Windows 7 with "vanilla" Framework 4.0. Users who experienced this issue with our Shared Memory ADO.NET technology (`libctsqlshm.dll` vs. `libctsqlshm.so`) will be happy to see this compatibility adjustment.
- .NET Core is now fully supported on all platforms including Windows and Linux. An exception with error code **4000** is returned when your project is not signed and linked to the FairCom DB assembly.

Additional .NET Drivers

The following .NET standalone record access drivers are now included in the FairCom PRO package in this location:

FairCom-DB.windows.64bit.v12.0.1.255\drivers\ctree.drivers\bin

- *FairCom.CtreesDb.dll* - Already in package
- *FairCom.CtreesDbSrv.dll* - Server DLL Edition

StandAlone PRO Package:

- *FairCom.CtreesDbLib.dll* - Local library (LOCLIB) Edition
- *FairCom.CtreesDbStd.dll* - StandAlone Edition



.NET - Added CTable.HasLocks()

SYMPTOM: The CTDB function `ctdbTableHasLocks()` was never implemented in .NET

CAUSE: Not yet implemented

SOLUTION: Implemented as boolean method to return true in case locks are present false otherwise

COMPONENTS: .NET CTable object

Method: HasLocks

Check if there is any lock acquired on the table in the current session

Parameters: none

Returns: Return true if locks are present.

.NET Stored Procedures

.NET Stored Procedures - Implement access to LVAR* fields

`ctsqlapi()` now implements 'long var' fields support for server side stored procedures.

- .Net stored procedures make calls to '`ctsqlapi()`' inside the server in order to interact with the SQL engine. In '`ctsqlapi()`', we had not implemented the server side support for the calls to get/put LVAR data. Therefore (until now) it was not possible to read/write LVAR* data from the stored procedure. Now, we have implemented the support and adjusted the code to satisfy this need.
- We have completed the implementation in the core server layer and in the C# layer to provide easy access to the LVAR* fields with stored procedures
- The access to a LVAR* field in a result set is possible through the `SqlSpCommand.CurrentRow[].Value` is always returned as `Byte[]`.
- The insertion is always made through parameterized prepared statements and assigned using `SqlSpCommand.Parameter[].Value`. The value must be passed as `Byte[]`.
- We manage all LVAR* (LVARCHAR included) fields through byte arrays so the end user has enough flexibility to encode/decode the contents in their preferred encoding without the need to force a default encoding in the SQL Stored Procedure `SqlSP*` functions.

.NET Stored Procedure - More information

Support for LVAR* fields has been added to the .NET Stored Procedure layer, and the existing `SqlSpCommand` has been updated to make `LVARCHAR` and `LVARBINARY` fields accessible through `SqlSpCommand.CurrentRow[].Value`.

Note: The value is always returned as `Byte[]`.

`SqlSpCommand` has also been updated to support the insertion and update of `LVARCHAR` and `LVARBINARY` fields as well. The insertion is made through parameterized queries, and the value must always be passed as `Byte[]` through `SqlSpCommand.ParameterCurrentRow[].Value`.



LVARCHAR and *LVARBINARY* are handled as parameters, so only prepared statements can be used to assign values. Due to this handling, **ExecuteNonQuery()** can't be used to assign *LVARCHAR* and *LVARBINARY*. **Execute()** is the only supported method for assigning these types of fields.

Below are examples of this new behavior.

Read Example:

```
SqlSpCommand cmdSelect = NewSqlSpCommand();
String strCommandSelect = "SELECT lvarbinfield,lvarcharfield,pk FROM lvartbl
where pk = 2";
cmdSelect.Prepare(strCommandSelect);
cmdSelect.Execute();
while (cmdSelect.MoveNext())
{
    //Reading LVARBINARY field as object. It may be casted to Array
    object xml = cmdSelect.CurrentRow[0].Value;
    //Reading LVARCHAR field and casting as Byte[]
    byte[] vs = (byte[])cmdSelect.CurrentRow[1].Value;
    if (vs != null)
    {
        //If the byte[] is not null convert the byte array using
        //the preferred encoding
        string text = Encoding.Default.GetString(vs);
    }
}
```

Write Example :

```
SqlSpCommand cmdInsert = NewSqlSpCommand();
String strCommandInsert = "INSERT INTO lvartbl(lvarcharfield,lvarbinfield)
VALUES (?,?)";
cmdInsert.Prepare(strCommandInsert);
cmdInsert.Parameter[0].Value = Encoding.Default.GetBytes("This will be a
LVARCHAR");
cmdInsert.Parameter[1].Value = Encoding.Default.GetBytes("We should insert
binary data here");
int rv = cmdInsert.Execute();
if (rv != 0)
    throw new Exception();
```



.NET Entity Framework

ADO.NET with EF6 Written Tables now Return Correct Identity Values

Previously, a “syntax error” was thrown when EF6 was used to write a into a table that defines an identity field. This error was caused by an internal incompatibility between sub-statements and FairCom's ADO.Net driver. This has since been corrected by modifying the logic to return a reader with the correct identity value.

Entity Framework Table Addition Speed Optimized

Previously, adding a table to an existing c-tree based entity data model could have taken up to 4 hours. This was due to an internal recursive call that caused the stack to grow increasingly large, and has since been fixed by modifying the code to avoid recursive calls and frequent memory allocations.

AFFECTED COMPONENTS: Entity framework 6 driver

Correct Entity Framework ADO.NET Parameter Handling Using INSERT Statements

An addition to an entity framework model failed with a “missing input parameters” exception. Entity Framework 6 always runs a returning SELECT to check and validate the primary key value of the row just inserted. This worked correctly for the INSERT statement, however, the ADO.NET provider was not setting parameters for the returning SELECT if parameters were used. To address this we replaced the INSERT sequence with two explicit commands, one for the INSERT statement and one for the SELECT statement whereby all SQL parameters are considered.

.NET Gui Tools

Dr. ctree does not show large DODA

Dr. ctree (.NET) did not show the DODA from a file (while ctutil does) when the DODA contains a large enough number of fields.

The code in FairComISAM assembly fails to retrieve the schema map if it takes more than 8192 bytes and not having the schema map the DODA is not properly retrieved as well. The schema map is not retrieved properly since the code assumes it is at maximum 8192 bytes large and does not realloc and retry in case is larger.

We fixed this issue by adding code to realloc the buffer where to store the schema map and redo the retrieval.

Dr. ctree: Header Values incorrectly shown

SYMPTOM: On 32bit platforms the header values are shown 4 bit off causing all the header values to be shown incorrectly.

CAUSE: Because of the removal of the ctalign member of the header (never present on 64bit) we need to skip a size of LONG.



SOLUTION: Added code to skip LONG size bytes on 32bit platforms.

Dr.Ctreet: Fix for header descriptions

SYMPTOM: Some of the headers description were not consistent with the current header specs

CAUSE: The header descriptions were based on an old version of the header struct

SOLUTION: Fix the header descriptions in both .NET and Java implementations, this required the update of the core libraries as well

.NET/Java/Web Tools: Added support for USERINFOX with user memory over 2GB

SYMPTOM: The user statistics shown in the monitoring tools may show negative user memory values when the user memory usage goes over 2GB

CAUSE: We found that the variable for user memory may overflow so we implemented a new version of USERINFO that has to be implemented in our monitoring tools

SOLUTION: Implemented the new USERINFOX in the graphical tools: .NET, Java and Web

.NET SQL Explorer: Forbid editing of autotimestamp fields

SYMPTOM: Impossible to enter a value for autotimestamp fields as any value was invalid and NULL was not admitted.

CAUSE: The SQL client (ADO.NET) correctly returned an error -20189 when trying to insert a value for an autotimestamp field. The NULL value was not admitted so with the records editor was not possible to insert the row.

SOLUTION: Copied the logic used to manage the identity fields and adapted to autotimestamp fields. This required a modification to the column class to have a new property to indicate whether the column is an autotimestamp field. The autotimestamp fields now behave correctly.

.NET SQL Explorer: Trigger information not shown

SYMPTOM: SQLEditor not showed the information of the selected trigger from the left tree.

CAUSE: The internal code tried to extract triggers information from admin.systrigcols rather than admin.systrigger.

SOLUTION: Changed the internal code to select from admin.systrigger, the remaining part of the internal code is working correctly even when selecting data from the new table.

.NET SQL Explorer: Unable to show Date.MinValue

SYMPTOM: In case a date equal to C# Date.MinValue needs to be shown in the SQL Queries tab results area a blank cell is shown instead.



CAUSE: In the cell population code there was a check to show an empty cell if the date value to be shown was equal to Date.MinValue.

SOLUTION: As there is not a way to set a Date or DateTime value to something less than Date.MinValue the check have been removed.

.NET Security Admin: Fixed on screen behavior of the servers' combobox

SYMPTOM: The servers combo didn't follow the top anchor causing the combo to stay in the same position when the window size was changed.

CAUSE: The combo box anchors were not correctly set.

SOLUTION: Fixed the combo box anchors.

.NET LogAnalyzer: Added support for new CTSTATUS.FCS single line format

SYMPTOM: Log Analyzer required changes to open newer CTSTATUS.FCS format with messages on a single line.

CAUSE: The parsing logic was designed to parse the old, multiline, format only.

SOLUTION: Added new logic to correctly distinguish between CTSTATUS.FCS formats and parse accordingly

.NET ISAM Explorer: Write Out ISAM Record Structures Containing SQL NUMBER Types

ISAM Explorer is a useful visual utility to view the internal FairCom DB attributes for tables and indexes. It can also output the record structure definitions, schema resource (*DODA*) and *IFIL* structures defining the table and indexes. It was determined the structure output of a file containing a CTNUMBER field was mapped as a "unknown type" which caused the header file to be unusable in C.

CTNUMBER is a relatively new field introduced after the original implementation of ISAM Explorer. The field type handling when writing out the record structure members was reviewed and now maps all FairCom DB types to their C structure definitions appropriate for including in external applications.

However, even if all the other record structure types are mapped to native C types CTNUMBER is a specific FairCom DB type with no native representation. To use a structure definition with this field type include the c-treeDB header *ctdbsdk.h*.

6. CTDB

6.1 c-treeDB Default Extent Sizes Supported Beyond 64K

ctdbSetTableDefaultDataExtentSize and **ctdbSetTableDefaultIndexExtentSize** set the data and index extension sizes in the *IFIL*. These functions were originally written in a way limiting their maximum values to 65,535, even though c-tree allows much larger extensions. The logic has been modified to allow effectively setting extension sizes larger than 65,535.

6.2 Set First Extent Size in FairCom DB API

To reduce file fragmentation on disk, customers may want to set a large file extent size. c-tree and FairCom DB API allow this possibility, however as soon as a larger extent size has been set, the minimum initial size of the file is equal to the extent size. This results in a huge initial disk usage. c-tree allows setting the "first" extensions size to limit this problem, but FairCom DB API did not, prior to this enhancement.

The following functions have been added to expose this feature to FairCom DB API:

ctdbGetTableFirstDataExtentSize

Retrieve the table first data extent size.

```
NINT ctdbDECL ctdbGetTableFirstDataExtentSize(CTHANDLE Handle)
```

ctdbGetTableFirstIndexExtentSize

Retrieve the table first index extent size

```
NINT ctdbDECL ctdbGetTableFirstIndexExtentSize(CTHANDLE Handle)
```

ctdbSetTableFirstDataExtentSize

Set table first data extent size (creation time size).

```
CTDBRET ctdbDECL ctdbSetTableFirstDataExtentSize(CTHANDLE Handle, NINT size)
```

ctdbSetTableFirstIndexExtentSize

Set table First index extent size (creation time size).

```
CTDBRET ctdbDECL ctdbSetTableFirstIndexExtentSize(CTHANDLE Handle, NINT size)
```



6.3 Improved c-treeDB Truncate Table Functionality

The c-treeDB truncate table function has been improved to intelligently handle specific situations that previously caused it to fail. These include:

1. **ctdbTruncateTable** is unsupported (because it is turned off internally in the code via `#define's`)
2. There was a pending update to a transaction controlled table.

ctdbTruncateTable() has been updated such that if it is not supported or fails with **TRNU_ERR**, **ctdbAlterTable(..CTDB_ALTER_TRUNCATE)** is instead called attempting to carry on the truncate function. Truncating a table using **ctdbAlterTable(..CTDB_ALTER_TRUNCATE)** is less efficient and cannot be "rolled back" by aborting the transaction; however, it works in the cases listed above.

A new method with the same behavior, **CTTable::TruncateTable**, is added to the c-treeDB C++ API.

6.4 Change SQL Database Path Utility

The **ctpath** utility allows you to adjust the path in the dictionary after extracting the SQL database directory from a backup and renaming it. This utility is command-line only, which allows it to run on all supported operating systems. It also includes a command line switch **-v** for verbose output.

6.5 ctpath - Change Internal (SQL) Database Paths

The **ctpath** utility allows you to adjust the path in the dictionary after extracting the SQL database directory from a backup and renaming it.

Changes the internal FairCom DB SQL dictionary paths of database locations.

Syntax

```
ctpath [-s server] [-u user] [-p password] [-d database] [-v] from-path-prefix to-path-prefix
```

Options:

- **-s server** - FairCom DB server name
- **-u user** - user name
- **-p pw** - user password
- **-d database** - database name
- **-v** - verbose
- **from-path-prefix** - path prefix to be replaced
- **to-path-prefix** - path prefix to use as replacement



Description

- Command-line switches may *not* have optional spaces between the switch and the argument. Example: `-s FAIRCOMS` is *not* the same as `-sFAIRCOMS`.
- Command-line switches may be entered in any order, but the *from-path-prefix* must appear *before* the *to-path-prefix*.
- Command-line switches should start with a '-' or '/' character. Command line switches accept both lowercase and uppercase characters, e.g. `-s` or `-S` are the same.
- `ctpath` returns 0 when the execution detected no errors. Non-zero values are returned when errors are detected. Error messages are written to `stderr`.
- If you omit the `-d` database switch, all databases in the session will be scanned.
- The `-r` switch indicates to repeat the substitution and replace all occurrences of the searched path and not just the first one.
- The `-v` command-line switch indicates verbose output.

6.6 Set Table Partition Base in FairCom DB API

A new c-treeDB C API function, `ctdbSetTablePartitionBase(CTHANDLE Handle, LONG base)`, has been added to set the partition base number when a table is created or altered (if table is reconstructed) after full rebuild.

Parameters:

- *Handle* - [IN] `CTTABLE` table handle
- *base* - [IN] `LONG` Partition number to set as base

Returns: c-treeDB error code.

6.7 Support for Modifying file Partition Added to c-treeDB

`ctdbSetTablePartitionRule` and `ctdbSetTablePartitionIndexNbr` now allow users to modify the partition size of files that have already been partitioned.

6.8 Getter Setter Methods for Blobs as base64 Strings

With this revision, methods have been added in c-treeDB C++ `CTBlob` class to set blob content from a base64 string and to retrieve the blob content as a base64 string.

Below are the added methods:



```
CTBlob::SetBlobAsBase64(const CString& str)
CTBlob::SetBlobAsBase64(const char *str)
void CTBlob::AsBase64String(CString& str) const
```

The following new functions were also added:

ctdbBlobSetAsBase64()

Description

Sets the blob data from a buffer encoded in Base64*

Parameters

- *pBlob* [IN] - Blob pointer*
- *buffer* [IN] - Base64 encoded buffer*

Return

CTDBRET_OK on success

ctdbBlobGetAsBase64()

Description

Clear a blob variable by releasing memory associated with *data, setting data to NULL and size to zero*

Parameters

- *pBlob* [IN] - Blob pointer*
- *pbuffer* [OUT] - pointer to data*
- *pSize* [IN/OUT] - number of bytes in buffer*

Return

CTDBRET_OK on success, *pSize* is set to the actual number of bytes used

6.9 CTPP - CTEException::GetErrorMsg() may return a pointer to released memory

The following notes are a compilation of a series of updates regarding **CTException::GetErrorMsg()**.

Internal testing revealed that when the stack variable was passed to the **CTException()** constructor (which stores the pointer value for later use), it went out of scope before the



CTException object referenced its pointer value, but after the exception was thrown. To correct this, **CTException** implementation has been updated to duplicate the error message and/or the file name, and to release the memory in the object destructor.

CTException::GetSourceFile() and **CTException::GetErrorMsg()** have also been updated to ensure they do not return NULL. If the message or the filename are missing, they now return an empty string.

Additional testing revealed that in some cases, the allocator and de-allocator were different depending on the context where the **CTException** was allocated or released, causing crashes or memory corruption. The **CTException** class was updated to have the message and file strings as predefined arrays, and code was added to copy in the message and file name (when appropriate) with truncation, if necessary.

Lastly, the following updates have been made to the **CTException** class destructor:

- Pre-allocated string sizes are now handled appropriately
- when CTEXCEPTION_ALLOC is turned on, it uses free instead of _ctdb_free.

6.10 Copy Constructor added to CTException Class

A crash was detected in CTException destructor when the code caught a CTException and then re-threw the exception, and allocated memory that was freed was dereferenced.

This crash was due to the CTException class not having a copy constructor, which has been added with this update.

6.11 ctdbGetFieldAs* and ctdbSetFieldAs* from RTG column with dbtype set

ctdbGetFieldAsString() on a RTG type mapped to a date by dbtype=date did not return a string representing a date.

In cobol RTG there is no specific time for dates/times but rather a cobol base type is used and then its value gets interpreted as a date/time depending on the dbtype and the format. Therefore the first implementation was to retrieve the data as the base type however that is not very convenient.



Therefore, considering that when checking for NULL value we already interpreted the field value as the remapped type, we now changed the behavior and when a dbtype• is specified we interpret the content as the specified type in all

ctdb[G|S]etFieldAs* functions with the exception of ctdb[G|S]etAsBlob and ctdb[G|S]etAsBinary where instead we return the actual record buffer content (which seems appropriate).worth noticing the ctdbGetFieldType() return the RTG base type also for remapped fields and ctdbGetFieldNAVProperties() returns NAV_DATE/NAV_TIME/NAV_DATETIME depending on the dbtype set.

6.12 CTDB Filter support for BTRIEVE date and time data types

We continue to support our BTRIEVE driver technology. The CT_BT_DATE and CT_BT_TIME data types were not supported by our filters. We have now added this support.

6.13 ctdbNumberOfEntries() Added to CTDB

A new function, **ctdbNumberOfEntries()**, has been added to the c-treeDB C API. This function retrieves the number of key entries in an index file, and has the following parameters:

- *Handle* [IN] - the table handle
- *index* [IN] - the index number. The first index number is always zero.

ctdbNumberOfEntries() returns the number of keys in a 64-bit integer, preventing overflow. The function also accepts special index numbers, and mandates the use of an index to count the entries. If there is no available index, the function errors out instead of either trusting the recordcount held in the records header or counting the records.

6.14 CTDB - alter table from attribute - alter table support for RTG

One idea to be able to alter the table in RTG is to modify the COBOL source code, generate the XDD and pass the new XDD to ctutil that then identifies what changed and apply the changes. Since the XDD gets converted into attributes and CTDB is aware and can handle the attributes, it seemed natural to have CTDB compare the attributes and perform the alter table. To this end we added to CTDB a new function that perform this operation. At this time limited to the RTG necessities/capabilities.

```
/^*****\
```

Function: ctdbAlterTableFromAttr - Rebuild existing table based on changes indentified comparing the attributes



passed in against the one in the table

Parameters: Handle [IN] - Table handle
 TblAttrHdl [IN] - Table attribute handle
 Action [IN] - Alter table action. (see ctdbAlterTable)

Returns: Return CTDBRET_OK on success

*****/

CTDBRET ctdbDECL ctdbAlterTableFromAttr(CTHANDLE Handle, CTATTR_HANDLE TblAttrHdl, NINT Action)

6.15 CTDB - New function ctdbInsNAVField()

We added a new function in CTDB to handle NAV types. Similar to ctdbInsField() for “regular” c-tree field types, we added this while working on our REST API.

/*****\

Function: ctdbInsNAVField

Insert a field before the field number given by Index to the table using high level data types*

Parameters: Handle [IN] - Table handle

index [IN] - Field index

FieldName [IN] - Name of field

FieldType [IN] - The field type

FieldWidth [IN] - The field width (in bytes, words for N*CHAR fields, precision for MONEY and NUMBER)

FieldScale [IN] - The field scale (applies only to NAV_NUMBER and NAV_MONEY)

nullable [IN] - The field nullability (YES if the field can be set to NULL)

The field nullability (YES if the field can be set to NULL)*

Returns: Return a field handle on success or NULL on failure.

*****/

CTHANDLE ctdbInsNAVField(CTHANDLE hTable, NINT Index, pTEXT FieldName, NAVDataTypes FieldType, NINT FieldWidth, NINT FieldScale, CTBOOL nullable)

6.16 ctdbSetDefaultValueAsBinary Added to CTDB

A new c-treeDB C API function was added to be able to set the default value passing the binary representation of the field content. Previously, setting binary values with **ctdbSetFieldDefaultValue()** would consider the value as a string with unexpected errors such as 749.



```
CTDBRET ctdbDECL ctdbSetFieldDefaultValueAsBinary(CTHANDLE Handle, pTEXT value,
VRLEN length)
```

ctdbbSetFieldDefaultValueAsBinary() sets the field default value as a binary value. The default value of a field is used during an alter table operation when a full table rebuild is performed. During a full alter table rebuild, and after the old record buffer data is moved to the new record buffer, the new record buffer is scanned and if a NULL field is found and that NULL field has a default value, the default value is copied to the field buffer. The field default value is kept as a binary representation of the data.

Parameters

- *Handle* - field handle
- *value* - pointer to a buffer containing the value to be used as default in its binary representation
- *length* - length of the buffer

6.17 NULL KEY VALUES Support - CTDB level metadata stored file's unified resource

For RTG to support a compatible BTRIEVE driver, we needed to implement NULL KEY VALUES Support in the CTDB/ISAM layers. Internal modifications have been made as follows:

We decided to use the attribute resource (unified resource) to store the nullkey information at c-tree ISAM level. There it is difficult to distinguish between indexes created by a specific API (CTDB/RTG) from indexes

created by the user in his application, yet we need to identify the attributes for the index we are interesting into. The natural idea is to use the relative index number, but that account for “internal indexes” while originally in RTG we

stored information only for user created indexes and in CTDB we did not store any index related information (unless altering an RTG table). However both were coded to consider that the index information in the attributes were only for “user defined” indexes.

Thus, for CTDB/ISAM we have no index information in the attributes, from RTG we have only user index information. When at ISAM level we store the nullkeyvalue we may pick the wrong index or in adding the information create empty index placeholder for all indexes in the IFIL, not only the “user defined” ones causing then misalignment and interpretation issues.

Therefore we decided to change the logic in order to store in the attributes information for all indexes in both CTDB and RTG mark internal indexes so that we can distinguish them (added a property on purpose) in CTDB add index information to the attributes at creation time.

Added to attributes and XDD the possibility to store the null value at segment level Attributes (for RTG RTGI information) and XDD (for RTG ctutil -make) needed to be extended to allow the possibility to specify the null value at segment level. To this end we added a new segment attribute and a new segment attribute in the XDD specification.



6.18 CTREE/CTDB/RTG - support for CT_BT_BIT field type

We added a new c-tree data type CT_BT_BIT which indicates a field occupying a bit in a byte. If the field is “isolated” that it takes one entire byte otherwise if there are multiple CT_BT_BIT in sequence it takes 1 bit within the same byte of the previous/next field.

In the DODA the length of this field should be set to 0 when taking the same byte of the field it precedes or 1 byte if it is isolated or the last byte of a sequence (of maximum 8 fields). It is mandatory to have field attributes setting the significant bit number (numbered from 0 right to left) the first field in the sequence (or the isolated one) takes bit 0, the second bit 1 etc.

Index creation is not supported.

6.19 CTDB - improved batch processing

SYMPTOM: batch operation are slow and make inefficient server calls:

any ctdbNextBatch call that does not have a record available and so needs to make a server call does a CHGBAT call even if the current batch is the one we need.

once the batch is finished, we need to make one final dobatch() to identify that we are done, one to cancel the batch and one to free the batch.

CAUSE:

1. Previously we didn't have logic to keep track of the “current batch number” to aid with avoiding useless calls.
2. Previously we didn't leverage the BAT_CLSE support that would signal the end of the batch avoiding all final calls to close out the batch.

SOLUTION:

implemented current batch number tracking

implemented the use of BAT_CLSE. This (by c-tree design) is effective only if the “COMPATIBILITY BATCH_SIGNAL” is specified in ctsrvr.cfg

here are the number of function calls before and after the modifications using a test program.

before:

CHGBAT	177400
FREBATN	162600
BATSETX	472800

after (with COMPATIBILITY BATCH_SIGNAL)



CHGBAT	35200
FREBATN	15000
BATSETX	177600

6.20 CTDB/CTREE/RTG - support for BTRV BLOB and CLOB

We have implemented support for two new data types CT_BT_BLOB and CT_BT_CLOB to interpret and update PSQL/BTRIVE's BLOB and CLOB data types. This required new attributes and new XDD elements (LOBheader attributes in fields) to point to the 8 bytes lob handle that BTRV puts in the record buffer.

6.21 CTDB - New functions to retrieve all dictionary information for tables

CTDB does not expose a method to retrieve all tables information stored in the dictionary but just a few methods to retrieve some of them. Both for efficiency and to retrieve information not yet exposed, we added two new functions to allow retrieving all table information from the dictionary during a dictionary scan, `ctdbFirstTableXtd` and `ctdbNextTableXtd` with these prototypes.

```
/^*****\
Function:  ctdbFirstTableXtd - Get the first table in a database dictionary.
Parameters:
    Handle [IN] - Database handle.
    data [OUT] - dictionary data for the table
Returns:   Returns CTDBRET_OK on success or the c-tree error code on failure.
*****/

CTDBRET ctdbDECL ctdbFirstTableXtd(CTHANDLE Handle, pCTDBDICTDATA data)

/^*****\
Function:  ctdbNextTableXtd - Get the next table in a database dictionary.
Parameters:
    Handle [IN] - Database handle.
    data [OUT] - dictionary data for the table
Returns:   Returns CTDBRET_OK on success or the c-tree error code on failure.
```



```

*****/
CTDBRET ctdbDECL ctdbNextTableXtd(CTHANDLE Handle, pCTDBDICTDATA data)

```

CTDB dictionary now also stores the table owner.

This new function `ctdbFindTableXtd` does it while searching for a specific table.

```

/^*****\
Function:  ctdbFindTableXtd - Find a table in a database dictionary
Parameters: Handle [IN] - Database handle
              Name [IN] - Table name
              data [OUT] - dictionary data for the table
Returns:    Returns CTDBRET_OK on success or the c-tree error code on failure.
*****/
CTDBRET ctdbDECL ctdbFindTableXtd(CTHANDLE Handle, pTEXT Name, pCTDBDICTDATA
data)

```

At table open time ensure we set the table owner in the `CTHANDLE` structure as defined in the dictionary, so that this information is available in the table handle itself.

6.22 Added new batch mode "BAT_AUGFIX"

Previously, a c-treeDB batch insert call failed with **NLEN_ERR[634]** on files with fixed length record size that were created with a *ctFLEXREC* mode. *ctFLEXREC* and other new modes may fall into a category of file modes we call augmented and providing advanced record handling features.

To correct this, a new batch mode *BAT_AUGFIX* has been added. It indicates that, for files with *ctAugmentedFixed* mode enabled, the record size is not specified in the batch buffer. In turn, c-treeDB has been modified to use this new mode to prevent **NLEN_ERR[634]** from occurring.

6.23 Performance Enhanced During Batches that Filter out Fields

Internal testing revealed a loss in performance when field filtering (field mask) was used with batches. The logic has since been updated, resulting in a net 10% performance gain.



6.24 Performance Enhanced for Batch Variable Length Record Read

Poor performance was reported for batch operations. New code has since been added to increase performance for variable length record reads.

6.25 Retrieve the conditional expression function

We added the function `ctdbGetIndexCndxExpr()` to retrieve the conditional expression given the index handle.

```
/*****  
Function:   ctdbGetIndexCndxExpr - Get the conditional expression for this index  
Parameters: Handle [IN] - Index handle  
Returns:    Return Conditional expression  
*****/  
pTEXT ctdbDECL ctdbGetIndexCndxExpr(CTHANDLE Handle)
```

6.26 ctdbGetNAVFieldProperties maps unsigned integer c-tree field to signed larger numeric

Both JSON NAV and SQL do not have unsigned integer types, however c-tree and CTDB do. When accessing a file with an unsigned integer from SQL to and from JSON NAV the data type mapping is somewhat difficult. By default we always mapped to the same signed type, but that means that some value show up as negative which is really inconvenient. For SQL we had an option at import time to “promote” the unsigned type to a signed larger type thus covering all possible value it may be set and, again optionally, have constraints to verify the boundaries. Now that we introduced boundary checks directly in CTDB instead of forcing them in SQL as check constraints, It makes much more sense that we default the other way around. That is by default we now promote unsigned integers to the next integer value and CTDB checks boundaries. In order to avoid this promotion it is possible to set the promoted members of the CTDBFIELD structure to HNO before calling `ctdbGetNAVFieldProperties`. `ctdbGetNAVFieldProperties` sets the promoted member to YES or NO depending on whether the promotion occurred or not (it leaves it untouched if set to HNO and the promotion does not happen). We may add in the future a way to set/get the promoted member without the need to access the structure directly.

This brakes compatibility with previous version since in SQL tables may be imported with different integer types, however the new mapping is a lot safer and should not cause any major issue. In any event we provide a way to use the old mapping with minimal effort.



6.27 CTDB - ctsqlimp utility and "sqlimp" functions default to promote unsigned integers

CONEXT: While working on CT_CHARU field causes problems

DETAILS: ctsqlimp utility has been updated as follows: In interactive mode the suggested default answer for "Do you want Unsigned integers to be promoted?" has been changed to YES

- -p parameter now legacy since that's the default.
- -P parameter still behave the same way yet the check constraint is useless as the check is performed at CTDB level now.
- -t parameter to revert to the "not promote" old behavior.
- -P, -p, -t are mutually exclusive.

SQL import functionality in CTDB/CTREE (either explicit or implicit) now default to promote the unsigned integers unless a "sql import callback" is provided that explicitly set the integer promotion differently.

This breaks compatibility since the behavior is now different but there are easy way to set it back to the previous behavior (if desired).

6.28 c-treeDB API Functions for User-Defined Hot Alter Table Field Conversion Callback

FairCom DB now supports passing the name of a shared library and name of a field conversion function to the hot alter table function. This permits application developers to set a user-defined field conversion function to convert field values from one version of a schema to another version.

We now extend our c-treeDB API to specify the conversion callback library at a table level and the callback conversion function name at a field level. If the callback library name at table level is not specified we retrieve the last one used. If no callback library has ever been specified we return an error.

c-treeDB C API Functions

ctdbSetFieldHotAlterCallbackFunction() sets the field hot alter table callback function.

```
CTDBRET ctdbDECL ctdbSetFieldHotAlterCallbackFunction(CTHANDLE Handle, pTEXT  
functionName)
```

Parameters: Handle [IN]*

Field handle **

functionName [IN]* The name of the function in the library specified in the table
handle**

Returns: CTDBRET_OK on success

ctdbSetTableHotAlterCallbackLib() sets the callback library name for hot alter table



```
CTBOOL ctdbDECL ctdbSetTableHotAlterCallbackLib(CTHANDLE Handle, pTEXT libName)
```

Parameters: Handle [IN]

Table handle.*

libName [IN]

Library name**

Returns: c-treeDB YES or NO

ctdbGetTableHotAlterCallbackLib() returns the callback library name for hot alter table

```
pTEXT ctdbDECL ctdbGetTableHotAlterCallbackLib(CTHANDLE Handle)
```

Parameters: Handle [IN]

Table handle.*

Returns: pointer to the internal library name buffer

6.29 C++/CTPP - CTable Open method to open tables using the table UID

In our C++ API (CTPP) We added the ability to open a table using the UID. Previously CTable object does not expose a method to do so therefore we added it.

```
/^*****\
Method:      Open - Open an existing table
Parameters: uid [IN] - The table UID
              OpenMode [IN] - The table open mode
Returns:     none
*****/
void CTable::Open(const ULONG uid, const CTOPEN_MODE OpenMode)
```

6.30 CTDB - New function ctdbDeleteBackgroundLoadStatus

Added a new function ctdbDeleteBackgroundLoadStatus to delete the background index load status information.

```
/^*****\
Function:    ctdbDeleteBackgroundLoadStatus - deletes the resource that contains the status of
background load
              operations for the specified entry of the specified table
Parameters: Handle [IN] - Table handle
              status [IN/OUT] - status information
              In input the function uses status->loadid
              if != 0. Otherwise it fetches the status.
```



Returns: Returns CTDBRET_OK on success

*****/

CTDBRET ctdbDECL ctdbDeleteBackgroundLoadStatus(CTHANDLE Handle, pBGLDINF status)

6.31 CTDB - added support for changelD field

CONTEXT: Implement change id field to support optimistic locking

DETAILS: added to CTDB support to define changelD field

/^*****\

Function: ctdbSetChangelDField - Set ChangelD Field for the table*

Parameters: Handle [IN] - Table handle*

FieldName [IN] - The identity field*

Returns: CTDBRET_OK on success or c-tree error code on failure.

*****/

CTDBRET ctdbDECL ctdbSetChangelDField(CTHANDLE Handle, cpTEXT FieldName)

^*****\

Function: ctdbGetChangelDField

Get the name of the changelD field.*

Parameters: Handle [IN]

Table Handle.*

Returns: changelD field name or NULL, in this case use ctdbGetError() to verify if there was an error or simply there is no identity field defined.

*****/

pTEXT ctdbDECL ctdbGetChangelDField(CTHANDLE Handle)

6.32 CTDB - Alter Table - ability to create/update conditional indices with table open shared

Prior to this change, in order to create a conditional index we required the table to be opened in exclusive mode. With the introduction of c-tree's internal PRMIIDX82 function this is not the case anymore. Therefore we updated CTDB's alter table code to take advantage of this new functionality.



6.33 CTDB - Set scanner cache feature on table

We added to CTDB a new function to be able to set/reset the scanner cache feature on a table.

```
/^*****\  
Function:   ctdbSetScannerCache - turn on/off the scanner cache for the table  
Parameters: Handle [IN] - Table handle  
            flag [IN] - YES to turn it on, NO to turn it off  
Returns:    CTDBRET_OK on success or c-tree error code on failure.  
*****/  
CTDBRET ctdbDECL ctdbSetScannerCache(CTHANDLE Handle, CTBOOL flag)
```

6.34 CTDB - ability to set a index as primary key index

We need to be able to identify an index as the primary key index and persist the information. Therefore we added a number of CTDB functions for this purpose.

```
/^*****\  
Function:   ctdbGetIndexPrimaryFlag - Retrieve the flag that indicates this index as primary  
Parameters: Handle [IN] - Index handle  
Returns:    Return the primary flag  
*****/  
CTBOOL ctdbDECL ctdbGetIndexPrimaryFlag(CTHANDLE Handle)  
  
/^*****\  
Function:   ctdbSetIndexPrimaryFlag - Set the primary flag for this index  
Parameters: Handle [IN] - Index handle  
            PrimaryFlag [IN] - Primary index flag  
Returns:    Return CTDBRET_OK on success.  
*****/  
CTDBRET ctdbDECL ctdbSetIndexPrimaryFlag(CTHANDLE Handle, CTBOOL PrimaryFlag)  
  
/^*****\  
Function:   ctdbGetPrimaryKeyIndexNbr - Retrieve the primary key index number  
Parameters: Handle [IN] - Table handle  
Returns:    Return the index number on success or -1 on error.  
*****/  
NINT ctdbDECL ctdbGetPrimaryKeyIndexNbr(CTHANDLE Handle)  
  
/^*****\  
Function:   ctdbGetPrimaryKeyIndex - Retrieve the primary key index handle
```



Parameters: Handle [IN] - Table handle

Returns: Return the index handle on success or NULL on error.

*****/

CTHANDLE ctdbDECL ctdbGetPrimaryKeyIndex(CTHANDLE Handle)

New errors:

```
CTDBRET_PKEXIST = 4188,          /* PrimaryKey Index already exists /
CTDBRET_PKINDEXCLASH = 4189,     / PrimaryKey Index definition and other index settings
clash */
```

Internally we added a new index attribute to keep track of the fact that the index is a primary key index.

6.35 CTDB - SQL import logic improved to take primary key index information into account

Added logic into SQL import to take the primary key index information stored in the data file into consideration. If there is a primary key index signaled in the data file that is imported as primary index.

6.36 CTDB - alternative fixed string padding

We have a customer who padded fixed length string in a different way than c-tree and CTDB. In particular assuming 0x00 is the delimiter and 0x20 is the padding, on a 5 bytes fixed length string set to "A" ctree's and CTDB's buffer after padding is 0x61 20 20 20 00 while the customer's is 0x61 00 20 20 20. The difference is the position of the delimiter, c-tree and CTDB use the delimiter at the end of the buffer, the customer at the end of the significant portion of the string (thus from the 'C' point of view having a variable length string in a fixed size buffer with a maximum length).

We added logic to define the delimiter position for a table which then get stored as a table attribute (schema attribute) so that whenever that table is opened we know how to pad fixed length strings.

We ensured that internal tables (ctdb's dictionaries and SQL system tables) use the "good old c-tree and ctdb" way.

There is logic to retrieve the value stored in the table resource and to use it for the table so that the chance to have mixed strategies on different records is minimal and due only to an explicit setting rather than a silent error.



For the time being the only way to determine the delimiter position is through #defines and it is effective on existing tables with no indication (in which case we use the default) and on newly created tables.

#define CTDB_DEFAULT_DLM_POS_STR_END changes the default from 'good old c-tree' to 'customer' way. This #define is OFF by default.

6.37 Improved Change Batch Performance

When working with multiple c-treeDB batch contexts was found to impact performance when switching between batches. FairCom DB did not have a mechanism to "piggyback" a change batch call automatically along with the batch create call resulting in an additional round trip to the server. A new BAT_EXTENSIONS mode *BAT_EXT_CHANGEBAT* is added enabling such support. When this mode is provided, the *PKEYREQ2* *bavail* member is set to the batch number. This avoids the additional **CHGBAT()** call.

c-treeDB batch APIs have been enhanced to take advantage of this feature from the following functions:

ctdbSetBatchFilter

ctdbSetBatchRangeOn

ctdbAddToFieldMask

Note, this support is on by default in V12.6.0 builds forward.

Other c-treeDB batch enhancements reduce batch time consuming allocations and deallocations, due to recent modifications, when the operation was not always required. In other cases, significant time was observed where client-server round trips took place.

To address these, a new batch mode *CTBATCH_KEEPPBUFFER*, was added. When this mode is OR'd into **ctdbSetBatch()** mode parameter then the code reuses the batch buffer if already allocated and during **ctdbEndBatch()** does not release it. If this mode is not OR'd in than **ctdbSetBatch()** behaves as before and if for any reason there is a batch buffer still allocated it gets freed and reallocated. **ctdbBatchEnd()** was modified to avoid making client-server round-trips to free resources as it is now aware of pending batch states.

6.38 FairCom DB Batch Calls now Return Number of Records Rejected from Active Filter

A customer requested to have for batch retrieval the number of records that were rejected as they do not pass the filter criteria. *BAT_EXT_FLTR_OUT* *batmode2* mode can now be used with BAT_EXTENSIONS. When this is specified the first 4 bytes of the response buffer contain the number of records filtered out by FairCom DB's filter. The value is the number of records filtered from the current returned amount (per batch return, not batch total).

c-treeDB was also updated with a corresponding batch mode *CTBATCH_RET_FILTER*.



6.39 CTDB Function to Retrieve Number of Filtered Records

A new c-treeDB function, **ctdbBatchFiltered()**, to retrieve the number of records filtered out during batch operation has been added. A batch call needs to be started using the mode or the count maintained. This function can be called any time a batch is active and returns the number of records that the server skipped up to that point in the batch retrieval operation.

The function prototype is as described below.

```
/^*****\

Function:    ctdbBatchFiltered
*
*           ctdbBatchFiltered retrieve the number of records rejected so far
*           as not matching active filter by a batch retrieval operation.
*           If a batch operation is not active, ctdbBatchTotal return zero.
*           The value is significant only if the CTBATCH_RET_FILTER mode has
*           been used in ctdbSetBatch() call.
*
Parameters:  Handle [IN]
*           Handle must be a record handle associated with an open table.*
Returns:     Return the number of records rejected so far because not matching
*           active filter by a batch retrieval operation.
*           If a batch operation is not active, ctdbBatchTotal return zero.
*****/
```

6.40 CTDB - new function ctdbBlobReserveSize

CONTEXT: Working on binary fields in JSON DB DETAILS: When using a CTBLOB and allocating it on the heap, the function **ctdbBlobAlloc()** allows specifying a size which is used to allocate the blob buffer. When a CTBLOB is allocated on the stack, Object does not provide any function to allocate the internal buffer. Having the internal buffer allocated with a known size is very useful when the blob content needs to be build “in chunks” like for instance when retrieving it from a stream or, in my case, from a JSON array. In such scenario is possible to get the pointer to the internal buffer (**ctdbBlobGetData**) and fill it directly from the source of content instead of allocation a temporary buffer and once filled use one of the **ctdbBlob** function to copy the content into the internal buffer.

SOLUTION: Therefore we added a new ctdb function to allocate the internal CTBLOB buffer.

```
/^*****\

Function:    ctdbBlobReserveSize - Ensure that the blob is large enough to contain the specified
size
Parameters:  pBlob [IN] - Blob pointer
              size [IN] - size in bytes to allocate for the blob data
Returns:     Return error code
*****/
```



```
CTDBRET ctdbDECL ctdbBlobReserveSize(pCTBLOB pBlob, VRLEN size)
```

6.41 CTDB - Performace - avoid useless SWTCTREE call during ctdbGetFieldAs() and ctdbSetFieldAs()

SYMPTOM: One customer noticed a very large amount of SWTCTREE() calls in their application, in particular when performing ctdbGetFieldAs* operations.

CAUSE: CTDB , when calling ctdbGet*Handle(), makes sure to always call SWTCTREE() function so that it ensure that the active c-tree instance is the one on which the handle retrieved operates. This way the c-tree calls acts in the proper instance. This could be completely avoided if we knew that there is only one c-tree instance defined, but unfortunately this is not easy to track outside c-tree itself. In any case even if there are multiple c-tree instances there are cases where the SWTCTREE() call is useless since no c-tree call is performed. This is very true for the ctdbGetFieldAs() and ctdbSetFieldAs() functions which do not perform any c-tree call.

SOLUTION: We changed the _ctdbGetFieldCheck() and _ctdbSetFieldCheck() functions (that are called by ctdbGetFieldAs and ctdbSetFieldAs to retrieve the various handles) to not call ctdbGetRecordHandle() function. A new internal function _ictdbGetRecordHandle() is now used that allows specifying if the SWTCTREE is required or not. In this way we avoid one SWTCTREE() call every time a field content is retrieved. Worth noting that CTDB used inside the SQL server does not perform any SWTCTREE() calls for we know that only one c-tree instance is present for a SQL thread. This change had an immediate positive affect on performance.

6.42 CTDB - new functions to retrieve primary key candidate index

There are cases (i.e. existing tables) where the primary key for the table has not been defined. In such scenarios CTDB is not able to determine a Primary Key. However we could determine if there is an index that could act as primary key and, if so, consider it as the primary key “candidate”. Therefore we added new function to retrieve a “candidate primary key”. The function return the actual primary key when defined otherwise they attempt to identify an index which features allow to use it as primary key.

Two new functions:

```
/*****\
```

Function: ctdbGetPrimaryKeyCandidateIndexNbr

Retrieve the key index number of an index the is or can act as primary key*

Parameters: Handle [IN] - Table handle



```

Returns:      Return the index number on success or -1 on error.
*****/

NINT ctdbDECL ctdbGetPrimaryKeyCandidateIndexNbr(CTHANDLE Handle)

/^*****\

Function:      ctdbGetPrimaryKeyCandidateIndex
                Retrieve the index handle of an index the is or can act as primary key*

Parameters:    Handle [IN] - Table handle

Returns:      Return the index handle on success or NULL on error.
*****/

CTHANDLE ctdbDECL ctdbGetPrimaryKeyCandidateIndex(CTHANDLE Handle)

```

6.43 ctSQLImportTable() function and ctsqlimp utility support for TLS

SYMPTOM: Reported lack of TLS support in the ctSQLImportTable() function and ctsqlimp.

CAUSE: ctsqlimp is a simple wrapper around the ctSQLImportTable() function which takes care of performing the connection to the server (unless instructed to use the current one). In ctSQLImportTable() there is no code to support TLS connections.

SOLUTION: Added to ctSQLImportTable possibility to setup a TLS connection by specifying in CTSQLIMPOPTS a new member "pTEXT tls_cert".

This 'tls_cert' member can be set to:

- NULL to not use TLS,
- SQLIMP_BASE_TLS to use basic TLS connection with encryption but no certificate checking.
- a string containing the name of a certificate file to use TLS with certificate.

The ctsqlimp utility now has two new command line options:

- --tls - use tls connection with no certificate check (mutually exclusive with --tls_cert)
- --tls_cert cert - use tls connection with certificate 'cert' (mutually exclusive with --tls)



6.44 Record buffer optimization - memset in record handling may have significant performance impact

SYMPTOM: Bad performance after reading large record

CAUSE: CTDB's record buffer does not get reallocated. If a records in a table contain very large data blocks the record buffer grows to it's size. There are 'memset()' calls in the code to clean the record buffer. When the allocated record size gets very large these 'memset()' operations may take a long time.

SOLUTION: Do 'memset()' on record buffer in 3 cases:

1. at buffer allocation time to ensure it "0x00" filled.
2. in `ctdbClearRecord()`.
3. after reading a new record to ensure that the part of the buffer that it is not used is reset

We see no easy and safe way to avoid the memset in case #1 and #2 however for case #3 given that we should never access memory over the established record length, it seems that we could avoid the memset completely, however we historically saw cases where the last field of a variable region was a string (CT_STRING) and was not terminated. CTDB should have code to guarantee that we do not reach over the end of the record when reading this last field, however there may be a case that we did not cover therefore setting the rest of the buffer to the string terminator is a CTDB good safety measure.

Nevertheless it seems reasonable to set only a few bytes so that we keep the safety "just in case" but we avoid large probably useless memset. Therefore:

- a) changed code to not reset the entire used buffer to the string terminator but rather just at maximum `ctdbMAX_REC_PADDING` byte (defined in `ctdbred.c` defaulting to 4).
- b) avoided internal calls to `ctdbClearRecord()` in `ctdbFindRowid` (no other "ctdbFind" functions do that) and in `ctdbNextBatch()` (no other "record movement" function do that).

6.45 CTDB - `ctdbClearRecord()` performance improvement

CONTEXT: Customer noted that the `ctdbClearRecord()` uses a inefficient (to large) record buffer size (`"recbuf_size"`). We addressed `memset()` optimization in our record handling.

DETAILS: Similarly to what the "move record" functions do after reading the record, `ctdbClearRecord()` calls `memset` on the entire record buffer allocated even if it is not entirely used. Previously we changed the `memset()` in the record moving functions gaining significant performance when the record buffer is very large. Now we adjusted `ctdbClearRecrod()` by calculating the size of a clear record and ensuring we call `memset()` just for that size (plus minimal extra padding bytes).

7. CTREE

7.1 Alter Table Compression - modes to copy compression from existing open file and to reset to the default

We resolved an issue with `ctdbAlterTable()` for compressed files. The alter table was losing the original compression characteristics and replacing them with the current setting or the default settings. We needed a way to be able to copy the compression setting from an existing open table.

Now using `ctSETCOMPRESS` :

- The first parameter (comptype) can be set to `ctCMPREC_DEF` to reset the compression to the default.
- The first parameter (comptype) can be set to `ctCMPREC_CPY` to copy the compression settings from another file. All other parameters are ignored.
- The second parameter (compvrsn) needs to be set to either `ctCMPRECusrno` or `ctCMPRECsysno` (server side conde only) to indicate if the file number passed in is a user file number or a system file number (server only).
- The third parameter (def_cmprec_dll) is a string containing the datno from which to copy the setting. Other parameters are ignored.

7.2 Prevent Updates and Deletes for Tables

A new file mode (*ctlInsertOnly*) has been introduced to prevent updates and deletes to data files. This mode also prevents all changes using the low-level WRTREC/WRTVREC. To enable this mode at create time, include the *ctlInsertOnly* bit in *XCREBLK.splval* passed to **CreatelfilXtd8()**.

PUTHDR() has been extended to support the new mode: *ctlInsertOnlyhdr*.

To enable this mode on existing files, call **PUTHDR(datno, YES, ctlInsertOnlyhdr)**. To disable on existing files, call **PUTHDR(datno, NO, ctlInsertOnlyhdr)**.

This requires exclusive access to the file.

An ADMIN (or other user) must call **PUTHDR** to disable *ctlInsertOnly* mode to be able to update the file.

PUTHDR calls for *ctlInsertOnly* can be replicated.



New Table Create Mode

A new Table Create mode has been added at the FairCom DB API level: *CTCREATE_INSERTONLY*. This mode allows tables to be created at the FairCom DB API level with this optional behavior.

7.3 Support adding identity field and auto system time field to partition host that has existing members

SYMPTOM: A call to `ctdbAlterTable()` to make a table partitioned fails with error `PMXS_ERR(948)` if the table has an identity field or an auto system time field.

CAUSE: The add identity field and add auto system time fields intentionally fail if called for a partition host that has existing partition members. The alter table creates the new partitioned table and adds records to it before it adds the identity and auto system time fields to the new table.

SOLUTION: Implement support for adding identity field and auto system time fields to a partitioned file that has existing partition members.

We noticed that the `ctinfo` utility did not display the header value indicating that the file has an identity field, so we added that to the `ctinfo` output.

7.4 Support Added to hot Alter Table for non-Schema-Based key Segment Modes.

With this revision, hot alter table now allows adding fields after all existing fields on tables that use non-schema-based key segment modes. This is allowed because adding new fields after the current fields will not affect any key segment offset calculations.

AFFECTED COMPONENTS: server

7.5 Support new types of null key value checks for c-tree indexes

DETAILS: In order to support a feature that the Btrieve database supports, we implemented in c-tree support for two new null key value checks when adding a key to an index. The two new



modes, which are specified in the index definition structure's (IIDX) `inulkey` field when creating an index file, are:

CT_NULL_KEY_ALL: if the entire key matches the specified null key value, consider the key to be null.

CT_NULL_KEY_ANY: if any segment of the key matches its corresponding null key value, consider the key to be null.

The null key value for an index is set by a call to the new function `ctSetNullKeyValue()`. Its prototype is as follows:

```
ctCONV NINT ctDECL ctSetNullKeyValue(FILNO keyno, cpTEXT nullkey, ULONG keylen);  
keyno is the index file number.  
nullkey is the null key value.  
keylen is the size of the null key value in bytes, which must match the defined key length, including the  
tie-breaking bytes for an index that allows duplicates, although those bytes are not considered in the null  
key comparison.
```

The function returns zero on success or a non-zero c-tree error code on failure.

We also added a function that can be used to read the null key value for an index:

```
ctCONV NINT ctDECL ctGetNullKeyValue(FILNO keyno, pTEXT nullkey, ULONG keylen);  
keyno is the index file number.  
nullkey is the buffer where the key value is returned.  
keylen is the size of the null key buffer in bytes, which must match the defined key length, including the  
tie-breaking bytes for an index that allows duplicates.
```

COMPATIBILITY NOTE: An index file that uses this feature, and its corresponding data file, have a feature bit set in their header so that a server or client or standalone library must support this feature in order to open the files. If not, the file open fails with error 744.

7.6 Record Update Callback - Support record update callback that is called at the start of an add or update operation, allowing the callback to modify the record image

DETAILS: The server's record update callback feature now supports a callback that is called at the start of record add and update operations and that allows the callback to update the record image. By comparison, the record update callbacks that existed prior to this revision are called later in the add and update routines and do not support the callback changing the record image.

To use this feature, when calling `ctRecordUpdateCallbackControl()` to add a record update callback function, set the `calltm` field of the `RUCBACB` structure to `RUCBprerecupd`.

Recall that the prototype for the user-defined record update callback function is as follows:

```
NINT rucbRecordUpdateCallbackFunction(pRUCBF prucbf, pRUCBO prucbo, pRUCBSTT  
prucbstt);
```

We added fields to the `RUCBSTT` structure. When the structure version (indicated by the `version` field value) is `RUCBSTT_VERSION_V02`, the `RUCBSTT` structure includes fields that hold the



record length, record image pointer, and function pointers to the server's memory allocation and free functions:

```
/* State information to pass to record update callback function: */
typedef struct rucbstt_t {
    LONG version; /* structure version */
    VRLEN reclen; /* record length in bytes */
    LONG8 tranno; /* transaction number for the operation */
    pTEXT recbuf; /* buffer containing record image */
    rucbAllocFunction_t allocfn; /* memory allocation function */
    rucbFreeFunction_t freefn; /* memory free function */
} RUCBSTT, *pRUCBSTT;
```

To update the record image in your user-defined record update callback function, if you are increasing the record length, use the memory allocation function to reallocate the record buffer and set `prucbstt->recbuf` to the new buffer and `prucbstt->reclen` to the new record length. If you are not increasing the record length, you can modify the record image in the existing record buffer.

Note that for a fixed length data file or a `ctAugmentedFxd` variable length data file, the record update callback function is not allowed to change the record length. Attempting to do so causes the add or update operation to fail with error `DSIZ_ERR(443)`.

Also, record update callbacks now receive the partition host file name and file number. The record update callback functions were receiving the file name and file number of the partition member. We have changed the behavior so that the host name and file number are passed if available. In our testing, we found that the host info might not be available when closing a partition member because the host might already be closed.

LIMITATIONS: The changes to the record image made by the record update callback function are not visible to the caller of the add or the update operation until the caller re-reads the record.

7.7 User-Defined Hot Alter Table Field Conversion Callback

It was requested to include the ability to define custom field conversions during alter table. It is now possible to include a custom shared library to be used with `ctAlterSchema()`.

This function accepts parameters that use the record descriptor base structure of type `RCDSC`, and an array of record descriptor structure of type `RCSCHD`. To support this field conversion feature, we define a new structure, `RCDSC2`, that contains a field `pTEXT dllname` to hold the field conversion DLL name, and we defined a new structure `RCSCHD2`, that contains a field `pTEXT fname` to hold the field conversion function name.

The data type of the `precpsc` pointer in `ctAlterSchema()` was changed to `pVOID` such that either a pointer to the `RCDSC` structure or the `RCDSC2` structure can be specified:

```
ctCONV NINT ctDECL ctAlterSchema(FILNO datno, pVOID precpsc);
```



To use version 2 of these structures, use the *RCDSC2* and *RCSCHD2* structures and set the *RCDSC2* *rdtype* field to *ctRDSCver2*.

The function prototype for the user-defined field conversion function **convertfield()** is shown here. Note the use of the *pRCSCHDN* data type, which is a pointer to the current version of *RCSCHD* structure. The field conversion function should check that the version value passed to the function matches the version of the *RCSCHD* structure (1 or 2) that the conversion function is using.

```
NINT convertfield(NINT version, pRCSCHDN pschema, ppTEXT pdp, pVLEN premlen, ppTEXT psp, VLEN datlen, TEXT dbyte, TEXT pbyte)
```

[IN] version: Version of pschema structure.

[IN] pschema: Record descriptor for this field.

[IN,OUT] pdp: Destination data pointer.

[IN,OUT] premlen: Remaining length in output buffer.

[IN,OUT] psp: Source data pointer.

[IN] datlen: Number of bytes from source data pointer to end of record image.

[IN] dbyte: String delimiter byte.

[IN] pbyte: String padding byte.

Returns a c-tree error code. NO_ERROR indicates success.

Diagnostics

We added a companion utility function, **ctGetRecordConverters()**, that can be used to read the record converter information that is stored in the data file by a hot alter schema operation. The **ctinfo** utility uses this function to read and display the record converter information.

```
ctCONV NINT ctDECL ctGetRecordConverters(FILNO datno, pTEXT buffer, pVLEN pbuflen);
```

[IN] datno- Data file number.

[OUT] buffer- Output buffer.

[IN/OUT] pbuflen- On input, holds the size of the output buffer in bytes. If the output buffer is too small to hold the output data, the function returns VBSZ_ERR and sets *pbuflen to the required buffer size in bytes. On success, holds the size of the data that was written to the output buffer.

ctGetRecordConverters() reads the record converters from the specified data file. The record converters are written to the output buffer as a null-terminated string in JSON format.

Returns a c-tree error code. NO_ERROR(0) indicates success.

Example code for a field conversion function can be found in

ctree\source\ctrucbdl.c in the function **qaflexrec015_convertfield()**.

Compatibility

Our implementation approach for version 2 of the hot alter feature allows existing code to operate without any changes. If a new client attempts to use the new feature with an old server, its function call fails with error **FVER_ERR** (43). Even if a call to **ctAlterSchema()** specifies version



2 of the hot alter structures, the function call data is sent to the server in version 1 format unless the new structure fields are used, and the record converter is stored in version 1 format in the data file. This makes it possible to preserve backward compatibility of a v2 client with v1 servers.

The server writes a version 1 format hot alter schema replication transaction log entry unless the new structure field are used, in order to preserve backward compatibility with a v1 replication agent and v1 target server.

7.8 CTREE - Implement change id field to support optimistic locking

We now support defining a field that is automatically assigned a value when the record is added or updated in a transaction. We call this type of field a change id field. The field is required to be defined as a CT_INT8U data type with field length of 8.

When a record is added or updated, the change id field is set to the current transaction number. For a file that is not under transaction control, it's possible to use the change id feature by using ctPREIMG transactions when adding and updating records.

When a record is updated, the change id in the record image that is passed in is compared to the current change id of the record in the data file. If the values match, the update sets the change id to the current transaction number. If they differ, the update fails with error CHANGEID_MISMATCH_ERR (1192).

New functions:

```
/*
Add a change ID field to the specified data file.
[IN] datno- Data file number.
[IN] fieldno- The field number in the DODA for the change ID field.
Returns NO_ERROR on success, or non-zero on failure.
*/
ctCONV NINT ctDECL addChangeIDfield (FILNO datno, NINT fieldno);

/*
Get the relative field number of the specified data file's change id field.
[IN] datno- Data file number.
[OUT] pFieldNumber- On success, the zero-based field number in the DODA for
the change ID field is returned here. If the data file has no change id
field, -1 is returned.
Returns NO_ERROR on success, or non-zero on failure.
*/
ctCONV NINT ctDECL getChangeIDfieldNumber (FILNO datno, pNINT pFieldNumber);

/*
Get the value of the specified data file's change id header field.
[IN] datno- Data file number.
[OUT] pChangeIdValue- On success, the change id header value is returned here.
```



```
Returns NO_ERROR on success, or non-zero on failure.
*/
ctCONV NINT ctDECL getChangeIDheaderValue(FILNO datno, ctCHGID *pChangeIDValue);
/*
Delete the change ID field from the specified data file.
[IN] datno- Data file number.
Returns NO_ERROR on success, or non-zero on failure.
*/
ctCONV NINT ctDECL delChangeIDfield (FILNO datno);
```

New error codes:

- CHANGEID_NOTSUPPLIED_ERR = 1188: Record update failed because changeid was not supplied by the caller.
- CHANGEID_NOTREAD_ERR = 1189: Record update failed because the changeid value was not read at the time the record was read. One possible cause is that the record was read without locking it.
- CHANGEID_ALREADY_EXISTS_ERR = 1190: The specified data file already has a change id field.
- CHANGEID_DOES_NOT_EXIST_ERR = 1191: The specified data file does not have a change id field.
- CHANGEID_MISMATCH_ERR = 1192: The change id value passed to an update operation does not match the record's current change id value.
- CHANGEID_TOO_LOW_ERR = 1193: The generated change id value is less than the last change id assigned to a record in the data file. This can be caused by deleting the transaction logs or copying the file to another server, then opening the file within an active transaction.

LIMITATIONS: Note that this feature is only supported in client/server mode for the following reasons:

- The single user standalone model doesn't support this feature because it only supports a single connection accessing a table at one time.
- The multi user standalone model doesn't support this feature because it doesn't support transaction control, and this feature uses the current transaction number as the change id value.

7.9 File Block Enhancement: Support a file block mode that allows active transactions time to complete before blocking the file

The file block function immediately blocks a file. However, we would like to support a file block that gives pending transactions time to complete before the file is blocked. In order to support this feature, we introduced an extended version of the file block function that supports specifying a transaction timeout in seconds. When a non-zero transaction timeout is specified, the server



optionally blocks new transactions and then waits for all active transactions to complete for those connections that have the file to be blocked open, then unblocks new transactions once the startup phase of the file block has been set up.

The function prototype for the new file block function is:

```
ctCONV NINT ctDECL ctFILBLKX (const ctFILE_BLOCK_XTD_OPTIONS
*pctFileBlockXtdOptions);
typedef struct ctFileBlockXtdOptions_t {
    UCOUNT structVersion; /* [IN] version of this structure */
    UCOUNT tranTimeoutSec; /* [IN] optional transaction timeout in seconds */
    ULONG    action; /* [IN] file block action bits */
    cpTEXT   fileName; /* [IN] name of file to block or unblock */
} ctFILE_BLOCK_XTD_OPTIONS;
```

To set a transaction timeout, set the tranTimeoutSec field of the ctFILE_BLOCK_XTD_OPTIONS structure to a non-zero timeout value in seconds. A tranTimeoutSec value of zero indicates no transaction timeout.

To block new transactions at the start of the file block, OR in the ctFBblockNewTransactions mode bit to the action field of the ctFILE_BLOCK_XTD_OPTIONS structure.

Here is an example showing blocking a file with a 10 second transaction timeout and blocking new transaction begins until the file block startup has completed:

```
ctFILE_BLOCK_XTD_OPTIONS fileBlockXtdOptions = { 0 };
NINT ret;
fileBlockXtdOptions.structVersion = ctFILE_BLOCK_XTD_OPTIONS_VERS_V01;
fileBlockXtdOptions.fileName = "myfile.dat";
fileBlockXtdOptions.action = ctFBblock | ctFBisam | ctFBblockNewTransactions;
fileBlockXtdOptions.tranTimeoutSec = 10;
ret = ctFILBLKX(&fileBlockXtdOptions);
```

We added option -t tranTimeoutSec to the ctfilblkif utility and option -n to block new transactions at the start of the file block. Example:

Block file mark.dat, blocking new transactions and allowing active transactions 3 seconds to complete before blocking the file:

```
ctfilblkif -f mark.dat -n -t 3
```

7.10 Support adding a condition to an index when creating the index with the data file open in shared mode

An index can be created with the data file open in shared mode. Now we also support adding a condition to the index when creating the index with the data file open in shared mode. In order to use this feature, call the new c-tree API function PRMIIDX82().



PRMIIDX82() is used to add a permanent index to a data file. The function accepts ifilptr and pxcreblk parameters that are identical in usage to PRMIIDX8(). The two additional parameters indicate the number of extended create blocks in the pxcreblk array and provide the desired index attributes:

```
extern ctCONV CTERR ctDECL PRMIIDX82(pIFIL ifilptr, pXCREblk pxcreblk, ULONG  
numberOfExtendedCreateBlocks, ctINDEX_ATTRIBUTES *pIndexAttributes);
```

The ctINDEX_ATTRIBUTES structure contains a version so that it can be modified in future versions to support passing additional index attributes. Version 1 of the structure is defined as follows:

```
/* index attributes for PRMIIDX82 */  
typedef struct ctIndexAttributes_t {  
    ULONG        structureVersion;  
    ULONG        numberOfExtendedKeySegments;  
    ULONG        numberOfIndexConditions;  
    pctKSEGDEF   *extendedKeySegments;  
    cpTEXT       *indexConditions;  
} ctINDEX_ATTRIBUTES;
```

To use the index attributes structure, set structureVersion to ctINDEX_ATTRIBUTES_VERS_V01, set numberOfExtendedKeySegments to the number of extended key segments in the extendedKeySegments array (specify zero and NULL if no extended key segments are supplied), and set numberOfIndexConditions to the number of index conditions in the indexConditions array (specify zero and NULL if no index conditions are specified).

- extendedKeySegments is an array of pointers to key segment definitions, one for each key segment definition (ISEG structure element) specified in the array of IIDX structures specified in the ifilptr parameter. If a given key segment has no extended key segment definition, specify NULL for that array element.
- indexConditions is an array of pointers to index conditions, one for each index definition (IIDX structure element) specified in the ifilptr parameter. If a given index has no index condition, specify NULL for that array element.

7.11 Support changing a file's password or permission mask when the file is open in shared mode

Prior to this revision, attempting to change a file's password or permission mask with the file open shared failed with error 62 (LERR_ERR). Now the server permits this.

Note: All connections that have the file already open are unaffected by the changes. But after the security attributes have been changed, subsequent file open requests are subject to the modified security attributes.



7.12 CTREE - New function to generate a sequence name that does not conflict with existing sequences

CONTEXT: Working on enhancing sequence support in SQL

SYMPTOM: On a SQL server with more than one database it was not possible to create in two different database with the same name by the same user. At SQL level the sequence name is unique per database, user, name while at c-tree level is unique server wide.

DETAILS: c-tree has sequence support, wherein each sequence needs to have a unique name of 32 chars (plus terminator). In SQL we allow sequence to be unique per database and per user, moreover their name can be up to 64 chars. So at c-tree level we have 32 chars. In SQL, to have a unique sequence name server wide we would need to “glue” the database name, the user name and the sequence name. This definitely larger than the 32 chars we have in c-tree. One possibility to solve this problem was to change the sequence logic to allow larger names, which has backward compatibility concerns. Another idea one was to have a sequence unique name generator at c-tree level and have a way in SQL to track the “c-tree unique sequence name” assigned to the sequence in the SQL system table. We opted for this solution. Therefore we added to c-tree a new function.

Function Name: `ctNextSequenceUniqueName`

Purpose: returns the nextavailable to be used UNIQUE

Parameters:

`buffer` buffer that will be populated with the name of the last sequence.

`bufsiz` buffer size

Return values:

`NO_ERROR` Success.

`VBSIZ_ERR` buffer too small. `bufsiz` contains the minimum required size.

`ctCONV NINT ctDECL ctNextSequenceUniqueName(pTEXT buffer, pVLEN bufsiz)`

This function determines the next available unique sequence name of the form “\$\$seq_XXXXXXXXXX” where each ‘X’ is a digit and returns it. Multiple calls to `ctNextSequenceUniqueName` would return the same name until a new sequence with that name (or one of the same form) gets created.

7.13 Partition: support for non-schema key segment modes (REGSEG, INTSEG, etc.)

A file partition conditional expression rule can refer to index key fields explicitly by DODA names only if their key segment mode is such so that the field position is interpreted as field ordinal number (SCHSEG, USCHSEG, etc.). The segment modes such as REGSEG, INTSEG, etc. that interpret the field position as field offset from beginning of record buffer were not supported until now.



We added support for segment modes that interpret the field position as field offset from beginning of record buffer. Those are REGSEG, INTSEG, UREGSEG, SRLSEG, SGNSEG, FLTSEG, DECSEG, BCDSEG.

7.14 CTREE/CTDB - support for unixtime stamp data type (time_t)

We now implemented support for unix time stamp data format by adding two new data types:

CT_UNIXTIME_T (32 bit time_t)

CT_UNIXTIME64_T (64 bit time_t)

Efforts included:

- Implemented indexing support, both types require a SNGSEG mode or SCHSEG.
- Implemented support in expression parsing (conditional indices, filters, partitioned files...)
- CTDB, SQL, c-tree core when need to interpret the time_t value perform conversions to GMT (no local time support).
- Implemented functions in c-tree and in CTDB to handle the new types. Updated ctdbGetFieldAs* and ctdbGetFieldAs* to deal with the new types.
- Mapped to NAV_TIMESTAMP (for all APIs using NAV types, like JSON DB, CTDB with NAV...) and in SQL to timestamp.

The following functions have been added to the CTDB API.

```
/^*****\nFunction:    ctdbUnixTimeToDateTime\n            convert a unixtime into a CTDATETIME type value.*\n\nParameters: pDateTime [OUT] - CTDATETIME value\n            unixtime [IN] - time_t\n            tolocaltime [IN] - convert to localtime before packing\nReturns:    Return CTDBRET_OK on success\n*****/\nCTDBRET ctdbDECL ctdbUnixTimeToDateTime(pCTDATETIME pDateTime, time_t time,\nCTBOOL tolocaltime)\n\n/^*****\nFunction:    ctdbDateTimeToUnixTime\n            convert a CTDATETIME type value into a time_t unixtime value.*\n\nParameters: DateTime [IN] - CTDATETIME value
```



```

        unixtime [OUT] - time_t
        tolocaltime [IN] - convert from localtime
    Returns:    Return CTDBRET_OK on success
    *****/
CTDBRET ctdbDECL ctdbDateTimeToUnixTime(CTDATETIME DateTime, time_t* time, CTBOOL
fromlocaltime)

/^*****\
    Function:    ctdbUnixTimeGetDate
                Retrieve a CTDATE type value from a unix time type value*

    Parameters: DateTime [IN] - unix time time_t type value
                pDate [OUT] - Packed CTDATE type value
    Returns:    Return CTDBRET_OK on success
    *****/
CTDBRET ctdbDECL ctdbUnixTimeGetDate(time_t DateTime, pCTDATE pDate)

/^*****\
    Function:    ctdbUnixTimeGetTime
                Retrieve a CTTIME type value from a unix time type value*

    Parameters: DateTime [IN] - unix time time_t type value
                pTime [OUT] - Packed TIME type value
    Returns:    Return CTDBRET_OK on success
    *****/
CTDBRET ctdbDECL ctdbUnixTimeGetTime(time_t DateTime, pCTTIME pTime)

/^*****\
    Function:    ctdbUnixTimeToString
                Convert a unix time_t type value to string*

    Parameters: time [IN] - unix time_t type value
                DateType [IN] - One of the the date types
                TimeType [IN] - One of the time types
                pStr [OUT] - Pointer to a string buffer to received the converted date and time
                size [IN] - size in bytes of the memory area pointed by pStr
    Returns:    Return CTDBRET_OK on success
    *****/
CTDBRET ctdbDECL ctdbUnixTimeToString(time_t time, CTDATE_TYPE DateType,
CTTIME_TYPE TimeType, pTEXT pStr, VRLEN size)

/^*****\
    Function:    ctdbStringToUnixTime
                Convert a date and time in a string to a packed CTDATETIME type value

```



```

Parameters: pStr [IN] - String containint the date and time
            unixtime [OUT] - Pointer to a time_t type value to received the converted string
date and time
            DateType [IN] - One of the date types
            TimeType [IN] - One of the time types
Returns:    Return CTDBRET_OK on success
*****/
CTDBRET ctdbDECL ctdbStringToUnixTime(cpTEXT pStr, time_t* unixtime, CTDATE_TYPE
DateType, CTTIME_TYPE TimeType)

```

7.15 Retry opening transaction log when updating deferred index log entry status

DETAILS: We added a retry of the transaction log open call in the function that updates the status of the first deferred index log entry in a transaction when the transaction commits. A customer reported a case where the file open call failed with a sharing violation error on Windows. We added a retry up to 10 times, with 500 millisecond defer between retries, and we create a process stack dump if we exhaust the retries. We also added a log message in this case.

7.16 When changing security attributes for a partition host file, change the attributes for all existing partition members

A newly created partition member file inherits the security attributes of the partition host file. But if the SECURITY() function is used to change the security attributes of a partition host file, the existing members' security attributes are not updated.

We have changed the SECURITY() function behavior for a partitioned host file. Now a call to SECURITY() for a partition host calls SECURITY() for each existing partition member file so that the changes to the partition host's security attributes are applied to all the existing member files.

If a SECURITY() call for one of the partition member files fails, the operation terminates and returns the error code. In this situation, if the partition host file uses transaction control, all the security attribute changes that had been made up to that point in that call to SECURITY() are undone. If the partition host file does not use transaction control, all security changes that have been applied to the host and members remain in effect. To restore consistency, another SECURITY() call can be made after resolving the issue that caused the SECURITY() call to fail.



7.17 Read Permissions - Server now checks for read permission on function calls that read data or key values

We have added the ability to prevent clients from reading records from a table. The server allowed read operations on the table even if the file permissions did not grant read permission. We changed the server's behavior to deny read access if the file permission does not grant read access to the caller.

This support was added to support a dbNotify client's ability to add data to a FairCom MQ broker, yet restrict that data from being read.

7.18 NULL Key Support - Improve indexing of null values

Our ISAM level index support had the following limitations involving indexing of NULL values:

- we did not provide a way to indicate which key segments are NULL, and
- we did not support enforcing uniqueness constraints on an index while allowing multiple key values to exist if they consist of any NULL key segments or all NULL key segments.

These limitations cause some SQL operations to return incorrect results or to return an error when an index is created on a column that contains NULL values.

We resolved these limitations by introducing a key segment mode that references a NULL field bitmask value in the record image in order to provide the NULL indicator value for a key segment, and two key segment modes that can optionally be include in the index definition to either allow duplicate key values when all of the indexed fields are NULL, or when any of the indexed fields are NULL.

The new key segment modes are as follows:

- **BITSEG**: A bitmask segment. To define a key segment that uses this mode, set the ISEG structure's smode to BITSEG, the soffset to the field number of the field that holds the NULL bitmask value in the record image, and slength to the bit number in the bitmask of the field whose NULL bit is referenced. In the index definition a BITSEG key segment adds one byte to the key length. The BITSEG value is set to 0 if the value is NULL and 0xff if the value is not NULL, so that the NULL values collate before non-NULL values.
- **ALLNULLDUPSEG**: Allow duplicate key values when all of the indexed fields are NULL. To define a key segment that uses this mode, set the ISEG structure's smode to ALLNULLDUPSEG, soffset to 0 (this value is ignored), and slength to 8 if the file is a huge file or a partitioned file, or to 4 otherwise.
- **ANYNULLDUPSEG**: Allow duplicate key values when any of the indexed fields are NULL. To define a key segment that uses this mode, set the ISEG structure's smode to ANYNULLDUPSEG, soffset to 0 (this value is ignored), and slength to 8 if the file is a huge file or a partitioned file, or to 4 otherwise.

We added new API functions `ctGetNullBit()` and `ctSetNullBit()`:



```
/*^*****^
```

Function: ctGetNullBit - Read the state of the specified bit in the specified bit array.

Parameters: pArray [IN] - The bit array to read

BitNumber [IN] - Zero-based bit number to read

Returns: Return YES if bit is set or NO if bit is cleared.

```
*/
```

```
ctCONV TEXT ctGetNullBit(const UTEXT* pArray, NINT BitNumber);
```

```
/*^*****^
```

Function: ctSetNullBit - Set or clear the specified bit in the specified bit array.

Parameters: pArray [IN/OUT] - The bit array to update

BitNumber [IN] - Zero-based bit number to set or clear

SetBit [IN] - If non-zero, set the bit; if zero, clear the bit

Returns: none

```
*/
```

```
ctCONV void ctSetNullBit(pUTEXT pArray, NINT BitNumber, TEXT flag);
```

We added new key segment modes BITSEG, ALLNULLDUPSEG, and ANYNULLDUPSEG.

Mode value | Symbolic constant | Segment Position interpretation | Explanation

258 | BITSEG | Schema field number of bitmask field | bitmask segment: set soffset to schema field number of null bit mask; set slength to schema field number of target field

259 | ALLNULLDUPSEG | ignored | allow duplicate key values when all of the indexed fields are NULL: set soffset to zero (it is ignored); set slength to size of record offset (4 or 8)

260 | ANYNULLDUPSEG | ignored | allow duplicate key values when any of the indexed fields are NULL: set soffset to zero (it is ignored); set slength to size of record offset (4 or 8)

New c-treeDB key segment modes CTSEG_BITSEG, CTSEG_ALLNULLDUPSEG, and CTSEG_ANYNULLDUPSEG.

c-treeDB API Segment Modes | c-treeDB API .NET Segment Modes | Explanation

CTSEG_BITSEG | (none) | bitmask segment: soffset holds field number of null bit mask; slength holds field number of target field

CTSEG_ALLNULLDUPSEG | (none) | allow duplicate key values when all of the indexed fields are NULL

CTSEG_ANYNULLDUPSEG | (none) | allow duplicate key values when any of the indexed fields are NULL

7.19 ctinfo: Retrieve Assigned OS System File IDs

The **ctinfo** utility has been enhanced to output the DEVID and INODEID values of the system file ID stored in each c-tree file. The new output appears as:



```
System id (device) = 0xc28befb3
```

System id (inode) = 0x34900000000060e

GetCtFileInfo() has been enhanced to support two additional mode parameters:

- *DEVID* - Returns a 4-byte integer holding the device ID component of the system file ID.
- *INODEID* - Returns a 4-byte integer holding the low-order 4 bytes of the 8 byte *inode* component of the system file ID. Use **ctGETHGH()** to retrieve the high-order 4 bytes.

8. ISAM

8.1 Return Server's Read-Only Mode with GETNAM()

A mode has been added to the **GETNAM()** function to return the server's read-only attribute. To use this mode, call **GETNAM()** with filno -1 (it is ignored), buffer, size of buffer, and mode of READONLYSERVER.

On success, a value of "y" is written to the buffer to indicate that the server is in read-only mode or a value of "n" is written to the buffer to indicate that the server is in read/write mode.

A server that doesn't support this mode returns error code **IMOD_ERR** (116).

Example:

```
{code}
NINT    rc;
TEXT    buf[2];

if (!(rc = GETNAM(-1, buf, sizeof(buf), READONLYSERVER))) {
    if (!strcmp(buf, "y")) {
        printf("server is read only\n");
    } else {
        printf("server allows writes\n");
    }
} else {
    printf("Error: Failed to get server read only attribute: %d\n", rc);
}
{code}
```

Affected Components: Client and server

8.2 Automatic Temporary Directory Cleanup After Crash

The **ctTempDir()** API call was extended to support automatic removal of temporary directories. Directories may now include the new option *ctTDRautodelete* at create time: (*ctTDRcreate* | *ctTDRautodelete*). At next server startup, these directories are deleted if they haven't been explicitly removed. This ensures eventual cleanup in the event of a crash, unless *TMPDIRDT.FCS* is deleted.



A number of existing bugs in this area were also cleaned up during testing.

- *ctTDRdelete* may fail due to differences in path separators and paths are now correctly formed. .
- *errorbuffer* was not returned to the client if a callback was not defined due to an incorrect parameter length and was corrected..
- If deleting a directory listed in *TMPDIRDT.FCS* fails because it does not exist on disk, we now remove the entry from *TMPDIRDT.FCS*
- Calls to *ctTempDir()* opened *TMPDIRDT.FCS* without closing it, consuming three FairCom DB file handles until disconnected. This file is now closed after each access.
- Existing legacy *TMPDIRDT.FCS* versions had binary data at record offset zero, leading to insertion errors. A \$DELFLD\$ field was added to improve this

8.3 Improve ctQuiet Timeout

When a FairCom DB quiesce (*ctQUIET()* call) is called, it can eventually fail with error 843 as it was unable to block all thread activity. There is a fixed number of internal attempts, however, it may take a very long time (5+ minutes) to return the error before it gives up with a large number of connections. During this waiting period, all server activity is blocked.

CTSTATUS.FCS message

```
"ctQUIET Note: no progress clearing threads from core. Abort block attempt.: 843"
```

An internal loop must acquire and release mutexes for each user, and if a large number of users are active, this may become the primary source of delay. To alleviate this point of delay, a server configuration option has been added specifying a wait interval and this value is checked on each loop until the maximum value is encountered.

QUIET_MAX_WAIT <seconds>

When exceeded, *ctQUIET* fails with error 843. The default value is 60 seconds with a minimum value of 5 seconds.

Note: This value only applies after any initial wait value specified in the originating *ctQUIET()* API call

Diagnostic

A new diagnostics configuration was also added to aid in debugging log timeouts

DIAGNOSTICS CTQUIET

When enabled, this diagnostic creates a stack dump when a *ctQUIET()* fails with 843. This information can help FairCom support determine why a thread is not exiting the core as expected causing the long delay window.

9. StandAlone

We continue to support our StandAlone model, wherein our database technology can be linked directly into the application (Embedded). In fact, we believe we offer the most extensive set of database capabilities on the market in this unique embedded model.

9.1 StandAlone Full Source Code Package - Porting to alternative platforms made easier

Available on request, we now offer a complete source code package for our StandAlone technology. This give users the freedom to port to alternative environments as needed. This technology, of course, is subject to our distribution license requirements. Please contact FairCom if you are interested in this flexibility.

9.2 FairCom SQL Service - Simpler StandAlone SDK Packages

Because our "non-Client-Server" embeddable model is intended to be linked directly into applications, we refer to this as our StandAlone model.

Offered with full source code, we have added a specific packages to our lineup for your convenience: FairCom-DB-StandAloneSDK.

FQL Server Available: One of our most popular StandAlone models has been our Multi-User implementation known as FPUTFGET. This non-client-server model supports simultaneous applications to maintain (read/write) the same set of files on disk. Although caching and transaction processing are not supported, many users have enjoyed this model for years.

We would like to remind these users of our SQL/ODBC support for this model. Named after our term FPUTFGET, we call this our FQL Server. This process acts like any other multi-users application by (no-caching) "force-write" "force-read" data to/from disk. In fact, you have a complete SQL server using your preferred underlying non-client-server data model. Take a look!



9.3 Performance enhancement: Turn on log writethru for Single-User TRANPROC library

We now turn on the log writethru option in the single-user tranproc library when the operating system supports opening a file in a mode that writes through the file system cache. This change significantly speeds up transaction processing performance. If you want to disable the log writethru option, add `#define NO_ctBEHAV_LOG_WRITETHRU` to `ctoptn.h` and recompile the library.

9.4 Updated Standalone Rebuild and Compact Utilities for Data Encryption

V12 introduced data encryption to standalone FairCom DB usage. However, the compact and rebuild utilities were not updated with required support. This support is now added to these important utilities.

Advanced encryption requires a master key to be supplied. Standalone utilities prompt for the master key, or a master key file can be created, they will optionally use it. To create a master key file, follow the steps below.

1. Create a master key file, following the prompts::
`ctcpvf -k mykey -s mykey.fkf`
2. Set this environment variable to the name of your master key file:
`CTREE_MASTER_KEY_FILE=mykey.fkf`

The environment variable must be present in the process environment of your FairCom DB application. If using these steps, then the compact and rebuild utilities reads the master key from the file instead of prompting for the key.

9.5 Partitioned file support in multi-user StandAlone operational model

We enabled partitioned file support in the multi-user standalone library, both the single-threaded and the multi-threaded standalone models.

A partitioned file logically appears to be one file (or more accurately one data file and its associated index files), but is actually a set of files whose contents are partitioned by the value of the partition key. Both the data files and index files are partitioned. This permits data with a defined range of values for the partition key to be rapidly purged or archived (instead of having to delete record-by-record each record within this range).

A partitioned file is comprised of a host data file and its associated indexes combined with a rule. The rule determines how to map the partition key value (e.g., a date, or invoice number, or some



other characteristic) into a particular partition member of the host. The host file does not contain any actual data, but serves as a logical file that appears to contain all the data.

A partition member file has the same definition as the host, but only holds data whose partition key maps to the member. For example, customer orders may be partitioned by calendar quarter. All orders booked in the same quarter are stored in the same member. A member is comprised of a standard FairCom DB data file and its associated indexes.

To add data to the partitioned file, simply call an add routine for the host. The code will add the record in the proper partition member, creating the partition member if necessary. Rewriting a data record may move the record between partitions if the partition key entry for the record is changed. Under transaction control, such a delete/add record operation is done atomically.

9.6 Change FPUTFGET model's ctREADFIL and ctSHARED open to a shared read-only open

SYMPTOM: In v11 on Unix systems, in the FPUTFGET operational model, if a process has a c-tree file open in ctSHARED mode, another process attempting to open the file in ctREADFIL | ctSHARED mode fails with error 920, and vice-versa.

CAUSE: This is by design, because in FPUTFGET mode the ctREADFIL | ctSHARED mode is converted to ctREADFIL by removing ctSHARED from the file mode, and ctREADFIL opens are not compatible with the ctSHARED filemode, because ctREADFIL caches index node reads, so the FPUTFGET c-tree library does not permit other opens with write access when the file is open with ctREADFIL.

SOLUTION: We introduced a compile-time option, `#define ctBEHAV_FPG_SHARED_READFIL`, which can be specified in `ctoptn.h` when compiling the FPUTFGET library. This option causes a ctREADFIL | ctSHARED open to be treated as a shared open with read-only access. Note that this means that since other processes are allowed to open the file for write access, this type of shared read-only open does not enable optimizations of caching reads of index nodes. Note that this option is off at compile time by default.

Here we list the compatible file open modes, where:

- `r` is ctREADFIL
- `rs` is ctREADFIL | ctSHARED
- `s` is ctSHARED
- `x` is ctEXCLUSIVE.

Original behavior:

- `r` and `rs` are compatible
- `s` is compatible with `s`
- `x` is compatible with none

New behavior:

- `r` is compatible with `r`



- rs and s are compatible
- x is compatible with none

9.7 Fixed index file error 14 in FPUTFGET after index update in exclusive writethru mode

A problem was seen in the multi-user standalone (FPUTFGET) operational model when an index file was opened in exclusive mode with write-through mode (ctWRITETHRU) and was updated. When the index file was closed, the following message was logged to *CTSTATUS.FCS* and the file was marked corrupt:

```
Error on close file. locale:13 sysiocod:0 uerr_cod: 527
<indexname>
```

An FPUTFGET optimization introduced in V11.2.1 caused an unexpected condition in certain situations involving an exclusive ctWRITETHRU open. The logic has been modified to correct this.

Affected Components: Multi-user Standalone (FPUTFGET) library, V11.2.1 and later.

9.8 Compact Error RCPT_ERR (1156) Fixed in Multi-user Standalone Models

File compact failed with **RCPT_ERR** (1156) using the FPUTFGET library. The logic has been modified to correct this.

9.9 StandAlone batch calls with locks fail with LEOF_ERR(30)

SYMPTOM: Batch call with record locks fails with LEOF_ERR(30) using non-server model.

CAUSE: In ifrebat(), the wrong sysno is passed to iLOKREC to free the record locks. If the batch was conditioned on an index member the LEOF test finds the lock offset is greater than -1 (numrec for an index member).

SOLUTION: Adjusted logic so the correct sysno value is used.



9.10 ctREADFIL may be ignored by FPUTFGET

SYMPTOM: An update may be allowed to a file opened with ctREADFIL using a library compiled with `#define FPUTFGET`.

CAUSE: In FPUTFGET, the ctREADFIL state is stored in CTFIL.fmode. A call to redhdr() may overwrite the file mode used at open time.

SOLUTION: At open, we already saved the ctPERMANENT file mode bit in a separate variable (CTFIL.savprm). During redhdr(), we use this variable to preserve this bit state in fmode. This revision extends this to include some other fmode bits set at open time: ctREADFIL | ctSHARED. This is controlled by `#define ctBEHAV_FPG_OPENMODES`, which is enabled by default.

9.11 File Close Silently Failing

SYMPTOM: In the multi-threaded FPUTFGET operational model on Unix, a file may unexpectedly remain open after a file close call.

CAUSE: In the multi-threaded FPUTFGET model on Unix systems, we keep a file descriptor open until all references to the file at the c-tree level have been closed. In order to track open references to a file, we put entries into an in-memory list, with a reference count on each entry. However, the function that removes an entry from the list had a bug: if removing the last entry from the list, the list tail pointer was always set to NULL instead of being set to the previous entry in the list. The NULL tail pointer caused the list add function to believe the list was empty, and so the next time an entry was added to the list, the head pointer was set to point to the new entry and the existing entries were no longer part of the list. As a result, a subsequent search for the file in the list did not find it, causing the closing of the file to not find the entry and the error caused the file to be kept open.

SOLUTION: Correct the setting of the tail pointer when deleting an entry from the list. Therefore this bug has since been corrected, and files are now properly closed after a file close call.

AFFECTED COMPONENTS: Standalone Multi-threaded multi-user library on Unix platforms

10. Utilities

10.1 Dynamic Dump - Online Backups

Improved Dynamic Dump Performance

When a dynamic dump was run and the client requested receiving dump status messages, the performance of the dump may have been slow, especially if the client was on a different system than the server (even if the client requested using a large buffer size for the streaming of the dump from the server).

This drop in performance was due to a large amount of network calls between the server and the client. The server's logic has since been updated to process these calls more efficiently, preventing performance from dropping.

Hot Backup Without Quiet Checkpoint Requirement

Dynamic Dumps (Hot Backups) have required stopping all new transactions until outstanding transactions complete, allowing a checkpoint with no active transactions to be written to the transaction log. But this could cause an arbitrarily long delay if large transactions are active. This quiet checkpoint was required for a forward roll to be able to have a valid starting point. This revision adds support for creating dynamic dumps without a quiet checkpoint requirement. Prior to this revision,

To request a backup without stopping all new transactions, add the new keyword `!ALLOW_TRAN` to your dynamic dump script. This eliminates the delay caused by waiting for active transactions to complete, but causes the backup to be larger and additional processing to occur when restoring from the backup. This added overhead is proportional to the size of the active transactions at the time of the backup.

`!ALLOW_TRAN` eliminates the need for a quiet checkpoint during dynamic dump, unless server configuration option `PREIMAGE_DUMP` is active. When `PREIMAGE_DUMP` is active, `!ALLOW_TRAN` is ignored. When `!ALLOW_TRAN` is in effect, it modifies the behavior of the existing `!DELAY <interval>` dynamic dump option: if `!DELAY` is non-zero, block new transactions for up to `<interval>` seconds, waiting for a quiet transaction state.

If a quiet transaction state is reached, we create a regular backup. If `<interval>` expires and active transactions remain, continue the backup with active transactions. This allows the use of a short delay to possibly get a smaller and faster restoring backup if large transactions are unusual.

When a backup created with `!ALLOW_TRAN` is restored, any changes made by transactions that were active at the beginning point of the backup will be rolled back. A forward roll that begins from such a backup will apply these earlier transactions.



Redirect All Restored Files to a Specified Path

The `!REDIRECT` option used in a dynamic dump restore script configuration file now supports an option to redirect all directory names to the specified directory name. To use this option, specify `<entirepath>` as the first path name in the `!REDIRECT` option, and specify the desired new directory name for the second path name. The new directory name can be a full or a relative path.

Example, using this option in a dynamic dump restore script will cause all files to be restored to the directory `C:myrestoredir`:

```
!REDIRECT <entirepath> C:myrestoredir
```

ctfdmp and ctldmp no longer fail with 1166 (ENCL_ERR) on non-encrypted logs

ctfdmp failed with the message: "FWD: Roll-Forward Error Termination...1166". **ctldmp** failed to run on non-encrypted logs. The logic has been corrected to read the encryption settings from the log.

Dynamic dump error message improvement

We improved in the error message from the dynamic dump execution in our internal function `_isamDumpFiles()` in order to be able to distinguish if the error is returned on the `dyndmp()` call or the in callback thread checking its progress status.

Dynamic Dump Hangs

Previously, the Dynamic Dump hung without creating a backup. Additionally, `CTSTATUS.FCS` contained one of the following messages for a prior dump:

1. "DD: could not process script file: -1"
2. "could not get next available file number"

The causes of the above messages are detailed below:

1. When a dynamic dump is begun, the dump script is opened and parsed in two phases. If this second open of the dump script failed, or an error occurred during the second script parsing, the backup would fail without releasing a mutex.
2. If all of the file handles specified by the `FILES` keyword (as specified in `ctsrvr.cfg`) are currently in use, the backup would fail without releasing a mutex.

In both cases, all future backups would then wait indefinitely to acquire the mutex.

The work around for this issue, if encountered in a previous server, is to stop and restart the c-tree Server process and then suspend backups until both conditions above can be corrected.



Recovery Script Configurations Ignored from Backup Scripts

Most ctrdmp keywords are allowed, but ignored in backup scripts. The following dump restore script keywords would cause a dynamic dump backup to fail if present at backup time.

!CONVERT_PATHSEP

!REDIRECT_IFIL

!OLD_REDIRECT

!DIAGNOSTICS

!RECOVER_DETAILS

!RECOVER_MEMLOG

These restore only options are now correctly ignored during backup.

Update to Dynamic Dump Keys

Files restored from dynamic dump may have been missing data or keys, with probable 160 errors after a restore. This was due to cache pages failing to be flushed due to the page being in use (contended). This has been fixed such that if a backup checkpoint fails to flush transaction files cache pages, all active transaction logs are included in the backup.

SYMPTOM:

1. Files restored from dynamic dump may be missing data or keys. Probable 160 errors.
2. When using !ALLOW_TRAN, the backup may fail with error 96 on L0000000.FCS

CAUSE:

1. The BDUMP checkpoint marking the beginning of the dynamic dump backup attempts to flush all data and index cache pages so all prior updates are on disk. It is possible that some pages may fail to be flushed due to the page being in use (contended). The transaction logs included in the dynamic dump begin with the BDUMP checkpoint, so they don't have sufficient data at restore time to fill in the cached updates that did not go to disk.
2. After server startup, ct_rcvlog may be initialized to 0 until the first ENDLOG is written. If a ADUMP checkpoint was written in this interval, ctdlgnum was set to zero. This caused the dynamic dump to attempt to include log #0 into the backup, which never exists.

SOLUTION:

1. If a BDUMP checkpoint fails to flush ctTRNLOG cache pages, we include all active transaction logs in the backup.
2. When ctdlgnum is set, if ct_rcvlog is zero, compute the log requirement based on ct_logprg.

Improved Hot Backup Quiet Checkpoint Requirement

Prior to this update, a forward roll would fail with **RFCK_ERR** if the starting checkpoint indicated unflushed updated data or index cache pages, unless the checkpoint was a Restore point. All flushed states are now guaranteed such that the unflushed TRNLOG cache pages are recovered from the transaction logs, making it valid to use this as a forward roll starting log position.



Dynamic dump missing data or keys for non-ctTRNLOG Files

SYMPTOM: Non-ctTRNLOG Files restored from dynamic dump may be missing data or keys when using !PROTECT option. Probable 160 errors.

SOLUTION: Since there are no transaction logs, the only action we can take is to retry the cache flush in the event that some pages are not flushed. Checkpoint flushes now loop up to MAX_NONTRAN_CHECKPOINT_FLUSH_ATTEMPTS or MAX_TRAN_CHECKPOINT_FLUSH_ATTEMPTS.

Since flushing pages removes them from the update list, the retries will only attempt to flush pages that failed the prior attempt, plus any pages that might be added during the flush attempt. We apply this retry to all checkpoints that flush pages (since we think it important to flush), but we are only aware of a vulnerability for BDUMP_CHECK (or the new ADUMP_CHECK) checkpoints.

When #define ctBEHAV_CHECKPOINT_FLUSH_RETRY is enabled, we set macros MAX_NONTRAN_CHECKPOINT_FLUSH_ATTEMPTS = 5 and MAX_TRAN_CHECKPOINT_FLUSH_ATTEMPTS = 1 (1 = no retries, behavior not changed).

If the final flush attempt fails to flush all pages for BDUMP_CHECK, we now log a message to CTSTATUS like: "Non-tran index buffers unflushed in dump checkpoint" We can use these messages to identify how frequently this issue may occur in practice, and possibly revise these retry values.

NOTE: Dynamic dump backups of Non-ctTRNLOG files created while updates are in progress without using the !PROTECT option can always exhibit data loss or index inconsistencies.

Prevent Unix Dump Restore Hang During Large Deployment

A dump restore when deploying a replication plan might hang on Unix systems. This happened as the deploy operation was waiting for the restore process to terminate before its dump output was read. However, only a certain amount of data can be written by the dump restore process to its standard output stream without having been read by the parent process before a write to standard output blocks. If a large number of files are being restored, this limit is reached.

Dump restore output is now read immediately, rather than waiting for the child process to terminate avoiding the hang.

Correct Dump Restore Redirect Log Messages to Include Filename

The **ctrdmp** utility was found to incorrectly log redirection messages. A redirect message was logged three times instead of logging a message and including the old and the new file names.

Incorrect Log Messages



```
Wed May 25 16:07:51 2022
User# 00001    DR: file will be redirected....
Wed May 25 16:07:51 2022
User# 00001    DR: file will be redirected....
Wed May 25 16:07:51 2022
User# 00001    DR: file will be redirected....
```

Correct Log Messages

```
Wed Jul  6 12:00:00 2022
User# 00001    DR: file will be redirected....
Wed Jul  6 12:00:00 2022
User# 00001    C:/abac/db/zent/qatruncatefile.dat
Wed Jul  6 12:00:00 2022
User# 00001    ./tempdump_ReplPlan1_43352/tempqatruncatefile.dat
```

A prior modification to logging passed an incorrect parameter value to the logging function and proper file name values are now used in this logging function.

Prevent Dynamic Dump Restore Errors 499, 12, and Files Missing after Restore

A dynamic dump restore might fail with error 499 or 12, or files might be missing after a successful dump restore. This situation was determined to be when a file being backed up was truncated or deleted during the backup process.

There are two cases to consider:

1. If file name redirection (!REDIRECT) is used in the dump restore script, the dump restore might not perform the proper transaction-dependent file undo or redo actions for file create, rename, and delete operations that occurred during the dynamic dump. This is because the dump recovery functions that perform these undo and redo actions do not apply the file name redirection rules to the file names that are read from transaction logs. As a result, the recovery function might not find the file that is to be acted upon.
To prevent file redirection interference, the transaction-dependent file undo and redo functions used by the restore utility now also apply file name redirection rules to file names read from transaction logs.
2. The server prevents file truncate and delete operations from occurring during a dynamic dump while the file is being backed up. A file truncate fails with error **DDDM_ERR(440)** in this situation, and the delete sleeps until the dump has backed up the file. However, after the file has been backed up, if transaction-dependent rename or delete operations occur on the file, the dynamic dump restore recovery might not be able to undo or redo these operations because the temporary files (filename Z*) created during these operations are not included in the dump backup. Also, the temporary files are deleted when the transaction commits, unless the file uses a restorable delete (*ctRSTRDEL*) extended file mode.

To prevent truncate and delete operations interfering, these operations are now rejected for the entirety of the backup operation, even after those specific files have been backed up.



Diagnostics

For help with additional backup and recovery diagnostics !KEEP_LOGS is now an allowed dump restore script option. Presence of this option causes the dump restore to retain transaction logs extracted from the dump stream file instead of deleting these logs before exit.

Server wait on external ctrdmp or disk full action process may hang on Windows

SYMPTOM: On Windows, when the server is reading the output of an external ctrdmp process or a disk full action process, the thread may hang on the read call.

CAUSE: The handle that the parent process passed to the child for use in writing to standard output is still open by the parent process. According to Microsoft's documentation (see example <https://docs.microsoft.com/en-us/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output>), the parent process should close the handles that it passes to the child process in order to be able to know that the child process has terminated.

SOLUTION: Close the handle after creating the child process.

Dynamic dump !REPLICATE

Enables replication of the files specified by the !FILES (/doc/faircom-database-backup-guide/database-backup-option-files.htm) command, including a superfile host and its members.

You can enable replication on one or more files during your dynamic dump "hot" backup operation. This allows an easy way to begin syncing files between systems.

A dynamic dump script can specify the !REPLICATE option in the !FILES section to instruct the dynamic dump to enable replication support for the files whose names follow the !REPLICATE option. Replication is commenced after the dynamic dump has achieved a quiet transaction state. As are other files listed in the !FILE section, these files are also backed up by the dynamic dump. For example, the following script will cause the dynamic dump to enable replication for the file *test2.dat* if it does not already have replication enabled:

```
!DUMP mybackup.fcd
!FILES
FAIRCOM.FCS
test1.dat
test1.idx
test2.idx
!REPLICATE
test2.dat
!END
```

If enabling replication fails for any file, the dynamic dump logs an error message to *CTSTATUS.FCS* and terminates. Possible causes of such an error include:



1. Specifying the name of a non-ctree file or a file that does not meet the requirements for replication after the !REPLICATE keyword.
2. If a file is open in exclusive mode at the time of the dump, the dynamic dump is not able to enable replication for the file.

Dynamic dump !REPLICATE option now applies to superfile host and its members

The !REPLICATE dynamic dump option now enables replication for a superfile host and its members.

!REPLICATE option in dynamic dump script can cause backed up file to fail to open with error 14 or 413

SYMPTOM: When using the !REPLICATE option in a dynamic dump script to enable replication for a c-tree data file when a dynamic dump is performed, if the file is open at the time of the dump but has not been updated, after the file is restored it fails to open with error 14, or error 413 for a superfile.

CAUSE: If the file is open and has not been updated at the time of the dump, the enabling of replication for the file sets the update flag in the file's header to a non-zero value. The dynamic dump sees the update flag is set and writes the header to disk before backing up the file. However, no transaction log entries have been written for the file, and so during restore the file's update flag value is not reset to zero.

SOLUTION: When enabling replication for a file during a dynamic dump, write the header with the replication bit set and skip turning on the file's update flag.

Update to Dynamic Dump Error Message

When a file without an extended header is specified in the !REPLICATE section of a dynamic dump backup script, the dump now fails with **error 734**, and the following message is logged in *CTSTATUS.FCS*:

```
DD: Failed to enable replication for file that does not have an extended header
```

10.2 ctinfo: Retrieve Assigned OS System File IDs

The **ctinfo** utility has been enhanced to output the DEVID and INODEID values of the system file ID stored in each c-tree file. The new output appears as:



```
System id (device) = 0xc28befb3
```

```
System id (inode) = 0x34900000000060e
```

GetCtFileInfo() has been enhanced to support two additional mode parameters:

- **DEVID** - Returns a 4-byte integer holding the device ID component of the system file ID.
- **INODEID** - Returns a 4-byte integer holding the low-order 4 bytes of the 8 byte *inode* component of the system file ID. Use **ctGETHGH()** to retrieve the high-order 4 bytes.

10.3 ctinfo: File encryption and file name information

The ctinfo utility has been enhanced to include information about the encryption cipher and length. It also now outputs the full data file name and default index file name.

10.4 ctinfo improved: Show DODA type names; Pause before dumping attributes content

SYMPTOM:

- ctinfo did not show the data type for new cobol types
- ctinfo did not pause after DODA and before attributes making it hard to read the DODA.

CAUSE:

- new data type not mapped
- missing logic to pause.

SOLUTION:

- added the “case”s for the missgin field types
- added calls to ctinfo_pause() around attributes output

10.5 ctinfo: display record update callback information

The ctinfo utility already has logic to display the information about full text indexes. We now updated this utility to include generic record update callback details for callbacks other than full text indexes.

This may help to understand an error 981 (CDLL_ERR) when attempting to open a file. The new output includes the callback name, library, call time, function names, and parameter values stored in this resource.



10.6 ctinfo: display distinct key counts for indexes

When ctinfo is run on an index, we now display any full and partial distinct key counts for each member.

```
Header Count
5202 keys in index 0
5202 distinct keys in index 0
partial distinct counts for index 0: 5197
5202 keys in index 1
5035 distinct keys in index 1
5202 keys in index 2
6 distinct keys in index 2
```

10.7 ctVerifyFile: Improved Index Leaf Nodes Key Mark Checks

The **ctvfyfil** utility exhibited unexpected behavior in client/server mode: for a particular index, the utility failed with error **601** when using the **-chkidxrec** option alone, but when **-chkdatkeys** and **-chkidxrec** were both used, the utility was not reporting certain unexpected marks. The logic has been modified so that the index verify always reports cases of unexpected key marks. This modification is enabled only for key lookup operations that are made during a call to **ctVerifyFile()**.

Affected Components: Server, standalone library

10.8 ctVerifyFile: Additional Test for keys Referencing Deleted Records

Previously, **ctVerifyFile(ctVFchkIndexWithRecord)** did not report errors for keys referencing a deleted fixed length record. To correct this, an additional test for this situation was added that reports **IREN_ERR (114)**.

AFFECTED COMPONENTS: ctvfyfil utility, ctVerifyFile Isam api

10.9 ctVerifyFile: Check for unexpected key marks in index leaf nodes crashed on variable-length data file

SYMPTOM: When running ctvfyfil on a variable-length data file, the server (or the ctvfyfil utility if run in standalone mode) might crash.

tp is past the end of the node buffer.



CAUSE: Previously we modified the index verify leaf node scan to force evaluation of key marks when scanning right from the current position in a leaf node even if the mark count in the node header is zero. However, we did not exclude a variable-length data file's space management index from this checking of transaction marks. The space management index is not under transaction control, and so it does not reserve space in the index node for a transaction mark region. As result, we read past the end of the index node buffer.

SOLUTION: Exclude variable-length data file space management indexes from the leaf node key mark check.

10.10 ctVerifyFile: Internal Index Node Validation Improvements

An **ctVerifyFile()** API function mode bit is available enabling additional index validation steps. *ctVFchkIndexInternals* can be specified and this mode specifically detects the same index error that may be encountered during node pruning, which is logged to *CTSTATUS.FCS* with the message "ERROR: Invalid node links". Future additional index checks could be added under this bit.

An End of File validation now also occurs with the default open mode (Read Only). Previously it was skipped if not in Exclusive mode.

The **ctvfyfil** utility was updated use this mode with the **-chkidxint** command line option.

Usage

```
ctvfyfil [<option> ...] <file name>
where <option> is one or more of the following:
  -<page size> Use the specified page size
  -chkdat Read data file only (default)
  -chkdatkeys Read data file and check keys in index
  -chkdeletedspace Check deleted space
  -chkidx Read index file
  -chkidxrec Read index file and check records in data file
  -chkidxint Additional index checks
  -excl Open the file in exclusive mode
  -int Interactive mode: stop on each error
  -index=<N> Check only the Nth index (N=1,2,...)
```

10.11 ctVerifyFile/ctvfyidx - Add distinct key count verification

Distinct key counts are an optional file feature used by the SQL optimizer to provide improved index selection. These are approximate counts of unique keys (or partial keys) maintained in the file header for non-unique (and/or multi-segment) indexes. Approximation occurs when the key is first or last in a node, as we don't check neighboring nodes. We examined a table & indexes with poor SQL query performance caused because the distinct key counts were wrong to a degree far in excess of what we expect due to this approximation, causing a suboptimal index to be used. Development logs indicated a similar issue was encountered in a transaction recovery case, but



this was never reproduced. This enhancement adds checking of the distinct key counts to existing index verification utilities so we can more easily detect problems with the distinct key counts, which may provide more insight on how to reproduce this problem.

ctvfyfil - File Verify Utility

The utility ctfyfil now has the -chkdistinct option to include these checks. If combined with the -excl option the distinct counts will be updated to the correct value.

```
usage: ctfyfil [<option> ...] <file name>
where <option> is one or more of the following:
-<page size>      Use the specified page size
-chkdat           Read data file only (default)
-chkdatkeys       Read data file and check keys in index
-chkdeletedspace  Check deleted space
-chkidx           Read index file
-chkidxrec        Read index file and check records in data file
-chkidxint        Additional index checks
-chkdistinct      Distinct key counts
-excl             Open the file in exclusive mode
-int             Interactive mode: stop on each error
-index=<N>        Check only the Nth index (N=1,2,...)
```

The function ctVerifyFile() now supports option ctVFchkIndexDistinct to check index distinct key counts.

ctvfidx - Index Verify Utility

The utility ctfyidx now has the -distinct option, which is enabled by default. If combined with the -excl option the distinct counts will be updated to the correct value.

```
usage: ctfyidx [<option> ...] [-u <userid>] [-p <password>] [-s <servername>] <file name> <member #>
where <option> is one or more of the following:
-excl            Open the file in exclusive mode
-delstk          Check delete stack (on by default)
-link            Check links (on by default)
-leaf            Check leaf nodes (on by default)
-chkkey          Check keys in leaf nodes
-space           Check for lost nodes
-distinct        Check distinct key counts (on by default)
```

The function ctVERIFYidx() now supports option ctVFYIDXdistinct to check index distinct key counts.



10.12 Block Logon when Quiesce is Abandoned

When the server has been quiesced and the connection that established the quiesce disconnects without undoing the quiesce, subsequent connection attempts fail with error **QABN_ERR** (898), except for special ADMIN user logons like the **ctadmn** utility uses, which are permitted to log on to undo the quiesce.

The server now supports a configuration option to change the behavior in this situation to blocking the connection request instead of returning **QABN_ERR**. To enable this feature, add the following option to the server configuration file:

```
ABANDONED_QUIET_SUSPEND_LOGON YES
```

This option defaults to NO. It can be changed at runtime using **ctadmn** option 10.

10.13 Retrieve MRT Host Table Name via SQL Stored Procedure

A new stored procedure, **fc_get_hosttablename**, retrieves the host table name of a Multi-Record Type (MRT) table. As an input parameter, it receives an integer that is the table ID for which it will retrieve the MRT host name and UID. It returns a result set with exactly 1 row containing four columns:

1. The table UID or the host UID in case of ,ulti-record type (MRT) table
2. The c-treeDB table name
3. The filesystem path where the table is located
4. The filesystem name of the table

10.14 ctCreateAuthFile() Creates Encrypted Auth Files

This modification introduces a function call, **ctCreateAuthFile()**, to create encrypted authentication files similar to the FairCom **ctcmdset** utility.

Affected Components: **ctcfgset**, **ctcmdset**, core library

ctCreateAuthFile

Parameters

- pTEXT *sn* - the input script name
- pTEXT *outfile* - the name of the output file
- VRLEN *outsiz* - the size of the output buffer for file name
- pTEXT *errbuf* - a buffer where a textual error message is shown in case of error
- VRLEN *errbuflen* - the length of the error log



Return

The function returns a NINT value, 0 in case of success an error number otherwise.

10.15 ctpath - Change Internal (SQL) Database Paths

The **ctpath** utility allows you to adjust the path in the dictionary after extracting the SQL database directory from a backup and renaming it.

Changes the internal FairCom DB SQL dictionary paths of database locations.

Syntax

```
ctpath [-s server] [-u user] [-p password] [-d database] [-v] from-path-prefix to-path-prefix
```

Options:

- **-s server** - FairCom DB server name
- **-u user** - user name
- **-p pw** - user password
- **-d database** - database name
- **-v** - verbose
- **from-path-prefix** - path prefix to be replaced
- **to-path-prefix** - path prefix to use as replacement

Description

- Command-line switches may *not* have optional spaces between the switch and the argument. Example: **-s FAIRCOMS** is *not* the same as **-sFAIRCOMS**.
- Command-line switches may be entered in any order, but the **from-path-prefix** must appear *before* the **to-path-prefix**.
- Command-line switches should start with a '-' or '/' character. Command line switches accept both lowercase and uppercase characters, e.g. **-s** or **-S** are the same.
- **ctpath** returns 0 when the execution detected no errors. Non-zero values are returned when errors are detected. Error messages are written to **stderr**.
- If you omit the **-d** database switch, all databases in the session will be scanned.
- The **-r** switch indicates to repeat the substitution and replace all occurrences of the searched path and not just the first one.
- The **-v** command-line switch indicates verbose output.

10.16 ctscmp - Corrected IAIX_ERR (608) Error While Compacting V6 Superfiles

The **ctscmp** utility could fail with error **IAIX_ERR** (608) when compacting a V6 superfile with an **SRLSEG** index. The logic has been modified to correct this error.



10.17 ctrbldif - Toggle Index Null Key Support with Rebuild Utility

The ctrbldif utility supports command-line options to turn null key support on or off for all indexes or selected indexes. It can be desirable to turn off null key support for an index that doesn't use it so that it can qualify as a replication unique index. The options are implemented in the **ctrbldif** utility, not in the server or the standalone library.

Here are the newly-added command-line options:

- **-set-null** Turn on null key support for all indexes.
- **-set-null=<N>** Turn on null key support for specified index numbers.
First index is index zero. Example: -set-null=0,2,4
- **-unset-null** Turn off null key support for all indexes.
- **-unset-null=<N>** Turn off null key support for specified index numbers.
First index is index zero. Example: -unset-null=0,1,5

10.18 Support Added to hot Alter Table for non-Schema-Based key Segment Modes.

With this revision, hot alter table now allows adding fields after all existing fields on tables that use non-schema-based key segment modes. This is allowed because adding new fields after the current fields will not affect any key segment offset calculations.

AFFECTED COMPONENTS: server

10.19 Update to ctsqlcdb Utility

The database maintenance utility, **ctsqlcdb**, supports **-add** switch to add a SQL database, but **ctsqlcdb** used with **-add DATABASE** assumes the database dictionary has a name like **./DATABASE.dbs/SQL_SYS/DATABASE.fdd**. A more general form of the **-add** command was needed and now available.

```
-addctdb <dbname> <path> [<servername> [<user> <password>]]
```

This variation supports a reference to an existing c-treeDB database located at *<path>*

Example

```
ctsqlcdb -addctdb test ./somewhere/
```

This links a database dictionary *./somewhere/test.fdd* into the session dictionary making tables available at the database layer.



10.20 Segmented Sort Work Files for Rebuild

A rebuild of a compressed record data file was reported to fail with error 496, with internal error 674 (additional segments needed) and 39 (no more space left in file). The rebuild operation creates sort work files as segmented files overcoming legacy 2 GB filesystem limits. The maximum required number of sort work files was calculated using the physical size of the actually data file. However, for a compressed record data file, the estimate might be too small resulting in the unexpected error. Disabling segmented sort work files and always writing into HUGE files (64-bit addressable) resolves this problem as on modern operating systems, files can be arbitrarily large.

10.21 Retain Existing Permissions and Synonyms on SQL Import

The existing SQL Import utility had ability to retain existing permissions and synonyms however, was not exposed in the command line usage. This option is now available with the command line utility switch "-y".

Usage

```
ctsqlimp <filename> [-d database] [-s srvname] [-u userid] [-a password] -y
```

10.22 ctfchk Utility added to packages

ctfchk is a standalone c-tree utility that calculates the checksum on all active records in a fixed-length c-tree data file. It reads active records from a fixed-length c-tree data file in physical order and outputs a checksum and active record count. The utility uses the CRC-32 algorithm to compute the checksum.

This utility can be used to compare the record contents of two fixed-length data files. Two c-tree data files that contain the same active record images in the same order will generate identical checksums. Two c-tree data files whose active record contents differ will generate different checksums (subject to the limitations of the CRC-32 algorithm). See FairCom documentation for more information.

10.23 PTADMIN() option to purge all existing partitions below the new partition base value when increasing the partition base

A call to PTADMIN() to change a partition host's partition base value fails with error 718 (PUSD_ERR, partition member in use) if any active partitions exist below the new base value. We now provide a way to successfully increase the base value in this situation, by automatically purging all active partitions that are below the new base value.



To use this feature, specify both the ptADMINbase and the ptADMINpurge modes in the ptmode parameter when calling PTADMIN(). Here is an example:

```
NINT rc;
FILNO datno; /* set to partition host data file number */
/* Change partition base to 15345, and purge all existing partitions below this base partition. */
rc = PTADMIN(datno, NULL, 15345, ptADMINbase | ptADMINpurge | ptADMINraw);
```

Note that we have preserved the existing behavior: when only ptADMINbase is specified in ptmode, the PTADMIN() call still fails with error 718 if active partitions exist below the new base value.

10.24 ctpartadmin utility now supports partitioning an existing data file

This modification introduces a new command 'create' to ctpartadmin utility. The '-c create' command creates a partition index and partition the data file based on the partition rule expression. The command requires the following options to be specified:

- -r rule_expression: the partition rule expression string. If the expression string contains space characters, the string should be specified within quotes. e.g., ctpartadmin -c create -r "year(co_ordrdate) - 2000"
- -i index_name: the name used to reference the index. The index name is combined with table name to make up the index file name. e.g., ctpartadmin -c create -f custordr -i "part" results in index file custordr_part.idx
- -I (uppercase i) index_name: the same as -i but the partition index is created to allow only unique keys
- -g field_name: the name of the field used to make the partition key sorted in ascending order.
- -G field_name: the name of the field used to make the partition key sorted in descending order.

The -g and -G options can be combined and specified multiple times to create a multi-segment key. e.g., ctpartadmin -c create -f customer-orders -G customer_category -g order_date

If the '-c create' command is successful the 'file partitioned' message is displayed. e.g.,

```
>ctpartadmin -c create -f custordr.dat -u admin -p ADMIN -r year(co_ordrdate) -i part -g co_ordrdate
file partitioned
>ctpartadmin -f custordr.dat -u admin -p ADMIN -c status -n 2002
partition active
```



10.25 ctldmp utility now decodes DIFIMAGE entries into old and new image values

For DIFIMAGE log entries, the ctldmp utility now deconstructs the DIFIMAGE data into the old and new image values, showing the offset and length of each pair of old/new images.

10.26 ctldmp/ctlchg: Support 6-byte transaction numbers

We added support to the ctldmp and ctlchg utilities for specifying a 6-byte transaction number value for the TRAN option.

The utilities support two formats:

TRAN <high2bytes>:<low4bytes>, as displayed by the ctldmp utility, and

TRAN <64bit_integer>

10.27 ctcompare utility - Support running with a server that has a different path separator

The ctcompare utility can now be run as a client with a server that uses a different path separator than the client system.

10.28 Specialized Superfile Compact Options

This revision adds a **-v11** command-line switch to the superfile utility **ctscmp.exe** (**ctscmp** on Linux/Unix) utility. This switch avoids introducing V13/V12-specific features to the resulting file, allowing new files created with V13/V12 to remain backward-compatible with V11.

Usage:

```
ctscmp.exe filename [-v11] [sectors]
```

11. Diagnostics

11.1 Suppress DIAGNOSTICS READ_ERR stack traces when no error is returned

When using the `DIAGNOSTICS READ_ERR` server configuration option, the server may create process stack traces and log **READ_ERR** diagnostic messages to *CTSTATUS.FCS* even if no **READ_ERR** errors were returned from c-tree function calls.

This could occur when a read at the end of the file returned fewer bytes than was requested, which is a normal situation (if sysiocod was zero, the read was successful).

Examples were found in the replication log read function, `rlctio()`, and a physical read of a file, which ignores **READ_ERR**.

The logic has been modified to take these situations into account and therefore will no longer return these false diagnostic messages.

Affected Components: Server, standalone library

11.2 Robust System Log Features Secures SQL Database Auditing

The FairCom DB System Log - SYSLOG - is an extensive audit log subsystem capturing multiple categories of database events, including user defined entries. As SYSLOG is utilized more, several limitations became apparent with respect to size, and search performance. These have been reviewed updated to handle greater volume of activity.

Increased size

The server now creates system log files as HUGE files so they are not limited to 4GB. HUGE files can be as large as 16 exabytes. At startup, if the server finds existing non-huge system log files, it renames them to *SYSLOGDT.FCA* and *SYSLOGIX.FCA* (first deleting any existing files having those names), and then the system log thread re-creates the system log files as huge files.

To disable the creation of huge system log files, add `SYSLOG NONHUGE` to the server configuration file.

Faster purge and truncation



SYSLOG is a FairCom DB subsystem to secure database events for auditing. Over time this table can become quite large and existing purge functions are not efficient enough. The **SYSLOG()** function supports purging records, optionally filtered by time and by event code. A notable problem is even when all entries are "purged" storage device space is not released leaving potentially very large (up to 4GB) empty files on disk. The solution to this problem is a new SYSLOG truncate capability.

With `SYSLOG TRUNCATE`, present in your configuration file if a complete purge is requested (no filtering by time or event) we truncate the file rather than delete individual records.

This approach is much faster and avoids limitations with space reuse. Any users reading from **SYSLOG** will encounter error **FBLK_ERR** if the log is truncated while they have it open. The table must be closed and reopened after receiving this error.

ctadmn

The **ctadmn** utility has been enhanced to support **SYSLOG** purging with time-based filtering.

The "Monitor Server Activity" menu now has a new option, "Purge SYSLOG entries", which prompts for the number of days of activity to keep. "0 Days" results in a truncate, while larger values use the regular approach.

Efficient space reuse

After a **SYSLOG PURGE**, messages could fail to be inserted because they exceeded the largest available deleted space and the **SYSLOG** was at its non-huge size limit. This can occur because *ctTRNLOG* files do not coalesce adjacent deleted records by default unless a RECBYT index is available.

For efficient space reuse a RECBYT index has been added to the **SYSLOG** tables. At startup, existing **SYSLOG** tables are checked for the requested RECBYT setting, and recreated if necessary. **SYSLOG*.FCS** files are renamed to **SYSLOG*.FCA** before this conversion and can be removed once successful SYSLOG behavior is confirmed.

Prior behavior can be restored by specifying keyword `SYSLOG V11_REUSE`.

Selectively omit SQL user statement logging

When SQL query logging is enabled with `SYSLOG SQL_STATEMENTS`, by default all SQL statements are written to the SYSLOG table. A common use case involves two classes of SQL operations.

- The first are defined SQL statements issued directly (usually hard coded) by an application. These are quite numerous, repetitive, and well defined. Audit logging isn't of great value with these specialized application -specific SQL operations.
- A second more important class are "external" SQL queries such as those from a user via ODBC drivers. Many organization required these queries to be logged and audited for security review.

A new configuration option `SYSLOG_EXCLUDE_SQL_USER <name>`, can now exclude users from `SYSLOG SQL_STATEMENT` logging by user name greatly reducing volume of log entries in



the SYSLOG tables.. Multiple users may be excluded by specifying the keyword multiple times. No validation is made that the name specified matches an existing user name.

11.3 Configuration keyword CTSTATUS_MASK WARNING_PAGESIZE disables mismatch warning

In V13/V12, each open of an index with a page size smaller than the servers configured PAGE_SIZE setting generates a warning message like:

```
Mismatched PAGE_SIZE is wasting 75% of Index cache used by C:\\PathTofilename.idx
```

A new configuration option CTSTATUS_MASK WARNING_PAGESIZE has been added, disabling logging of warning messages such as the one above.

For applications that have large numbers of files, or that frequently open and close files, this may result in logging large numbers of messages to CTSTATUS. This revision disables this logging after the first 20 occurrences.

11.4 Configuration keyword disables mismatch warning

A new configuration keyword has been added:

```
CTSTATUS_MASK WARNING_PAGESIZE
```

Setting this keyword disables the logging of warning messages like:

```
Mismatched PAGE_SIZE is wasting 75% of Index cache used by C:\\Path Tofilename.idx
```

11.5 Mask PAGE_SIZE Warnings in CTSTATUS.FCS

FairCom DB V13/V12 defaults to a larger page size configuration recommended for modern client-server applications. However, existing index files must be rebuilt to take advantage of this larger index node size. Opening existing indexes smaller than the configured server value results in a warning message logged to CTSTATUS.FCS about wasted memory usage.

For applications that have large numbers of files, or that frequently open and close files, this may result in large numbers of messages to CTSTATUS.FCS, and this logging is disabled after the first 20 occurrences.

The following command turns off all PAGE_SIZE warnings so no warnings are produced:

```
CTSTATUS_MASK WARNING_PAGESIZE
```

Setting this keyword disables the logging of warning messages like the one shown above.



For debugging, if it is necessary to see all warnings, you can use the following keyword to log *all* the warnings to *CTSTATUS.FCS*:

```
CTSTATUS_MASK WARNING_PAGESIZE_SEEALL
```

11.6 Diagnostic Message for Transaction Log Deletion

It was reported that archived transaction logs could be unexpectedly deleted. To determine a cause, diagnostic logging of transaction log deletion has been added, and is enabled by the existing server configuration option `DIAGNOSTICS LOG_PURGE`.

Example

```
Mon Nov 15 13:14:29 2021
- User# 00004 ct_inactlog: prglog= 8 reduce_flag= 1 ct_logkep= 0 ctLnum->root1= 46 ct_logprg= 4
Mon Nov 15 13:14:29 2021
- User# 00004 ct_inactlog: (2) delete log L00000008.FCS
```

AFFECTED COMPONENTS: Server

11.7 Resource Read Errors (RRED_ERR, 407) Diagnostics

RRED_ERR(407) occurs when traversing a data (or index file's) resource chain if the data contains a resource mark value other than 0xfefe. To help diagnose occurrences of this error, diagnostic logging has been added so that when **RRED_ERR(407)** occurs, the following message is logged to *CTSTATUS.FCS*:

```
User# 00001RRED_ERR: file= RECBINDT.FCS clstyp= 2 flmode= 0x1037 offset= 32768 loc= 1 read= ffff
1f000000 09000000 00000000 afa10000 1f800000 00000000 00000000 : 407
```

This message shows the file name (file=), file type (clstyp=), file mode (flmode=), offset at which the resource read occurred (offset=), location in the code where the diagnostic function was called (loc=) and the resource data that was read (read=). The first two bytes of the resource data are the resource mark, which should be 0xfefe.

The server also creates a minidump when this error occurs if `DIAGNOSTICS RRED_ERR` is enabled in *ctsrvr.cfg*. This can be dynamically enabled at run time via administrator utilities.

AFFECTED COMPONENTS: standalone library, server

11.8 Failed Client Shared Memory Connection Diagnostics

A report was made concerning intermittent failed client connections using shared memory. Server side logs indicated a dropped client Unix socket. To understand an actual cause of failure and improve a reconnection attempt, additional information is needed. In many cases, an `strace` or



truss of the client process will show failed system calls. However, for rare events in production, a run time diagnostic is needed.

The ISAM client library on Unix systems now supports logging information to a file when a shared memory connection attempt fails. To use this feature, set the environment variable CTCLIENT_DEBUG_LOG to the name of a file before the process connects. If the shared memory connection fails, a message is written to the file. The message has a timestamp and process id (pid).

Example

```
export CTCLIENT_DEBUG_LOG=client.log
./ctixmg ADMIN ADMIN FAIRCOMSXXX
```

Contents of *client.log*:

```
Thu Mar 24 16:20:39 2022 [pid=18219176] at_SessionOpen: Could not get shared memory key;
ftok_name=/tmp/ctreedbs/FAIRCOMSXXX readKey=0 errno=2.
```

Analysis of this information should lead to modifications for a more robust connection sequence.

11.9 Added Diagnostic Logging for INIX_ERR

Diagnostic logging has been added to the server for **error 124 (INIX_ERR)**. To enable logging, add `DIAGNOSTICS INIX_ERR` to `ctsvr.cfg`. When logging is enabled and **error 124** occurs, a dump of the server process is created.

Additionally, the following diagnostic error logging options can now be changed at runtime:

- `DIAGNOSTICS FALG_ERR`
- `DIAGNOSTICS DMAP_ERR`
- `DIAGNOSTICS POPN_ERR`

AFFECTED COMPONENTS: server

11.10 Server Read-only Status Logged to CTSTATUS.FCS

`CTSTATUS.FCS` logging has been updated to record when a server is configured in read-only mode, or when read-only mode is turned off.

The following message is now printed when read-only mode is set:

```
server set in read-only mode
```

When read-only mode is turned off, the following is printed:

```
server set in read/write mode
```



11.11 Improved System File id Diagnostic Logging

System file id list logging has been improved in the following ways:

1. Added a location code in order to track the location of added and deleted entries
2. Added entry/exit logging for **iOPNFILX()**
3. Added logging of goto calls within **iOPNFILX()**

AFFECTED COMPONENTS: server

11.12 Diagnostic logging for commit delay livelock

In order to understand what can be causing the message "WARNING: COMMIT_DELAY livelock" in CTSTATUS.FCS, we now include the task id of the thread that is holding the log flush semaphore in the log message. We create a process stack dump of the server process. If we are able to reproduce this situation, we will want to see what the thread is doing that is holding the log flush semaphore. It could be just an unexpectedly slow write of the transaction log buffer to the transaction log on disk. However, we wonder if there could be cases where a deadlock has occurred.

Also, when the server's attempt to create a stack dump fails, we log the message "Failed to dump stack" instead of "Dumped stack" to make it clearer that the attempt failed.

11.13 Server logs diagnostic messages to help analyze slow or hanging plugin startup and plugin unload

We made the following improvements to the server, which will help us understand which plugins are taking a long time to start and to unload:

- When each plugin starts, we output the time it took to start.
- When each plugin unloads, we output the time it took to unload.
- When starting to shut down, the plugins now set their status to **PLUGIN_STOPING**, and when finished shutting down to **PLUGIN_LOADED**.

We implemented these changes to help us better diagnose slow or hanging server startup or shutdown that is caused by plugins.

This helps troubleshooting server startup or shutdown hanging or taking a long time.



11.14 Node Name Added to DLOK_ERR and Transaction Abort Messages

Per a customer request we added the connection node name to the following messages are logged to *CTSTATUS.FCS*:

a) **DLOK_ERR** diagnostic error messages now includes the node name of the the lock owner, if available. (Note: the **DLOK_ERR** logging is enabled by the `DIAGNOSTICS DLOK_ERR` configuration option.)

b) Transaction abort messages for maximum logs exceeded and other conditions now includes the node name of the connection whose transaction is being aborted, if available.

Here are some example messages showing the node name:

```
Mon May 1 09:38:09 2023 - User# 00026 DLOK_ERR: file=myfile.dat
offset=0x0000-000129af owner=21 node name="my node name"
Mon May 1 09:38:41 2023 - User# 00036 Transaction aborted at ct_mul_abandon3 for user# 21. Node name
"my node name": 821
If no node name is available because the connection has logged off already, the
log messages indicate this with (connection not active). For example:
Mon May 1 09:38:09 2023 - User# 00026 DLOK_ERR: file=myfile.dat
offset=0x0000-000129af owner=21 (connection not active)
Mon May 1 09:38:41 2023 - User# 00036 Transaction aborted at ct_mul_abandon3
for user# 21 (connection not active): 821
```

11.15 Log message to CTSTATUS.FCS when transaction commit fails

In order to help tracking down customer related issues, we added an additional diagnostics message to our server's log file. When a transaction commit fails (which is an unexpected situation), we log an error message in the following format to *CTSTATUS.FCS*:

```
Fri Feb 2 11:57:11 2024 - User# 00005 WARNING: Commit of transaction
<transactionNumber> failed: <errorCode>
```

12. Compatibility

12.1 BINARY CAST of Padding Spaces Retained in VARCHAR Fields

Prior to this modification, we truncated VARCHAR field padding (spaces) when cast from BINARY. Beginning with this modification, we correctly copy the cast buffer as is into the VARCHAR field, and not truncate any string padding with a NUL character, which broke expected data return from the field.

A new server configuration keyword reverts to the old behavior:

STRING TO BIN TRIM SPACES

Note: This is a *compatibility change*. The new configuration option should be considered a backward compatibility option. That is, we have NEW correct behavior by default. If anyone depended on the old behavior, the configuration keyword can be used to revert to that behavior.

12.2 SQL - Correct Numeric String Conversion

Converting this SQL string to a numeric value resulted in a loss of 2 significant digits.

```
'-0.999999999999999999999999999999999999'
```

The number of 9s after the decimal point was 30 instead of 32. The conversion logic from string to numeric in both c-treeDB and SQL counted leading '0' as significant digits, which caused the loss of precision. Proper string parsing prevents this situation.

12.3 Background Transaction Data and Index Flush Threads no Longer Default to Immediate

FairCom DB server inadvertently defaulted to IMMEDIATE mode for TRAN_DATA_FLUSH_SEC and TRAN_INDEX_FLUSH_SEC configuration options even though these options are documented as disabled by default.

The defaults have been modified to set `TRAN_DATA_FLUSH_SEC` and `TRAN_INDEX_FLUSH_SEC` to `rate=1/1`, meaning defer 1 millisecond after every flush.



12.4 TLS/SSL Correctly Configured from Encrypted Configuration File

TLS/SSL connections failed with **SSLCONN_ERR** (1104) when `SUBSYSTEM COMM_PROTOCOL SSL` was configured in `ctsrvr.set`. The logic has been modified to correct this.

Security Considerations: Setting `SSL_CONNECTIONS_ONLY YES` in `ctsrvr.set` was also ignored, allowing non-SSL connections.

Workaround: Configure SSL in `ctsrvr.cfg`

12.5 Expanded Windows Broadcast Machine Name Length

When the `COMPUTERNAME` environment variable is specified in `ctsrvr.cfg`, a Windows API function is now used to get the computer's DNS host name.

Prior to this modification, the Windows `COMPUTERNAME` environment variable was used, which limited the name to 15 bytes. If the system has a computer name longer than 15 characters, only the first 15 were broadcast. This modification supports broadcasting computer machine names longer than 15 characters.

12.6 Mask `PAGE_SIZE` Warnings in `CTSTATUS.FCS`

FairCom DB V13/V12 defaults to a larger page size configuration recommended for modern client-server applications. However, existing index files must be rebuilt to take advantage of this larger index node size. Opening existing indexes smaller than the configured server value results in a warning message logged to `CTSTATUS.FCS` about wasted memory usage.

For applications that have large numbers of files, or that frequently open and close files, this may result in large numbers of messages to `CTSTATUS.FCS`, and this logging is disabled after the first 20 occurrences.

The following command turns off all `PAGE_SIZE` warnings so *no* warnings are produced:

```
CTSTATUS_MASK WARNING PAGESIZE
```

Setting this keyword disables the logging of warning messages like the one shown above.

For debugging, if it is necessary to see all warnings, you can use the following keyword to log *all* the warnings to `CTSTATUS.FCS`:

```
CTSTATUS_MASK WARNING PAGESIZE SEEALL
```

12.7 32K Page Size now Always Set as Default

The default `PAGE_SIZE` for V13/V12 and later is 32K. However, it was noticed in selected cases where the default might not be set to 32K, unless specifically set in `ctsrvr.cfg`. Starting with



releases Build 211022 and later, the default is now set to 32K regardless if the value is set in *ctsrvr.cfg*. Setting `PAGE_SIZE` in *ctsrvr.cfg* is available to override the 32K default page size.

Note: This modification affects compatibility.

12.8 V6 to V7 SDK Mapping Macros Disabled

Certain legacy V6 to V7 macros have been disabled as one of them, `SHARED`, is also used in Solaris system headers. The inclusion of a Solaris system header, such as *dlfcn.h*, after including c-tree headers caused `SHARED` to be redefined without warning, as the default compiler disables warnings in system headers.

12.9 Recovery Script Configurations Ignored from Backup Scripts

Most `ctrdmp` keywords are allowed, but ignored in backup scripts. The following dump restore script keywords would cause a dynamic dump backup to fail if present at backup time.

!CONVERT_PATHSEP

!REDIRECT_IFIL

!OLD_REDIRECT

!DIAGNOSTICS

!RECOVER_DETAILS

!RECOVER_MEMLOG

These restore only options are now correctly ignored during backup.

12.10 Default COMPATIBILITY BATCH_SIGNAL Behavior

`COMPATIBILITY BATCH_SIGNAL` server configuration option enables the following two behaviors:

1. allow the use of the `BAT_CLSE` option in a batch call (otherwise it is ignored), and
2. set `sysiocod` to `BTNO_COD` when the server automatically closes a batch.

It seems reasonable for the `BAT_CLSE` mode specified by the caller to take effect unconditionally rather than depending on a server configuration option, and the setting of `sysiocod` does not need to depend on a configuration option because the client can ignore the value. Also, standalone mode batch code does not condition these actions on a configuration option. For these reasons, we have changed the server's batch function to always enable the two behaviors that were previously enabled by the `COMPATIBILITY BATCH_SIGNAL` configuration option.



Note: FairCom DB still accepts `COMPATIBILITY BATCH_SIGNAL` (we made this choice so that someone using this option will not encounter an error at server startup due to an unrecognized configuration option), however, it is no longer required to enable these behaviors.

12.11 Enable Default Data Cache Empty Lists

Cache improvements introduced in v11 changed the data cache from an LRU list approach, which involved taking pages on and off the LRU lists on each access, to a sequential search over lists of all cache pages. On these lists, the next page that is found is used, regardless of whether it is currently in use or not. As a result, no priority is given to assigning an empty page to a cache request.

Reports described slowdowns as expected data was not retained in cache. This was due to the fact that "hot" cache pages were being reused rather than obtaining pages from the unused pages available.

FairCom DB supported maintaining a list of empty cache pages, however, the feature was not enabled. Prior to v12.5.0 it was possible to add `USE_EMPTY_LISTS YES` to `ctsrvr.cfg` and optionally enable the empty data cache list feature. The empty list feature is now enabled by default as it is much more efficient cache usage. To disable this feature if needed, add `USE_EMPTY_LISTS NO` to `ctsrvr.cfg`.

12.12 Default COMPATIBILITY FILE_DESCRIPTOR_LIMIT Configuration

`COMPATIBILITY FILE_DESCRIPTOR_LIMIT` is now included as a default commented configuration for convenience. After increasing our default `FILES` value to 4096, our QA was getting file descriptor errors on some Linux test environments. This configuration allows an immediate short term work around as needed,

Revised Default

; Maximum number of c-tree files

FILES	4096
-------	------

; FILES - Operating System file descriptor limits

; When the file descriptor limit, set by the operating system, is set too low, a
; message is written to standard output indicating that system-dependent
; initialization failed and that details are in CTSTATUS.FCS.

; The `COMPATIBILITY FILE_DESCRIPTOR_LIMIT` configuration option can be used to
; allow the server to start in this situation. Use the `COMPATIBILITY`
; `FILE_DESCRIPTOR_LIMIT`, in case it is not convenient for a system administrator
; to set the Operating System value of the file descriptor limit.



; Or, you can lower the value of your FILES keyword above.

; Running FairCom Server with insufficient file descriptors can lead to errors
; opening files. Use of this keyword is generally discouraged. It is provided for
; short-term use until site administrators are able to increase the appropriate
; file descriptor limits for the operating system.

;COMPATIBILITY FILE_DESCRIPTOR_LIMIT



12.13 Stricter Transaction Dependent Behavior for all Table Definition Operations

In October 2023, we introduced an option that can be specified when creating a non-transaction-controlled file. This option causes the database engine to delete the file when the transaction aborts. To avoid complications with internal transaction processes, we elevate this operation to function as a full transaction-dependent file would.

We then introduced a feature to automatically enable transaction-dependent behavior for all full transaction-controlled (*ctTRNLOG*) files. Now file operations such as file create, delete, and rename automatically start a transaction if a transaction has not yet been started and perform the operation as a transaction-dependent operation. The goal of this change is to prevent automatic recovery errors on transaction-controlled files that are created without the transaction-dependent feature.

Such recovery errors could appear as an automatic recovery failing with error 12 or 499, and logging the following messages to *CTSTATUS.FCS*:

```
Recovery may proceed by adding 'SKIP_MISSING_FILES YES' *** ** to the server
configuration file. ***
```

```
List of (auto skip) missing files without TRANDEP activity:
```

```
<files>
```

With the use of auto transaction-dependent operations, we do not expect these rare but occasional recovery errors to occur anymore as long as all changes to the files (create, delete, rename) are done under the control of the database engine. *(It is still possible to experience such a recovery error if files are renamed or deleted using O/S operations outside of the control of the database engine which is never advised.)*

To disable the auto transaction-dependent operations in the server model, add

```
AUTO_TRANDEP NO
```

to your server configuration file. To disable this feature in single-user tranproc mode, add

```
#define NO_ctBEHAV_AUTO_TRANDEP
```

to *ctoptn.h* and recompile your standalone library.

In client/server mode, a client can check if the server has enabled auto transaction-dependent support by calling **SYSCFG()** and checking if the `cfgCFGCHKautoTrandep` bit is set on the *cfgCFGCHK* element of the array returned by **SYSCFG()**. This bit is only set when the server is compiled with this support enabled and it is enabled at runtime.

As currently implemented, this option cannot be changed at runtime in the server model and is enabled for all transaction-controlled tables.

13. Security

13.1 Source Code Security - Coverity Scan

FairCom has invested in an industry leader in software source code security scanning software. Using Coverity, FairCom performs continuous security scans of our internal database kernel's source code.

13.2 Apple Packages now signed - 'Contents' directory introduced

The Apple macOS packages are now fully compliant with Apple macOS software signatures. This allows the FairCom technology to be installed and executed on today's modern Apple macOS operating system without security authorizations.

In order to fully support the macOS software signatures the directory structure inside the package has been adjusted. Everything has been packed inside a "Contents" directory.

Inside the "Contents" directory you'll find sub- directories named "Home", "MacOs" and "_CodeSignature".

"MacOs" and "_CodeSignature" are required to support the code signature. These folders do not contain anything that's directly used but they are still required to support the software signatures.

All the end user contents are now stored inside the "Contents:Home" directory. The remainder of the FairCom directory structure remains the same with "config", "data", "drivers", "server", "tools" and "tranlogs" directories.

13.3 Security Improvements for LDAP Authentication

LDAP client authentication requires passing an actual client password to the FairCom DB server. This requires additional encryption to protect the password in transit. Unique public/private key pairs are generated at runtime and are only used for this one particular connection request.

The encryption algorithm that the client library uses to securely pass the user password to the server when using LDAP authentication was updated to use an AES-GCM encryption with a random initialization vector (iv). Previously, it used AES-CBC encryption with a hard-coded vector.



The following Framework client interfaces are affected:

- All C library based ISAM components
- All SQL drivers including ODBC, Direct SQL (and other drivers using ctsqlapi)
- JAVA JDBC
- ADO.NET

If a client library that uses AES-GCM for LDAP authentication connects to a server that does not support it, the connection attempt now fails with error code **1179**, **CLIENT_LOGON_REQUIRES_AES_GCM** at the ISAM level, or **error -18179** at the SQL level.

If a client library that does not use AES-GCM for LDAP authentication connects to a server that use AES-GCM authentication at the ISAM level, the connection attempt now fails with **error code 941** or **1180 (SERVER_LOGON_REQUIRES_AES_GCM)**, or error code **-17941** or **-18180** at the SQL level.

Examples

Prior client against new server:

```
./ctstat -vas -u "IT Bob,ou=support" -p "bobber1" -s FAIRCOMS
./ctstat: INTISAMX(FAIRCOMS) = 941
```

New client against prior server:

```
./ctstat -vas -u "IT Bob,ou=support" -p "bobber1" -s FAIRCOMS2
./ctstat: INTISAMX(FAIRCOMS2) = 1179
```

LIMITATIONS:

- The ADO.NET driver uses AES-CBC with a random iv, since AES-GCM is not supported by the .NET Framework (only .NET Core supports it).
- The JDBC driver can be compiled with Java 1.7, but it requires Java 1.8 or later in order to use AES-GCM.

AFFECTED COMPONENTS: ISAM and SQL server and client, JDBC and ADO.NET drivers.

LDAP authentication failure no longer incorrectly indicates timeout error on Linux/Unix

LDAP authentication failure could incorrectly indicate a timeout error on Unix/Linux. A timeout error may be logged incorrectly if a different LDAP error occurred. In this situation, all connections could begin failing with the following *CTSTATUS.FCS* message:

```
LDAP_AUTH: LDAP authentication of application ID failed: Timed out (4294967291)
```

The logic has been modified to correct this.



LDAP_LOCAL_PREFIX option to filter LDAP authentication by username

When LDAP_SERVER is enabled, all non-admin users authenticate using LDAP. The LDAP_LOCAL_PREFIX <prefix> keyword allows filtering certain users to use local c-tree authentication if their user name begins with <prefix>. This prefix comparison is case-insensitive.

Note: If the user account existed prior to this V12 change, local authentication may fail with **LDRQ_ERR** (985) or SQL error (-17985): CT - Logon is denied because this user account requires LDAP authentication, but c-tree Server has not enabled LDAP authentication. If this occurs, the user account will need to be deleted and re-created.

LDAP errors generate correct error messages at login

With LDAP_GROUP_CHECK enabled, a login attempt failed with an unusual error(-17032): CT - Attempt to delete record twice in a row. Other incorrect error returns were also possible. The correct LDAP errors were already logged in *CTSTATUS.FCS* and the logic has been modified to now return the correct LDAP-specific errors.

LDAP - Server crash on Linux using LDAP with SSL fixed

With LDAP and SSL authentication enabled, a server crash was seen on Linux if a high rate of connection attempts was made. The logic has been modified to correct this.

LDAP support for Solaris

LDAP support has been enabled for the Solaris platform. We have added support for either native Solaris LDAP or OpenLdap.

Note: This introduces a runtime dependency on *libssl.so* for Solaris.

A new keyword has been added:

```
LDAP_MODULE <name>
```

This keyword allows specifying the LDAP shared library to load. The default value is *libldap.so*. On Solaris, *libldap.so* is the native LDAP library, so this should be used to specify the name of the OpenLdap version of *libldap.so*.

The LDAP_MODULE keyword is not supported on windows.

Note: To use LDAP authentication, you must updating the client side as well as the server.



LDAP Connection and File Open Passwords Memory Leaks Fixed

Memory leaks were detected with LDAP connections or file open passwords:

1. When using LDAP authentication, memory was leaked on each connection (both client and server).
2. If using a file password, memory was leaked on client and server on each file open.

Certain client-side and server-side functions were not calling a function that frees memory allocated by OpenSSL when encryption and OpenSSL ciphers are enabled (the default in V12). The logic has been modified to correct this.

Workaround: OpenSSL-based encryption can be disabled on the server-side with the *ctsrvr.cfg* keyword `OPENSSL_ENCRYPTION NO`, avoiding the memory leak on the server-side. However, if otherwise using `ADVANCED_ENCRYPTION`, encrypted files are incompatible with mismatched `OPENSSL_ENCRYPTION` settings.



13.4 Transport Layer Security Secures Data in Transit between Network FairCom DB Clients and Servers

Note: FairCom DB 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. FairCom DB 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. FairCom DB 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 *ctsvr.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_tcpip6"
```

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_tcpip6
```

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 FairCom DB 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 FairCom DB 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 ctreesql 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 TCP/IP_CHECK_DEAD_CLIENTS
(</doc/ctserver/compatibility-tcpip-check-dead-clients-config.htm>)

Advanced SSL Certificate Options

The `ssl_certificate` keyword in the `config/cthhttpd.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/cthhttpd.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.



OpenSSL Updated to 3.0 for Latest Connection Security

FairCom DB and other FairCom products such as Replication Manager, are now updated to OpenSSL 3.0 for data in transit security (TLS). This provides additional updated levels of security with minimal if any changes to your current application TLS support. Notably, selected algorithms have been added and/or replaced. It is a good time to review your cipher usage when moving to this latest support.

Our OpenSSL 3.0 support now also includes fips-140-2 conformance levels for advanced sensitive applications such as those in financial and health care.

Contact your FairCom support team should you have questions regarding this updated library support.

OPENSSL_ENCRYPTION keyword no longer controls internal cipher

The OPENSSL_ENCRYPTION keyword no longer modifies the internal cipher.

Troubleshooting and Testing

The following are recommendations to ensure a functioning TLS connection in FIPS mode. These are only the most basic steps to cover. Specific encryption and TLS cipher requirements are the sole responsibility of the end user administrator.

TLS is only supported over TCP/IP connections. If your server and the client are running on the same machine (perhaps for testing purposes), then to configure and test a FIPS mode TLS connection requires disabling default localhost shared memory connections. To do so, comment the `COMM_PROTOCOL FSHAREMM` line in your `ctsrvr.cfg` server configuration by adding a semi-colon (;) at the beginning of the line:

```
;COMM_PROTOCOL FSHAREMM
```

This will force all client connections to use TCP/IP and TLS.

Startup Checklist

Upon a successful start of the FairCom DB Server, log entries will be located in `CTSTATUS.FCS` indicating that FIPS is enabled. For example:

```
- User# 00001      OpenSSL FIPS Cryptographic provider in use
```

For diagnostic logging the following environment variable can be set for your FairCom DB client application process:

```
FAIRCOM_FIPS_CLIENT_LOG = <logfile>
```

When enabled this logs verification status from the client application side to the specified log file. This is included primarily for development purposes to verify FIPS enabled OpenSSL encryption is in use.

When the client is trying to connect to the server...



- **Error #0** means the client is attempting to use FIPS mode. That is, `FAIRCOM_FIPS_CLIENT_MODE` environment variable is defined as `YES`, but the application isn't finding `fips.dll` (or `libfips.so`) in its working directory.
- **Error #133** happens when the server is not running. You may be using an evaluation license that has a 3 hour time-out, and the Server has timed out and stopped running.
- **Error #1104** is reported when the server requires TLS logins. That is, `SSL_CONNECTIONS_ONLY` is configured as `YES` within the `SUBSYSTEM COMM_PROTOCOL SSL` block of your server configuration. However, the client is not specifying a TLS communication string. For example, `^fssltcp` for ISAM connections and `ssl:` for SQL connections.

Client FIPS Configuration for TLS Connections

The FairCom provided `fips.dll` (and `libfips.so`) is also required to enable FIPS TLS on the FairCom DB client side. This file can be copied from your FairCom DB Server directory (`\FairCom-DB*\server`) into your client application area. Include this file with your application(s) and tools linked with FairCom client-side libraries.

When started in FIPS mode, the client library attempts to load the OpenSSL fips module (`fips.dll` or `libfips.so`) from its working directory during initialization. If an error occurs during FIPS initialization, **FIPS_INIT_ERROR (1182)** is returned from the FairCom DB initialization routine `ctThrdInit()`.

To enable FairCom linked client-side applications to use FIPS, set the following environment variable for your FairCom DB client application process:

```
FAIRCOM_FIPS_CLIENT_MODE=YES
```

For diagnostic logging the following environment variable can be set for your FairCom DB client application process:

```
FAIRCOM_FIPS_CLIENT_LOG = <logfile>
```

When enabled, this logs verification status from the client application side to the specified log file. FIPS initialization messages (and other errors) will be written to this file, and successful FIPS initialization will log the following messages:

```
Using OpenSSL configuration file ./faircomssl.cnf
OpenSSL FIPS Cryptographic provider in use
```

This is included primarily for development purposes to verify FIPS enabled OpenSSL encryption is in use.

FIPS initialization creates the mandatory configuration file `./faircomssl.cnf`, if it doesn't already exist.

This client environment configuration applies to all connections using FairCom DB SQL drivers and FairCom DB client libraries. Note that some FairCom DB APIs, such as the Java-based c-treeDB Java API JTDB, use a "native code" FairCom library interface (`mtclient.dll`) to connect with the database. These should enable `FAIRCOM_FIPS_CLIENT_MODE=YES`.



Note that C# ADO.NET and Java JDBC drivers must enable FIPS support as provided by those API standards. These drivers are developed entirely in their language respectively and no other underlying FairCom library is used.

Compatibility

There are a number of compatibility effects. When FIPS mode is enabled in the server, the following are **not** supported:

- The FairCom DB user and group credentials file, *FAIRCOM.FCS*, created prior to V10 are not compatible when operating under FIPS mode. You will need to repopulate a new *FAIRCOM.FCS* with your user and group credentials.
- Client Connections from prior client versions will not be able to connect in a FIPS compliant mode.
- Encrypted Files created using KDFv1. This includes all encrypted files created with an earlier version of FairComDB. *If this applies to you, contact FairCom for migration information. Documentation is available upon request.*

FIPS-compliant OpenSSL 3.0 libraries for Windows and Linux

This delivery includes OpenSSL 3.0 static libraries for Windows and Linux that are compliant with the Federal Information Processing Standard (FIPS). (The Linux version was built on CentOS 6 x64 and the Windows version was built on Visual Studio 2017.)

Be sure to see the note in the installation, which includes the following:

With the default OpenSSL installation comes a FIPS provider module, which needs some post-installation attention, without which it will not be usable.

This involves using the following command:

```
$ openssl fipsinstall
```

See the `openssl-fipsinstall(1)` manual for details and examples.

The install file also includes the following information:

Setting the FIPS HMAC key

```
--fips-key=value
```

As part of its self-test validation, the FIPS module must verify itself by performing a SHA-256 HMAC computation on itself. The default key is the SHA256 value of "the holy handgrenade of antioch" and is sufficient for meeting the FIPS requirements.

To change the key to a different value, use this flag. The value should be a hex string no more than 64 characters.

Client-side Support Added for FIPS

To enable FIPS mode on the client side, set the environment variable `FAIRCOM_FIPS_MODE=Y`. This applies to connections using native SQL and FairCom DB client libraries. Non-native clients (e.g. C# or JDBC) must enable FIPS support as provided by those environments. Note that some non-native APIs (e.g. JTDB) use a native interface to connect with the database. These should



enable `FAIRCOM_FIPS_MODE=Y`, and also look at enabling FIPS support within the non-native environment.

If desired, it is possible to compile FairCom DB clients with macro `#define ctFeatFIPS_ONLY_CLIENT` to always enable FIPS mode.

When started in FIPS mode, the client library will attempt to load the OpenSSL fips module from its working directory (`fips.dll` or `libfips.so`) during initialization. If an error occurs during FIPS initialization, **FIPS_INIT_ERROR (1182)** will be returned from `ctThrdInit()`. To learn more about FIPS initialization, the environment variable `CTSTATUS` may be set to a file name. FIPS initialization messages (and other errors) will be written to this file, and successful FIPS initialization will log the following messages:

```
Using OpenSSL configuration file ./faircomssl.cnf
OpenSSL FIPS Cryptographic provider in use
```

FIPS initialization will also create the mandatory configuration file, `./faircomssl.cnf`, if it doesn't already exist.

Note: Enabling FIPS mode only controls the OpenSSL encryption routines that are used, it does NOT require their usage or modify existing default settings. For example, TLS encrypted TCP/IP communication must still be explicitly enabled.

SSL_CIPHERS

SSL_CIPHERS

(optional) - If specified this option sets the encryption ciphers that are allowed to be used for encrypting the SSL connection. Ciphers are separated by a colon, (:). An exclamation point (!) explicitly disables a cipher. @STRENGTH sorts the list in order of encryption algorithm key length. For more information, see <https://www.openssl.org/> <https://www.openssl.org/>

Default SSL_CIPHERS (as of FairCom DB V12.5)

```
ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```



Ephemeral Elliptic Curve Diffie-Hellman Cipher (ECDHE) now Supported for TLS Connections

Attempting to establish a TLS/SSL connection using an ephemeral elliptic curve Diffie-Hellman cipher such as ECDHE-RSA-AES256-SHA384 failed with error **1104**. Diagnostic logging showed the server's SSL accept call failed with a "no shared cipher" error. This was because the server did not enable any elliptic curves to use for SSL connections. The server now enables a default set of elliptic curves.

Specify Diffie-Hellman Parameters in a Server Certificate File

A subset of SSL ciphers require Diffie-Hellman key exchange parameters. Supply these parameter values with `KEY_EXCHANGE_PARAMS` in your SSL parameter configuration subsystem block. For complete flexibility, it is also possible to specify your Diffie-Hellman parameters directly in your server certificate file. If they are present in the certificate file, they are used. If they are not present in the certificate file and `KEY_EXCHANGE_PARAMS` is configured, the `KEY_EXCHANGE_PARAMS` configuration parameters are used.

The following SSL ciphers require Diffie-Hellman parameters:

- `ssl_ciphers DHE-RSA-AES128-SHA256`
- `ssl_ciphers DHE-RSA-AES256-SHA256`
- `ssl_ciphers DHE-RSA-AES128-GCM-SHA256`
- `ssl_ciphers DHE-RSA-AES256-GCM-SHA384`
- `ssl_ciphers DHE-DSS-AES128-SHA256`
- `ssl_ciphers DHE-DSS-AES256-SHA256`
- `ssl_ciphers DHE-DSS-AES128-GCM-SHA256`
- `ssl_ciphers DHE-DSS-AES256-GCM-SHA384`

Affected Components: Server

Server-side Diagnostics

Logging SSL diagnostic messages is improved with a configurable log file. To enable server-side SSL diagnostic logging add `DEBUG_LOG` to `SUBSYSTEM COMM_PROTOCOL SSL` block in `ctsrvr.cfg`. The log file is created in your server's `LOCAL_DIRECTORY` directory. For example:

```
SUBSYSTEM COMM_PROTOCOL SSL {  
  DEBUG_LOG ssl.log  
}
```

TLS/SSL Client Diagnostics

To enable SSL logging for a client, set the `CTSSL_DEBUG_LOG` environment variable to a log file name.



Diffie-Hellman parameters can now be read from server certificate file if present

A subset of SSL ciphers require Diffie-Hellman key exchange parameters. Supply these parameter values with `KEY_EXCHANGE_PARAMS` in your SSL parameter configuration subsystem block. For complete flexibility, it is also possible to specify your Diffie-Hellman parameters directly in your server certificate file. If they are present in the certificate file, they are used. If they are not present in the certificate file and `KEY_EXCHANGE_PARAMS` is configured, the `KEY_EXCHANGE_PARAMS` configuration parameters are used.

The following SSL ciphers require Diffie-Hellman parameters:

- `ssl_ciphers DHE-RSA-AES128-SHA256`
- `ssl_ciphers DHE-RSA-AES256-SHA256`
- `ssl_ciphers DHE-RSA-AES128-GCM-SHA256`
- `ssl_ciphers DHE-RSA-AES256-GCM-SHA384`
- `ssl_ciphers DHE-DSS-AES128-SHA256`
- `ssl_ciphers DHE-DSS-AES256-SHA256`
- `ssl_ciphers DHE-DSS-AES128-GCM-SHA256`
- `ssl_ciphers DHE-DSS-AES256-GCM-SHA384`

Affected Components: Server

FairCom DB Server FIPS Support Details

Support for using the FIPS module of OpenSSL Version 3.0 has been added, requiring OpenSSL Version 3.0 or later.

FIPS-140-2 is a U.S government computer security standard for a cryptographic module validation program. OpenSSL provides a specific validated FIPS Object Model and FairCom DB now incorporates this support.

https://en.wikipedia.org/wiki/FIPS_140-2 (https://en.wikipedia.org/wiki/FIPS_140-2)

FairCom DB FIPS-140-2 encryption support is strictly qualified to the AES encryption implementation. When operating in FIPS mode, the only supported encryption algorithm is AES as implemented in the OpenSSL FIPS module. SHA2 and SHA3 are the only available secure digests when operating in FIPS mode. All prior encryption and secure digest usage is disabled when operating in FIPS mode.

AES Data at Rest Encryption

Enabling FairCom DB FIPS mode requires a single configuration option to be set. Once set, FairCom DB only allows a limited complement of encryption cipher. These are provided in a specific OpenSSL module.

The new FairCom provided OpenSSL FIPS module (*fips.dll* or *libfips.so*) is located in the FairCom DB Server folder. Be sure this is present when starting your “**faircom.exe**” (**faircom** on Linux) binary.



Note that a new license file is required for the FIPS support. The necessary license (ctsrvr*.lic) is included in this package. Do not use any previous licenses with this version.

To enable FIPS mode, use the new *ctsrvr.cfg* configuration `FIPS_ENCRYPTION YES` (default NO). FIPS enabled mode only restricts and validates specific and limited encryption routines that are called from the FIPS certified module (AES symmetric encryption SHA2 and SHA3 secure digests, and a limited set of TLS ciphers). It does not enable disk or network encryption by itself.

Upon a successful start of the FairCom DB Server, log entries will be located in *CTSTATUS.FCS* indicating that FIPS is enabled. For example:

```
- User# 00001      OpenSSL FIPS Cryptographic provider in use
```

The following pre-existing server configurations are used to enable various FairCom DB file encryption:

- `ADVANCED_ENCRYPTION YES`
- `LOG_ENCRYPT YES`
- `ADMIN_ENCRYPT YES`

Additionally, file encryption must be enabled for each individual file used by the application.

(*) `LOG_ENCRYPT` is not functionally required to enable FIPS mode data encryption. However, transaction logs contain raw application data. For an application to be considered truly FIPS compliant, it is strongly recommended to encrypt FairCom DB transaction logs.

IMPORTANT: It is not possible to encrypt existing FairCom DB transaction logs. You must stop your FairCom DB server process and ensure all current logs are fully replayed for all transactions to be applied to all data and index files and then the existing logs are deleted. Only freshly created logs will be encrypted. Further, do not mix unencrypted and encrypted logs in a transition phase, or a window of non recoverability is opened exposing your environment to potential data loss,

FIPS mode uses only the OpenSSL FIPS module for server cryptographic functionality with the following exceptions:

- `SUBSYSTEM EXTERNAL_KEY_STORE AWS` uses AWS provided libraries to interface with AWS.
- Creating or using `MASTER_KEY_STORE` on Microsoft Windows uses the Windows Data Protection API (DPAPI).

TLS Network Encryption

TLS FIPS mode for network encryption requires configurations on both the server and client application side. Server configuration is described here. Client configuration will be found in the following section.

1. Enable `FIPS_ENCRYPTION YES` in your server configuration.
2. Ensure *fips.dll* or *libfips.so* is located in the server process environment.
3. Uncomment and configure your `SUBSYSTEM COMM_PROTOCOL SSL` subsection with appropriate options for your organization.

Note that FIPS mode limits available ciphers for your security administrator to review.

Restart your server and TLS connections should be available with no further considerations.



NetworkTLS encryption is enabled as discussed here: Transport
(/doc/ctserver/tls-encryption.htm)Layer Security Secures Data in Transit between Network
FairCom DB Clients and Servers

Quick Start - Steps to Enable FIPS-140-2 AES and SSL Encryption Modules

This section provides the minimum steps to enable FIPS verified implementations of OpenSSL
AES and SSL encryption for the FairCom DB Server.

*For conversions of existing encrypted files, contact FairCom
(https://www.faircom.com/company#contact_us) for additional documentation and procedures
specific to your version and environment.*

AES Data Encryption

1. As previously supported, first enable advanced data encryption with the following server
configuration in *ctsrvr.cfg*
ADVANCED_ENCRYPTION YES
2. Optionally, enable transaction log encryption
LOG_ENCRYPT YES
3. Enable FIPS AES encryption with the following additional configuration
FIPS_ENCRYPTION YES
4. When advanced data encryption is enabled, FairCom DB requires a master password phrase
at server startup. Create a master password verification file, *ctsrvr.pvf*
(/doc/ctserver/34330.htm) ,using the “ctcpvf” utility and place that pvf file in the server folder.
DO NOT FORGET YOUR MASTER PASSWORD PHRASE
5. Start your FairCom DB Server and provide your password phrase at startup. A key store file
is recommended for automated unattended Server startup in a “lights out” environment. See
Master Key Management section below for details and usage.

Once the server is started in Advanced Encryption mode, files are then capable of encryption,
however, are not encrypted by default. File encryption is a file by file administrative decision and
can be applied at file create time by an application developer. Data at rest encryption is
transparent to an application. For example, for ISAM level creates use **SetEncryption()**
(<https://docs.faircom.com/doc/ctreeplus/setencryption.htm>), or see Storage Attributes
(<https://docs.faircom.com/doc/sqlref/56079.htm>) for SQL or later applied by an administrator (see
the “-encrypt” switch for *ctcmpcif.exe* (https://docs.faircom.com/doc/cmdline/ctcmpcif-util_1.htm)
for encrypting existing files).

There are no client side configurations required to enable advanced data encryption for data at
rest.

Full data encryption details are available here: Advanced Data Encryption
(<https://docs.faircom.com/doc/ctserver/advanced-encryption.htm>)

Master Key Management

It's important to ensure that end user administrators in the field do NOT lose their FairCom DB
Server master password. FairCom can not, nor has any ability to, recover this password. If a
master password is lost, their corresponding encrypted data is lost. See the following for methods
FairCom provides to aid with this task:

FairCom DB Key Store - This method provides a local protected key for unattended key
presentation to the server. However, note that on Windows this support is provided with the



Microsoft Windows Data Protection API (DPAPI). FIPS conformance level should be determined by the local system administrator.

AWS Secrets Manager (<https://docs.faircom.com/doc/ctreeplus/76777.htm>) - This provides remote “escrowed” key storage. However, for full FIPS conformance, there may be additional integration provisions to consider. For example, does AWS key storage provide FIPS compliance? This exceeds the scope of this vendor’s control of master key management.

Encrypted Data Master Key Library

(<https://docs.faircom.com/doc/ctreeplus/EncryptedDataMasterKeyLibrary.htm>) - This provides custom application vendor key management capabilities. However, for full FIPS conformance, there may be additional integration provisions to consider. This exceeds the scope of this vendor’s control of master key management.

TLS Communications Encryption

FairCom DB TLS ((SSL) support requires both server and client configuration.

1. Enable FIPS AES encryption with the following additional server configuration
`FIPS_ENCRYPTION YES`
2. Uncomment the SSL subsystem communication protocol configuration block in *ctsrvr.cfg* and configure with your organizational certificates and ciphers before starting your FairCom DB Server.
`SUBSYSTEM COMM_PROTOCOL SSL`
3. To enable FairCom linked client-side applications to use FIPS, set the following environment variable for your FairCom DB client application process:
`FAIRCOM_FIPS_CLIENT_MODE=YES`
 The *fips.dll* or *libfips.so* object **must** be present in your client application executable process space when this option is enabled.
4. To request an SSL-enabled ISAM client connection, add “^fssltp” to the end of your ISAM connection string. For example:
`FAIRCOMS@localhost^fssltp`
5. To request an SSL-enabled SQL connection, prepend ssl: to your SQL connection string. For example:
`isql -u admin -p ADMIN ssl:6597@localhost:ctreesql`

Review specific connection string formats for your application development framework.

Full TLS details are available here: Transport Layer Security

(<https://docs.faircom.com/doc/ctreeplus/tls-encryption.htm>)

Client FIPS Configuration for TLS Connections

The FairCom provided *fips.dll* (and *libfips.so*) is also required to enable FIPS TLS on the FairCom DB client side. This file can be copied from your FairCom DB Server directory (*\FairCom-DB*\server*) into your client application area. Include this file with your application(s) and tools linked with FairCom client-side libraries.

When started in FIPS mode, the client library attempts to load the OpenSSL fips module (*fips.dll* or *libfips.so*) from its working directory during initialization. If an error occurs during FIPS initialization, **FIPS_INIT_ERROR (1182)** is returned from the FairCom DB initialization routine **ctThrdInit()**.



To enable FairCom linked client-side applications to use FIPS, set the following environment variable for your FairCom DB client application process:

```
FAIRCOM_FIPS_CLIENT_MODE=YES
```

For diagnostic logging the following environment variable can be set for your FairCom DB client application process:

```
FAIRCOM_FIPS_CLIENT_LOG = <logfile>
```

When enabled, this logs verification status from the client application side to the specified log file. FIPS initialization messages (and other errors) will be written to this file, and successful FIPS initialization will log the following messages:

```
Using OpenSSL configuration file ./faircomssl.cnf
OpenSSL FIPS Cryptographic provider in use
```

This is included primarily for development purposes to verify FIPS enabled OpenSSL encryption is in use.

FIPS initialization creates the mandatory configuration file `./faircomssl.cnf`, if it doesn't already exist.

This client environment configuration applies to all connections using FairCom DB SQL drivers and FairCom DB client libraries. Note that some FairCom DB APIs, such as the Java-based c-treeDB Java API JTDB, use a “native code” FairCom library interface (`mtclient.dll`) to connect with the database. These should enable `FAIRCOM_FIPS_CLIENT_MODE=YES`.

Note that C# ADO.NET and Java JDBC drivers must enable FIPS support as provided by those API standards. These drivers are developed entirely in their language respectively and no other underlying FairCom library is used.

Compatibility

There are a number of compatibility effects. When FIPS mode is enabled in the server, the following are **not** supported:

- The FairCom DB user and group credentials file, `FAIRCOM.FCS`, created prior to V10 are not compatible when operating under FIPS mode. You will need to repopulate a new `FAIRCOM.FCS` with your user and group credentials.
- Client Connections from prior client versions will not be able to connect in a FIPS compliant mode.
- Encrypted Files created using KDFv1. This includes all encrypted files created with an earlier version of FairComDB. *If this applies to you, contact FairCom for migration information. Documentation is available upon request.*

Troubleshooting and Testing

The following are recommendations to ensure a functioning TLS connection in FIPS mode. These are only the most basic steps to cover. Specific encryption and TLS cipher requirements are the sole responsibility of the end user administrator.

TLS is only supported over TCP/IP connections. If your server and the client are running on the same machine (perhaps for testing purposes), then to configure and test a FIPS mode TLS connection requires disabling default localhost shared memory connections. To do so, comment



the `COMM_PROTOCOL FSHAREMM` line in your `ctsrvr.cfg` server configuration by adding a semi-colon (;) at the beginning of the line:

```
;COMM_PROTOCOL FSHAREMM
```

This will force all client connections to use TCP/IP and TLS.

Startup Checklist

Upon a successful start of the FairCom DB Server, log entries will be located in `CTSTATUS.FCS` indicating that FIPS is enabled. For example:

```
- User# 00001      OpenSSL FIPS Cryptographic provider in use
```

For diagnostic logging the following environment variable can be set for your FairCom DB client application process:

```
FAIRCOM_FIPS_CLIENT_LOG = <logfile>
```

When enabled this logs verification status from the client application side to the specified log file. This is included primarily for development purposes to verify FIPS enabled OpenSSL encryption is in use.

When the client is trying to connect to the server...

- **Error #0** means the client is attempting to use FIPS mode. That is, `FAIRCOM_FIPS_CLIENT_MODE` environment variable is defined as `YES`, but the application isn't finding `fips.dll` (or `libfips.so`) in its working directory.
- **Error #133** happens when the server is not running. You may be using an evaluation license that has a 3 hour time-out, and the Server has timed out and stopped running.
- **Error #1104** is reported when the server requires TLS logins. That is, `SSL_CONNECTIONS_ONLY` is configured as `YES` within the `SUBSYSTEM COMM_PROTOCOL SSL` block of your server configuration. However, the client is not specifying a TLS communication string. For example, `^fssltcp` for ISAM connections and `ssl:` for SQL connections.

13.5 X509 Certificate based authentication

DETAILS: Added support for X509 based client/server authentication. This initial revision only supports X509 authentication when using SSL encrypted TCP/IP for ctree.

To use x509 authentication, the client must provide a PEM formatted x509 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's x509 will be parsed by the server to extract a username which determines the database permissions that are 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 an x509 certificate is provided by the client, the username and password arguments passed to `INTISAMX()` are ignored.



New Server keywords:

Added under SUBSYSTEM COMM_PROTOCOL SSL

VERIFY_CLIENT_CERTIFICATE { YES | NO } default NO

Server requires the client to supply a x509 certificate. Unless x509_AUTHENTICATION is enabled, a username and password must still be supplied by the client.

x509_AUTHENTICATION { YES | NO } default NO

If the client provides a x509 certificate at logon, use it for authentication and database authorization rather than a username/password. By default, the only trusted CA is the root CA of the SERVER_CERTIFICATE_FILE. So the client x509 certificate must be signed by the same root CA as the server. Combine with VERIFY_CLIENT_CERTIFICATE YES to make the x509 mandatory.

If x509_AUTHENTICATION is enabled, the following keywords are used to extract a username from the subject field of a successfully authenticated client x509 certificate.

x509_PATH: Mandatory. Specifies which relative distinguished name (RDN) component of the full distinguished name (DN) to parse. This should be specified using the short form name (e.g. CN).

x509_PREFIX: Optional. If specified the value found in x509_PATH must contain this prefix, which is removed when forming the username.

x509_DELIMITER: Optional. If specified the value found in x509_PATH must contain this suffix, which is removed when forming the username.

x509_REQUIREMENT_PATH: Optional. If specified, the DN must contain this RDN.

x509_REQUIREMENT: Optional. If specified, the RDN specified by x509_REQUIREMENT_PATH must match this value.

NOTE: all name matching is case insensitive, as is the resulting user name used for login.

Example:

Server has the following configuration:

x509_AUTHENTICATION YES

x509_PATH CN

x509_REQUIREMENT_PATH O

x509_REQUIREMENT faircom inc

A valid certificate with the subject:



Subject: C=US, ST=Missouri, O=FairCom Inc, OU=R&D, CN=John Doe,
emailAddress=john.doe@faircom.com

would resolve the user name as "John Doe"

A valid certificate with the subject:

Subject: C=US, ST=Missouri, O=Acme Inc, OU=R&D, CN=admin,
emailAddress=john.doe@gmail.com

would not resolve a user name, as "Acme Inc" does not match the x509_REQUIREMENT
"FairCom Inc" and an error would be returned.

Use the SUBSYSTEM COMM_PROTOCOL SSL keyword DEBUG_LOG = <log file> to aid in
debugging certificate name matching.

The following function has been enhanced for the client to configure x509 authentication

NINT ctDECL ctSetCommProtocolOption (NINT option,pVOID value)

New options:

ctCOMMOPT_FSSLTCP_CLIENT_CERTIFICATE - value references a char * referencing the
filename of a PEM formatted file with the full certificate chain the client will present for
authentication. This file name can be overridden by the environment variable
CTSSL_CLIENT_CERTIFICATE

ctCOMMOPT_FSSLTCP_CLIENT_KEY - value references a char * referencing the filename of a
PEM formatted file containing the private key used to sign the client certificate. This file name can
be overridden by the environment variable

CTSSL_CLIENT_KEY. The PEM may optionally be encrypted.

ctCOMMOPT_FSSLTCP_CLIENT_PASSPHRASE - value references a char * used to decrypt
the client's private key if it is encrypted.

x509 authentication is not supported in combination with LDAP authentication.

13.6 HTTPD Service Extra Headers Configuration

FairCom DB includes an httpd service enabling web-based services including a REST API, web
socket protocols, browser-based utilities, and other future support. Secure sockets over HTTPS is
a core feature of this support. It was noted that HSTS strict transport security policy could not be
enforced as the appropriate headers fields were not included. Specifically,
"Strict-Transport-Security" is defined for HSTS support.



An additional *services.json* configuration has been added to include custom site defined headers for the HTTPD service. When included and the user agent (UA) conforms to this policy, HSTS enforcement will be in effect.

```
"extra_headers": [  
  "Strict-Transport-Security: max-age=31536000; includeSubdomains; preload",  
  "Special-Custom-Header: FairCom",  
  "Access-Control-Allow-Origin: https://localhost:8443"  
],
```

Refer to https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security <http://www.> for additional details about this HTTP security policy.

Note: This support may be revised in future FairCom DB releases as core HTTPD service components are updated.

14. Configuration

14.1 Important - PAGE_SIZE Conversions

FairCom V13/V12 has a new default page size setting of 32K. In an effort to simplify the upgrade process for this new support, FairCom has implemented automatic conversions of its internal control files. This section describes this new support, and includes the recommended conversion process for converting application data.

In V12.0, to change the PAGE_SIZE required manually calling rebuild procedures to convert the FairCom Server's superfiles. With V12.0.1 and later, this is done automatically during the first FairCom Server startup.

Background

Operating systems have a default file block size they read and write data in. Most recent Linux and Windows operating systems are 4K. Other operating systems will vary. Setting the index page size to match the OS or multiples of OS file block size can improve I/O performance. FairCom Database Engine supports setting PAGE_SIZE from 512 bytes up to a maximum of 64K (65,536).

The PAGE_SIZE setting used by FairCom DB can have a dramatic effect on database performance. In most applications, a larger page size is a benefit, especially if index keys are relatively large (> 20 bytes or so), or there is a reasonable number of key values (consider 100K key values or greater).

You should set your c-tree server PAGE_SIZE to at least the OS file block size, nothing smaller. For example, if you set c-tree's PAGE_SIZE to 2K, then on Windows you are reading 4K at the OS level and throwing away 2K every time.

For most applications, 32K is a well performing value. If you are looking to fully optimize your application, and if your files are fairly large, you might gain performance by increasing this setting, in multiples of your operating system page size value. The best method for tuning this value is to try a few different sizes with exact application configurations and a copy of your data. For example, increase from 8K to 16K and if your performance increases, then continue and test with 32K. Once your performance starts dropping, then scale back in smaller increments looking for the sweet spot for your application. It's possible your application sweet spot might be less than 32K.

For V13/V12, FairCom has increased the default PAGE_SIZE setting to 32K (V10 and V11 defaulted to 8K). There are two options for addressing this topic:

1. Staying with your current PAGE_SIZE setting is an option. You can set PAGE_SIZE X to your current value (where X is your current PAGE_SIZE). This will prevent the need to rebuild your application indexes.



2. Rebuild your application indexes to utilize either FairCom's new 32K default (or a `PAGE_SIZE` better suited to your application) by following the testing procedure above.

Changing `PAGE_SIZE`

Upgrading FairCom Servers to a different page size is a maintenance task that should be done carefully after a full, reliable backup. Failure to perform the procedures (provided later in this section) will cause the system to be halted with a `KSIZ_ERR`, error **40**.

In V12.0.1 and later, the server will automatically convert its internal superfiles to the server's current page size. However, you are responsible for converting your application index files and any superfiles created by your FairCom DB based application.

When the `PAGE_SIZE` server configuration option is changed, FairCom applies this change to all FairCom server-controlled files. At server startup we check the page size of the following FairCom files:

- *ctdbdict.fsd*
- *Database dictionaries*
- *FAIRCOM.FCS*
- *SEQUENCEDT.FCS*
- *DFRKSTATEDT.FCS*
- *SYSLOGDT.FCS*
- *REPLFFCHGDT.FCS*
- *REPLOGDT.FCS*
- *REPLOGSHIPDT.FCS*
- *REPLSTATEDT.FCS*
- *REPLSYNCDT1.FCS*
- *REPLSYNCDT2.FCS*

If any do not match the newly specified `PAGE_SIZE`, a backup copy of the existing file is made with an *.FCB* extension. The server process then attempts to invoke the FairCom **ctscmp.exe** (**ctscmp** on Linux/Unix) superfile rebuild utility with the new configured `PAGE_SIZE`.

Superfiles require that the `PAGE_SIZE` at open matches the `PAGE_SIZE` at file creation time, or a **KSIZ_ERR** (40) or **SPAG_ERR** (417) error occurs at file open. For *FAIRCOM.FCS*, this prevents the server from starting when operating with a different configured `PAGE_SIZE` setting.

A new configuration keyword has been added to control the automatic page size conversion at startup:

```
PAGE_SIZE_CONVERSION {YES|NO}
```

Default: YES

Limitations:

1. We expect **ctscmp.exe** (**ctscmp** on Linux/Unix) to exist in the process working directory and have permission to be executable by the server process. If this utility does not exist, the server fails to start with the following messages likely logged to *CTSTATUS.FCS*:
 - User# 00001 Wrong PAGESIZE for FAIRCOM.FCS, attempting to convert file from PAGESIZE 8192 to 32768



- ```

- User# 00001 Did not find ctscmp in working directory for Superfile
conversion: 2
- User# 00001 Could not process User/Group Information in
FAIRCOM.FCS: 417
- User# 00001 Could not initialize server. Error: 417
- User# 00001 01 M0 L74 F417 P0x (recur #1) (uerr_cod=417

```
- The `LOCAL_DIRECTORY` keyword must be set and must be different from the working directory. Otherwise, `ctscmp.exe` will encounter a `TCOL_ERR` (537) and fail.
  - Superfile conversion occurs only after auto-recovery. Recovery is likely to fail if run with a different `PAGE_SIZE` setting. We expect any needed recovery should occur before making a major configuration impacting the physical attributes of critical operational files such as authentication information and SQL database dictionaries (system tables).
  - NO FURTHER ATTEMPT IS MADE to convert any other existing indexes. All other application created indexes will require manual rebuilding to convert to a new page size. See the steps below for the best practice procedures.**

The following messages may be found logged to `CTSTATUS.FCS` after a successful conversion:

```

- User# 00001 Wrong PAGESIZE for FAIRCOM.FCS, attempting to convert
file from PAGESIZE 8192 to 32768
- User# 00001 Backup created ../data/\FAIRCOM.FCS =>
../data/\FAIRCOM.FCS.1621883226.FCB
- User# 00001 Superfile conversion successful
- User# 00001 Wrong PAGESIZE for ctddbdict.fsd, attempting to
convert file from PAGESIZE 8192 to 32768
- User# 00001 Backup created ../data/\ctddbdict.fsd =>
../data/\ctddbdict.fsd.1621883232.FCB
- User# 00001 Superfile conversion successful
- User# 00001 Wrong PAGESIZE for
.\ctreeSQL.dbs\SQL_SYS\ctreeSQL.fdd, attempting to convert file from
PAGESIZE 8192 to 32768
- User# 00001 Backup created
../data/\.\ctreeSQL.dbs\SQL_SYS\ctreeSQL.fdd =>
../data/\.\ctreeSQL.dbs\SQL_SYS\ctreeSQL.fdd.1621883237.FCB
- User# 00001 Superfile conversion successful

```

Once correct server operations are confirmed after conversion, the `*.FCB` files can be removed and deleted.

## Changing Application `PAGE_SIZE` Manually

Changing the application index page size is a maintenance task that should be done carefully after a full, reliable backup. When making a `PAGE_SIZE` configuration change with existing indexes, those indexes must be rebuilt using the new node size before they can take advantage of the new configured size.

- `ctrbldif.exe` can be used to make this modification for fixed-length files
- `ctcmpcif.exe` must be used for variable-length files.



Existing indexes with nodes smaller than a configured `PAGE_SIZE` will display the following informational messages in *CTSTATUS.FCS* upon file open:

```
- User# 00027 Mismatched PAGE_SIZE is wasting 75% of Index cache used by
.\ctreeSQL.dbs\admin_custmast.dat
```

This message appears because the smaller node size of the existing index fits into the larger page size value as allocated in the index buffer cache with the remainder of the cache page space wasted. Note that even though in this case the file will open successfully, you are wasting memory. Consider with the message above, if you are using an 8K `PAGE_SIZE` in the FairCom DB Server, and your application index file is set to 2K, then 6K of memory is wasted with each index page loaded into the Index Cache.

FairCom DB V13/V12 defaults to a larger page size configuration recommended for modern client-server applications. However, existing index files must be rebuilt to take advantage of this larger index node size. Opening existing indexes smaller than the configured server value results in a warning message logged to *CTSTATUS.FCS* about wasted memory usage.

For applications that have large numbers of files, or that frequently open and close files, this may result in large numbers of messages to *CTSTATUS.FCS*, and this logging is disabled after the first 20 occurrences.

The following command turns off all `PAGE_SIZE` warnings so *no* warnings are produced:

```
CTSTATUS_MASK WARNING PAGESIZE
```

Setting this keyword disables the logging of warning messages like the one shown above.

For debugging, if it is necessary to see all warnings, you can use the following keyword to log *all* the warnings to *CTSTATUS.FCS*:

```
CTSTATUS_MASK WARNING PAGESIZE_SEEALL
```

**Note:** Existing indexes with a larger page size than the configured `PAGE_SIZE` continue to result in an error **KSIZ\_ERR** (40) on open.

## Rebuilding Application Files

**Recommendation:** If you are considering changing your `PAGE_SIZE` setting, be sure to rebuild ALL of your index files.

If you are reducing your `PAGE_SIZE` setting, and you miss rebuilding an index, and its value is set larger than your current `PAGE_SIZE` setting, then you will get an error 40 (**KSIZ\_ERR**, Index node size too large) when this file is opened and it will need to be rebuilt to resolve this error.

If your index file(s) have a `PAGE_SIZE` set less than your current `PAGE_SIZE` setting in *ctsrvr.cfg*, then you will be wasting memory. The file will open fine, because a smaller index `PAGE_SIZE` will fit in the larger `PAGE_SIZE` set in *ctsrvr.cfg*.

**Recommendation:** Experiment on *copies* of your data and server folder. When you are confident with the results, back up your source data and then change the live production files.



## Compatibility

The page (node) size is a permanent attribute of an index when it is created. Index nodes remains that size until rebuilt or compacted with a different size. There are limitations with existing files using a new larger page size.

- c-tree can open existing indexes with a *smaller* page size than currently configured. There is a minor memory use trade off in doing so. As each index node maps to a single server cache page, and the server cache page is allocated as a page size, the unused space in the cache page is lost. This can be significant. If a server configured for a 32K page size index cache is at 100% capacity of 8K index nodes, *up to 75% of cache memory is unused*. For very large caches this is significant.

**It will be extremely beneficial to rebuild existing indexes and take full advantage of both increased index page size benefits as well as full cache memory usage.**

Opening indexes created with page sizes larger than currently configured results in error 40 (**KSIZ\_ERR**).

- *Superfiles can ONLY be opened with the same configured page size as they were originally created.* This has important implications with critical c-tree housekeeping files:
  - *FAIRCOM.FCS* - This file maintains all user, group and password hash information.
  - *ctdbdict.fsd* - This file maintains the catalog of available databases (SQL and c-treeDB).
  - *<database\_name>.fdd* - This file maintains the SQL database system tables (catalog or dictionary).

Using these existing superfiles with a different page size will result in a server startup failure. Opening a superfile with a different page size also results in error 40 (**KSIZ\_ERR**). You will find this message logged in *CTSTATUS.FCS* should the server make this failed attempt.

## Rebuilding Existing Indexes

The following options are available for rebuilding indexes to take advantage of increased page size.

### ALWAYS MAKE CLEAN BACKUP COPIES OF THESE FILES BEFORE YOU BEGIN THESE PROCEDURES

The index rebuild utility, **ctrblidf**, is the quickest easiest option for most indexes. Simply pass the new **sect** size.

*Remember, each sect = 128 bytes. For 32768 your sect size will be 256.*

```
> ctrblidf mytable.idx -256 ADMIN ADMIN FAIRCOMS
```

For superfiles, use the superfile compact utility, **ctscmp**.

```
> ctscmp ctdbdict.fsd Y 256
```

To make superfiles that can be opened by versions earlier than V12, use **ctscmp** with the **-v11** option. This option forces the utility to avoid introducing V12-specific features to the resulting file.  
Usage: **ctscmp** filename [-v11] [sectors]

## PAGE\_SIZE Change Procedures

For procedures explaining how to rebuild files affected by a change of **PAGE\_SIZE**, see **Adjusting PAGE\_SIZE** ([/doc/FairCom-Installation/AdjustingPAGE\\_SIZE.htm](/doc/FairCom-Installation/AdjustingPAGE_SIZE.htm)) in the *FairCom Installation Guide*.



## 14.2 32K Page Size now Always Set as Default

The default `PAGE_SIZE` for V13/V12 and later is 32K. However, it was noticed in selected cases where the default might not be set to 32K, unless specifically set in `ctsrvr.cfg`. Starting with releases Build 211022 and later, the default is now set to 32K regardless if the value is set in `ctsrvr.cfg`. Setting `PAGE_SIZE` in `ctsrvr.cfg` is available to override the 32K default page size.

**Note:** This modification affects compatibility.

## 14.3 Mask `PAGE_SIZE` Warnings in `CTSTATUS.FCS`

FairCom DB V13/V12 defaults to a larger page size configuration recommended for modern client-server applications. However, existing index files must be rebuilt to take advantage of this larger index node size. Opening existing indexes smaller than the configured server value results in a warning message logged to `CTSTATUS.FCS` about wasted memory usage.

For applications that have large numbers of files, or that frequently open and close files, this may result in large numbers of messages to `CTSTATUS.FCS`, and this logging is disabled after the first 20 occurrences.

The following command turns off all `PAGE_SIZE` warnings so *no* warnings are produced:

```
CTSTATUS_MASK WARNING_PAGESIZE
```

Setting this keyword disables the logging of warning messages like the one shown above.

For debugging, if it is necessary to see all warnings, you can use the following keyword to log *all* the warnings to `CTSTATUS.FCS`:

```
CTSTATUS_MASK WARNING_PAGESIZE_SEEALL
```



## 14.4 Single-line Status Log Messages Improves External Integration

Historically, *CTSTATUS.FCS* log messages listed a timestamp as one text line and the log message on a separate text line. Now it's possible to configure the server to write a single-line message. This feature streamlines integration with third-party monitoring tools that expect messages on a single line.

```
Thu Apr 7 17:46:09 2022 - User# 00005 Thread Start: ctqchkp
Thu Apr 7 17:46:09 2022 - User# 00006 Thread Start: Recycle Bin
Thu Apr 7 17:46:09 2022 - User# 00007 Thread Start: ctdnode
Thu Apr 7 17:46:09 2022 - User# 00009 Thread Start: Non-Transaction Index File Flush
Thu Apr 7 17:46:09 2022 - User# 00010 Thread Start: Transaction Data File Flush
Thu Apr 7 17:46:09 2022 - User# 00008 Thread Start: Non-Transaction Data File Flush
Thu Apr 7 17:46:09 2022 - User# 00011 Thread Start: Transaction Index File Flush
Thu Apr 7 17:46:09 2022 - User# 00012 Thread Start: Non-Tran File Deferred Indexing Thread
Thu Apr 7 17:46:09 2022 - User# 00013 Thread Start: Tran File Deferred Indexing Thread
Thu Apr 7 17:46:09 2022 - User# 00014 Thread Start: Background Index Load Thread
Thu Apr 7 17:46:09 2022 - User# 00015 Thread Start: ctKEEPOPENClosethrd
Thu Apr 7 17:46:09 2022 - User# 00016 Thread Start: ctPART_AUTO_PURGEthrd
Thu Apr 7 17:46:09 2022 - User# 00017 Thread Start: ctDISTcountsThrd
Thu Apr 7 17:46:09 2022 - User# 00018 Thread Start: ctsyslog
Thu Apr 7 17:46:09 2022 - User# 00019 Thread Start: ct_replflushlog_thd
Thu Apr 7 17:46:09 2022 - User# 00020 Thread Start: ctrspac
```

Use `MULTILINE_STATUS_LOG_MESSAGE NO` in *ctsrvr.cfg* for single-line messages. The default is remains for compatibility with the multi-line messages in prior releases.

## 14.5 BINARY CAST of Padding Spaces Retained in VARCHAR Fields

Prior to this modification, we truncated VARCHAR field padding (spaces) when cast from BINARY. Beginning with this modification, we correctly copy the cast buffer as is into the VARCHAR field, and not truncate any string padding with a NUL character, which broke expected data return from the field.

A new server configuration keyword reverts to the old behavior:

```
STRING_TO_BIN_TRIM_SPACES
```

**Note:** This is a *compatibility change*. The new configuration option should be considered a backward compatibility option. That is, we have NEW correct behavior by default. If anyone depended on the old behavior, the configuration keyword can be used to revert to that behavior.



## 14.6 FairComConfig Utility now detects prior .NET configurations

The ADO.NET driver integration could not be used in Visual Studio because Visual Studio was unable to locate the correct driver to use. The following error could be seen: "Unable to find the requested .Net Framework Data Provider. It may not be installed."

A duplicate entry in *machine.config* caused the .NET framework to be unable to identify the correct driver to use with *Ctree.VisualStudio.Data*. Although **FairComConfig** has logic to find duplicate entries in *machine.config*, a change in the display name of the driver between V11 and V12 meant it was unable to locate V11 installations.

**FairComConfig** has been modified to use a different attribute to locate earlier installations: The invariant driver name (*ctree.data.sqlclient*) is not expected to change among versions.

## 14.7 Block Logon when Quiesce is Abandoned

When the server has been quiesced and the connection that established the quiesce disconnects without undoing the quiesce, subsequent connection attempts fail with error **QABN\_ERR** (898), except for special ADMIN user logons like the **ctadmn** utility uses, which are permitted to log on to undo the quiesce.

The server now supports a configuration option to change the behavior in this situation to blocking the connection request instead of returning **QABN\_ERR**. To enable this feature, add the following option to the server configuration file:

```
ABANDONED_QUIET_SUSPEND_LOGON YES
```

This option defaults to NO. It can be changed at runtime using **ctadmn** option 10.

## 14.8 Shared Memory Key Configuration Through Environment Variables

The **SHMEM\_KEY\_ISAM** and **SHMEM\_KEY\_SQL** server configuration options support specifying an environment variable, whose value is substituted for the configuration option value when the server starts up. For example, if the environment variable **MY\_ISAM\_KEY** is set to a numeric value such as 12345 or 0xabcdef before starting the server process, then the following option can be specified in the server configuration file to use this environment variable value for the **SHMEM\_KEY\_ISAM** configuration option value:

```
SHMEM_KEY_ISAM %MY_ISAM_KEY%
```

**Note:** When **SHMEM\_KEY\_ISAM %MY\_ISAM\_KEY%** is active in *ctsrvr.cfg*, the FairCom DB Server process will not start unless the environment variable is also defined. This is the expected behavior for all the environment variable options.



## 14.9 Disable Diagnostic pstack Generation From Shared Memory Errors

A pstack was generated when the error message "unexpected shared memory connection state" was logged, even if `STACK_DUMP OFF` was specified in `ctsrvr.cfg`. The logic has been modified to eliminate this stack.

## 14.10 Background Transaction Data and Index Flush Threads no Longer Default to Immediate

FairCom DB server inadvertently defaulted to IMMEDIATE mode for `TRAN_DATA_FLUSH_SEC` and `TRAN_INDEX_FLUSH_SEC` configuration options even though these options are documented as disabled by default.

The defaults have been modified to set `TRAN_DATA_FLUSH_SEC` and `TRAN_INDEX_FLUSH_SEC` to `rate=1/1`, meaning defer 1 millisecond after every flush.

## 14.11 File Open Automatically Retries on Pending Creation State

The server now supports an extended file open mode that causes an ISAM-level file open call to detect that it failed because the file's creation is pending. In this case, it will retry up to a specified number of times.

To use this feature, call `SetXtdFileOpenMode(ctXOPN_RETRYCREPNDG)` before calling the ISAM-level file open function such as `OPNIFILX()` or `OPNRFILX()`. The extended file mode is only in effect for the next ISAM-level file open call.

The number of retries and delay between retries is configurable using the following `ctsrvr.cfg` server configuration options:

- `BLOCKING_OPEN_RETRY_LIMIT n`, where `n` is the maximum number of retries. A value of zero means infinite retries. Note that the retries will be stopped if the server shuts down or if an administrator terminates the connection.
- `BLOCKING_OPEN_RETRY_DEFER n`, where `n` is the number of milliseconds to delay between retries.

These settings can also be changed at runtime.

Fields have been added to the `ctGSMS` server structure and to the snapshot text output for these settings. Due to these changes, the `ctGSMS` structure version has been changed from 23 to 24.

**Note:** This feature requires a server that supports the `ctXOPN_RETRYCREPNDG` extended open mode. If the server does not support this extended open mode, the extended open mode has no effect.

**Affected Components:** Server, `ctsnpr` utility, c-tree headers used by client application.



## 14.12 Avoid OPEN I-O Exclusive Open Errors

Two or more OPTIONAL OPEN I-O operations on the same file failed with a locked file error and the following error message logged in the FairCom RTG log:

```
ERROR 11:12:-8 CREIFILX(custmasI0.CAD) 0#-1:custmasI0.CAD.dat
```

The same scenario succeeded with the native Micro Focus file system.

If more than one user is attempting to create the same file, the first user that successfully creates the file keeps the file open exclusively for a short amount of time while the file is populated with FairCom RTG internal resources. If another user attempts to create or open the same file while it is open exclusively, the operation fails with c-tree error **12** sysiocod **-8** indicating that the file is locked.

The logic has been modified to change the way files are open in FairCom RTG. With this change, an attempt to open a file that is being created will not fail but instead will wait until the file is available.

The number of retries and delay between retries is configurable using the following *ctsrvr.cfg* server configuration options:

- BLOCKING\_OPEN\_RETRY\_LIMIT *n*, where *n* is the maximum number of retries. A value of zero means infinite retries. Note that the retries will be stopped if the server shuts down or if an administrator terminates the connection. Default value is 10.
- BLOCKING\_OPEN\_RETRY\_DEFER *n*, where *n* is the number of milliseconds to delay between retries. Default value is 100.

### Runtime Configuration

These two settings can be changed at runtime using the **ctadmn.exe** utility (option 10 - "Change Server Settings") or by calling **SetSystemConfigurationOption** (</doc/ctreeplus/setsystemconfigurationoption.htm>) (**ctSETCFG()**).

### Example

```
/* set blocking open retry limit to 20 */
ctSETCFG(setcfgCONFIG_OPTION, "BLOCKING_OPEN_RETRY_LIMIT 20");

/* set blocking open retry defer to 50 milliseconds */
ctSETCFG(setcfgCONFIG_OPTION, "BLOCKING_OPEN_RETRY_DEFER 50");
```



## 14.13 Create system log files as huge files to remove 4GB file size limit

The server now creates system log files as HUGE files so they are not limited to 4GB. HUGE files can be as large as 16 exabytes. At startup, if the server finds existing non-huge system log files, it renames them to *SYSLOGDT.FCA* and *SYSLOGIX.FCA* (first deleting any existing files having those names), and then the system log thread re-creates the system log files as huge files.

To disable the creation of huge system log files, add `SYSLOG NONHUGE` to the server configuration file.

## 14.14 SYSLOG improved reuse of disk space

After a `SYSLOG PURGE`, messages could fail to be inserted because they exceeded the largest available deleted space and the `SYSLOG` was at its non-huge size limit. This can occur because *ctTRNLOG* files do not coalesce adjacent deleted records by default unless a `RECBYT` index is available.

For efficient space reuse a `RECBYT` index has been added to the `SYSLOG` tables. At startup, existing `SYSLOG` tables are checked for the requested `RECBYT` setting, and recreated if necessary. *SYSLOG\*.FCS* files are renamed to *SYSLOG\*.FCA* before this conversion and can be removed once successful `SYSLOG` behavior is confirmed.

Prior behavior can be restored by specifying keyword `SYSLOG V11_REUSE`.

## 14.15 SYSLOG - Efficiently Truncate

`SYSLOG` is a FairCom DB subsystem to secure database events for auditing. Over time this table can become quite large and existing purge functions are not efficient enough. The `SYSLOG()` function supports purging records, optionally filtered by time and by event code. A notable problem is even when all entries are "purged" storage device space is not released leaving potentially very large (up to 4GB) empty files on disk. The solution to this problem is a new `SYSLOG` truncate capability.

With `SYSLOG TRUNCATE`, present in your configuration file if a complete purge is requested (no filtering by time or event) we truncate the file rather than delete individual records.

This approach is much faster and avoids limitations with space reuse. Any users reading from `SYSLOG` will encounter error `FBLK_ERR` if the log is truncated while they have it open. The table must be closed and reopened after receiving this error.

### ctadmn

The `ctadmn` utility has been enhanced to support `SYSLOG` purging with time-based filtering.



The "Monitor Server Activity" menu now has a new option, "Purge SYSLOG entries", which prompts for the number of days of activity to keep. "0 Days" results in a truncate, while larger values use the regular approach.

## 14.16 SYSLOG\_EXCLUDE\_SQL\_USER

When SQL query logging is enabled with `SYSLOG SQL_STATEMENTS`, by default all SQL statements are written to the SYSLOG table. A common use case involves two classes of SQL operations.

- The first are defined SQL statements issued directly (usually hard coded) by an application. These are quite numerous, repetitive, and well defined. Audit logging isn't of great value with these specialized application -specific SQL operations.
- A second more important class are "external" SQL queries such as those from a user via ODBC drivers. Many organization required these queries to be logged and audited for security review.

A new configuration option `SYSLOG_EXCLUDE_SQL_USER <name>`, can now exclude users from `SYSLOG SQL_STATEMENT` logging by user name greatly reducing volume of log entries in the SYSLOG tables.. Multiple users may be excluded by specifying the keyword multiple times. No validation is made that the name specified matches an existing user name.

## 14.17 Server Startup Error 90 Fixed

The FairCom DB Server startup terminated with the following error message in `CTSTATUS.FCS`:

```
User# 00001 01 M0 L33 F0 P0x (recur #1) (uerr_cod=90): 90
```

The logic has been modified to correct this problem.

**Workaround:** Add `COMPATIBILITY KEEP_EMPTY_NODE_ON_READ` to `ctsrvr.cfg`.

**Affected Components:** Server

## 14.18 File Copying/Quiesce Updates

After quiescing the server or blocking a file using the `ctFBsysclose` file block option, if the file is replaced with a different version of the file, having a different c-tree or system file id, when the quiesce or file block ends, a call to open the file may fail with **error 1130** or may hang (if the `PENDING_FILE_OPEN_RETRY_LIMIT` server configuration option is set to zero, which is currently the V13/V12 default). When a file block or quiesce has physically closed a file but is keeping it logically open, if a file that has the same name but a different c-tree file id in its header is copied over the already-open file, and then the unblock or quiesce is undone, an attempt to open the file with a different file name specification might cause the server to consider the file pending open. This is due to the file having the same system file id but different c-tree file id. (By



different file name specification we mean a logically equivalent name to the already open file but specified in a different way, for example `test.dat` and `.test.dat`.) As a result, the open request waits for the pending file open to resolve itself, but in this case the open is not actually pending. This causes the open to hang or return **error 1130** depending on the `PENDING_FILE_OPEN_RETRY_LIMIT` configuration option value.

The file open hanging or failing with **error 1130** is resolved as follows: The server is now able to determine for a system file id list entry whether or not the open of that file is still pending. If an open request does not find the file already open based on file name and c-tree file id and an entry already exists in the system file id list and it is not pending open, the server fails the open call with unique file id **error 463 - UQID\_ERR**.

In addition to the bug fix to resolve the file open error, FairCom has made it possible for the server to check if the file's c-tree or system file id changed when it reopens a file that it closed at the system level. This option is on by default in V13.0 and later. The option is controlled by the server configuration option `SYSCLOSE_REOPEN_CHECK_DIFF`. To enable the option, add `SYSCLOSE_REOPEN_CHECK_DIFF YES` to `ctsrvr.cfg`. To disable use a value of `NO`. The option can also be changed at runtime using the **ctSETCFG()** API function or **ctadmn** utility's change configuration option menu option. When this option is enabled and the server finds that the file's c-tree or system file id has changed, it keeps the file closed, discards updates that have been made to the file in cache pages, prevents all active transactions that updated the file from committing (they will fail with **error 94**), and prevents the file from being updated until it is closed and reopened. The goal is to reduce the likelihood that changes to the file could cause a fatal error for the server if updates are allowed on the changed file.

Attempting to update or open a file that has been found to be different when reopening it fails with the new **error code 1178** (`SYSCLOSE_REOPEN_DIFF_ERR`, A reopen of a file that was closed at system level found a different file). To clear the error, ensure that all connections have closed the file so that the server physically closes the file. Then the newly-copied-in file can be opened.

**AFFECTED COMPONENTS:** server

## 14.19 Default COMPATIBILITY BATCH\_SIGNAL Behavior

`COMPATIBILITY BATCH_SIGNAL` server configuration option enables the following two behaviors:

1. allow the use of the `BAT_CLSE` option in a batch call (otherwise it is ignored), and
2. set `sysiocod` to `BTNO_COD` when the server automatically closes a batch.

It seems reasonable for the `BAT_CLSE` mode specified by the caller to take effect unconditionally rather than depending on a server configuration option, and the setting of `sysiocod` does not need to depend on a configuration option because the client can ignore the value. Also, standalone mode batch code does not condition these actions on a configuration option. For these reasons, we have changed the server's batch function to always enable the two behaviors that were previously enabled by the `COMPATIBILITY BATCH_SIGNAL` configuration option.



Note: FairCom DB still accepts `COMPATIBILITY BATCH_SIGNAL` (we made this choice so that someone using this option will not encounter an error at server startup due to an unrecognized configuration option), however, it is no longer required to enable these behaviors.

## 14.20 Support setting server configuration directory with environment variable

The FairComDB server now supports setting the `FCSRVR_CFG_DIR` environment variable to a directory name that the server is to use as the directory in which it looks for its configuration files.

Note that the `FCSRVR_CFG` environment variable is still supported. If both `FCSRVR_CFG_DIR` and `FCSRVR_CFG` are set, the value of `FCSRVR_CFG` is used for the server configuration file name, and `FCSRVR_CFG_DIR` is used for other configuration file names.

We still need to modify other code such as the server plugin code to use the `ctConfigDir` server global variable as the configuration directory.

The server writes a log message indicating the configuration directory it is using.

For example, if using the default configuration directory the server logs the following message to `CTSTATUS.FCS`:

```
Using configuration directory
```

And if using the value from the `FCSRVR_CFG_DIR` environment variable the server logs this message:

```
Using configuration directory from environment: <config_dir_value_from_environment>
```

## 14.21 Configure `DIAGNOSTICS FULL_DUMP` at Runtime

FairCom DB now supports changing the `DIAGNOSTICS FULL_DUMP` configuration option at runtime, using either the `ctadmn` server administrator utility or `ctSETCFG()` C API function in the same manner we document changing other configuration options.

## 14.22 Server Configuration - Default `FILES 4096`. Was `FILES 1024`.

We have changes the default value in our default `ctsvr.cfg` for the `FILES` keyword:

Was: `FILES 1024`

Now: `FILES 4096`



## 14.23 services.json replaces cthttpd.json Configuration

Compatibility Change Notice: Internal support that loads cthttpd settings from the *cthhttpd.json* file has been removed. From build dates 230301 forward, the only supported settings file is *services.json*. *services.json* will be used going forward for additional services besides just HTTP/HTTPS.

## 14.24 Set CHECKPOINT\_INTERVAL to 12th of total log space in server configuration file

Our previous default on the ctsrvr.cfg file was using a 10 MB checkpoint interval values, even though we set log space to 1 GB. This means that we are creating checkpoints more often than we generally recommend. We generally recommend 3 checkpoints per transaction log file.

We changed the code so that we use a checkpoint interval value in the configuration file of 1/12th the total log space.

```
; Transaction log size
LOG_SPACE 1 GB
CHECKPOINT_INTERVAL 85 MB
CHECKPOINT_FLUSH 17
LOG_TEMPLATE 2
```

## 14.25 Improve ctQuiet Timeout

When a FairCom DB quiesce (**ctQUIET()** call) is called, it can eventually fail with error 843 as it was unable to block all thread activity. There is a fixed number of internal attempts, however, it may take a very long time (5+ minutes) to return the error before it gives up with a large number of connections. During this waiting period, all server activity is blocked.

*CTSTATUS.FCS* message

```
"ctQUIET Note: no progress clearing threads from core. Abort block attempt.: 843"
```

An internal loop must acquire and release mutexes for each user, and if a large number of users are active, this may become the primary source of delay. To alleviate this point of delay, a server configuration option has been added specifying a wait interval and this value is checked on each loop until the maximum value is encountered.

QUIET\_MAX\_WAIT <seconds>

When exceeded, ctQUIET fails with error 843. The default value is 60 seconds with a minimum value of 5 seconds.

**Note:** This value only applies after any initial wait value specified in the originating ctQUIET() API call



## Diagnostic

A new diagnostics configuration was also added to aid in debugging log timeouts

DIAGNOSTICS CTQUIET

When enabled, this diagnostic creates a stack dump when a **ctQUIET()** fails with 843. This information can help FairCom support determine why a thread is not exiting the core as expected causing the long delay window.

## 14.26 Docker uses NAT - New property "dbengine\_behind\_nat" in ctagent.json

CONTEXT: Customer testing docker with Replication Manager

Customer is trying to use Replication Manager to replicate to FDB servers inside a Docker Container. They get timeout errors from Memphis trying to connect back to the FDB servers. We reproduced this issue. We figured that 'dbengine\_conn\_by\_ip' might be the solution. We added this line to the ctagent.json files inside the containers with the actual IP of the container host machine. The containers also have the FDB ports mapped to the host machine. This should allow Memphis to connect back to the FDB servers. What we found was that Memphis ended up recording an IP for the FDB servers that is a virtual IP that is internal to the container. This does not resolve to the correct machine, and the connection fails between Memphis and the FDB server. Adding 'dbengine\_conn\_by\_ip' did not change this. Memphis still recorded the wrong IP. Therefore we introduced new property in ctagent.json named "dbengine\_behind\_nat".

SYMPTOM: When a second instance of the docker starts, it overwrites the first one in Replication Manager.

CAUSE: During the ctagent initialization, if we have two OpSystem's with different hostnames, but same IP address. They can't coexist today in Replication Manager. As the second instance starts with a new hostname, which could not be found yet in Replication Manager, it tries to find if there is already another machine with the same IP address, and then overwrites it. This is important in case the existing registered machine in Replication Manager renames its hostname or because of the cases, specially Unix, where the hostname was not set. But it does not work in case of NAT, where it is expected to have more than one machine with the same IP address.

Network address translation (NAT) is a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.

If the Replication Manager environment has NAT involved, like multiple containers running under the same real IP address, each server instance should have the "dbengine\_behind\_nat" set to true in the ctagent.json settings file. If it is not set, Replication Manager will recognize all containers as the same OpSystem instance, under the real IP address.

SOLUTION: Added support to a new property in ctagent.json named "dbengine\_behind\_nat". Default is FALSE. But in case it is true, it will relay only on the hostname for distinguishing OpSystem instances under Replication Manager.



## 14.27 Server rebuild options may be set at runtime - SORT\_MEMORY and MAX\_HANDLES

Server keywords SORT\_MEMORY <value> [SCALE] and MAX\_HANDLES <value> may be modified at runtime using SetSystemConfigurationOption().

```
NINT SetSystemConfigurationOption(NINT option, pTEXT value);
```

### Example:

```
SetSystemConfigurationOption(setcfgCONFIG_OPTION, "SORT_MEMORY 100 MB");
Optional scale = KB,MB,GB,TB
SetSystemConfigurationOption(setcfgCONFIG_OPTION , "MAX_HANDLES 30");
```

## 14.28 Configuration option MAX\_REBUILD\_QUEUE is runtime configurable

Server configuration keyword MAX\_REBUILD\_QUEUE may now be set at runtime via ctSETCFG\_MAX\_REBUILD\_QUEUE().

### Prototype:

```
NINT ctSETCFG_MAX_REBUILD_QUEUE(cpTEXT maximum_queue_size,NINT logerror);
```

### Example:

```
ctSETCFG_MAX_REBUILD_QUEUE("10MB",YES);
<maximum_queue_size> is interpreted as bytes by default, or it can be specified with a suffix of
KB, MB, or GB (e.g., 10 MB).
```

## 14.29 V12/V11 Encryption Backward Compatibility

### Encryption backward compatibility

In order for servers to support file encryption options that remain compatible with pre-V12 servers we have restored support for our PRE-V12 (legacy) encryption routines. This disables FIPS support, which FairCom does not support in pre-V12 this support.

Removed the (previously commented out) keyword "FIPS\_ENCRYPTION YES" from your ctsrvr.cfg.

To force new files to be encrypted with the V11 Master Cipher use this ctsrvr.cfg option:



## COMPATIBILITY PREV\_V12\_MASTER\_CIPHER

Add this "COMPATIBILITY PREV\_V12\_MASTER\_CIPHER" to your ctsrvr.cfg.

To force new files to use keys derived compatible with V11 (using MD5) requires creating the Master Key verification file (ctsrvr.pvf) with the -kdfv1 option Master Password Verification Options (faircom.com):

```
ctcpvf -kdfv1
```

Summary:

If you want newly created file encryption to be backward compatible with V11 servers:

1. create ctsrvr.pvf with -kdfv1: `ctcpvf -kdfv1`
2. use ctsrvr.cfg option: COMPATIBILITY PREV\_V12\_MASTER\_CIPHER

ctinfo (V12.5+) outputs detailed encryption information (if it can read the file).

```
reading a AES file created by V11 ctinfo output:
Data visibility: AES encrypted
Encryption Key length: 256 bits
Encryption Attributes: 0x1
 256 bit master key
 KDF uses MD5
```

## 14.30 OPENSSL\_ENCRYPTION keyword no longer controls internal cipher

The OPENSSL\_ENCRYPTION keyword no longer modifies the internal cipher.

## 14.31 Ephemeral Elliptic Curve Diffie-Hellman Cipher (ECDHE) now Supported for TLS Connections

Attempting to establish a TLS/SSL connection using an ephemeral elliptic curve Diffie-Hellman cipher such as ECDHE-RSA-AES256-SHA384 failed with error **1104**. Diagnostic logging showed the server's SSL accept call failed with a "no shared cipher" error. This was because the server did not enable any elliptic curves to use for SSL connections. The server now enables a default set of elliptic curves.

### Specify Diffie-Hellman Parameters in a Server Certificate File

A subset of SSL ciphers require Diffie-Hellman key exchange parameters. Supply these parameter values with KEY\_EXCHANGE\_PARAMS in your SSL parameter configuration subsystem block. For complete flexibility, it is also possible to specify your Diffie-Hellman parameters directly



in your server certificate file. If they are present in the certificate file, they are used. If they are not present in the certificate file and `KEY_EXCHANGE_PARAMS` is configured, the `KEY_EXCHANGE_PARAMS` configuration parameters are used.

The following SSL ciphers require Diffie-Hellman parameters:

- `ssl_ciphers DHE-RSA-AES128-SHA256`
- `ssl_ciphers DHE-RSA-AES256-SHA256`
- `ssl_ciphers DHE-RSA-AES128-GCM-SHA256`
- `ssl_ciphers DHE-RSA-AES256-GCM-SHA384`
- `ssl_ciphers DHE-DSS-AES128-SHA256`
- `ssl_ciphers DHE-DSS-AES256-SHA256`
- `ssl_ciphers DHE-DSS-AES128-GCM-SHA256`
- `ssl_ciphers DHE-DSS-AES256-GCM-SHA384`

**Affected Components:** Server

### Server-side Diagnostics

Logging SSL diagnostic messages is improved with a configurable log file. To enable server-side SSL diagnostic logging add `DEBUG_LOG` to `SUBSYSTEM COMM_PROTOCOL SSL` block in `ctsrvr.cfg`. The log file is created in your server's `LOCAL_DIRECTORY` directory. For example:

```
SUBSYSTEM COMM_PROTOCOL SSL {
 DEBUG_LOG ssl.log
}
```

### TLS/SSL Client Diagnostics

To enable SSL logging for a client, set the `CTSSL_DEBUG_LOG` environment variable to a log file name.

## 14.32 Recovery Configuration File for Secondary Synchronous Server Coordination

In V12.0.1 and later, the server now supports an optional automatic recovery configuration file. This file is used to provide commit position information for synchronous replication to the server when the server is no longer running as a primary server, so that it can properly undo or redo transactions whose commit state on the secondary server is not known based on the source server's transaction log entries.

The file name defaults to *recovery.json* in the server's current working directory. The configuration option `RECOVERY_CONFIG <filename>` can be used to specify a different file name.

The recovery configuration file contains a list of Replication Agents, with the unique ID and the transaction ID or log position of the last commit applied by each Replication Agent. If both transaction ID and log position are specified for an agent, the transaction ID takes precedence.

**Example file:**

```
{
 "agentrecoveryposition": [
 {"agentid":"agent1", "transactionid":111},
 {"agentid":"agent2", "lognumber":2, "logoffset":65536}
]
}
```

When recovery has completed after having read at least one Replication Agent's commit position from this file, the server deletes this file so that it is not used on the next startup.

## 14.33 Expanded Windows Broadcast Machine Name Length

When the `COMPUTERNAME` environment variable is specified in `ctsrvr.cfg`, a Windows API function is now used to get the computer's DNS host name.

Prior to this modification, the Windows `COMPUTERNAME` environment variable was used, which limited the name to 15 bytes. If the system has a computer name longer than 15 characters, only the first 15 were broadcast. This modification supports broadcasting computer machine names longer than 15 characters.

## 14.34 Return Replication Log Scan Last Log Entry Offset

A server configuration option has been added that allows you to change the log offset returned by the replication log scan for an operation that consists of multiple log entries, such as `REPL_ADDREC`. By default, we return the offset of the *first* log entry that comprises that operation. This configuration option makes it possible to change the behavior so that the offset of the *last* log entry that comprises the operation is returned. The option is:

```
REPL_USE_LAST_LOGPOS <YES | NO>
```

The default is NO.

This option can be changed at runtime with the server administrator utility `ctadmn`.

This option should be used when applications expect replication operations to be returned in ascending log offset order.

Note that in addition to this option, the option `COMPATIBILITY NO_REPL_DEFER_TRAN` must be used to avoid reordering of transaction begin operations that is caused by deferring the return of the begin transaction operation until an operation of interest to the replication reader has been read from the log.

## 15. Communication Protocols

### 15.1 Shared Memory

#### Shared Memory protocol connect function on Windows now uses the optional socket timeout value

On Windows, a shared memory connection attempt by a client may need to retry the wait on the named pipe that the server uses. The connect function can retry once a second for up to 240 seconds (4 minutes). Because of this retry, a connection attempt by a Replication Agent thread from within a server that is shutting down takes a long time to return. The server is waiting for the thread to exit, so the server takes a long time to shut down.

The logic has been modified so that the shared memory client connect function stops retrying the wait on the server named pipe after a specified timeout period elapses.

The Replication Agent defaults to a 5 second socket timeout. This value can be changed using the `socket_timeout` option in `ctreplagent.cfg`.

**Affected Components:** Server

#### Shared Memory Key Configuration Through Environment Variables

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

```
SHMEM_KEY_ISAM %MY_ISAM_KEY%
```

**Note:** When `SHMEM_KEY_ISAM %MY_ISAM_KEY%` is active in `ctsrvr.cfg`, the FairCom DB Server process will not start unless the environment variable is also defined. This is the expected behavior for all the environment variable options.

#### Unix Shared Memory connection errors logged by server now include pipe or socket method in use

The Unix shared memory connection logic now indicates (on the server side) whether the old named pipe method or the current Unix domain socket method was in use when the error



occurred. The old method is expected to be used only by old clients that do not support the Unix domain socket method. Some example messages:

```
User# 00022 FSHAREMM: The client's shared memory version read from client pipe (1258973544) is not compatible with the server's shared memory version (4)
User# 00022 FSHAREMM: Detected an incomplete connection request (socket seqnum 3) (server seqnum 4)
```

**Affected Components:** Server

## Expanded Windows Broadcast Machine Name Length

When the `COMPUTERNAME` environment variable is specified in `ctsrvr.cfg`, a Windows API function is now used to get the computer's DNS host name.

Prior to this modification, the Windows `COMPUTERNAME` environment variable was used, which limited the name to 15 bytes. If the system has a computer name longer than 15 characters, only the first 15 were broadcast. This modification supports broadcasting computer machine names longer than 15 characters.

## Named Pipe connection logic removed from Unix shared memory

Originally, our Unix shared memory communication protocol used named pipes for the client to communicate with the server before the shared memory region had been set up. As a result of optimizations, the named pipe method is no longer supported. This modification ensures that a process cannot write to the named pipe, which would cause the server to log unexpected shared memory connection error messages to `CTSTATUS.FCS`.

The `COMPATIBILITY SHMEM_PIPE` configuration option no longer has any effect.

**Affected Components:** Server, client

## Improved Shared Memory Connections When Permissions Discrepancy Between Server and Client

A **978 error** occurs when the user account under which the server process is running does not have permission to duplicate the client process handle.

A shared memory connection attempt on Windows failed with **error 978** when client and server processes were run as Windows services under the same non-administrator user account. However, if the processes were run as non-service processes, the shared memory connection attempt succeeded. This error can occur in other situations as well.

To address this class of permissions when a shared memory connection fails with **error 978** FairCom DB clients now grant the duplicate process handle permission to the user account under which the server is running, and then the client retries the connection.

There was also a discrepancy in failed FairCom DB SQL connections on Windows when permissions prevented a shared memory connection. Previously, the connection was retried as TCP/IP. Now, to match ISAM client behavior, a **-17978 error** is returned and the connection fails



such that a user is explicitly informed the shared memory connection could not be established due to permissions.

**Compatibility Note:** Both client and server must be updated with this bug fix to resolve the error. Although there is no compatibility issue between the old and new client and server (the new server accepts connections from the old client and the old server accepts connections from the new client), both client and server must be updated to resolve this error.

**Note:** This is a change in behavior for SQL clients.

**Affected components:** Client library, Server, SQL Client libraries

**Environment:** Windows

**Diagnostic Tip:** If an error occurs when the client library attempts to retry the **978 error** and the environment variable SHMEM\_DEBUG\_LOG is set to a log file name, the client library writes an error message to that log file.

## Improved SQL Shared Memory Connections When Permissions Discrepancy Between Server and Client

Previously, when a SQL shared memory connection failed due to Windows permission restrictions, the SQL client automatically retried a TCP/IP connection. For consistency with the ISAM connection behavior, we changed the SQL behavior to fail with error **-17978** in this situation. This permits the user to know that the shared memory connection could not be established due to the permission restriction.

**Note:** This is a change in behavior.

**Affected Components:** SQL server and client

## Failed Client Shared Memory Connection Diagnostics

A report was made concerning intermittent failed client connections using shared memory. Server side logs indicated a dropped client Unix socket. To understand an actual cause of failure and improve a reconnection attempt, additional information is needed. In many cases, an strace or truss of the client process will show failed system calls. However, for rare events in production, a run time diagnostic is needed.

The ISAM client library on Unix systems now supports logging information to a file when a shared memory connection attempt fails. To use this feature, set the environment variable CTCLIENT\_DEBUG\_LOG to the name of a file before the process connects. If the shared memory connection fails, a message is written to the file. The message has a timestamp and process id (pid).



## Example

```
export CTCLIENT_DEBUG_LOG=client.log
./ctixmg ADMIN ADMIN FAIRCOMSXXX
```

Contents of *client.log*:

```
Thu Mar 24 16:20:39 2022 [pid=18219176] at_SessionOpen: Could not get shared memory key;
ftok_name=/tmp/ctreedbs/FAIRCOMSXXX readKey=0 errno=2.
```

Analysis of this information should lead to modifications for a more robust connection sequence.

## Configurable Shared Memory Connection Timeout

It was reported an application could return a **133 error** after waiting the maximum 10 second timeout on a shared memory initial connection attempt. While the root cause of the timeout remains under investigation, it was requested to be able to raise the timeout limit as needed to avoid the **133 error** and maintain application continuity. Additionally, Unix shared memory communication protocol login timeouts are now configurable at runtime.

### Server Configuration

The server's Unix domain socket read timeout can be set using a server configuration option:

```
SHMEM_CONNECT_TIMEOUT <n>
```

This option sets shared the memory connection timeout to *<n>* seconds and defaults to 25s for Bank of New York Mellon (BNY Mellon). The server setting can be changed at runtime by calling **ctSETCFG()** or using **ctadmn** to adjust on-the-fly.

### Client Configuration

The following environment variables can be set before establishing a client shared memory connection on a Unix system:

- CTREE\_SHMEM\_CONNECT\_TIMEOUT=<n>  
Sets socket timeout to *<n>* seconds. Defaults to 25 for BNYM
- CTREE\_SHMEM\_CONNECT\_SEM\_TIMEOUT=<n>  
Sets logon semaphore timeout to *<n>* seconds. Defaults to 70 for BNYM

These values can be programmatically set in client code by calling **ctSetCommProtocolOption()** with the option *ctCOMMOPT\_SHMEM\_CONNECT\_TIMEOUT* or *ctCOMMOPT\_SHMEM\_CONNECT\_SEM\_TIMEOUT* before connecting to the server. Setting these values by the API call overrides any environment variable settings.

**AFFECTED COMPONENTS:** client library, server

## 16. Replication

### 16.1 Support more than one replication plan between two FairCom DB Servers

Previously Replication Manager did not support more than one Replication Plan between two DB Engines. We have now replaced this restriction and allow for multiple plans to be defined and activated.

### 16.2 Support replication of superfile hosts and members

We now support replicating create and delete of superfile hosts and operations on superfile members. Add the option "replicate\_data\_definitions yes" to ctreplagent.cfg.

Also, we have new replication agent configuration options which enable diagnostic log messages to ctreplagent.log:

- diagnostics file\_open: log file open/close activity
- diagnostics tran\_state: log transaction state
- diagnostics dependency\_graph: log dependency graph details
- diagnostics file\_open\_err: log errors opening files
- diagnostics analyzer: log transaction analysis details

These diagnostics options can be changed at runtime using the repadm utility. To turn off an option at runtime, prefix the option value with ~. For example:

```
repadm -c setconfig:"diagnostics ~file_open" -s REPLAGENT@localhost
```

### 16.3 Support Publication "By Database"

Supporting advanced publication rules by database. Our original replication solution required the user to select on a file-by-file basis the tables to be replicated. Now, we introduce the ability to specify a database to be replicated.



## 16.4 Support Publication "By Dictionary"

Our original replication solution required the user to select on a file-by-file basis the tables to be replicated. Now, we introduce the ability to specify a either a SQL or a CTDB dictionary as your list of replicated files.

## 16.5 Support multiple folder rules per Publication

Previously we supported setting only one folder rule per publication. But now, in Replication Manager's database + folder publication UI, we were support multiple folders in the same publication.

## 16.6 Non-recursive filename wildcard matching "by folder" replication plans

We implemented a non-recursive filename matching wildcard option for filename matching function.

The internal function `icrt_filmtc()` accepts a new option bit, `WC_NORECURSE`, that causes the function to only evaluate a wildcard character (asterisk) for the current path component in the target string. So for example, when using this option a wildcard specification of:

```
/Users/FairCom/data/*.dat
```

matches:

```
/Users/FairCom/data/test.dat
```

but not:

```
/Users/FairCom/data/subdir/test.dat
```

We added a "recurse" property to the replication file filter. It defaults to "y" for backward compatibility. We now use `recurse="n"` in the replication file filter for a "by folder" replication plan.



## 16.7 Support for "recursive" parameter for Publication "by folder"

Prior publication "by Folder" only published the files from the given folder level and ignore any children folders. We added a new parameter in the publication named "recursive". This new property can be set by the ctMemphisPersistPublication web API, but also by the FCREPLPublication::SetRecursive() C++ API. This information is persisted in a new field named "IsRecursive" in Memphis' Publication table.

## 16.8 Support Regular Expression (regex) rules in Publication

We have added support for Regular Expression (regex) rules in our publication definition.

Regular Expression (regex) Examples

### Example 1

Match filenames starting with admin\_ followed by numeric digits followed by .dat

For example it matches admin\_001.dat

`admin_([0-9]+).dat`

### Example 2

Match filenames followed by .data, .data, or .db.

For example, it matches c:\my path\my file.dat

`(.*(data|dat|db))`

### Example 3

Match filenames except those that end in .temp, .tmp, .test, and .tst.

For example, it does not match myfile.temp

NOTE: this regex needs work because an extra dot in the filename causes a match.

`.*(\.|\V)(?!temp|tmp|test|tst).*$`

### Example 4

Matches any filename that ends with .d followed by one or more numeric digits.

For example, it matches c:\my path\my.test file.d01

`^.*(\.|\V)(d[0-9]+)$`

NOTE: a user may or may not supply the ^ character at the beginning and the \$ character at the end to match the whole name. Either way, we need to match the entire name.



## 16.9 Replication agent sync shutdown

The repadm utility now supports an option to cause the replication agent to finish applying all changes it has read from the source server's transaction logs and then shut down.

The repadm utility's -n option can also be used with this option.

Sample output:

```
C:\> repadm -c syncshutdown -s AGENTSERVERNAME
```

Requesting replication agent to shut down with the specified options. Successfully shut down the replication agent.

```
Log copy position=<6,309897979> Last commit position=<6,309888349> Log read
position=<6,309897979> Transaction ID=808025
```

NOTE: If the replication agent was not able to connect to the target server before it is shut down, it reports zeroes for the log positions and transaction id.

See the repadm utility's repadmShutdownWithOptions() function for an example showing how to use this functionality programmatically.

## 16.10 Replication deployment now copies indexes instead of "Rebuild on Deploy"

We have changed the default in our replication plan deploy logic related to indexes. Before, we would not copy the indexes from the source server to the target server. We would rebuild the indexes on the target server after we have copied over the data file.

We determined that it is much more efficient if we add the index files into the dynamic dump and restore them in the target, instead of copying only the data files over and rebuilding the indexes. This improved the performance of the deploy.

## 16.11 Return Replication Log Scan Last Log Entry Offset

A server configuration option has been added that allows you to change the log offset returned by the replication log scan for an operation that consists of multiple log entries, such as REPL\_ADDREC. By default, we return the offset of the *first* log entry that comprises that operation. This configuration option makes it possible to change the behavior so that the offset of the *last* log entry that comprises the operation is returned. The option is:

```
REPL_USE_LAST_LOGPOS <YES | NO>
```

The default is NO.

This option can be changed at runtime with the server administrator utility **ctadm**.



This option should be used when applications expect replication operations to be returned in ascending log offset order.

Note that in addition to this option, the option `COMPATIBILITY NO_REPL_DEFER_TRAN` must be used to avoid reordering of transaction begin operations that is caused by deferring the return of the begin transaction operation until an operation of interest to the replication reader has been read from the log.

## 16.12 Replicate Restore Point Operations

The creation of a restore point is now replicated. The target server writes a log entry that the Replication Agent reads. The Replication Agent performs the same restore point operation on the target server in the same transaction order as on the source server.

## 16.13 Support FileSystem scan of non-ctree files

In the past, we had disabled support for non-ctree files in Replication Manager. Now we support the scan of non-ctree files to be selected for a replication plan.

## 16.14 Automatic Purge and Archive of Replication Logs

Some tables in Replication Manager (e.g., Action, Log, Replication Statistics, Exceptions) can grow without any cleanup. Although some of these tables were auto archived, the history table is never purged, so it can grow indefinitely. The logic has been modified to add keywords for configuring the archiving and purging feature in Replication Manager. These keywords are in *ctReplicationManager.cfg* in the *config* directory. The default configuration is:

```
DAYS_BEFORE_ARCHIVING_LOG_RECORDS 14
DAYS_BEFORE_ARCHIVING_ACTION_RECORDS 14
DAYS_BEFORE_ARCHIVING_EXCEPTION_RECORDS 14
DAYS_BEFORE_ARCHIVING_STATISTIC_RECORDS 14

DAYS_BEFORE_PURGING_LOG_RECORDS 90
DAYS_BEFORE_PURGING_ACTION_RECORDS 90
DAYS_BEFORE_PURGING_EXCEPTION_RECORDS 90
DAYS_BEFORE_PURGING_STATISTIC_RECORDS 90
```

Note that the Replication Statistics has 2 tables, one for the single-threaded replication (REPLSTAT) and another for the parallel replication (REPLPARALLELSTAT). The second table has 2 extra children tables: REPLPARALLELANALIZER and REPLPARALLELAPPLY. When the



Replication Statistics are archived or purged, all of these tables are impacted to avoid breaking the database constraints.

Both archive and purge are executed by a background thread that is always running in the Replication Manager instance. This thread constantly verifies the status for DBEngines and Replication Plans. It is able to identify that the current iteration is the first one for the current date, which is the time it checks all the possible archive and purge information.

## 16.15 Log Only Failed Operations to Exception Log

By default, when the Replication Agent fails to apply a replicated operation to the target server, it writes an entry to the replication exception log for every operation in the transaction where the failure occurred. This makes it possible to see what operations were not applied due to the failure. But this can generate a lot of log entries and may make it difficult to see the failed operations.

To address these issues, we added a Replication Agent configuration option to instruct the agent to only record exception log records for those operations that failed. To use this feature, add the following option to the Replication Agent configuration file, *ctreplagent.cfg*:

```
exception_logging_errors_only
```

## 16.16 Replication Command Line Utility - ReplUtil

Our powerful Replication Command Line Utility (ReplUtil) continues to improve.

This utility (replutil.exe) can be found withing your Replication Manager package in the ReplicationManager base folder.

Highlights include:

- Inconsistencies in how the wildcards are obtained from the JSON commands file.
- Missing a way to get names and/or list the subscriptions.
- Reviewed the implementation of the wildcards
- List the details of a plan which shows the involved subscriptions.
- Changed the “norebuild” option to “rebuild” to avoid misunderstanding.
- Make sure only a single target is specified
- Error checks for NULL pointers
- Improved the plan show function by using built in functions
- Now displays messages indicating the action that it is performing and the progress of the action.
- Wildcard exclusions with folder publications
- Exclude support for db publications



- The output of replutil was in clear text without using a formatting scheme this made replutil outputs difficult to parse. We implemented a new replutil boolean option called "jsonoutput" which forces the replutil output to always be in JSON format.

## 16.17 repadm - Replication Administrator Utility Updates

### Force All Database Changes Before Synchronous Server Shutdown

The **repadm** utility now supports an option to cause the Replication Agent to finish applying all changes it has read from the source server's transaction logs and then shut down.

The **repadm** utility's -n option can also be used with this option.

Sample output:

```
C:\> repadm -c syncshutdown -s AGENTSERVERNAME
Requesting replication agent to shut down with the specified options.
Successfully shut down the replication agent. Log copy position=<6,309897979> Last commit
position=<6,309888349> Log read position=<6,309897979> Transaction ID=808025
```

**Note:** If the Replication Agent was not able to connect to the target server before it is shut down, it reports zeros for the log positions and transaction ID.

See the **repadm** utility's **repadmShutdownWithOptions()** function for an example showing how to use this functionality programmatically.

### Toggle READONLY Target Server Mode on Replication Shutdown

The **repadm** utility now supports an option, **-settargetserverwritable**, that can be used with the **shutdown** or **syncshutdown** commands. This option causes the Replication Agent to connect to the target server and set it writable (READONLY\_SERVER NO). This is done at the end of shutting down the Replication Agent. Note that the call to shut down returns to the **repadm** utility before this is done, since it is done by the main Replication Agent thread rather than the server thread that initiates the shutdown of the Replication Agent.

Example Replication Agent shutdown command:

```
repadm -c syncshutdown -settargetserverwritable -s FAIRAGENT
```



## Check for Active Replication

The **repadm** utility now supports an option to indicate if the Replication Agent is operational. Operational means the agent threads are running and are connected to the source and target server as appropriate. This simple point-in-time check is able to detect when Replication Agent threads have exited due to an error and when Replication Agent threads are not connected to their servers. It cannot detect a situation where Replication Agent threads are alive but are in a hung state.

Example **repadm** command-line usage:

```
repadm -c isactive -s AGENTSERVERNAME
```

The utility outputs a value of "y" if the agent is operational, "p" if the agent is paused, and "n" if the agent is not operational.

Function call usage:

**ctReplAgentOp()** with mode of *ctRAOPcheckifactive*.

Example code:

```
NINT rc;
TEXT stateBuffer[2];
VRLen bufsize = 2;
rc = ctReplAgentOp(ctRAOPcheckifactive, NULL, 0, stateBuffer, &bufsiz);
```

When the **repadm** utility's *isactive* command is used, the utility now sets the return code to indicate the agent's status as of the last check made by the utility. Status codes are:

- 1: error checking agent status
- 2: agent is operational
- 3: agent is paused
- 4: agent is not operational

More status codes may be added in the future.

## Reset Replication Statistics

The **repadm** utility can be used to reset the Replication Agent statistics (counts of passed/failed operations). Use the option **repadm -c resetstats**.

This is in addition to the existing support for resetting Replication Agent function statistics (**repadm -c resetfuncstats**).



## 16.18 repadm utility can be used to reset the replication agent statistics

The repadm utility can be used to reset the replication agent statistics (counts of passed/failed operations). Use the option repadm -c resetstats.

This is in addition to the existing support for resetting replication agent function statistics (repadm -c resetfuncstats).

## 16.19 repadm Utility - option to display key value and record image in exception log record

The repadm utility now supports an option to show the key value and record image in the exception log record for the getlog and getlogtail commands.

The syntax for this new option is:

-showdetails <option>

For the getlog and getlogtail commands, show key value and/or record images.

Supported values: all, keyval, oldkey, oldrec, recimg, currecimg

### Example:

```
repadm -c getlog -s replagent -showdetails all
```

## 16.20 Encrypted Password File Support for Replication Diagnostic Utility ctrepd

The **ctrepd** utility now supports the *-1<settings\_file\_name>* command-line option used by other command-line utilities. This option specifies a file containing an encrypted user name and password for authenticating to a FairCom DB server.

## 16.21 ctrepd utility: Add option to set node name

We added an option -nodename:<nodename> to the ctrepd utility. This option allows the node name for the log read connection to be set. This can be useful for treating the connection like a replication log reader by setting the node name like this: -nodename:"ctreplr 10.0.0.1"



## 16.22 Display Replication Latency from repadm Utility

The replication administration utility, repadm now supports an option to display the replication agent's latency. A command option showlatency is added to provide detailed output.

### Usage

```
repadm -c showlatency -u ADMIN -p ADMIN -s TARGET_SERVER_NAME -a AGENTNAME
```

### Output

```
Tue May 9 15:25:46 2023
 lognum logpos pending latency
agent: 103 418609 20 3 (not caught up)
server: 107 941839

Tue May 9 15:25:47 2023
 lognum logpos pending latency
agent: 104 1756920 14 3 (not caught up)
server: 107 1668498

Tue May 9 15:25:48 2023
 lognum logpos pending latency
agent: 105 3296971 26 3 (not caught up)
server: 107 1668498

Tue May 9 15:25:49 2023
 lognum logpos pending latency
agent: 107 1105515 30 3 (not caught up)
server: 107 1668498

Tue May 9 15:25:50 2023
 lognum logpos pending latency
agent: 107 1668701 0 0 (caught up)
server: 107 1668498
```

## 16.23 Replicate File Copy Operations

FairCom DB now supports replicating file copy operations. Prior to this modification, the file copy was not replicated, which could cause the Replication Agent to terminate with error **446** with these messages in *ctreplagent.log*:

```
Wed Jun 9 09:24:56 2021: Logread: ERR: Create file handle for operation 18 is not yet implemented: 446
Wed Jun 9 09:24:58 2021: Logread: ERR: Failed to create file handle when processing SETDEFBLK log entry: 446
```

Now the file copy operation is properly replicated. If the Replication Agent uses redirect rules in *ctreplagent.cfg*, these rules are applied to the source and destination file names that are specified in the file copy function call.



## 16.24 Additional Replication Extension Event Callbacks

### Synchronous Shutdown Event

The Replication Agent now supports a callback function (as part of the replication extension library feature). This function is called when a sync shutdown has occurred if an extension library is loaded (using the `extension_library` option in the replication agent configuration file) and the extension library exports a function named **`rxOnSyncShutdown`**. See *ctrepluser.c* for the function definition:

```
rxEXPORT VOID rxOnSyncShutdown (prxEVENT prxevent);
```

The replication state as of the shutdown is passed to this function in the *prxevent->ev.shutdownInfo* structure; *prxevent->unqid* holds the agent unique ID, and state info is also in the JSON file *recovery\_<agentid>.json*.

### Create IFIL Event

The replication agent extension library now supports a function named **`rxAfterCREIFIL()`**, which the replication agent calls after successfully replicating a CREIFIL operation.

Additionally, the `RXF_OnTruncateFile`, `RXF_OnChangeSerialNumber`, `RXF_OnCopyFile`, `RXF_OnPUTIFIL`, and `RXF_OnPUTHDR` callbacks are now correctly called by the replication agent.

## 16.25 Replication callback called after replicating a CREIFIL operation

We add replication callback function to be called after successfully replicating a CREIFIL operation. The replication agent extension library now supports a function named `rxAfterCREIFIL()`, which the replication agent calls after successfully replicating a CREIFIL operation.

We also noticed that the `RXF_OnTruncateFile`, `RXF_OnChangeSerialNumber`, `RXF_OnCopyFile`, `RXF_OnPUTIFIL`, and `RXF_OnPUTHDR` callbacks were not getting called by the replication agent, so we fixed that.



## 16.26 Replication Manager files to be created under data directory

Some customers facing challenges when server tries to write on read-only directories (server's binary folders). Most enterprise customers don't give write permission under the server's binary folders. Replication manager creates the "replmanager\_local\_replplan" and several JSON files like dbengine.json, replplan.json, and availgroup.json files under the server's working directory, which can be read-only.

We have moved a number of files now to the "data directory", as follows:

- Modified the RestartReplAgents() to dump and restore the active replication plan information as a JSON file in <data directory>/replmanager\_local\_replplan/replplan/<repl plan name>.json.
- Moved the "dump" directory, where the replication plan deployment writes the dyndmp to <data directory>/replmanager\_local\_replplan/replplan/dump.
- Moved Replication Plan exception json file <data directory>/replmanager\_local\_replplan/replplan/dump.
- Moved dbengine.json and availgroup.json files that dumps Memphis detailed information to allow the server to restart later without Memphis to <data directory>/replmanager\_local\_replplan.
- Modified the logic where the Replication Agent's filter list generates the XML file to move its temporary directory to <data directory>/replmanager\_local\_replplan/replmanager\_temp
- Modified the plugin to receive the local server's data directory in its structure. It is required for correctly calculating the internal directories under the data path.

## 16.27 Improve performance of dump restore when deploying a publication with many files

When deploying a publication that has many files, the dump restore phase of the deploy can spend a significant amount of time reading the dump restore script file for each file that it restores, in order to determine if the file is in the list of files to be restored. However, when deploying a publication we want to restore all of the files that were included in the dynamic dump, so we don't need to check if each file name is in the restore script.

We modified server code to create a restore script from the dump script and use this restore script when restoring the files. The restore script contains a single file name of \*, which matches all file names.



## 16.28 Improved Replication Manager Plan Shutdown

Stopping a server that has many replication plans running could take a relatively long time. Replication Manager always uses a synchronous shutdown. Synchronous shutdown means that the replication agent waits to complete its shutdown until it has applied all the changes that it has read from the transaction logs. However, a synchronous shutdown is only required if using a Replication Manager availability group and current shutdowns now behave in this manner.

### Other Replication Manager Improvements

- When Replication Manager process restarts, the replication plan statistics are now properly updated
- Active clients are now properly cleared when shutting down and multiple replications plans are running
- Shutdown times are improved with ctHttpd and ctAgent plugins active
- Overall server shutdown stability has been improved in numerous areas with both FairCom DB servers and Replication Manager.

## 16.29 Reduce Number of Parallel Replication Apply Thread Target Server Connections

When the replication agent is using parallel replication, each apply thread connects to the target server twice: once through the client interface to apply operations, and once through the server DLL interface to write to the

exception log. To reduce the number of threads, the embedded replication agent uses the client connection to the target server for both purposes.

A replication agent that is running outside of the target server can enable this behavior by using the following configuration option in *ctreplagent.cfg*:

```
exception_logging target_server
```

When a non-embedded replication agent uses this configuration option, the replication exception log is created on the target server instead of on the replication agent's local server. To access the replication exception log in

this situation, one must connect to the target server instead of the replication agent server. For example:

```
repadm -c getlog -s TARGET_SERVER_NAME
```

This support is only available in FairCom DB builds 12.6.0 and later.



## 16.30 Reduce Number of Log Read Thread Connections to Target Server when Using Embedded Replication Agent

The replication log read thread has a connection to its local server in order to access the replication filter change data file and a connection to the target server to apply replicated changes. But in the embedded replication model the replication agent runs in the target server, so these two servers are the same server.

We reduced the number of connections used by an embedded replication agent by changing the code that accesses the file filter change data file to use the connection to the target server.

We automatically enable this optimization for an embedded replication agent.

For an external replication agent, this option can be enabled from *ctreplagent.cfg* with the following configuration option :

```
file_filter_target yes
```

## 16.31 Initialize Replication File Handle Values with ctReplInitFileHandle()

FairCom DB Replication architecture depends on extended state information of many internal structure members in newer versions, and will continue to do so in the future. Users accustomed to using the Replication SDK to build specific log reading applications may not be aware of the initial expected values required in these structures. To ensure consistent usage a new Replication API was added to initialize the File Handle structure. **ctReplInitFileHandle()** takes an allocated *ctFILH* structure and initializes structure members to expected starting values.

### ctReplInitFileHandle

```
NINT ctReplInitFileHandle(REPL_INIT_FILE_HANDLE_OPTIONS *replInitFileHandleOptions, pctFILH pfilhnd);
```

Initializes the specified replication file handle.

- [IN] *options*: A versioned structure containing the options for the function call
- [OUT] *pfilhnd*: The replication file handle to initialize

Options for the **ctReplInitFileHandle()** API function:

```
typedef struct replInitFileHandleOptions_t {
 REPL_INIT_FILE_HANDLE_VERSION structVersion; version of this structure */
 const char fileName; / name of the file */
} REPL_INIT_FILE_HANDLE_OPTIONS;
```

Values for structVersion field in REPL\_INIT\_FILE\_HANDLE\_OPTIONS structure



```
typedef enum replInitFileHandleVersion_t {

 REPL_INIT_FILE_HANDLE_VERSION_1 = 1, /* initial version of structure */

 REPL_INIT_FILE_HANDLE_VERSION_CURRENT = 1 /* current version of structure */

} REPL_INIT_FILE_HANDLE_VERSION;
```

Returns NO\_ERROR on success, or a non-zero c-tree error code on failure.

## 16.32 Update Replication Manager Publication Files and Folders When Server Moved to Alternate Environment

After a cluster failover event moved a FairCom DB source server to another machine and attempted to restart an existing Replication Plans, it failed and a timeout error was noted in the logs. Although the server had been correctly updated to the new machine in the Replication Manager "Memphis" database, the published files, folders, and dictionaries used on the existing Replication Plans were still considered in the old machine. When starting a Replication Plan it scans published folders, and in this case times out as the original machine is not answering anymore.

When the FairCom DB starts and it finds itself in the Replication Manager "Memphis" database by UI, it is able to detect if it has been moved to another machine. At this point, it updates the DBEngine's OpSystem table to the new one and also resets the previous IP address set. All the physical files involved in any publication rules are then updated according to the following::

- "By file" Plans - loop through all files in the list and make sure they exist under the same volume name, path, name in the new machine, if it doesn't exist, add it. Update the publication list to replace them with the new ones.
- "By dictionary" Plans - (by database): replace the dictionary physical file path to be in the new machine.
- "By folder" Plans - the same as "by file" adjustments.

In addition, retry logic was added to up to 5 seconds in case it is not able to retrieve Memphis directory information, in case Memphis is still starting.

**Note:** This attempt to automatically update the list of published rules only works if the source server has been moved to another machine, *and it keeps the same physical file path*. In this paths does not match in the new machine, an administrator will be required to correct the list of published files manually.

## 16.33 Obtain Oldest Uncommitted Transaction Log Position from a Replication Agent

We added a replication agent command to return information about the oldest transaction that the replication agent has read from the source server's transaction log but that the replication agent has not yet applied to the target server. This feature is supported in the following ways:



## API

The **ctReplAgentOp()** function now accepts a new operation code (*opcd* parameter), **ctRAOPgetOldestUncommittedPosition**. The information is returned in the format of the **OLDEST\_UNCOMMITTED\_TRAN\_STATS** structure, which is defined as follows:

```
/* oldest uncommitted transaction information */
typedef struct oldestUncommittedTranStats_t {
 LONG structVersion; /* version of this structure */
 ULONG uncommittedTransactionCount; /* number of transactions read but not yet committed */
 ctLOGP lastProcessedLogPosition; /* last log position whose entry log read processed */
 ctLOGP oldestUncommittedPosition; /* lowest log position of transaction commit we have read but
not yet applied */
} OLDEST_UNCOMMITTED_TRAN_STATS;
```

### repadm Utility Oldest Committed Transaction

The **repadm** utility supports an option "getoldestuncommittedtran" that displays the oldest uncommitted transaction information. Here is an example command usage and its output:

```
./repadm -c getoldestuncommittedtran -s replagent -a agent1 -t -h 1

Thu Sep 7 14:09:54 2023
last processed log position: 5 2399527
oldest uncommitted tran position: 5 2325186
uncommitted tran count: 97

Thu Sep 7 14:09:55 2023
last processed log position: 6 2069252
oldest uncommitted tran position: 6 2009289
uncommitted tran count: 80
```

### repadm Utility getstats command

For a parallel replication agent, the **repadm** utility's "getstats" option now displays the oldest uncommitted transaction log position below the last processed log position, as shown in the following output. In this example, the log read thread has processed up to log 30 offset 3300473, and the oldest uncommitted transaction is at log 30 offset 3251827. If the oldest uncommitted transaction log position is zero, it means that the replication agent has committed all transactions up to the point of the last processed log position.

```
--logread-----
sid tid lognum logpos state seqno time error func
23 32 30 3300473 source 398228 2 0 ctReplGetNextChange
30 3251827
```

An application can use this information to determine if the replication agent has applied all the transactions up to a particular point in the transaction logs.

**Compatibility Note:** We updated the log read statistics structure to a new version in order to include the oldest uncommitted transaction log position. As a result of this update, a **repadm** utility that does not support the new version of the structure will fail to read the log read thread statistics. In this situation, the **repadm** utility outputs the error message "Error: Replication agent log read stat structure version (1) does not match repadm utility structure version (2)." and exits. The solution is to use a version of the **repadm** utility that supports the new structure version.



## 16.34 Replication Agent Log File now Created in Local Executing Server Directory

In our new replication model in which a replication agent manager controls the replication agents, the replication agent log file was created in the working directory of the process in which it is running. However, this directory might not be writable. The replication agent manager now prepends the server's local directory to the replication agent log file name so that the log file is stored in the server's local directory.

**Replication Agent:** For our standalone replication agent (not running inside a server), the replication agent manager's log file (*ctreplagentmanager.log*) remains created in the working directory of the process because we haven't loaded the server DLL when we create that log file.

## 16.35 Replication manager logging improvements

We improved replication manager's logging in the following ways:

- a) Ensure that the local path value is set for a replication manager configuration object that is copied from another configuration object. This enables some logging of memory grid operations to agentLog.txt that was not previously enabled because the log path value was not getting set.
- b) Change the binding of the agentType value to the agentType parameter when adding a row to the logtable memphis table so that a row having agentType of -1 (RCES\_LOGAGENT\_INVALID) can be logged.
- c) Change the memory grid jar file to create the ctMemGridNode.log file in the server's LOCAL\_DIRECTORY directory instead of in the server process' working directory. We added an init2 function to the memory grid jar that takes the log path.

## 16.36 Change replication agent exception logging default option to log errors only

We changed the replication agent to log only errors to the exception log by default. Previously, the replication agent logged every operation of a failed transaction. To restore the previous behavior, use "exception\_logging on" to the replication agent configuration file.

## 16.37 Add replication agent configuration option to write error 1105 occurrences to ctreplagent.log and apply the transaction anyway

The replication agent now supports a configuration (ctreplagent.cfg) option "check\_update log\_only". When this option is used, the replication agent checks for conflicts when replicating a record update or delete operation, like it does with the "check\_update yes" configuration option,



but when it detects a conflict the replication agent logs a message about detected conflict to `ctreplagent.log` and proceeds to replicate the operation.

Note that this mode of operation doesn't guarantee that the replicated data remains consistent with the original data if for some reason a conflict is detected. But it allows replication to attempt to proceed, rather than aborting the entire transaction due to the conflict.

## 16.38 ctAgent - forcing IP address option ignored

CONTEXT: Customer trying to run FDB with ctAgent in a docker.

SYMPTOM: After DBEngine (FDB) running in a docker connects to Replication Manager, it is mark as inactive (grayed out).

CAUSE: When forcing Replication Manager to use a specific IP address while attempting to connect to a DBEngine, through the `"dbengine_ip_address"` property in `ctagent.json`, it is ignored. So, in case of docker instance, the DBEngine registers the "virtual" IP address in Replication Manager, which is not achievable from outside the docker, so the real IP address, which is achievable from outside the docker, should be configured to be used while connecting to the DBEngine. The problem was that in the `_ctConnect()` function in the `fcrcesdbengine_ctree.cpp`, the `"dbengine_ip_address"` property was ignored when building the "server name" in the `<name>@<ip address>` format.

SOLUTION: Modified the `_ctConnect()` function in the `fcrcesdbengine_ctree.cpp` to pass in the `"dbengine_ip_address"` property, in case it was set, when calling method `RCESDBEngine::GetServName()`, so, it is always used in the first attempt to connect to the server. In case it failed, it still retries with the other possible IP addresses, but the `"dbengine_ip_address"` property is not changed in this case.

## 16.39 Docker uses NAT - New property "dbengine\_behind\_nat" in ctagent.json

CONTEXT: Customer testing docker with Replication Manager

Customer is trying to use Replication Manager to replicate to FDB servers inside a Docker Container. They get timeout errors from Memphis trying to connect back to the FDB servers. We reproduced this issue. We figured that `'dbengine_conn_by_ip'` might be the solution. We added this line to the `ctagent.json` files inside the containers with the actual IP of the container host machine. The containers also have the FDB ports mapped to the host machine. This should allow Memphis to connect back to the FDB servers. What we found was that Memphis ended up recording an IP for the FDB servers that is a virtual IP that is internal to the container. This does not resolve to the correct machine, and the connection fails between Memphis and the FDB server. Adding `'dbengine_conn_by_ip'` did not change this. Memphis still recorded the wrong IP. Therefore we introduced new property in `ctagent.json` named `"dbengine_behind_nat"`.



**SYMPTOM:** When a second instance of the docker starts, it overwrites the first one in Replication Manager.

**CAUSE:** During the ctagent initialization, if we have two OpSystem's with different hostnames, but same IP address. They can't coexist today in Replication Manager. As the second instance starts with a new hostname, which could not be found yet in Replication Manager, it tries to find if there is already another machine with the same IP address, and then overwrites it. This is important in case the existing registered machine in Replication Manager renames its hostname or because of the cases, specially Unix, where the hostname was not set. But it does not work in case of NAT, where it is expected to have more than one machine with the same IP address.

Network address translation (NAT) is a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.

If the Replication Manager environment has NAT involved, like multiple containers running under the same real IP address, each server instance should have the "dbengine\_behind\_nat" set to true in the ctagent.json settings file. If it is not set, Replication Manager will recognize all containers as the same OpSystem instance, under the real IP address.

**SOLUTION:** Added support to a new property in ctagent.json named "dbengine\_behind\_nat". Default is FALSE. But in case it is true, it will relay only on the hostname for distinguishing OpSystem instances under Replication Manager.

## 16.40 Automatically check for a write lock when updating a data file during synchronous replication

When the server detects that a data file is being updated in a synchronous commit transaction and that file is included in the replication activity for the transaction, the server now checks that the caller holds a write lock on the record it is updating or deleting. If not, it fails the operation with error 57.

## 16.41 Client failOver/Broadcast notifications

We have added a mechanism for Client side notification. We now have the ability for the MT Client LIB to automatically spin off a Thread that periodically looks for BROADCASTS UDP datagram notifications. We now have a way to send the Client Apps a UDP broadcast to indicate needs, such as a fail-over notification.



## 16.42 File Groups

### Replication File Groups

A challenge faced with multiple replication plans was that each plan instantiated a server thread. With a large number of threads, this directly impacted CPU usage. Many of these plans replicated the same files between the same set of servers. Collapsing these replication threads into the most minimal set was needed to reduce unnecessary resource consumption.

We have enhanced Replication support for managing independent groups of files, which we call File Groups. To implement this support, we added an abstraction layer to consolidate as many files into a single replication thread as possible. This additional layer accepts additional requests for operations and can start and stop Replication Agent threads as needed to perform operations.

Replication Agents in this testing model are embedded server threads and configured with an "unmanaged" replication configuration file. That is, in this phase of work, they do not require a Replication Manager Memphis database back end and manual configuration will be used.

After we have verified that these File Group enhancements meet the requirements, we plan in the future to move this support into Replication Manager for dynamic configuration.

## 16.43 Configure Embedded Replication Agent

The Replication Agent service is enabled in the replica server configuration file *config/services.json*. Enable the Replication Agent service with the following entry located in "other services". Set "enabled" to "true":

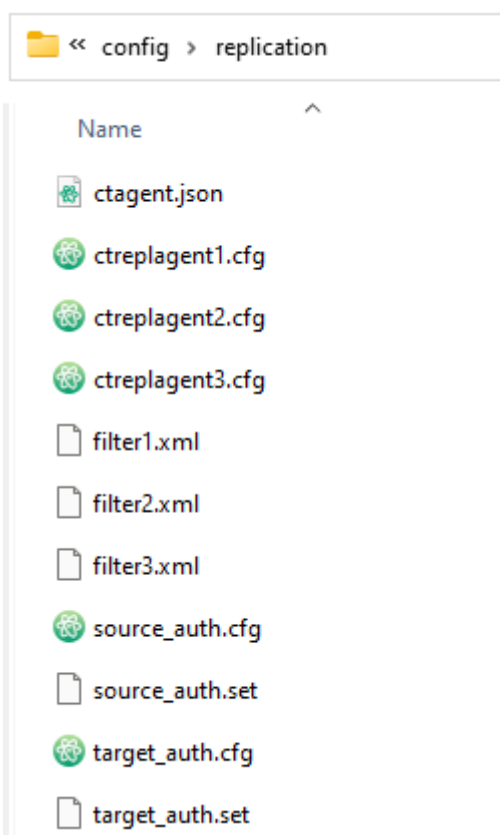
```
{
 "serviceName": "ctagent",
 "serviceLibrary": "./agent/ctagent.dll",
 "enabled": true
},
```

ctagent is the Replication Agent service. Its configuration file is *ctagent.json* located in the *config/replication* folder. In this model, "managed" should be set as "false" and set *use\_replication\_agent\_manager* to true to enable new File Group support.

```
{
 "managed": false,
 "use_replication_agent_manager": true
}
```

Other replication configuration files, such as replication file filter files will be located in the *config/replication* folder as referenced from *ctreplagent.cfg*. We will define these configurations in a later set. Example configuration files are included in this folder and can be ignored.

The *source\_auth.set* and *target\_auth.set* files are required by the replication agent thread.



## 16.44 Requirements for Source and Replica FairCom DB Servers

The source and replica configuration files (*ctsrvr.cfg*) must specify REPL\_NODEID values. For example, REPL\_NODEID 10.0.0.1 for the source server and REPL\_NODEID 10.0.0.2 for the replica server. These values will be referenced in the Replication Agent configuration file, *ctrepagent.cfg*.

## 16.45 Replication Agent Manager Supported Operations

Replication Agent Manager supports the same following operations currently supported by replication agents:

- Start a Replication Agent.
- Stop a Replication Agent.
- Pause a Replication Agent.
- Resume a replication Agent.

Replication Agent Manager also supports the following new operations:

- Add a Replication Agent.
- Remove a Replication Agent.



- Deploy File Groups to a Replication Agent.
- Remove File Groups from a Replication Agent.
- Check the status of a Replication Agent's current action.
- Cancel a Replication Agent's current action.

Typically, the steps that one will follow when setting up replication in this new model is:

- Add a Replication Agent.
- Deploy File Groups to a Replication Agent.
- At any time, additional File Groups can be deployed to a Replication Agent or removed from a Replication Agent. The new replication agent management component takes the necessary actions to update the Replication Agent state and deploy or remove the File Groups.

When adding a new agent, the agent ID and configuration file name are stored in *ctReplagentManager.json* located in the server's working data folder defined with LOCAL\_DIRECTORY).

### Add a Replication Agent

Use the following **repadm** command to add a Replication Agent:

```
repadm -c addagent <configurationFileName> -s <replicationAgentServerName>
```

<configurationFileName> is the name of the replication agent configuration file.

<replicationAgentServerName> is the server name specified in *ctsrvr.cfg* of the Replication Agent Manager process.

Replication Agent Manager adds the replication agent to the file *ctReplagentManager.json*.

### Example Replication Agent Configuration

Here is an example Replication Agent configuration file (*ctreplagent.cfg*) for a Windows system. (For Linux use forward slash instead of backslash for the path separator.)

```
unique_id agent1
file_filter <..\config\replication\filefilter.xml
resync_file_list ..\config\replication\deployresyncfilelist.txt
replicate_data_definitions yes
file_group_parallel_apply yes
parallel_apply yes
num_apply_threads 8
source_authfile ..\config\replication\source_auth.set
source_server FAIRCOMS
source_nodeid 10.0.0.1
target_authfile ..\config\replication\target_auth.set
target_server replica
target_nodeid 10.0.0.2
read_timeout_ms 1000
redirect C:\test\primary\data C:\test\replica\data
```

Note the following settings:

- The replication agent unique ID (`unique_id agent1`). This setting provides the name by which the Replication Agent is referred to when sending commands.



- The source and target server names (`source_server FAIRCOMS` and `target_server replica`) and replication node IDs (`source_nodeid 10.0.0.1` and `target_nodeid 10.0.0.2`). These must match the `SERVER_NAME` and `REPL_NODEID` values in the servers' configuration files.
- The `redirect` command to redirect file names from the primary server directory name to the replica server name. This is required if the replica has a different directory name than the primary.
- The `source_authfile` and `target_authfile` file names. These files provide the user name and password to log on to the source and target servers. They must be created using the `ctcmdset` utility.
- The `file_group_parallel_apply` option. This is a new option. If `file_group_parallel_apply no` is specified (this is the default), replication of transactions within a File Group is done serially. If `file_group_parallel_apply yes` is specified, replication within a File Group is done in parallel.

### Example initial Replication Agent file filter

Here is an example initial replication file filter (*filefilter.xml*) to use for a Replication Agent. Note that no File Groups are listed, because when the Replication Agent is first added, it has no File Groups.

```
<?xml version="1.0" encoding="us-ascii"?>
<replfilefilter version="1" recursive="y" persistent="y" agent="AGENT1">
 <purpose>open_file</purpose>
 <purpose>read_log</purpose>
 <purpose>create_file</purpose>
</replfilefilter>
```

### Example initial Replication Agent file list

Here is an example initial replication file list (*deployresyncfilelist.txt*) to use for a Replication Agent. Note that a dummy file name is listed as a placeholder.

none

### Deploy File Groups to a Replication Agent

To deploy File Groups to a Replication Agent, use the following **repadm** command.

```
repadm -c deployfilegroups agent1 <deployConfigurationFileName> -s <replicationAgentServerName>
```

`<deployConfigurationFileName>` is the name of the replication agent configuration file to use when deploying the File Groups.

`<replicationAgentServerName>` is the server name specified in *ctsrvr.cfg* of the Replication Agent Manager process.

### Example Replication Agent deploy configuration file

Here is an example *ctreplagent.cfg* configuration file for deploying File Groups:

```
unique_id agent1
file_filter <..\config\replication\deployfilefilter.xml
resync_file_list <..\config\replication\deployresyncfilelist.txt
replicate_data_definitions yes
parallel_apply yes
```



```

num_apply_threads 4
source_authfile ..\config\replication\source_auth.set
source_server FAIRCOMS
source_nodeid 10.0.0.1
target_authfile ..\config\replication\target_auth.set
target_server loc
target_nodeid 10.0.0.2
read_timeout_ms 1000
redirect C:\test\primary\data C:\test\replica\data

```

Note that the configuration file matches the Replication Agent configuration except for the following differences:

`file_filter` specifies the name of the file filter that contains the definitions of the File Groups to deploy.

`resync_file_list` specifies the name of the resync file list that contains the names of all the data files to deploy.

### Example Replication Agent deploy file filter

Here is an example file filter containing a definition of a File Group to deploy:

```

<?xml version="1.0" encoding="us-ascii"?>
<replfilefilter version="1" recursive="y" persistent="y" agent="AGENT1">
 <file status="include" fileGroup="nr7778">C:\abac\db\nr7778*.dat</file>
 <purpose>open_file</purpose>
 <purpose>read_log</purpose>
 <purpose>create_file</purpose>
</replfilefilter>

```

### Example Replication Agent deploy file list

Here is an example *deployresyncfilelist.txt* containing the list of all data files to deploy for the File Group (referenced in the above *ctreplagent.cfg*).

```

C:\abac\db\nr7778\abcd\customer.dat
C:\abac\db\nr7778\efgh\account.dat

```

### Remove File Groups from a Replication Agent

Use the following **repadm** command to remove File Groups from a Replication Agent:

```
repadm -c removefilegroups <agentId> "<fileGroup1|fileGroup2|...>"
```

<agentId> is the unique id of the Replication Agent to remove.

<fileGroup1|fileGroup2|...> is a list of File Group names to remove, separated by vertical bar characters "|".

### Remove a Replication Agent

Use the following **repadm** command to remove a Replication Agent:

```
repadm -c removeagent <agentId> -s <replicationAgentServerName>
```

<agentId> is the unique id of the Replication Agent to remove.

<replicationAgentServerName> is the server name specified in *ctsrvr.cfg* of the Replication Agent host process (FairCom DB Server).



## Check Replication Agent Current Action Status

Use the following **repadm** command to check the status of a Replication Agent's current action:

```
repadm -c getactionstatus <agentId> [<requestId>] -s <replicationAgentServerName>
```

<agentId> is the unique id of the Replication Agent to check.

<requestId> is an optional action request id.

<replicationAgentServerName> is the server name specified in *ctsrvr.cfg* of the Replication Agent Manager process.

## Check Replication Agent's Current Action Status Until it Completes

Use the following **repadm** command to check the status of a Replication Agent's current action until it completes:

```
repadm -c pollactionstatus <agentId> [<requestId>] -s <replicationAgentServerName>
```

<agentId> is the unique id of the Replication Agent to check.

<requestId> is an optional action request id.

<replicationAgentServerName> is the server name specified in *ctsrvr.cfg* of the Replication Agent Manager process.

The polling interval is currently 0.5 seconds and polling continues until the action completes (or the utility is terminated with CTRL-C).

## Replication File Group Fixes and Improvements

Further FairCom testing of new Replication File Group management demonstrated additional areas that were improved and corrected.

- When the replication agent reaches the maximum log read position when deploying a file group, the informational message that it writes to *ctreplagent.log* was marked as an error message and is now logged as "informational".
- Reduced filename filter hash memory usage by rearranging structure fields and reduced hash bins.
- Improved filename filter hash performance by skipping checks on the list when a filter count is zero.
- Improved file group deploy phase to use a new method of setting the source server's current log position.
- When the replication agent starts up the first time and the replication agent configuration doesn't specify a starting log read position and the source server has purged transaction log 1, the replication agent fails to start with error 96. We changed the replication agent to obtain the source server's current log position in this situation and use that position as its log read position.
- We improved error message logging for file group deploy file copy operations such as the file filter and resync list files.
- Exclude FAIRCOM.FCS superfile members from checking replication file filters.
- Add code to `checkReplAgentFileFilterLists()` to convert the file name to a full path before evaluating the first filter.



- Add a "hasfullpath" parameter to `ctReplAgentFileFilterGetEntry()` so we can avoid getting the full path when the filename already has a full path.

## 16.46 Replication Demos

### Replication Demo - Replication Using JSON RPC

The Replication Manager Engine provides the JSON RPC API, which allows an application to access and control replication across all registered databases. The replication API is a programmatic interface to the centralized replication metadata in the Replication Manager Database. Each FairCom DB instance can optionally register with Replication Manager. When it does, it transmits metadata about its databases, tables, files, and existing replication plans to Replication Manager. This makes it easy for Replication Manager to create and manage replication plans across all registered servers.

This tutorial may be found in this folder in your package: `drivers\java.rpc.replication`

This exercise demonstrates the use of this API to create and deploy replications plans. See the FairCom documentation for more information:

<https://docs.faircom.com/doc/tutorials/java-rest-replication-api>

### Replication Tutorial - Create test program to demonstrate `ctReplAgentOp()` opcode usage

We added a tutorial to demonstrate using the `ctReplAgentOp()` function. This tutorial can be found here in your package: `.\drivers\ctree.c.replication`. The source code is in `ctrepl_tutorial.c`.

See `.\drivers\ctree.c.replication\tutorials\readme.txt`

Run the tutorial program without any options in order to see its usage.

The example program defines a function that demonstrates a call for each supported `ctReplAgentOp()` opcode.

For example, here is the function that demonstrates the `ctRAOPstartAgent` opcode:



```
/*^*****
/*
Demonstrate calling ctReplAgentOp() with opcode of ctRAOPstartAgent.
This function call starts the specified replication agent.
[IN] agentId- Replication agent id.
Requirements:
1) The caller has connected to the server in which the replication agent
manager is running.
2) A replication agent with the specified replication agent id has
been registered with the replication agent manager (using ctRAOPaddAgent).
Returns a c-tree error code. NO_ERROR indicates success.
*/
static NINT ctRAOPstartAgent_Example(const char* agentId)
{
 NINT rc;
 VRLEN outlen;
 char errbuf[256];
 printf("Starting replication agent %s\n", agentId);
 errbuf[0] = '\0';
 outlen = (VRLEN)sizeof(errbuf);
 rc = ctReplAgentOp(ctRAOPstartAgent, agentId, (VRLEN)strlen(agentId) + 1, errbuf, &outlen);
 if (rc)
 printf("Error: %s: %d\n", errbuf[0] ? errbuf : "Could not start replication agent", rc);
 else
 printf("Successfully started replication agent.\n");
 return rc;
}
/*~*****
```

Example configuration files for the primary and the secondary server are in the config subdirectory.

An example bash shell script showing how to run the test program is in run\_tutorial.bash.

The file readme.txt lists the steps to follow to configure a secondary server to run an embedded replication agent in unmanaged mode (which means that the memphis higher level replication manager is not used in this configuration) in order to run the tutorial.

To run the examples that apply to the replication agent, run the following command:

```
run_tutorial.bash agent
```

The example script adds a replication agent, starts it, and runs commands to get and change its state.

# 17. High Availability

## 17.1 Pacemaker

FairCom's V13 represents a significant investment in support of the Pacemaker high availability solution.

Although we note a handful of the issues here, our efforts extend well beyond this list in order to offer a mission critical cluster platform.

- Improved embedded agent control to properly manage the promotion and the demotion of the ctrees OCF resource
- Improved integration of pacemaker events with c-tree Server startup
- Server role detection and configuration in a Linux cluster
- Cluster role detection logging and sync
- Error code in sync shutdown log messages
- repadm utility option and client API call to indicate if replication agent is operational
- repadm utility isactive command now uses process return code to indicate agent status
- repadm command to remove sync agent state from primary server
- Server's role is changed from secondary to primary, it broadcasts a failover message
- Ensure replication agent is configured on server startup when recovery doesn't need to know server's role
- Store last known primary server node name and its synchronous state in pacemaker cluster
- Support shutting down server when it is waiting to detect its role in pacemaker cluster at startup
- Server startup logic ensure proper behavior for server promotion
- Secondary server startup to properly integrate with promote action
- Avoid possible race condition in re-enabling sync replication on primary server
- Improve diagnostic logging of agent sync shutdown
- Improvements to pacemaker agent script - monitor; stop enhanced for configured timeout;
- Sync commit diagnostic logging of log position copied by replication agent
- Sync replication diagnostic logging of wildcard file name filter list comparisons
- Replication file filter diagnostic logging
- Support demote server function
- Implement new server role of pending secondary
- Deny server promotion if replication agent exception log contains records
- Sync commit diagnostic logging of log position copied by replication agent



- Replication file filter diagnostic logging
- CHECK\_CLUSTER\_ROLE is enabled, trigger server shutdown on SIGTERM
- Resource agent: adjust default timeout values

Please refer to our complete Pacemaker documentation for more information.

## 18. Platform/Compiler Support

### 18.1 Visual Studio 2022 Support

It is remarkable how many versions of the Microsoft compilers we have worked with over the years. The tradition continues with Visual Studio 2022:

```
MSVC++ 14.37 _MSC_VER == 1937 (Visual Studio 2022 version 17.7.0)
MSVC++ 14.36 _MSC_VER == 1936 (Visual Studio 2022 version 17.6.4)
MSVC++ 14.36 _MSC_VER == 1936 (Visual Studio 2022 version 17.6.2)
MSVC++ 14.36 _MSC_VER == 1936 (Visual Studio 2022 version 17.6.0)
MSVC++ 14.35 _MSC_VER == 1935 (Visual Studio 2022 version 17.5.0)
MSVC++ 14.34 _MSC_VER == 1934 (Visual Studio 2022 version 17.4.0)
MSVC++ 14.30 _MSC_VER == 1933 (Visual Studio 2022 version 17.3.4)
MSVC++ 14.30 _MSC_VER == 1932 (Visual Studio 2022 version 17.2.2)
MSVC++ 14.30 _MSC_VER == 1930 (Visual Studio 2022 version 17.0.2)
MSVC++ 14.30 _MSC_VER == 1930 (Visual Studio 2022 version 17.0.1)
MSVC++ 14.28 _MSC_VER == 1929 (Visual Studio 2019 version 16.11.2)
MSVC++ 14.28 _MSC_VER == 1928 (Visual Studio 2019 version 16.9.2)
MSVC++ 14.28 _MSC_VER == 1928 (Visual Studio 2019 version 16.8.2)
MSVC++ 14.28 _MSC_VER == 1928 (Visual Studio 2019 version 16.8.1)
MSVC++ 14.27 _MSC_VER == 1927 (Visual Studio 2019 version 16.7)
MSVC++ 14.26 _MSC_VER == 1926 (Visual Studio 2019 version 16.6.2)
MSVC++ 14.25 _MSC_VER == 1925 (Visual Studio 2019 version 16.5.1)
MSVC++ 14.24 _MSC_VER == 1924 (Visual Studio 2019 version 16.4)
MSVC++ 14.23 _MSC_VER == 1923 (Visual Studio 2019 version 16.3)
MSVC++ 14.22 _MSC_VER == 1922 (Visual Studio 2019 version 16.2)
MSVC++ 14.21 _MSC_VER == 1921 (Visual Studio 2019 version 16.1)
MSVC++ 14.2 _MSC_VER == 1920 (Visual Studio 2019 version 16.0)
MSVC++ 14.16 _MSC_VER == 1916 (Visual Studio 2017 version 15.9)
MSVC++ 14.15 _MSC_VER == 1915 (Visual Studio 2017 version 15.8)
MSVC++ 14.14 _MSC_VER == 1914 (Visual Studio 2017 version 15.7)
MSVC++ 14.13 _MSC_VER == 1913 (Visual Studio 2017 version 15.6)
MSVC++ 14.12 _MSC_VER == 1912 (Visual Studio 2017 version 15.5)
MSVC++ 14.11 _MSC_VER == 1911 (Visual Studio 2017 version 15.3)
```



```
MSVC++ 14.1 _MSC_VER == 1910 (Visual Studio 2017 version 15.0)
MSVC++ 14.0 _MSC_VER == 1900 (Visual Studio 2015 version 14.0)
MSVC++ 12.0 _MSC_VER == 1800 (Visual Studio 2013 version 12.0)
MSVC++ 11.0 _MSC_VER == 1700 (Visual Studio 2012 version 11.0)
MSVC++ 10.0 _MSC_VER == 1600 (Visual Studio 2010 version 10.0)
MSVC++ 9.0 _MSC_FULL_VER == 150030729 (Visual Studio 2008, SP1)
MSVC++ 9.0 _MSC_VER == 1500 (Visual Studio 2008 version 9.0)
MSVC++ 8.0 _MSC_VER == 1400 (Visual Studio 2005 version 8.0)
MSVC++ 7.1 _MSC_VER == 1310 (Visual Studio .NET 2003 version 7.1)
MSVC++ 7.0 _MSC_VER == 1300 (Visual Studio .NET 2002 version 7.0)
MSVC++ 6.0 _MSC_VER == 1200 (Visual Studio 6.0 version 6.0)
MSVC++ 5.0 _MSC_VER == 1100 (Visual Studio 97 version 5.0)
```

FairCom DB V13 formally supports Visual Studio 2015 through 2022. If you need support for an older version, please check with the FairCom Support Team for possible options.

## 18.2 Apple macOS Support: M1/M2/M3 - Code Signing - Monterey; Ventura; and Sonoma

### Apple Highlights:

- We offer support for either Intel or Apple's M1/M2/M3 processors. Our makefile maker utility (mtree) allows you to select your target.
- We properly "Code Sign" our Apps and binaries to assure users that they from a known source and have not been modified since it was last signed. Apple's modern security constraints requires "known source" binaries and users will find installation seamless with our "signed" binaries.
- Apple's hardware and development environment has always been a favorite of many developers. The evaluation and pace of the macOS operating system is consistent each year. FairCom supports these Apple upgrades, and since our last major worldwide release of V12, we have addressed three additional Apple versions: Monterey; Ventura; and Sonoma.



```
Operating Sys...: Apple MacOSX
Select the Version of Operating System:
1. macOS 14.0 (Sonoma)
2. macOS 13.0 (Ventura)
3. macOS 12.0 (Monterey)
4. macOS 11.0 (Big Sur)
5. macOS 10.15 (Catalina)
6. macOS 10.14 (Mojave)
7. macOS 10.13 (High Sierra)
8. macOS 10.12 (Sierra)
9. Mac OS X 10.11 (El Capitan)
10. Mac OS X 10.10 (Yosemite)
11. Mac OS X 10.9 (Mavericks)
12. Mac OS X 10.8 (Mountain Lion)
13. Mac OS X 10.7 (Lion)
14. Mac OS X 10.6 (Snow Leopard)
15. Mac OS X 10.5 (Leopard)
16. Mac OS X 10.4 (Tiger)
17. Mac OS X 10.3 (Panther)
18. Mac OS X 10.2 (Jaguar; Puma; Cheetah;...
 ...the Good Ole' Days...)
As you can see, we have been doing this for a long time!
```

## 18.3 Android Port - Code Compatible

We enjoyed the opportunity to help a customer with an interesting Android device. The device was a set of wearable glasses from a company named RealWear.

<https://www.realwear.com/>

The RealWear Navigator is a wearable Android device. We successfully ported the FairCom Edge Server and its MQTT messaging capabilities to this platform.



Although we do not offer direct Android support "out-of-the-box" (eg.: Tutorials, IDE Projects, etc) we encourage you to contact us if you have a Android needs and we will be happy to help.

This projected resulted in a number of Android adjustment to our technology noted below.

- Internal runtime library function adjustments for successful build with the Adroid ADK.
- 'CMakeLists.txt' - build all of the capabilities of all plugin support offered in a normal Linux.
- AndroidManifest.xml - Update the Manifest file.
- build.gradle - Update the Gradle file
- Build from Android v8.0 (Oreo) Api 26 to Android 10 (Q) Api 29.
- Connection pooling for REST in Android.
- ThingWorx: Ported the ThingWorx client so device can communicate with a ThingWorx server.

## 18.4 Windows ARM Native Port

We are pleased to offer support for the ARM processor on the Windows platform. Based on enhanced speed at lower costs, cloud based Windows virtual machines based on ARM processors have become popular.

This port is available upon request, so do not hesitate to contact FairCom if you have interest in this new platforms.



## 18.5 Linux MIPS Port

We have added the Linux MIPS to the options supported by our "make-maker" mtree tool.

Now select Option 10 to create a makefile for the Linux MIPS platform.

Select your Target:

- |                              |         |                                                               |
|------------------------------|---------|---------------------------------------------------------------|
| 1. Linux x64                 | - 64bit |                                                               |
| 2. Linux IA64                | - 64bit |                                                               |
| 3. Linux IBM P5              | - 64bit |                                                               |
| 4. Linux IBM P7              | - 64bit |                                                               |
| 5. Linux IBM P8              | - 64bit |                                                               |
| 6. Linux IBM P9              | - 64bit |                                                               |
| 7. Linux IBM Mainframe s390x | - 64bit |                                                               |
| 8. Linux ARM (generic)       | - 64bit |                                                               |
| 9. Linux Raspberry           | - 64bit |                                                               |
| 10. Linux Mips               | - 64bit | <b>&lt;&lt;&lt;&lt;&lt;&lt;&lt; Here &lt;&lt;&lt;&lt;&lt;</b> |

## 18.6 Legacy platforms remain supported

We would like to remind our customers as to the extent of platforms still supported by FairCom. Please feel free to contact us if you continue to have interest in any of these platforms:

- windows\win32 •| windows\win32.unicode •
- windows\winX64 • windows\winX64.unicode • windows\winARM64
- aix\aix.v4.3.32bit • aix\aix.v5.1.32bit • aix\aix.v5.1.64bit • aix\aix.v5.2.32bit • aix\aix.v5.2.64bit •
- aix\aix.v5.3.32bit • aix\aix.v5.3.64bit • aix\aix.v6.1.32bit • aix\aix.v6.1.64bit •
- aix\aix.v7.1.32bit • aix\aix.v7.1.64bit •
- freebsd\freebsd.v10.32bit • freebsd\freebsd.v10.64bit •
- freebsd\freebsd.v8.32bit • freebsd\freebsd.v8.64bit •
- hpux\hpux.v11.ia64.32bit • hpux\hpux.v11.ia64.64bit •
- hpux\hpux.v11.risc.32bit • hpux\hpux.v11.risc.64bit •
- linux\linux.ia64.64bit • linux\linux.mips.32bit • linux\linux.mips.64bit •
- linux\linux.p5.32bit • linux\linux.p5.64bit •
- linux\linux.p8.little.endian.32bit • linux\linux.p8.little.endian.64bit • linux\linux.p9.little.endian.64bit •
- linux\linux.v2.4.arm.32bit • linux\linux.v2.4.x86.32bit • linux\linux.v2.6.arm.32bit •
- linux\linux.v2.6.arm\_generic.32bit • linux\linux.v2.6.arm\_generic.64bit •
- linux\linux.v2.6.p7.32bit • linux\linux.v2.6.p7.64bit •
- linux\linux.v2.6.raspbian.32bit • linux\linux.v2.6.raspbian.64bit •
- linux\linux.v2.6.x86.32bit • linux\linux.v3.0.s390x.64bit •
- linux\linux.x64.64bit • linux\linux.x86.32bit •
- macosx\osx.v10.13.32bit • macosx\osx.v10.13.64bit •
- macosx\osx.v10.14.64bit • macosx\osx.v10.15.64bit •



- macosx\osx.v11.00.64bit • macosx\osx.v11.00.arm64 •
- macosx\osx.v12.00.64bit • macosx\osx.v12.00.arm64 •
- macosx\osx.v13.00.64bit • macosx\osx.v13.00.arm64 •
- macosx\osx.v14.00.64bit • macosx\osx.v14.00.arm64 •
- qnx\qnxrtp.v6.32bit • sco\sco.v6.32bit •
- solaris\solaris.v5.10.sparc.32bit • solaris\solaris.v5.10.sparc.64bit •
- solaris\solaris.v5.10.x64.64bit • solaris\solaris.v5.10.x86.32bit •
- solaris\solaris.v5.11.sparc.32bit • solaris\solaris.v5.11.sparc.64bit •
- solaris\solaris.v5.11.x64.64bit • solaris\solaris.v5.11.x86.32bit •
- solaris\solaris.v5.8.sparc.32bit • solaris\solaris.v5.8.sparc.64bit •
- solaris\solaris.v5.9.sparc.32bit • solaris\solaris.v5.9.sparc.64bit • solaris\solaris.v5.9.x86.32bit •

As this list verifies, our technology was architected for portability from its inception. If you need help with any specific operating system, please do not hesitate to contact us.

## 19. Docker Support

FairCom Technology (which includes FairCom DB, FairCom RTG, and FairCom Edge for purposes of this discussion) is easy to wrap up into a Docker Container. Given that installing FairCom Technology is simply expanding a gzipped tarball on Linux/Unix systems or expanding a .zip file on Windows, FairCom finds that most customers desire to create their own Docker Container from scratch. Therefore, this document provides a few tips to simplify the creation of Docker Containers with FairCom Technology.

Note that we are using the FairCom default ports throughout this discussion.

### 19.1 Docker: Scripts to create Docker image

We have added a Demo script to create a docker image containing the FairCom DB/Edge server.

The build\_image.sh script takes as a parameter the name of the FairCom distribution package to extract the server. So far the script and the package have been tested from the same directory. The script has logic to correctly name the script and tag with the version the resulting docker image. The docker image will be put in a tar.gz file.

Other support includes:

- Added a second parameter to the build\_image script that when specified replaces the default license in the source package with the specified license.
- Expose additional websocket port: Added the MQTT websocket port to the exposed ports list
- Created a new flavor of the image with OpenJDK enabled for stored procedures and triggers  
Realized it could be worth to create a docker image with the JDK to automatically enable Java stored procedures and triggers. Implemented a new docker flavor where the JDK is already configured in ctsrvr.cfg so the docker out of the box is able to run stored procedures and triggers
- Added text editor in docker image: The Docker image doesn't contain a text editor to change the configuration file inside docker containers. The image is built on a very thin image without any text editor shipped. We added instructions to the Dockerfile to make sure a default text editor (nano) is installed in the docker image.



## 20. Java JDK Version Considerations

### 20.1 Java Version Support

FairCom DB has been tested against and supports Java 8, 9, 11, and 17 for c-treeDB Java, JDBC, and SQL Stored Procedures, Triggers, and UDFs. For backward compatibility, some .jar files are maintained with Java 7 and prior.

Our packages now currently contain these five JDK Versions for JDBC and SQLSP:

```
ctreeJDBC_1_7.jar
ctreeJDBC_1_8.jar
ctreeJDBC_1_9.jar
ctreeJDBC_1_11.jar
ctreeJDBC_1_17.jar

ctreeSQLSP_1_7.jar
ctreeSQLSP_1_8.jar
ctreeSQLSP_1_9.jar
ctreeSQLSP_1_11.jar
ctreeSQLSP_1_17.jar
```

The default *ctreeJDBC.jar* is the *ctreeJDBC17.jar*.

### 20.2 Java Stored Procedures and Triggers Require JDK V1.7 or Newer

**Note:** The FairCom DB 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. FairCom DB 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.

# 21. Miscellaneous

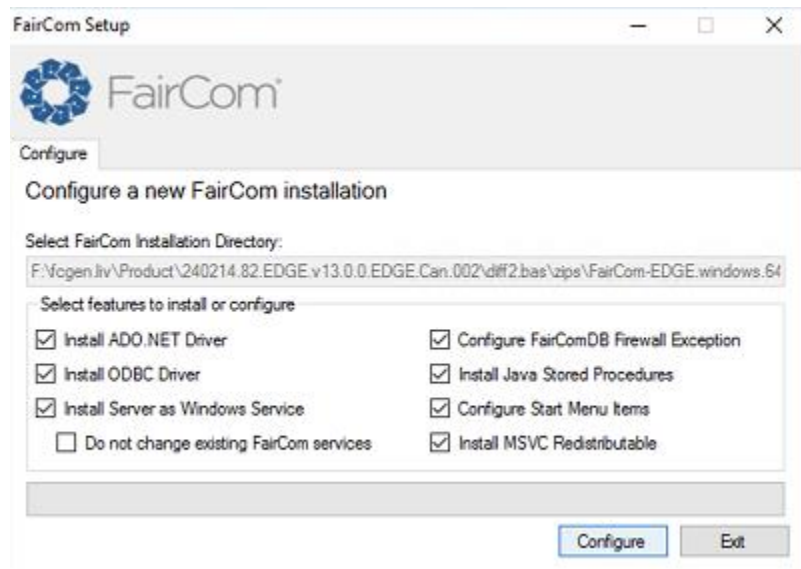
## 21.1 Windows Installers and FairComConfig Improved

FairCom distributes its Windows packages in two forms: Zip files or Windows .msi Installers.

Both packages are the same, yet the .msi files provide an "end-user" convenience.



User who prefer the Zip file format, can simply un-zip our package in any location and then execute the '\\tools\\SetUp\\FairComConfig.exe' Utility to configure the following options. Note, the .msi Installers execute the same 'FairComConfig.exe' Utility (internally) automatically to configure these options.



- Installs the FairCom DB/Edge Server as a Windows Service.
- Installs the ADO.NET Driver and sets the proper internal (machine.config) configuration files.
- Installs the ODBC driver AND updates the Windows Registry.
- Configures the FairCom DB Firewall Exception.
- Configures the Java JVM for FairCom DB Stored procedures.
- Configures the Windows Start Menu Items
- Checks for and Installs Microsoft's MSVC Redistributable modules.

The following changes have been made since our last major release of V12.

- ADO.NET and ODBC Installers were not present in the original V12 builds. This has been corrected.
- Firewall Options Handling: Improved firewall checkbox; checks; firewall rules; duplicate names.
- ADO.NET Improved GAC removal logic: Make sure to remove all possible drivers.
- ADO.NET Improved the removal from the registry: remove registry key only when it exists.

Other improvements:

- Preserve 'Config' folder when un-installing: Using the Windows un-install, we now will save your server's 'Config' folder in 'Config.sav'.
- Installers make sure the MSVC redistributable are installed when required.
- Support side-by-side Install - Added unique build reference numbering into each generation.



## 21.2 Windows Installer Error 1722 Fixed

When installing V12, the Windows installer failed with error **1722** while calling **FairComConfig.exe**.

On certain machines where Visual Studio 2015 has been uninstalled, some of its registry keys remain, causing **FairComConfig** to falsely detect the presence of Visual Studio 2015. The error is caused when **FairComConfig** attempts to install without a full set of Visual Studio registry keys. The logic has been modified to properly address this situation.

## 21.3 Install in Program Files

The ISAM Server installer (available upon request) supports installing in the *Program Files* folder on the Windows platform. The installer also inserts a firewall rule to add the FairCom DB server executable as an exception. The **FairComConfig.exe** utility has been updated to correctly handle the ISAM server name and to integrate capabilities to handle the Windows Firewall exception if changes are needed after the *.msi* installer has been executed.

## 21.4 VSS Writer: Volume Shadow Copy Service

We continue to support the Microsoft Volume Shadow Copy Service. Minimal changes have been applied since V12.

A shadow copy is a snapshot of a volume that duplicates all of the data that is held on that volume at one well-defined instant in time.

The FairCom Server is a Writer: Writers are applications or services that store persistent information in files on disk and that provide the names and locations of these files to requesters by using the shadow copy interface. During backup operations, writers ensure that their data is quiescent and stable—suitable for shadow copy and backup.

The FairCom VSS Writer support DLL is located in the server folder:  
.\server\c-treeACEVSSWriter.dll.

- Upgraded from Visual Studio 2012 to Visual Studio 2015 to build our 'c-treeACEVSSWriter.dll'. The 2015 compiler provides a good (backward) compatible base for this technology.
- VSS Writer name has not changed: When we changed our product name from c-treeACE to FairCom DB, we realized that if we changed the VSS Writer name, we would introduce a compatibility problem for our customers in the field. Therefore the VSS Writer name has not been changed and remains the same: 'c-treeACEVSSWriter'.

## 22. Bug Fixes

This chapter provides a list of the bugs we have corrected in this release. Although many of these were caught internally by our developers or QA team, we have chosen to provide the full list just in case you encounter an issue in an area we may have already corrected. We are proud to show the extent of our efforts over the last few years. Our progress has been substantial.

### 22.1 CTDB Fixes

#### Important Fixes

- CTDB - coding bug in `_ctdbAdjustTableSegmentCount` - r339252

#### Additional CTDB fixes

- CTDB - failed `ctdbAlterTable` may leave the table handle in inconsistent state leading to internal error - r324849
- CTDB - support for proper `ctdbSetFilter()` operations when using CTDBSAPP - r328673
- CTDB - new logic to automatically assign index names - r328687
- CTDB - alter table to add an index in standalone mode may succeed but the index is not created - r330339
- CTDB - `ctdbSetFieldType()` - r330709
- CTDB - `dbtype="date"` over `type="Alphanum"` not properly handled resulting in error 4028 - r335567
- CTDB - Attach session failing with CTDBRET\_INVATTACH in MTCLIENT model - r335868
- CTDB - possible crash setting a JSON field when CTDB\_BEHAVE\_CTJSON\_CHECK\_VALUE is turned on (default) - r337784
- CTDB - possible deadlock error running `ctdbDeleteTable()` or `ctdbDropTable` with SQL sessions - r338503
- CTDB - `ctdbRenameTable/ctdbMoveTable` may not rename indices - r340132
- CTDB - Heap corruption in `ctdbAttachSession` - r249674
- CTDB - `ctdbSetBatch` may cause crash when field mask is in use - r255324
- CTDB - `ctdbSetPathPrefix()` causes wrong symbolic alternate index name to be used - r255473
- CTDB - wrong data returned (and errors) if the table is VLEN with fixed record len > 0 and the very first field is a CT\_STRING - r255860
- CTDB - Memory leak in `LOCATE()` conditional expression function - r260131
- CTDB: Memory Leak in `ctdbAlterTable` using conditional indexes - r260349
- CTDB - `ctdbAlterTable()` fails with UALC\_ERR when adding a index with a JSON based column - r274206



- CTDB - ctdbCloseTable() mistakenly resets the value set by ctdbGetTableObjptr and returned by ctdbSetTableObjptr - r287060
- CTDB - numeric sign conversion missing in index information for CT\_SIGNEDCOMP2 and CT\_SIGNEDCOMP3 - r289167
- CTDB - Add overflow/underflow checks to c-treeDB functions that convert 64-bit integer or float to currency and convert currency to string - r289816
- CTDB- LLONG\_MIN handling for c-treeDB data conversion functions - r289875
- CTDB - overlapping fields in condition fields definition cause crash - r289997
- CTDB - ctdbSetFieldAsBlob() on UNICODE field sets wrong value - r292392
- CTDB - ctdbGetNAVFieldProperties wrong mapping for CT\_STRING fields - r292411
- CTDB - batch operation may crash the server - r293169
- CTREE - DoBatch/DoBatchXtd crash when using column filtering after a BAT\_CAN call - r293358
- CTDB - ctdbGetNAVFieldProperties returns wrong field type for NAV\_VARCHAR fields with length very close to the maximum permitted - r293173
- CTDB - ctdbNumberToString may overrun output buffer - r298064
- CTDB - Filter expressions leak memory if errors occur during parsing - r298065
- CTDB - Large filter expression may overflow the stack - r298067
- CTDB - Stack overflow in recursive calls to ctxmlFreeNode() when a large XML filter is freed - r298766
- CTDB - memory leaks in ctdbCreateTable - r313303
- CTDB - incorrect handling of ZSTRING BTRV data type in index - r315223
- CTDB - sqlimport logic may not set the index name properly in SQL - r315276
- CTDB - ctdbSetIndexDistinctKeyCountFlag([...], NO) on a just created index causes a crash - r316258
- CTDB - setting time/timestamps using ISO8601 string with fractional part produces wrong time
- CTDB - filter set by ctdbSetBatchFilter may not be reset at batch end - r318837
- CTDB - Crash in \_ctdbLocalNextBlockFSRead() after batch read call fails - r319786
- CTDB - ctsqlimp must no import indexes with ALTSEG, ENDSEG or not matching the database case sensitivity - r320644
- CTDB - ctdbAlterTable crash when using rowid segment in user index - r269373

## END OF CTDB FIXES



## 22.2 CTREE Fixes

### Important Fixes

- Rebuilding an index that doesn't allow duplicates on a low/high system doesn't properly sort duplicate keys by record offset, causing unexpected record purging behavior - r324331
- Failed system id call on file creation leads to stale cache pages associated with wrong file - r334168
- Renamelfil must check and maintain the system file id - r334323
- Recovery fails when using log encryption after setting OPENSSL\_ENCRYPTION NO - r272696
- Transaction recovery completes successfully, but data or index errors exist - r278399
- Recovery: Insufficient transaction logs in trandat() or tranidx() may be undetected - r278404
- Transaction log corruption during recovery operations - r277273
- Allow node pruning on replicated files - r277485
- Updates to variable length data file may be missing after crash if using deferred transaction begin and updating a record more than once in a transaction separated by savepoints - r311781
- Replication Deploy Effort - Fileblock may permit multiple threads in the core - r318498
- BLKIREC() Major Fix: A successful call to BLKIREC() that specifies a blocking condition unlocks the record that was read - r319412
- Deferred indexing thread hangs - r249023
- Server startup fails with RCHK\_ERR when using LOG\_ENCRYPT YES - r253250
- RDVREC may return truncated FLEXREC record without error code - r255840
- Crash while updating FLEXREC record - r255841
- Ctree file header updates may be missing at recovery time - r256158
- Crash on failed OPNISAM() or CREISAM() - Intel Solaris x86 - r335797
- Call to CLISAM() crashes in client library after a failed CREISAM() call - r335867
- Reading a record from a transaction-controlled fixed length record superfile member in physical order fails with error 158 - r339507
- When a record in a transaction-controlled file is updated more than once in a transaction with a pattern of "update, savepoint, update, update", reading the record may return an outdated record image - r339568
- Server crash if shutdown during plugin\_load - r249110
- Partition Host Fix: Incorrect locking on partition files when using RRDREC() - r265056
- Transaction log incorrectly encrypted after a failed master password change - r289656
- Mongoose crashes due to race conditions - Production Crash with OpenSSL openssl.1.0.2t - r303727
- Server or standalone crash may occur when inserting key into an index that uses our original key compression - r251653
- Server crash when server is using different MAX\_NAME definition than client - r249675



### Additional CTREE fixes

- Server crashes after a record add, delete, or update calls a record update callback function that opens a file - r322443
- Table lock counter is not incremented when the connection is requested to terminate right after its blocking table lock request is granted - r322616
- Optimistic Locking: Record read on a table that has a change id field fails with error 445 - r322764
- Optimistic Locking: Adding or updating a record in a table that has a changeid field can fail with error 1193 if the transaction started before file was opened - r322775
- Server crash if keepopen encounters pending index delete - r323245
- Wrong transaction log size message when increasing transaction log size - r323256
- ctQUIET() call may fail due to failure to clear threads from database core engine when using LDAP authentication - r323896
- Crash adding index when out of file control blocks - r324714
- Server crash when parsing a corrupted hot alter table resource - r327208
- Incorrect schema id value set in REPIIMAGE transaction log entry - r327747
- Possible buffer overrun in setDARfields() function in client library during record add or update when using identity or changeid field - r327749
- Record read on memory file that uses hot alter feature returns record image with additional 4 bytes at start of buffer - r327857
- Crash opening file with Corrupt DAR resource - r328320
- Server hang during abnormal server termination - r330314
- Automatic pagesize conversion failing on linux - r330707
- Internal functions for managing temporary directories affect lock state - r330715
- ISAM level file open call fails with error DMAP\_ERR (957) - r331265
- Shutdown leak - r331638
- CTREE - expression (filter/conditional index) execution enhancements - r330335
- CTREE - Fix REPEAT condition expression function bug introduced in revision 330335 - r330669
- File open fails with DMAP\_ERR due to stale ISAM key mapping preventing clearing of index reverse map pointer - r332705
- Distinct key estimates incorrect after recovery - r333086
- Server thread may hang in call to write to CTSTATUS.FCS; and startup status log write check silently fails - r335145
- Table locks, server might terminate with internal error 7031 or 7468, or stack overflow, or lock request hangs - r335871
- Hot Alter Table Field Conversion Callback - ctGetRecordConverters() call with output buffer size of zero causes client to become out of sync with server when using TCP/IP protocol - r335872
- SECURITY() call with mode of SEC\_CHECK\_ADVENC\_PASSWD fails with error 26 - r336573



- GETCURP() returns wrong record offset after successful EQLREC() call with NULL output buffer - r336763
- Server crashes in CREISAM or OPNISAM due to 4-byte file number changes - r336993
- COMPATIBILITY FORCE\_WRITETHRU - Recursive file header mutex request during online compact for ctWRITETHRU ctPREIMG file - r337804
- COMPATIBILITY WTHRU\_UPDFLG option can cause header changes for non-transaction-controlled file to be lost when file gets closed - r339148
- PREIMG index created in PREIMG transaction with data file open in shared mode doesn't get associated with data file - r338131
- Crash in client library on function call that references the last file number in the parameter file after OPNISAM or CREISAM call - r338531
- KMAT\_ERR on memory file on 32bit system - r338532
- Transaction that creates an index file, adds keys, and undoes the index file creation fails with error 69 - r339147
- Possible infinite loop in setDARfields() - r339149
- Transaction commit failure on source server can cause log scan to fail with error 749 - r339259
- File block of partition member may crash server when accessing the member after it is unblocked; file block of host causes threads to hang after unblocking - r339955
- FACS\_ERR(26) after canceling large rebuild - r340123
- Unneeded transaction log files may remain on disk after active transaction log count decreases - r340494
- Transaction Dependent File fixes
- Temporary files remain on disk after abort of transaction that contains a c-treeDB alter table operation - r331905
- Temporary file cleanup change can cause server to crash - r333423
- Client call hangs or returns error 128 after a call to ctDROPIDX() when using the TCP/IP protocol - r335978
- Support transaction dependent behavior for non-transaction-dependent files; and automatically enable transaction-dependent behavior for transaction-controlled files - r336041
- Support transaction dependent behavior for non-transaction-dependent files; and automatically enable transaction-dependent behavior for transaction-controlled files - part 2 - r336042
- Support transaction dependent behavior: ChangeISAMContext() fails with error 10 after drop of transaction-dependent index - r336178
- Support transaction dependent behavior: CTREE - PUTFIL to turn transaction on for partitioned files fails with error 99 when ctBEHAV\_AUTO\_TRANDEP is on - r336202
- Startup failure during automatic superfile pagesize adjustment - r249111
- CTREE - ctTruncateFile and compact functionalities in general lose changeid information - r312076
- Adding a record to a partitioned file with a change id field fails with error 1193 if using automatic transactions - r314904



- Hot Alter Table Field Conversion Callback - Record read fails with error 454 after hot alter is used on a table that already had hot alter used with conversion function - r305030
- Server Cache Enhancement: Server crash when configuration option is changed at runtime, caused by changes applied for cache resizing feature - r282772
- Superfile PAGE\_SIZE: - ctscmp utility leaves temporary files behind when changing page size- r253876
- Superfile PAGE\_SIZE: - automatic SQL/CTDB superfile pagesize conversion skipped - r271898
- Server crash trying to update file on readonly volume - r252819
- Crash in savicon() during OPNICON() call caused by previous context pointer used after being freed - r250938
- File copy fails with error 1014 when alternate index name contains a path that differs from the current location of the data file - r253886
- Reading a record in client/server mode from a superfile member that uses FLEXREC and CMPREC fails with error 43 - r252094
- SECURITY() function - Crash in ctchkadmn() when lctgv is NULL - r254217
- 4k drive sector size support for linux - r253552
- ctscmp utility does not properly copy compressed data records - r247197
- Recovery fails with error 12 / 413 after superfile host rename - r257738
- Calling PUTHDR() with mode of ctTIMEIDhdr on a transaction-controlled superfile member fails with error 48 - r257977
- Crash on memory allocation failure - r258085
- Large index rebuild may fail with memory allocation error or use excessive memory - r258415
- Possible data loss on Linux after system power loss for a transaction-controlled file that uses the ctDUPCHANNEL file mode - r259320
- READ ONLY Server Fix: Prevent SQL database create by client when server is read only - r261382
- File open optimization and system file id bug fixes possibly related to error POPN\_ERR(1130) - r263009
- Server file open call might fail with error ISRL\_ERR(554) when more than one connection opens the file at the same time - r263013
- Crash in client library frmkey() function after blocking and unblocking file - r263147
- Crash in client library when losing connection to server. - r263339
- File block of superfile host with ctFBautopen option does not reopen superfile members that were open when the file block occurred - r263148
- Using IICTs and deferred file closes and reopens can cause index to be inconsistent with data file - r263349
- File create fails with error 71 when using AUTO\_TRNLOG server configuration option - r263350
- Invalid memory accesses at shutdown - r263522
- Compacting a file with FILESYS\_COMPRESS\_FILE in effect does not enable Windows file system compression for the compacted data file - r263816



- Server's transaction data and transaction index background flushes unintentionally default to immediate - r264645
- Review tstupd() usage to ensure that return code is checked - r265027
- Deleted nodes in this file to be lost (and other issues) - ignore failure in putdelnod81 - r265041
- Slow server performance (logon, logoff, file open and close slow or appearing to hang) and increasing number of transaction logs due to log scan during file id reassignment - r265588
- PARTITIONED FILES FIX: Crash opening partition host with ctdbOpenTable(CTOPEN\_DATAONLY) - r266120
- OCON\_ERR(591) after adding or removing Index to existing data file. - r266369
- Connections to server hang after unexpected connection error occurs - r266527
- AUTO\_CLNIDX fails with error 69 on encrypted index file - r269008
- Table rename fails with RENF\_ERR (67) system error EXDEV (18) - r269475
- Unexpected file open behavior when using other extended file open mode with blocking file open retry extended file open mode - r270330
- Partial variable length record read may fail with error 160 when a key segment extends into the variable-length region of the record - r271582
- Server startup fails with error 90 - r271900
- PUTHDR() with mode of ctSERNUMhdr on ctTRNLOG data file fails with error 71 if no transaction is active - r272280
- Denial of service vulnerability - r272388
- Server crash after failed memory allocation - r272394
- Enabling compression when compacting data file causes error 198 - r275735
- Wrong record length passed to record update callback function - r276403
- COMPRESS\_FILE has no effect when XCREblk specifies ctAugmentedFxd or ctFLEXREC - r278009
- Fix for file open failing with error 1156 due to DEF\_MASK\_DODA2 bit not set in header - r278568
- Copying in different version of file that is closed by quiesce can cause file open to fail with error 1130 or hang - r279567
- ctQUIET() hangs when a rebuild is in progress - r279774
- Master password change may take 60 seconds to quiet server - r280612
- Client side hang or network error following call to Add, remove or modify a user account or group. - r281051
- Server crash in reader/writer lock release call from ctplugin\_logger() - r281143
- Notification messages not sent for partitioned files, causing BLKIREC() to time out - r281723
- Adding a record to a partitioned file that uses an extended key segment fails with error 707 - r281916
- Server connections hang due to file user count overflowing, and incorrect task id used for traversing multiple open links in record unlock routine - r282457
- Server crashes during space reclamation of replicated partitioned files - r282623
- Server crash due to heap corruption caused by double free when RCESDPConnect() fails - r283225



- RRED\_ERR (407) opening encrypted file after increasing page size - r283231
- PRMIIDX8() call fails with error 669 (XCRE\_ERR) in client/server mode. - r283932
- Server crash in \_consumerThrdReadFunction() - r284398
- Server crash reading malformed DODA - r286732
- ctFBsysclose file block on superfile host may cause server termination due to failed write of updated node while superfile host descriptor is closed - r287069
- Server crashes when using file block on a partitioned file - r287070
- READ\_ERR when opening a table that uses auto system time resource and 8-byte file offsets - r287343
- Crash during file block of superfile or partitioned file on system that requires pointer alignment - r287925
- Server crashes in iOPNFILX() during failed startup due to NULL system file ID list entry pointer - r288536
- ctChangeMasterPassword fails when LOG\_EVEN/LOG\_ODD in use - r289232
- Read next or previous on partitioned file may return incorrect record when using non-partition index - r289238
- BLKIREC() crashes in expression evaluation function when reading a variable-length record - r289754
- Records missing from query that uses a data filter on a partitioned table - r290280
- CTREE - ctAttrSetAttributeBinaryValue may store the wrong value (unified resource) - r295455
- NULL KEY VALUES Support - Check for invalid inulkey values when adding IFIL resource - r296438
- Rebuild fails with error 674 on compressed record data file - r296094
- Rebuild of fixed length file with record size over 32K fails with DREC\_ERR (21) - r297441
- Infinite loop in iclsfilx() when add of index member with host index open in shared mode fails - r297595
- Server crashes when custom function passes an out of range function number and function monitor is on - r298060
- Unicode server crash when environment variable value in configuration file exceeds 255 bytes - r298082
- Some code incorrectly checks isam\_err instead of uerr\_cod after a failed call to tstifnm() - r298396
- Server may hang when using forced blocking lock request feature - r299798
- Server logs system file id error messages to CTSTATUS.FCS during normal operation - r301249
- Replace vfork() with fork() in cases where the child process must make system calls - r302581
- Server hang on windows if unhandled exception occurs - r302591
- Automatic recovery may hang when checking index delete stack that is corrupt - r302818
- File open or create call might hang if file name exceeds maximum file name length - r303216



- Adding an index to a data file that is open in shared mode - Undo changes to connection's list of pending indexes when savepoint restore or transaction abort undoes add of index to transaction-controlled data file open in shared mode - r304779
- Adding record to ctAugmentedFxd file ignores the option to suspend automatic filling in of identity field value - r305208
- File with corrupt Compression resource crashes server - r308589
- Reading a queue message written by ctThrdQueueWriteDirect() leaks memory - r308633
- Limit shared memory read to size of shared memory region on Windows - r309206
- Crash in sndblk() in client library when using PRMIDX82() to specify extended create blocks or index conditions - r310147
- Source Code Security Project - locate() function in data filter does not find a string that matches the source string exactly - r266917
- Source Code Security Project - Memory leak or possible crash in \_ctdbPutDict() when using mirrored table with c-treeDB - r266989
- Source Code Security Project - Out of bounds array access in SQL parser for characters with high bit set - r271394
- Source Code Security Project - Clean up missing breaks in case statements, wrong sizeof, and incorrect variable assignment - r273077
- Source Code Security Project - Improper use of sizeof causes data truncation; possible crash in \_ctdbDUPIFIL() - r273079
- Source Code Security Project - Clean up out of bounds reads and writes and other issues reported by code analyzer - r273094
- Source Code Security Project - SQL server crash in sql\_tblinfo\_t destructor caused by delete[] changes - r273548
- PRINTF-LIKE FUNCTIONS: Add compiler check for printf-like functions to check consistency of format specifier and parameters - r270844
- Hot alter table does not update schema definitions in partition members - r265010
- CTDB - ctdbAlterTable() may mistakenly change field alignment - r265032
- Hot alter fails with error 554 when SRLSEG index exists - r296523
- Aborting a transaction that deletes and adds an identity field causes subsequent file open to fail with error 916 - ctBEHAV\_ABORT\_TRAN\_IN\_REVERSE - r301951
- ctAlterSchema - heap corruption - r306706
- CTDB - ctdbAlterTable error UNQK\_ERR adding an index to a replicated table - r306717
- c-treeDB hot alter table sets wrong default value for newly added field - r306990
- Improve performance of converting records to current schema version when reading records following a hot alter table operation - r307207
- Record read fails with error 153 after hot alter table adds a fixed length field with default value length less than the field length - r307552
- Field added by hot alter table that uses an RTG data type might have wrong default value - r307680
- Crash in cnvfld() due to invalid conversion function pointer when using a user-defined hot alter schema field conversion function - r309638



- Server crash in flxrConvertRecordToCurrentSchema() when using hot alter table on a replicated file that uses data compression - r311279
- Record read fails with error 4108 after hot alter changes size of length count for string field type - r315014
- Cache Fix: Enable data cache empty lists by default and implement cache resize support for data cache empty lists - r303525
- Index rebuild may hang when canceled by rebuild callback - r304067
- Allow file close record update callback function to call c-tree server functions during file block close of file - r304843
- Server crashes during batch column filtering when record read fails with error 160 - r306131
- Check for reaching end of record buffer when transforming extended key segment - r306426
- On Unix systems, library name conversion to standard format is incorrect if the library name includes a path - r306996
- Deferred indexing using wrong old record length - r306997
- Rebuild of index with data file open shared fails with error 62 (LERR\_ERR) - r307964
- Use of savepoints causes preimage space entries to accumulate until transaction commits - r308849
- Windows server crashes when viewing function monitor window after turning on function monitor logging to a file and turning it off - r308872
- Invalid field/key type error (-21019) trying to read UTF8 character fields on unix - r311675
- Rebuild fails with 39 error on compressed data file - r314117
- Inserting record to partitioned file with unique index on identity field fails with duplicate key error - r314677
- GETKSEGDEF() does not check that the extended key segment type matches the requested type - r316836
- GETKSEGDEF() bug fixes: output length; - r317133
- Move starting of deferred indexing threads after plugins have loaded - r317130
- Record update that does not change the key value fails with error 1001 for a file that does not allow ISAM key updates when the index allows duplicates - 317137
- Server terminates with fatal error for large transaction due to preimage swap feature discarding high word of offset to write - r317211
- Server DLL Interface Technology: ctSetConfigurationFile() crashes - r318810
- ADDVREC, RWTVREC, and RWTREC fail with error 1193 (CHANGEID\_VALUE\_TOO\_LOW\_ERR) when using automatic transactions - r319416
- Excessive and wasteful delete node transaction activity - r319772
- Server DLL Interface Technology: Server crash when calling ctdbEvalExprAsString() - r320937
- Support opening ctdb dictionary superfile more than once in a connection - r321688
- Server crashes in tstifnmcls() after record update callback function closes files during file block of partitioned file - r321698
- Server crash when logging connection limit error during failed call to ctThrdAttach() - r310343
- Server shuts down with internal error 8601 during batch call - r315936



### StandAlone

- CTDB - alter table to add an index in standalone mode may succeed but the index is not created - r330339
- Reading a record in physical order from a transaction-controlled superfile member data file in standalone mode fails with error 158 after record add is undone - r339531
- Undo of partition member create on Windows in standalone mode does not delete an index file that has index members - r339564
- PUTHDR() in non-tranproc library does not write header for tranproc file - r269370
- Stand-Alone ctcmpcif encrypt option parsing problem - r306704

### END OF CTREE FIXES

## 22.3 SQL Fixes

### Important Fixes

- SQL - UNICODE like '%' operator on fixed length column causes server crash - r251652
- SQL - unexpected error when parsing multiple SQL statements simultaneously - r286882
- SQL database upgrade hangs at startup - r270097
- SQL UNICODE query writes past end of buffer - r280108

### Additional SQL fixes

- SQL - wrong bigint value may get stored on data files on big endian machines - r324790
- SQL - query using view returns error and causes a panic situation - r325189
- SQL - query fails with error -16304 "LT\_ERR\_INTERNAL - Internal Latte error detected." - r333051
- SQL - fetch with scrollable cursor hides possible errors - r334311
- SQL - Fetch a tuple from a LATTE table given a tuple id fails with LT\_ERR\_LIMIT error (when multitbl is effective) - r334313
- SQL - query with where clause on datetime field on RTG table returns wrong result - r338220
- SQL Server Crash - PANIC - XEC fldref::execute - r249100
- SQL - possible memory overwrite in SQL concat and replace functions - r249112
- SQL - scalar subquery with outer references as parameter to scalar function may return internal error - r253537
- SQL - stored procedure deploy function succeed without actually deploying the stored procedure - r253543
- SQL - [N][VAR]CHAR conversion/cast to [VAR]BINARY right trimmed spaces - r250802



- SQL - possible crash using a constant in the group by clause - r255323
- SQL - not thread safe registering of logs buffer on client causes ODBC and DSQL multithreaded applications to crash - r258222
- SQL - LVARCHAR - SQLQA test jira131 fails on UNICODE servers - r258318
- SQL- wrong results for UNICODE server on Unix platforms - r258408
- SQL - server crash preparing a SQL query with case and parameters - r264011
- SQL- Wrong number of Parameters Returned when using Table Valued Function - r264653
- SQL - latte crashes on memory allocation failure - r265008
- SQL - query from MS Access causes server to crash. - r266508
- SQL - SQL UNICODE buffer overrun - r267926
- SQL - UNICODE server crash in psr\_env\_t::psr\_get\_sfnop() - r268126
- SQL - server exits when parsing query with large literal - r270949
- SQL - crash using scrollable cursors - r285108
- SQL: Shared memory - Invalid handle passed to socket functions in dxp\_rqtcp\_hdl\_t::dxp\_rqtcp\_hdl\_t() when using SQL shared memory protocol - r285932
- SQL: server crash getting list of built in procedures - r285746
- SQL - update statement crashes the server - r285954
- SQL - consolidated fixes to recursive queries - r286157
- SQL - DBSchema: Crash when trying to export SP/T - r289091
- SQL - crash - possible server crash when query specifies a field that does not exists. - r290292
- SQL - server crash at client connection time - r290293
- SQL - DSQL - error updating more than one long var field using parameters - r293015
- SQL - A query with IN over an INTERSECT returns wrong resultset. - r298285
- SQL - A query with IN over an INTERSECT returns wrong resultset. - r300236
- SQL - client side logging subsystem may crash when reusing a connection in a different thread. - r302686
- SQL query with aggregate returns incorrect results - r307064
- SQL - failed create table as select leaves the table name as in use - r308956
- SQL - Server crash in opt\_rm\_redundant\_idx\_canddts- r308965
- SQL - signal when "commit work" fails and actually aborts the transaction - r310892
- SQL - unexpected commit error dropping constraint - r311035
- SQL - multiple drop index leave the table in inconsistent index definition state - r311674
- SQL - Unicode server error -16106 caused by changes to latte stub function stub\_lt\_tpl\_hdl\_t::ReadField() - r313048
- SQL - query crashes when aggregating a sum of numeric values - r314660
- SQL - latte, and CTDB - Numeric Methods corrupt memory on unexpected inputs - r314682
- SQL - crash when query exceeds 250 table references - r316266
- SQL - query with subquery crashes the server - r318300
- SQL - JSON DB compatibility - alter table drop column causes table structure misalignment between SQL and CTDB - r319435



- SQL - JSON DB compatibility - sqlcolumns stored procedure returns wrong ORDINAL\_POSITION - r320867
- SQL - recursive query with parameter crashes the server - r321041

## END OF SQL FIXES

## 22.4 Utility Fixes

### Important UTILITY Fixes

- Dynamic Dump - Fatal error during dynamic dump - r327865
- Dynamic dump skips superfile host header write - r338138
- Dynamic Dump hangs - r274312
- Dynamic dump missing data or keys - r278231
- Dynamic dump missing data or keys - nontran files - r278398
- Dynamic dump skips superfile host header write - r338138
- RBLIFIL crash and heap corruption if incorrect IFIL->dreclen provided. - r261740

### Additional UTILITY fixes

- Dynamic dump restore fails with error 36 when writing CLSTRAN entry to transaction log - r330717
- Dynamic Dump: Only write messages to system monitor queue if a client has used SystemMonitor() to enable system monitoring - r330718
- Server crash creating backup using directory recursion option - r331632
- Dynamic dump with large number of files might continue running when server shuts down - r299503
- Dynamic dump and prime cache at startup with path and wildcard do not find the specified file - r303731
- Dynamic dump hang if dump script not found - r278181
- Prevent ctvfyl from crashing in standalone mode when opening files in exclusive mode; and QA test program improvements - r281918
- Check for unexpected key marks in index leaf nodes during ctVerifyFile() causes crash on variable-length data file - r301152
- Replace ctutilset() and util\_pre\_scan() functions with thread safe functions ctutilset\_xtd() and util\_pre\_scan\_xtd()
- ctfldmp and ctldmp fail with 1166 (ENCL\_ERR) on non-encrypted logs - r254510
- Dynamic Dump Fix: Using !REPLICATE option in dynamic dump script can cause backed up file to fail to open with error 14 or 413 - r263353
- Dump restore during deploy hangs on Unix systems when restoring a large number of files - r292259
- Dynamic dump fails with error 775 - r250234
- Dynamic dump restore errors 499, 12, or files missing after restore - r294079



- Server wait on external ctrdmp or disk full action process may hang on Windows - r296708
- Dynamic dump with large number of files might continue running when server shuts down - r299503
- Dynamic dump and prime cache at startup with path and wildcard do not find the specified file - r303731
- Dynamic dump hang if dump script not found - r278181
- ctVerifyFile(): Check for unexpected key marks in index leaf nodes during ctVerifyFile() causes crash on non-transaction-controlled file - r259693
- dfkctl utility: setposition option does not resume deferred index threads if error occurs after pausing threads - r264058
- ct\_tpc isam c crashes on Windows when exiting - r267525
- ctrdmp - dump restore errors using !CLNIDXX - r271021
- cttrap utility support for trapcomm log with both 2-byte and 4-byte file number connections - r279723
- ctscmp fails with 130 or 12 on superfile members - r294737
- ctscmp - 32 bit ctscmp fails with error 10 or 83 - r299799
- Prevent ctvfyl from crashing in standalone mode when opening files in exclusive mode; and QA test program improvements - r281918
- Check for unexpected key marks in index leaf nodes during ctVerifyFile() causes crash on variable-length data file - r301152

## END OF UTILITY FIXES

## 22.5 Other Fixes

### Communications

- TCP/IP - Incorrect client behavior in heterogeneous network environment - r265878
- Shared Memory: Possible read past end of descriptor array in server Unix shared memory protocol code caused by recent code changes - r271214
- Shared memory connection fails or crashes in LOCLIB model - r287923
- Unix SQL shared memory protocol does not detach from shared memory region when client disconnects from server - r287927
- Unix client connecting using localhost connects with TCP/IP instead of shared memory - r313179

### Drivers

- ADO.NET updates
- ADO Installers: 32 and 64 bit installer share the same registry area
- Entity Framework: Exception when trying to insert - r310327
- JDBC driver error -20008 connecting to SQL server - r304058

### Security

- SSL configuration ignored when SSL configured in ctsrvr.set - r268214



- `tls_cipher_suites` in `cthttpd.json` has no effect for `https`, `websocket`, or `mqtt` - r300249
- `sa_admin` may silently set the password to empty - r309897
- Server crashes when using LDAP authentication and an LDAP user logs on using the server DLL interface - r322761
- connection fails with `BTIP_ERR` (425) when LDAP authentication enabled - r251355
- LDAP for Solaris Effort - `SIGBUS` using `isql` with `ldap` authentication on `solaris` - r256839
- LDAP: Memory leaks with LDAP connections or file open passwords - r260133

## Misc

- Crash at startup when calling `GetVolumeInformationByHandleW` - r253265
- `AUTO_MKDIR` creation option not working on `unix` - r254821
- Crash or incorrect results when executing Java stored procedure that uses input variables - r273065

## Plug-Ins

- Plug-in Core Mechanism
- `PLUGINS` - `CTDatabasePooled` destructor causes crash
- `dbNotify` - support metadata
- `dbnotify` - crash with a wrong configuration at `"includeMetadata"`
- `dbnotify`: crash if `publishMqttMessages` is not an array
- `dbnotify` - Test `'topic'` variable before creating `pk` field list - r330674

## WebApps

- `CTHTTPD` mongoose leaking memory in `per_connection_thread_function` - r330778
- `DataExplorer`: `LVB` Types Not Saved in new `CRUD` Modal - r327641
- `DataExplorer`: Save (update) in Record Editor Not Working - r331641
- `DataExplorer`: Record Editor Fails to Render - r339718
- `MQTTEditor`: Clicking on Subscriptions shows the topic data grid - r333060
- `MQTTEditor`: Bug in Publish Persistence Topic When No Columns Defined - r257594
- `MQTTEditor`: Dashlane Extension Interfere With Proper Function of App - r267647
- `SQLEditor`: Create View Dialog Has Wrong List of Columns in `WHERE` Dropdown - r261997
- `SQLEditor`: Updating Record with `RowId 0` Results in Error - r267531
- `WebApps` - Common - (MM!) - `BinaryEditor`: Wrong error message when file is too large.
- `WebApps` - Common - Fixed problem where new accounts wouldn't save unless default "password" was changed.
- "Hashed" object memory leak
- Memory leak on `FCHashTable<T>::Add()` and `FCHashIntTable<T>::Add()` when key already exists
- `createSession` not reusing existing connection pool - user name case insensitive problem
- Connection pool memory leaks
- `DataExplorer`: Can't Create New Record if Only Values are Binary or



- Data Explorer: EDIT RECORD side panel - for floating point number , remove the two-digit maximum number of fractional digits
- DataExplorer: Some picker tables dont appear, and some do...
- DataExplorer - Fixed date save issue.
- DataExplorer: Data Explorer shows source\_payload blank
- DataExplorer: Rename view: refresh button mess tree and can't rename view with "minus" in name
- DataExplorer: SQL Scripts - "Run All Button" doesn't show results
- MQExplorer: Fix Inbound Message Widget on Dashboard
- MQExplorer - make it correctly handle upper/lower case in MQTT topics
- MQExplorer topics broken
- SQLEditor: Fix to prevent crashes while exporting view schema

### Installers

- Installers: Made sure the MSVC redistributable is installed when required
- Installers: ADO Installers: Redistributable fixes reflected in FairComConfig

### END OF OTHER FIXES

## 22.6 Replication

### Important Replication Fixes

- Replication Manager: - Folder Type Publication Should Allow Include/Exclude Designation - r254351
- ctMemphisGetFiles missing objects in recursive mode with a wildcard mask - r290564
- Prevent Replication Manager API to update DBEngine for Publication, Subscription, and Replication Plan - r292998
- Editing a subscribed publication in replication manager changes dbEngine to target dbEngine - r293001
- ReplicationManager: Selected Files Can't Be Deselected in 'New' Mode - r262607
- ReplicationManager: Editing a publication, from the subscription pane, breaks the UI - r270965
- ReplicationManager: Empty publications may be created - r272393
- ReplicationManager: Selected Files Can't Be Deselected in 'New' Mode - r262607
- ReplicationManager: Editing a publication, from the subscription pane, breaks the UI - r270965
- ReplicationManager: Empty publications may be created - r272393
- 2-way replication deployment - missing files - r249021



## Additional Replication fixes

### High Availability

- Use of uninitialized memory when updating server role during failover
- Secondary server crashes when setting availability group sync state during failover
- Adjustments to failover logic to work with replication agent ini file name change
- After failover, replication agent fails to start with error 96
- HA Failover Effort - Replication failover bug fixes
- Crash when freeing memory after call to `_sqlGetRedirections()`
- Failover bug fixes
- Server reports ctagent plugin load failed when using memphis failover
- Server crashes when unloading plugins due to failed call to wait for thread to terminate
- High Availability - Add read only server check to client library
- Bug fixes for changes to load memory grid earlier in server startup

### File Groups

- Support replication file groups - Replication file group deploy fails with error 96 on log 0 offset 0
- Replication agent manager action thread clears a mutex while it's in use when deploying a file group
- Thread handle leak on Windows when using replication agent manager
- Fix errors introduced with Replication Manager file group support
- Replication file group deploy hangs

### Replication Manager

- Server crashes during `FCParamHash::Add()` call
- Replication file system scan does not find subdirectories on Unix systems
- Dropping replication plan fails with error 540 when shared log ship support is turned off
- Clean up memory leaks in replication manager when errors occur
- Server crash in `_CheckDBEngineThrdFunction()`
- Replication manager shows replication plan as inactive when the replication agent is paused
- Bidirectional replication failing to deploy with error 13
- Replication Manager constraint violation on parallel replication archive

### Replication Engine

- Ensure that a client connection created in the master provider connection pool is independent of existing connections
- Setting replication file filter fails with error 749 when using synchronous replication if shared log ship thread compile time option is enabled



- Replication agent error 446 when savepoint restore undoes a file create
- Implement function to initialize replication file handle; fix crash and clear file number in `ctReplCloseFile()`; other replication bug fixes
- Replication of updates to a table that has a change id field fail with replication conflict error 1105
- Replication agent first time startup fails with error 96 opening transaction log 1
- Replication agent does not replicate file even though it matches a literal file name filter specification
- Replication agent log read at log 0 offset 0 and agent shutdown cleanup
- Fix bugs found during code review of replication uncommitted transaction position changes
- Replication agent fails to set log position with error 36, and resync errors 408 and 48 on superfile host
- Replication agent log read starts at oldest existing log instead of source server's current position if log ship has no saved log copy position
- Replication of create index call fails with error 46, and error 707 when replicating
- `REPLICATE_ALTER_TABLE YES` configuration option does not work due to bug in `c-treeDB` function that checks if file is replicated
- Use of uninitialized critical section in replication agent
- Replication agent sync shutdown hangs
- Replication agent sync shutdown hangs; and free mutex when clearing replication agent state
- Replication log read fails with error 2
- `ctAgent` crash when local server user doesn't match
- `ctAgent` failing to update local server user/password
- Replication: Server's `lstsuc` state variable is past last `SUCTRAN` entry in log
- Race condition on setting replication agent shutdown state variable when using replication agent
- Replication log read error caused by recently added forward scan phase
- Errors replicating operations while a file group is being deployed
- Replicated record delete fails with error 101 due to multiple operations on record in transaction
- Replication agent fails to set file filter with error 1115
- Error 733 and 12 replicating operations on partitioned files
- Failed deploy of replication plan might not set error message
- Replication plan deploy fails with error 1 on Unix system if `ctrdmp` utility is not in the server's working directory

## Replication Callbacks

- Replication Multi Callbacks - Replication agent displays negative replication operation counts
- Replication Multi Callbacks - failing to start `TranRepl` - 338511



## Reputil

- reputil crashes when attempting to resync a publication that does not exist

## Others

- ReplicationManager: drag & drop not working on new replication plans
- Server recovery as primary fails with TBAD\_ERR(76)
- Connection to server using cluster options returns PBAD\_ERR(749) if availability group is not deployed
- Connecting to server with cluster options fails with error 9022 if server is still initializing memphis agent
- Crash in GetClusterInformation()
- c-tree file relative path detection
- Replication Manager - Empty publication fails on editing
- Replication Manager buffer overflow
- ReplManager - Availability Group crashes Memphis
- ReplManager - Memphis database creation failed
- Memphis crash on checking DBEngine status
- Deadlock on server shutdown during plugin initialization
- Plugins hanging during server shutdown
- ReplManager - Error 9089 while deploying a bidirectional replication plan
- Memphis: Buffer overrun
- Memphis memory leaks on connection
- Memphis Replication is not correctly rebuilding deployed files
- Replication Deploy Effort - replication manager - crash when deploying a plan having "Rebuild on Deploy OFF"
- Error 87 deploying replication plan on Windows
- Detect deleted files that are still part of existing publications
- Read past end of buffer in \_ctCheckDBEnginePath()
- Replication Manager crash during shutdown with active Replication
- Availability group name pointer used after freed
- Replication Manager crash due to timeout
- Error during Replication by Database deployment
- replPlanMgr object getting out-of-sync in the action executors
- Automatically restart replication plans even if it is not connected to Memphis
- ctAgent crash if executed for first time without Memphis
- Server crashes when starting embedded replication agent on operating system that has case sensitive file names
- Automatically restart replication plans even if it is not connected to Memphis
- Server crashes at startup in ctPlugin\_start() if embedded replication agent can't connect to memphis
- Server crash in vsnprintf() called from RCESLogger::traceFormat()



- Deploy of bidirectional replication plan might fail with "Operation has been cancelled" error
- Server reports "too many open files" error due to file descriptor leak caused by memphis agent failing to close license file
- Use maximum read timeout of 5 seconds for memphis replication agent
- Deploy fails on Unix with error 1 in \_isamDumpFiles()
- Target server crashes when using multiple replication plans
- Memphis error "Not able to find connection pool for the given token"
- Replication agent bug fixes: Log ship thread may take long time to return from log read call, and missing mutex control on log read position
- Replication file system scan is slow due to checking all attributes of configuration in order to use an available connection from the pool
- Refreshing replication plan status can cause an action's status to be reported as cancelled
- Stack overflow when deploying a large number of replication plans
- Server crash in StopReplAgents() when stopping a memphis replication plan
- Replication deploy times out if file system scan takes more than 60 seconds
- Server hangs when shutting down during unloading of ctagent plugin
- Speed up starting of replication plan
- Replication plan action hangs due to deadlock on hash list mutex
- When deploying a replication plan, Replication Manager doesn't prompt to overwrite files when files exist on the target server
- Replication Enhancement: Target server crashes in RCESCheckReplConnected
- Replication: Improve speed of shutting down server when multiple memphis replication agents are running
- Replication Enhancement: Replication agent crashes when reading function statistics
- Possible race condition in update of memphis dbengine manager hash list
- When a replication plan is stopped, a connection to the memphis server remains active and memory is leaked in the secondary server
- Missing one or more characters at start of working directory in Memphis agent table
- Remove replication agent resources when deleting a memphis replication plan
- Memory leak in the memphis server's status check thread
- Memphis - Clean up memphis memory leaks
- Additional replication management memory leak cleanup
- ReplicationManager: drag & drop not working on new replication plans
- ReplicationManager: Editing an Existing Publication Lacks Some Pre-population of Fields
- ReplicationManager: Publication by File - selection graph issues
- ReplicationManager: File Directory Refresh Button Broken
- ReplicationManager: cannot select directory when creating Publication "By Folder"
- ReplicationManager: Memphis resync not showing files
- ReplicationManager: Displays Source server folder attempting to change Target server folder
- ReplicationManager: problem after Dropping availability group
- ReplicationManager: Error in finishWizard : onSaveCloseClick



- ReplicationManager: logs show wrong timestamp
- ReplicationManager: Manage Archived Plans not loaded
- ReplicationManager: UI not displaying errors from ctMemphisPersistReplPlan
- ReplicationManager - does not finish logging in
- Enabling replication for superfile host and members
- Replicate the changing of a file's serial number
- Enabling replication on open of superfile host and members
- Dynamic Dump Fix: Using !REPLICATE option in dynamic dump script can cause backed up file to fail to open with error 14 or 413
- Dynamic dump fails with error 775
- Replication log read error 749 caused by omission in support for replication of serial number change
- Replication agent file resync may fail without logging an error message to ctrepagent.log
- Replication resync file copy error 12
- repadm utility outputs garbage error message when replication resync is called on source server
- Embedded replication agent may crash the server if an error occurs when starting the agent
- Replication: Log read fails with error 2 when adding entry to node list after failover when using embedded parallel replication agent
- Shutdown of c-tree Server with embedded agent on Unix system hangs if triggered by signal
- Bidirectional replication with parallel embedded replication agent causes continuous transaction log writes on source and target servers
- Replication agent sync shutdown hangs if agent is paused
- Replication agent log ship thread repeatedly tries to connect to source server despite shutdown request
- Replication: Log read fails with error 2 when adding entry to node list after failover when using embedded parallel replication agent
- PAGE\_SIZE error during replication plan deployment
- Replication: Setting replication log read position by transaction id may return wrong log number
- Replication Publication by Folder - error 100
- Replication Effort - Socket close takes long time due to use of SO\_LINGER socket option
- Replication Push: Embedded parallel replication agent shutdown takes a long time
- Replication Push: Parallel replication agent exclusive file open errors
- Parallel embedded replication agent with many apply threads is slow to shut down when using shared memory on Windows
- Replication log scan reads partial record update if update is undone by savepoint restore followed by a special savepoint
- Replication of record in FLEXREC file may fail with error 1105
- repadm resetfuncstats option does not clear function stats for all replication agent threads
- Partitioned file replication bug fixes



- synchronous replication fix: A server that does not use CHECK\_CLUSTER\_ROLE configuration option does not operate in synchronous replication mode even when using a synchronous replication file filter
- Replication Testing: Default database creation at startup on cluster secondary may cause replication conflicts
- Replication agent does not apply redirect rules to index names for create file and index create operations, and to data and index names for rename file operation
- Replication Testing - Sync commit spin counter bug could disable spin performance optimization -
- Replication testing: Log read error 96 when starting parallel replication agent due to log ship purging required copy of transaction log
- Replication Testing: Parallel replication agent incorrect tracking of transaction commit state can cause transactions to be reapplied or skipped when agent restarts
- Replication Tests: Implement proper recovery of sync commit transactions during forward roll
- Replication Tests: - repadm -c getstate error 153 due to raSTAT structure change
- Replication Tests: Sync commit notification may delay transaction commit longer than necessary
- Replication tests - Replication resync using repadm utility fails with error 40
- Replication testing - Replication resync failed to resync replication files with error 101
- Replication test - Server startup may fail with error PBAD\_ERR if using multiple synchronous replication agents or transaction commit metadata size is large
- Replication test - Server crash in ctSyncReplAddAgentDependencyToTransaction() when more than one synchronous replication agent is registered with the server -
- Replication testing - Crash during transaction recovery
- Replication agent plugin may hang when starting up
- Replication resync does not properly resync files that are being updated during resync
- Replication resync fails with error 451 when connecting to target server -
- ctReplGetNextChange() error 2 caused by max response time limit -
- Replication tests - Server crash when two parallel embedded replication agents are using the same replication unique id
- Embedded replication agent crashes replicating ctCopyFile
- Replication agent run as Windows service crashes or fails to start with error 454
- repadm getstats command shows unexpectedly small function counts when using parallel replication agent
- Possible NULL pointer dereference in ctReplChangeRecord() when replicating low level file delete
- Parallel replication agent crash in ctRActapicb after losing connection to source server
- Server quiesce that waits for replication agent may time out
- Double free in ctReplGetNextChange
- Resync of replication plan fails with error 1108
- Memphis server takes a long time to shut down after embedded agent error occurs
- Server hangs after failed replication resync



- Server terminates due to replication agent plugin reader/writer lock initialization failing with error EBUSY on Mac OS X
- Server crashes when replication manager configuration file is missing
- Embedded replication agent reads memory from stack after going out of scope when memphis database creation fails
- Memphis server agent initialization error on Mac OS X: "Not able to retrieve OpSystem by IP address"
- Replication resync error "Get resync status return (0) operations, should return only one" on Windows
- Target server crashes in strlen() called from RCESReplPlanMgr::Resync() during file resync on Unix systems
- Slow memphis server shutdown due to plugins trying to connect to server during shutdown
- Memphis replication copy of files to secondary server sets default index extension on IFIL of copied data file
- Refreshing list of files in publication on Windows machine fails with error 4041
- Log reader does not close transaction log when log open fails with error 666
- Replication agent crashes when replicating multiple changes to a record followed by delete of the data file in a transaction
- Replicate of record update in same transaction that added the record may fail with error 101
- Replication Deploy Effort - Avoid unnecessary PUTIFIL call during replication deploy
- Replication Deploy Effort - Rebuild of index during deploy always uses default data file extension
- Replication agent sync shutdown hangs
- Replication agent logon error 470
- Log ship error 96 reading source server transaction logs when deploying replication plan after deleting and recreating the plan
- Updates to records missing from target file after deploy
- Parallel replication agent log reader startup fails with error 96 when the replication configuration specifies a starting log position
- Restart of original primary server after memphis failover uses wrong log position on new primary server
- Bug fix for replication agent shutdown hang did not set sysiocod value read by replication agent
- Bug fix for replication agent shutdown hang did not set sysiocod value read by replication agent
- Replication deploy does not detect failure of dump restore
- Low level read/write open of replicated data file was unintentionally allowed
- Remove open corrupt open mode from memphis lowlevel file open calls
- Server hangs or crashes at shutdown due to server side connections active during shutdown
- Replication of truncate file fails with error 46
- Use of uninitialized memory for stack based allocation of CTMemphisConnPuller object
- Improve parallel replication performance when exclusive access is required to a file



- Asynchronous replication agent skips sync commit transactions
- Transaction log activity on source server can cause replication agent startup error 96 when deploying a replication plan
- Set and clear log requirements during failover
- Replication agent shutdown fails with error 101 if shutdown is already in progress
- Replication agent apply thread reapplies committed transaction after losing connection to target server
- Replicate Superfiles Effort: Errors replicating superfiles, including possible replication agent crash or failure to open superfile host
- Replication errors after deploying a bidirectional replication plan
- Parallel replication agent leaks memory or crashes in ctreplophsh() when destroying dependency graph at shutdown
- Replication agent might crash if external client is querying its state while it is shutting down
- ctrepd and ctmtqa utilities fail to connect with error 540
- Two replication agents running in the same server interfere with each other
- Log ship thread fails to open newly-created source server transaction log with error 96
- Server startup error when using CHECK\_CLUSTER\_ROLE MEMPHIS and services.json instead of PLUGIN ctagent in ctsrvr.cfg
- Parallel replication agent memory use increases over time
- Parallel replication agent keeps excessive number of copied transaction logs
- Parallel replication agent might fail to open transaction log with error 96 after stopping during transaction activity
- Replication error 96 when starting up caused by recent changes to update minimum log requirement
- Log ship thread not purging some transaction logs after losing connection to source server
- Replication agent pause command hangs when using parallel replication with one apply thread
- Low level file copy fails with error 775 on replicated data file
- Replication resync hangs when parallel replication agent uses one apply thread
- repadm utility's resync option fails with error 36
- Replication Enhancement: ctReplGetNextChange() is slow when multiple replication agents are running in the same server
- Replication Enhancement: Improve performance of replication log read when many replication agents are running
- Replication agent memory use increases over time when using parallel replication with one apply thread
- Replication agent shutdown hangs when shutting down secondary server
- Server crashes when deploying availability group
- Replication statistics are not updated after memphis server restarts
- Speed up replication plan shutdown by using normal shutdown instead of sync shutdown
- Replication Enhancement: Replication agent that uses the shared log ship thread model might crash in RtlLeaveCriticalSection() when shutting down



- Replication Enhancement: Shared log ship thread might not stop when all log readers disconnect from it, and might not restart when new log reader connects
- Replication Deploy Effort - File block optimizations can cause server to crash in ctSETfblk() called from filblk\_clsfil(); and fix possible file block hanging
- Server hangs waiting for plugin initialization to complete if cthttpd initialization fails
- Server shutdown hangs when unloading cthttpd plugin if cthttpd plugin initialization failed
- Speed up shutdown of the cthttpd and ctAgent plugins at server shutdown
- Server takes long time to wait for plugins to unload at shutdown
- Replication agent does not detect loss of connection to target server
- Secondary server crashes when replication agent initialization fails
- repadm -c getstate may hang
- Server crashes in memphis replication agent at server startup
- Bug fixes for replication of data definition operations
- Embedded replication agent log ship thread may hang when it loses connection to source server
- ReplicationManager: Editing an Existing Publication Lacks Some Pre-population of Fields
- ReplicationManager: Publication by File - selection graph issues
- ReplicationManager: File Directory Refresh Button Broken
- ReplicationManager: cannot select directory when creating Publication "By Folder"
- ReplicationManager: Memphis resync not showing files
- ReplicationManager: Displays Source server folder attempting to change Target server folder
- ReplicationManager: problem after Dropping availability group
- ReplicationManager: Error in finishWizard : onSaveCloseClick
- ReplicationManager: logs show wrong timestamp
- ReplicationManager: Manage Archived Plans not loaded
- ReplicationManager: UI not displaying errors from ctMemphisPersistReplPlan
- ReplicationManager - does not finish logging in

## END OF Replication FIXES

## 23. Index

<b>!</b>		
!REPLICATE option in dynamic dump script can cause backed up file to fail to open with error 14 or 413.....	116	
<b>.</b>		
.NET.....	58	
.NET - Added CTTTable.HasLocks().....	60	
.NET Core.....	58	
.NET Core - Microsoft's free, open-source, general-purpose development platform.....	58	
.NET Entity Framework.....	62	
.NET Gui Tools.....	62	
.NET ISAM Explorer		
Write Out ISAM Record Structures Containing SQL NUMBER Types .....	64	
.NET LogAnalyzer		
Added support for new CTSTATUS.FCS single line format.....	64	
.NET Security Admin		
Fixed on screen behavior of the servers' combobox .....	64	
.NET SQL Explorer		
Forbid editing of autotimestamp fields .....	63	
Trigger information not shown .....	63	
Unable to show Date.MinValue .....	63	
.NET Stored Procedure - More information .....	34, 60	
.NET Stored Procedures .....	60	
.NET Stored Procedures - Implement access to LVAR* fields.....	33, 60	
.NET/Java/Web Tools		
Added support for USERINFOX with user memory over 2GB.....	63	
<b>3</b>		
32K Page Size now Always Set as Default ..	135, 167	
<b>A</b>		
Access c-treeDB Java API with Embedded Server Models.....	51	
Add replication agent configuration option to write error 1105 occurrences to ctreplagent.log and apply the transaction anyway.....	202	
Added Diagnostic Logging for INIX_ERR.....	131	
Added new batch mode.....	75	
Additional .NET Drivers .....	59	
Additional Replication Extension Event Callbacks .....	196	
ADO.NET with EF6 Written Tables now Return Correct Identity Values .....	62	
Alter Table Compression - modes to copy compression from existing open file and to reset to the default .....	87	
Android Port - Code Compatible.....	216	
Apple macOS Support		
M1/M2/M3 - Code Signing - Monterey; Ventura.....	215	
Apple Packages now signed - 'Contents' directory introduced .....	140	
Auto timestamp support.....	26	
Automatic Purge and Archive of Replication Logs .....	190	
Automatic Temporary Directory Cleanup After Crash.....	103	
Automatically check for a write lock when updating a data file during synchronous replication.....	204	
Avoid OPEN I-O Exclusive Open Errors.....	171	
<b>B</b>		
Background Transaction Data and Index Flush Threads no Longer Default to Immediate ..	134, 170	
BINARY CAST of Padding Spaces Retained in VARCHAR Fields.....	134, 168	
Block Logon when Quiesce is Abandoned ..	121, 169	
Bug Fixes .....	225	
<b>C</b>		
c.isam - C Language ISAM.....	40	
c.lowlevel - C Language Low-Level .....	40	
c.mqtt- C Language MQTT Client.....	40	
c.nav - C Language Record Navigation (CTDB) ..	41	
c.sql.direct - C Language Direct SQL .....	41	
C++ .....	54	
C++ CTPP - added overloads for CTSting Compare and CString .....	56	
CompareIC.....	56	
C++/CTPP - CTSession		
FindDatabase with char* parameters .....	55	
C++/CTPP		
Added CTTTable		
HasLocks().....	57	
C++/CTPP - Added SetBinaryFlag and GetBinaryFlag to the CTField class .....	54	
C++/CTPP - CTTTable Open method to open tables using the table UID .....	78	
Change FPUTFGET model's ctREADFIL and ctSHARED open to a shared read-only open ..	107	
Change replication agent exception logging default option to log errors only .....	202	
Change SQL Database Path Utility .....	66	
Changed Character Validation in ODBC Connection String .....	53	
Client failOver/Broadcast notifications .....	204	
Client FIPS Configuration for TLS Connections	149, 156	
Client-side Support Added for FIPS.....	150	



Communication Protocols.....	182	CTDB - New functions to retrieve all dictionary information for tables .....	74
Compact Error RCPT_ERR (1156) Fixed in Multi-user Standalone Models .....	108	CTDB - new functions to retrieve primary key candidate index .....	84
Compatibility .....	134	CTDB - Performance - avoid useless SWTCTREE call during ctdbGetFieldAs() and ctdbSetFieldAs() .....	84
Configurable Shared Memory Connection Timeout .....	185	CTDB - Set scanner cache feature on table .....	80
Configuration .....	162	CTDB - SQL import logic improved to take primary key index information into account .....	81
Configuration keyword CTSTATUS_MASK WARNING_PAGESIZE disables mismatch warning .....	129	CTDB Filter support for BTRIEVE date and time data types .....	70
Configuration keyword disables mismatch warning .....	129	CTDB Fixes.....	225
Configuration option MAX_REBUILD_QUEUE is runtime configurable .....	178	CTDB Function to Retrieve Number of Filtered Records.....	83
Configure DIAGNOSTICS FULL_DUMP at Runtime.....	175	CTDB/CTREE/RTG - support for BTRV BLOB and CLOB .....	74
Copy Constructor added to CTEException Class56, 69		ctdbGetFieldAs* and ctdbSetFieldAs* from RTG column with dbtype set .....	69
Correct Dump Restore Redirect Log Messages to Include Filename .....	113	ctdbGetNAVFieldProperties maps unsigned integer c-tree field to signed larger numeric .....	76
Correct Entity Framework ADO.NET Parameter Handling Using INSERT Statements.....	62	ctdbNumberOfEntries() Added to CTDB .....	70
Corrected ODBC Error -5 'wrong number of parameters' When Creating Stored Procedures.....	53	ctdbSetDefaultValueAsBinary Added to CTDB .....	71
cpp.nav - C++ Language Record Navigation (CTPP).....	41	ctfchk Utility added to packages .....	124
Create ISAM Compatible CT_MONEY Column Type from FairCom DB SQL.....	30	ctfdmp and ctldmp no longer fail with 1166 (ENCL_ERR) on non-encrypted logs.....	111
Create system log files as huge files to remove 4GB file size limit .....	20, 172	ctinfo display distinct key counts for indexes.....	118
csharp.nav - C# Language Record Navigation .....	42	display record update callback information.....	117
csharp.sql.ado.net - C# Language for ADO.NET ...	42	File encryption and file name information .....	117
csharp.sql.storedprocs - C# Language Stored Procedures.....	42	Retrieve Assigned OS System File IDs ...	101, 116
ctAgent - forcing IP address option ignored .....	203	ctinfo improved Show DODA type names .....	117
ctcompare utility - Support running with a server that has a different path separator.....	126	ctldmp utility now decodes DIFIMAGE entries into old and new image values .....	126
ctCreateAuthFile() Creates Encrypted Auth Files	121	ctldmp/ctlchg Support 6-byte transaction numbers.....	126
CTDB .....	65	ctpartadmin utility now supports partitioning an existing data file .....	125
CTDB - ability to set a index as primary key index .....	80	ctpath - Change Internal (SQL) Database Paths66, 122	
CTDB - added support for changelD field .....	79	CTPP - CTEException GetErrMsg() may return a pointer to released memory .....	55
CTDB - Alter Table - ability to create/update conditional indices with table open shared .....	79	GetErrMsg() may return a pointer to released memory .....	68
CTDB - alter table from attribute - alter table support for RTG .....	70	ctrldif - Toggle Index Null Key Support with Rebuild Utility .....	123
CTDB - alternative fixed string padding.....	81	ctREADFIL may be ignored by FPUTFGET .....	109
CTDB - ctdbClearRecord() performance improvement.....	86	CTREE .....	87
CTDB - ctsqlimp utility and .....	77	CTREE - Implement change id field to support optimistic locking.....	92
CTDB - improved batch processing.....	73	CTREE - New function to generate a sequence name that does not conflict with existing sequences.....	96
CTDB - new function ctdbBlobReserveSize .....	83	CTREE Fixes .....	227
CTDB - New function ctdbDeleteBackgroundLoadStatus .....	78		
CTDB - New function ctdbInsNAVField() .....	71		



CTREE/CTDB - support for unixtime stamp data type (time_t) .....	97
CTREE/CTDB/RTG - support for CT_BT_BIT field type.....	73
c-treeDB API Functions for User-Defined Hot Alter Table Field Conversion Callback .....	77
c-treeDB Default Extent Sizes Supported Beyond 64K .....	65
ctrepd utility	
Add option to set node name .....	194
ctscmp - Corrected IAIX_ERR (608) Error While Compacting V6 Superfiles .....	122
ctSQLImportTable() function and ctsqlimp utility support for TLS .....	85
ctVerifyFile	
Additional Test for keys Referencing Deleted Records.....	118
Check for unexpected key marks in index leaf nodes crashed on variable-length data file ...	118
Improved Index Leaf Nodes Key Mark Checks	118
Internal Index Node Validation Improvements..	119
ctVerifyFile/ctvfidx - Add distinct key count verification.....	119

## D

dbNotify - database triggers to send MQTT messages.....	23
Default COMPATIBILITY BATCH_SIGNAL Behavior.....	136, 174
Default COMPATIBILITY FILE_DESCRIPTOR_LIMIT Configuration.....	137
Diagnostic logging for commit delay livelock .....	132
Diagnostic Message for Transaction Log Deletion .....	130
Diagnostics .....	127
Diffie-Hellman parameters can now be read from server certificate file if present .....	153
Disable Diagnostic pstack Generation From Shared Memory Errors .....	170
Display Replication Latency from repadm Utility ..	195
Docker	
Scripts to create Docker image.....	220
Docker Support.....	220
Docker uses NAT - New property .....	177, 203
Dr. ctree	
Header Values incorrectly shown .....	62
Dr. ctree does not show large DODA .....	62
Dr.Ctrees	
Fix for header descriptions .....	63
Drivers .....	39
Drivers Overview .....	39
DSQL - Direct SQL .....	36
DSQL - New function	
ctsqlGetCommandFromCursor.....	36
DSQL - New function	
ctsqlGetParameterDirection.....	37

DSQL - New functions	
ctsqlGetConnectionFromCommand() and ctsqlGetParameterAddress() .....	38
DSQL - New functions to scroll the resultset using a scrollable cursor .....	36
Dynamic Dump - Online Backups.....	110
Dynamic dump !REPLICATE .....	115
Dynamic dump !REPLICATE option now applies to superfile host and its members.....	116
Dynamic dump error message improvement.....	111
Dynamic Dump Hangs .....	111
Dynamic dump missing data or keys for non-ctTRNLOG Files .....	113

## E

Enable Default Data Cache Empty Lists .....	8, 137
Encrypted Password File Support for Replication Diagnostic Utility ctrepd .....	194
Enhancements to Prime Cache at Server Startup.....	6
Entity Framework Table Addition Speed Optimized .....	28, 62
Ephemeral Elliptic Curve Diffie-Hellman Cipher (ECDHE) now Supported for TLS Connections .....	152, 179
Expanded Windows Broadcast Machine Name Length .....	135, 181, 183

## F

Failed Client Shared Memory Connection Diagnostics .....	130, 184
FairCom DB Batch Calls now Return Number of Records Rejected from Active Filter .....	82
FairCom DB Server FIPS Support Details.....	153
FairCom SQL Service - Simpler StandAlone SDK Packages .....	105
FairComConfig Utility now detects prior .NET configurations.....	169
File Block Enhancement	
Support a file block mode that allows active transactions time to complete before blocking the file .....	93
File Close Silently Failing.....	109
File Copying/Quiesce Updates .....	173
File Groups .....	205
File Names Supported up to 4K.....	14
File Open Automatically Retries on Pending Creation State .....	170
FindTarget Method Added to c-treeDB for Java ....	51
FIPS-compliant OpenSSL 3.0 libraries for Windows and Linux.....	150
Fixed index file error 14 in FPUTFGET after index update in exclusive writethru mode.....	108

## G

Getter Setter Methods for Blobs as base64 Strings .....	67
---------------------------------------------------------	----

**H**

High Availability .....	213
Highlights .....	1
Hot Backup Without Quiet Checkpoint	
Requirement .....	110
HTTPD Service Extra Headers Configuration .....	160

**I**

Important - PAGE_SIZE Conversions .....	162
Improve ctQuiet Timeout .....	104, 176
Improve performance of dump restore when	
deploying a publication with many files .....	197
Improved Change Batch Performance .....	82
Improved c-treeDB Truncate Table Functionality .....	54, 66
Improved Dynamic Dump Performance .....	110
Improved Hot Backup Quiet Checkpoint	
Requirement .....	112
Improved JDBC efficiency .....	51
Improved Replication Manager Plan Shutdown ...	198
Improved Shared Memory Connections When	
Permissions Discrepancy Between Server	
and Client .....	183
Improved SQL Shared Memory Connections	
When Permissions Discrepancy Between	
Server and Client .....	184
Improved System File id Diagnostic Logging .....	132
Initialize Replication File Handle Values with	
ctReplInitFileHandle() .....	199
Install in Program Files .....	224
Interface for Managing Plug-ins on-the-fly .....	24
ISAM .....	103

**J**

JAVA .....	50
Java JDK Version Considerations .....	221
JAVA JSON DB - Add new driver for Java	
Deveopers .....	50
Java Stored Procedures and Triggers Require	
JDK V1.7 or Newer .....	221
Java Version Support .....	221
java.jpa.nav - Java Language Persistence API	
(JPA) .....	43
java.json.db - Java Language over JSON DB .....	43
java.mqtt - Java Language MQTT Client .....	44
java.nav - Java Language Record Navigation	
(JTDB) .....	44
java.sql.hibernate - Java Language Hibernate	
Persistence .....	44
java.sql.storedprocs - Java Language SQL	
Stored Procedures .....	45
JDBC - allow URL without hostname .....	52
JDBC crash fixed .....	51
JDBC Installers now available .....	51
JDBC now Supports Setting Parameter to NULL .....	52
JDBC/JTDB .....	51
JTDB - Added CTable.HasLocks() .....	53

**L**

LDAP - Server crash on Linux using LDAP with	
SSL fixed .....	142
LDAP authentication failure no longer incorrectly	
indicates timeout error on Linux/Unix .....	141
LDAP Connection and File Open Passwords	
Memory Leaks Fixed .....	143
LDAP errors generate correct error messages at	
login .....	142
LDAP support for Solaris .....	142
LDAP_LOCAL_PREFIX option to filter LDAP	
authentication by username .....	142
Legacy platforms remain supported .....	218
Linux MIPS Port .....	218
Log message to CTSTATUS.FCS when	
transaction commit fails .....	22, 133
Log Only Failed Operations to Exception Log .....	191

**M**

Mask PAGE_SIZE Warnings in	
CTSTATUS.FCS .....	129, 135, 167
Miscellaneous .....	222

**N**

Named Pipe connection logic removed from	
Unix shared memory .....	183
Node Name Added to DLOK_ERR and	
Transaction Abort Messages .....	133
nodejs.json.db - JSON DB Client for Node.js .....	45
nodejs.mqtt - MQTT Client for Node.js .....	45
Non-recursive filename wildcard matching .....	187
NULL Key Support - Improve indexing of null	
values .....	100
NULL KEY VALUES Support - CTDB level	
metadata stored file's unified resource .....	72

**O**

Obtain Oldest Uncommitted Transaction Log	
Position from a Replication Agent .....	200
ODBC .....	53
Online Compact and Rebuild .....	8
On-line Compact for Background Table	
Defragmentation .....	9
Online Rebuild Support .....	10
OpenSSL Updated to 3.0 for Latest Connection	
Security .....	148
OPENSSL_ENCRYPTION keyword no longer	
controls internal cipher .....	148, 179
Other Fixes .....	238
Overcome 128K record size limit .....	27

**P**

Pacemaker .....	213
PAGE_SIZE Changes Automatically Applied to	
all FairCom Controlled Files .....	15
Partition	



support for non-schema key segment modes (REGSEG, INTSEG, etc.) .....	96
Partitioned file support in multi-user StandAlone operational model .....	106
Performance Enhanced During Batches that Filter out Fields .....	75
Performance Enhanced for Batch Variable Length Record Read.....	76
Performance enhancement Turn on log writethru for Single-User TRANPROC library .....	106
php.sql - SQL for PHP .....	46
php.sql.pdo - SQL for PHP PDO .....	46
Platform/Compiler Support .....	214
Prevent Dynamic Dump Restore Errors 499, 12, and Files Missing after Restore .....	114
Prevent Unix Dump Restore Hang During Large Deployment.....	113
Prevent Updates and Deletes for Tables .....	87
PTADMIN() option to purge all existing partitions below the new partition base value when increasing the partition base.....	124
PYTHON .....	58
Python APIs .....	58
python.json.db - Python Language over JSON DB .....	46
python.mqtt - MQTT Client for Python.....	46
python.nav - Python Language Record Navigation (CTDB).....	47
python.sql - Python Language SQL.....	47
python.sql.sqlalchemy - Python Langage for SQLAlchemy.....	47

**Q**

Quick Start - Steps to Enable FIPS-140-2 AES and SSL Encryption Modules .....	155
----------------------------------------------------------------------------------	-----

**R**

r288975 - DSQL - New function ctsqlGetConnectionFromCursor .....	37
Read Permissions - Server now checks for read permission on function calls that read data or key values .....	22, 100
Record buffer optimization - memset in record handling may have significant performance impact .....	86
Record Update Callback - Support record update callback that is called at the start of an add or update operation, allowing the callback to modify the record image .....	89
Recovery Configuration File for Secondary Synchronous Server Coordination.....	180
Recovery Script Configurations Ignored from Backup Scripts .....	112, 136
Recycle Bin Preserves Deleted Files and Protects Against Accidental Deletes.....	11

Redirect All Restored Files to a Specified Path ...	111
Reduce Number of Log Read Thread Connections to Target Server when Using Embedded Replication Agent .....	199
Reduce Number of Parallel Replication Apply Thread Target Server Connections .....	198
Remove JDBC setEscape Processing Flag Exception .....	53
repadm - Replication Administrator Utility Updates.....	192
repadm Utility - option to display key value and record image in exception log record.....	194
repadm utility can be used to reset the replication agent statistics.....	194
Replicate File Copy Operations .....	195
Replicate Restore Point Operations .....	190
Replication .....	186, 240
Replication Agent Log File now Created in Local Executing Server Directory .....	202
Replication agent sync shutdown .....	189
Replication callback called after replicating a CREIFIL operation .....	196
Replication Command Line Utility - ReplUtil .....	191
Replication Demo - Replication Using JSON RPC.....	211
Replication Demos.....	211
Replication deployment now copies indexes instead of.....	189
Replication File Group Fixes and Improvements.	210
Replication File Groups .....	205
Replication Manager files to be created under data directory .....	197
Replication manager logging improvements .....	202
Replication Tutorial - Create test program to demonstrate ctReplAgentOp() opcode usage .	211
Resize Data and Index Cache for Elasticity .....	3
Resource Read Errors (RRED_ERR, 407) Diagnostics .....	130
Retain Existing Permissions and Synonyms on SQL Import.....	124
Retrieve MRT Host Table Name via SQL Stored Procedure .....	27, 121
Retrieve the conditional expression function .....	76
Retry opening transaction log when updating deferred index log entry status .....	99
Return Replication Log Scan Last Log Entry Offset.....	181, 189
Return Server's Read-Only Mode with GETNAM() .....	103
Robust System Log Features Secures SQL Database Auditing.....	127
ROWID - CTDB/SQL - Expose 'rowid' as a field in SQL .....	30

**S**

Security .....	140
----------------	-----



Security Improvements for LDAP Authentication .	140
Segmented Sort Work Files for Rebuild .....	124
Sequence Number Entities now Unique	
Between Databases.....	29
Server .....	3
Server Configuration - Default FILES 4096. Was	
FILES 1024.....	175
Server internal processing files moved from	
working directory to LOCAL_DIRECTORY .....	21
Server logs diagnostic messages to help	
analyze slow or hanging plugin startup and	
plugin unload .....	132
Server Plug-Ins.....	23
Server Read-only Status Logged to	
CTSTATUS.FCS.....	21, 131
Server rebuild options may be set at runtime -	
SORT_MEMORY and MAX_HANDLES.....	178
Server Startup Error 90 Fixed.....	173
Server wait on external ctrdmp or disk full action	
process may hang on Windows.....	115
services.json replaces cthttpd.json Configuration	176
Set CHECKPOINT_INTERVAL to 12th of total	
log space in server configuration file .....	176
Set First Extent Size in FairCom DB API.....	65
Set Table Partition Base in FairCom DB API .....	67
Shared Memory .....	182
Shared Memory Key Configuration Through	
Environment Variables.....	169, 182
Shared Memory protocol connect function on	
Windows now uses the optional socket	
timeout value .....	182
Single-line Status Log Messages Improves	
External Integration.....	168
Source Code Security - Coverity Scan .....	140
Specialized Superfile Compact Options .....	126
SQL.....	26
SQL - Correct Numeric String Conversion .....	32, 134
SQL - support to map money type to c-tree's	
CT_MONEY .....	30
SQL Fixes .....	235
SQL Import Utility (ctsqlimp) - Option to keep	
existing permissions and synonym .....	28
SQL Performance Improvements .....	26
SQL Stored Procedures - Informational Methods	
Added to DhSqlException .....	32
SQL Windowing Functions for Analytics.....	32
sql.cli - SQL Language Interactive SQL	
Command-Line Interface (iSQL).....	48
sql.jdbc - Java Language Database Connectivity	
(JDBC) .....	48
sql.odbc - SQL Language Open Database	
Connectivity (ODBC) .....	48
SSL_CIPHERS .....	151
StandAlone .....	105
StandAlone batch calls with locks fail with	
LEOF_ERR(30) .....	108
StandAlone Full Source Code Package - Porting	
to alternative platforms made easier.....	105
Stricter Transaction Dependent Behavior for all	
Table Definition Operations .....	139
Support Added to hot Alter Table for	
non-Schema-Based key Segment Modes. 88, 123	
Support adding a condition to an index when	
creating the index with the data file open in	
shared mode .....	94
Support adding identity field and auto system	
time field to partition host that has existing	
members .....	88
Support changing a file's password or	
permission mask when the file is open in	
shared mode .....	95
Support FileSystem scan of non-ctree files .....	190
Support for .....	188
Support for Modifying file Partition Added to	
c-treeDB.....	67
Support for Multiple DISTINCT Keywords in a	
Single Query .....	27
Support more than one replication plan between	
two FairCom DB Servers .....	186
Support multiple folder rules per Publication .....	187
Support new types of null key value checks for	
c-tree indexes .....	88
Support Publication.....	186
Support Publication.....	187
Support Regular Expression (regex) rules in	
Publication .....	188
Support replication of superfile hosts and	
members .....	186
Support setting server configuration directory	
with environment variable .....	175
Suppress DIAGNOSTICS READ_ERR stack	
traces when no error is returned.....	127
SYSLOG - Efficiently Truncate .....	20, 172
SYSLOG improved reuse of disk space .....	20, 172
SYSLOG SQL_STATEMENT - Exclude users	
from logging by name .....	27
SYSLOG_EXCLUDE_SQL_USER.....	173
<b>T</b>	
thingworx.always-on - ThingWorx Connector .....	49
TLS/SSL Correctly Configured from Encrypted	
Configuration File .....	135
Transport Layer Security Secures Data in	
Transit between Network FairCom DB Clients	
and Servers.....	144
Troubleshooting and Testing .....	148, 157
Tuple Expression Enhancements .....	33
<b>U</b>	
Unexpected 64-bit ODBC Driver Overflow Error ...	53



Unix Shared Memory connection errors logged by server now include pipe or socket method in use .....	182
Update Replication Manager Publication Files and Folders When Server Moved to Alternate Environment.....	200
Update to ctsqlcdb Utility .....	123
Update to Dynamic Dump Error Message.....	116
Update to Dynamic Dump Keys .....	112
Updated Standalone Rebuild and Compact Utilities for Data Encryption .....	106
User-Defined Hot Alter Table Field Conversion Callback .....	90
Utilities .....	110
Utility Fixes .....	237

## V

V12/V11 Encryption Backward Compatibility .....	178
V6 to V7 SDK Mapping Macros Disabled.....	136
vb.nav - Visual Basic Language Record Navigation .....	49
Visual Studio 2022 Support .....	214
VSS Writer Volume Shadow Copy Service .....	224

## W

When changing security attributes for a partition host file, change the attributes for all existing partition members .....	99
Windows ARM Native Port .....	217
Windows Installer Error 1722 Fixed.....	224
Windows Installers and FairComConfig Improved .....	222

## X

X509 Certificate based authentication.....	158
--------------------------------------------	-----