FairCom White Paper

FairCom DB Constraints and Capacity Planning

Audience

Developers

Subject

Considerations when planning your FairCom DB

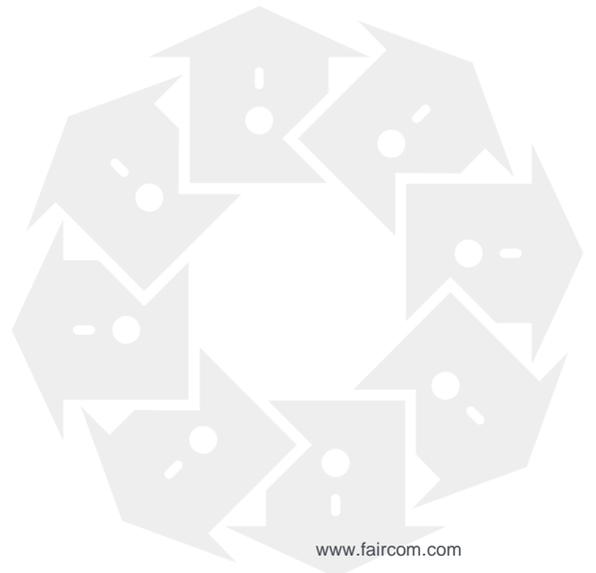**application**

Contents

1. Considerations When Planning Your Application

An important part of architecting any computer software project is to understand the constraints of that system and ensure that they do not impede operation of the system. Constraints come from many sources:

- The physical limitations of the hardware (hard disk sizes, amount of memory, etc.)
- Limitations of the operating system (address space, number of processes/threads, etc.)
- Limits of other software with which it interacts (memory requirements, restrictions in the amount of data that can be passed between processes at any one time).
- And, finally, the software itself, which may have its own internal limits.

Any piece of computer software must be designed with these constrains in mind.

FairCom DB is designed to minimize the constraints it adds to the above list. As an industrial-strength, enterprise-ready database, FairCom DB is intended to be used in demanding applications. Developed by and for developers, FairCom DB imposes very few limitations on the application you design.

In addition to understanding FairCom DB constraints from a development point of view, this information can help system administrators keep the system running at peak performance. Several aspects of operation can be monitored so that action can be taken to avoid limits long before they are reached.

This document outlines the constraints of FairCom DB so you can better understand the best way to work with it.

Constraints

While c-tree is designed to minimize artificial limitations, there are a few constraints that should be acknowledged to optimize operation and allow developers to plan an efficient application. These few constraints are listed below and described in more detail afterward.
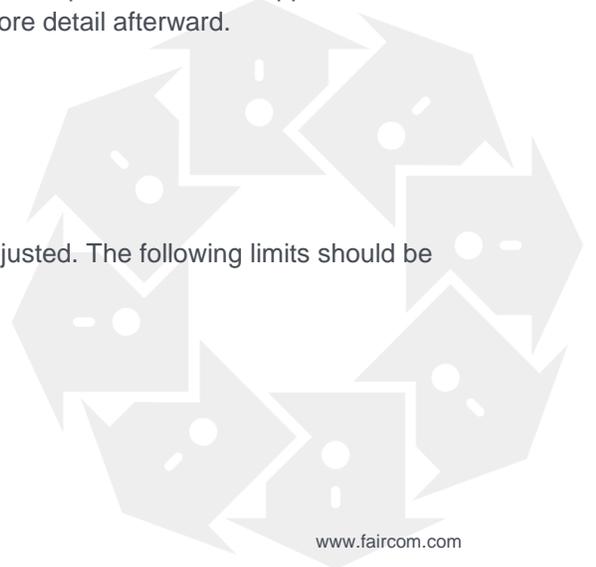
1.1 File-Related Limits

FairCom DB has very few hard limits, and some can be adjusted. The following limits should be kept in mind as you work with FairCom DB.

Max fixed length record size: 64KB

Max variable length record size: 4GB

Max number of records per file: No limit

Maximum file size: 16 Exabytes (18 million terabytes)

Maximum number of open files: 32,767

Maximum number of indices per data file: Defaults to 64 (can be increased using `MAX_DAT_KEY` in *ctsrvr.cfg* to a theoretical limit of 32767, although a practical limit exists well before this value)

Maximum number of segments per index: Defaults to 16 (can be increased using `MAX_KEY_SEG` in *ctsrvr.cfg* to a theoretical limit of 32767, although a practical limit exists well before this value)

1.2 File ID Overflow

Each time a transaction controlled c-tree data file or index file is opened, the value of its file ID number is increased. If your system has a large number of files, this value can increase a fair amount with each day of processing.

- The upper limit for this value is: 4,294,963,200

> If the upper limit is hit, the Server process will shut down.

- The value at which a "Pending File ID Overflow" warning message first appears is: 4,227,858,432

  The message "Pending File ID Overflow" will be written to *CTSTATUS.FCS*. A new entry will be logged every time another 10,000 numbers are used.

  From the time the first warning message appears, you have at most 67,104,768 additional data file and index file opens before this value hits this limit.

When the transaction file numbers have been exhausted, error **534** and the following message will be logged in *CTSTATUS.FCS*:

```
- User# 00018              Pending File ID overflow: 534
- User# 00018              O18 M18 L58 F-1 Pfffff003x (recur #1) (uerr_cod=534)
```

If you get error **534**, you must do a transaction log reset.

Pending File ID Overflow

The following warning message indicates that the FairCom DB Server internal file ID array is getting close to its upper limit:

```
Pending File ID Overflow
```

This issue is more common on older FairCom DB Server versions (before V9.3) because they used file ID numbers each time a c-tree data or index file was physically opened, even if it was just read, not written. Now FairCom DB uses a file ID number only when a file is physically opened and then updated.

Understanding the Warning Message

The "Pending File ID Overflow" message indicates that the FairCom DB Server internal file ID numbers are getting close to the upper limit.

Each time a transaction controlled c-tree data file or index file is opened, the value of its file ID number is increased. If your system has a large number of files, this value can increase a fair amount with each day of processing.

- The upper limit for this value is: 4,294,963,200

  If the upper limit is hit, the Server process will shut down.

- The value at which a "Pending File ID Overflow" warning message first appears is: 4,227,858,432

  The message "Pending File ID Overflow" will be written to *CTSTATUS.FCS*. A new entry will be logged every time another 10,000 numbers are used.

  From the time the first warning message appears, you have at most 67,104,768 additional data file and index file opens before this value hits this limit.

When the transaction file numbers have been exhausted, error **534** and the following message will be logged in *CTSTATUS.FCS*:

```
- User# 00018              Pending File ID overflow: 534
- User# 00018              O18 M18 L58 F-1 Pfffff003x (recur #1) (uerr_cod=534)
```

If you get error **534**, you must do a transaction log reset.

Determining the Current File ID

To determine the current value of your system's file ID number, you can use the **ctstat** transaction snapshot (**ctstat** *-vat*). The file ID number is shown as the `tfil` value (the sample below shows `tfil` of 233):

```
ctstat -vat -h 1 -i 1 1 -t -s FAIRCOMS


lowlog    curlog     lstent     lstpnt     lstsuc      tranno      tfil
    46        49     3217645    3217445         0     1045589       233
```

See Also:

- For more about **ctstat** *-vat*, see *Transaction Statistics Example* (*http://docs.faircom.com/doc/ctserver/60944.htm*) in the *FairCom DB Server Administrator's Guide*.

- For more about monitoring, see *Monitoring FairCom DB Transaction Numbers and Transaction File Numbers* (*http://docs.faircom.com/doc/knowledgebase/70287.htm*) in the *Monitoring Performance* section of the *Knowledgebase*.

Recommended Actions

The following actions are suggested when the file ID warning message is seen.

First, determine how much time you have before the upper limit is hit and the server shuts down:

1. Use the **ctstat** *-vat* command (as shown in the *previous section* (page 3)) on one of your highest number transaction logs to get the current file ID setting. Notice that this setting is only captured at the initial log creation, so it will increase during the processing of the active log.
2. Execute the **ctstat** *-vat* command on transaction logs from the previous day, first with the earliest log for the day and then with the last log for the day.
3. Calculate the difference in the file IDs. This will give you an idea of how many file IDs you have consumed during a given day so you can determine if you can safely wait until the next scheduled system restart.

Once you can safely shut down the system, be sure to shut it down cleanly. The best practice recommendation for shutting down FairCom DB Server is as follows:

1. Cleanly shut down the FairCom DB Server.
2. Restart the FairCom DB Server and prevent any users from connecting.
3. Cleanly shut down the FairCom DB Server a second time.

   This second shutdown ensures that any pending transactions in the current logs are processed.
4. Now you may safely move the existing transaction logs to a new location. Move the following files: **.FCS* and **.FCT*
5. Copy the *FAIRCOM.FCS* file back to its original location.

   The only **.FCS* to keep in your current directory is *FAIRCOM.FCS*. Keeping the *FAIRCOM.FCS* file will NOT impact the File ID setting.

   > *FAIRCOM.FCS* stores user information such as user IDs, so if you don't keep this file, you will have to recreate your users.

6. Restart the FairCom DB Server and it will create new transaction logs from scratch.

   You may confirm this by looking at the file names of the transaction logs (on Unix/Linux: **ls** *L*.FCS*) and you should see the first *L*.FCS* has been reset to number *L*00001.FCS*

If you would like to confirm that the file ID value has been reset, you can execute the **ctstat** *-vat* command again. You should see the file ID value is now a very low number.

1.3 Serial Number Segments

The ISAM-level segment mode 3, *SRLSEG*, indicates that the key segment will automatically be filled with a signed 4 or 8-byte sequence number. An **OSRL_ERR** (44) error is returned when the sequence number overflows. When using a signed 4-byte value, the value is limited to 2 GB. 4-byte sequence numbers should not be used in situations where more than 2^{31} (or 2,147,483,648) sequenced entries will be required. When using a signed 8-byte value, the sequence number value is limited to 2^{63} (or 9,223,372,036,854,775,808).

GetSerialNbr() returns the current sequence number for data file *datno* used in ISAM applications using a *SRLSEG* key segment mode. **GetSerialNbr()** returns a long integer containing the current sequence number, so use **ctGETHGH()** to obtain the high word portion of

the 8-byte value. If an error occurs (such as data file not in use), **GetSerialNbr()** returns a zero value. Check *uerr_cod* for the error code.

If a *SRLSEG* exists, **RebuildIFile()** finds the highest serial number in use and updates the file header with that value.

To manually manage serial numbers, use the *OPS_SERIAL_UPD status_word* described in the **SetOperationState()** function description.

1.4 Transaction High-Water Marks

The FairCom DB Server and the single-user standalone operational model with transaction processing use a system of transaction number high-water marks to maintain consistency between transaction-controlled index files and the transaction log files. When log files are erased, the high-water marks maintained in the index headers permit the new log files to begin with transaction numbers that are consistent with the index files.

With FairCom DB Server, if a transaction high-water mark exceeds 0x3ffffff0 (1,073,741,808) for version 7.x servers and earlier or 3ffffffffff0 (70,368,744,177,648) for version 8.x servers and later, then the transaction numbers will overflow causing problems with index files. On file open, an error **MTRN_ERR** (533) is returned if an index file's high-water mark exceeds this limit. If a new transaction causes the system's next transaction number to exceed this limit, the transaction will fail with an **OTRN_ERR** (534).

This should be an unusual occurrence except on systems that are continuously processing significant volumes of transactions.

Before these errors occur, the FairCom DB Server issues warnings when the transaction numbers approach the transaction limit. These warnings are issued periodically to the system monitor and the *CTSTATUS.FCS* file.

To aid in debugging if spurious **MTRN_ERR**s occur, the server configuration keyword TRAN_HIGH_MARK takes a long integer as its argument and specifies a warning threshold. If an index file's header contains a high-water mark in excess of this threshold, the file's name is listed in *CTSTATUS.FCS*.

Use the **CleanIndexXtd()** function or the **ctclntrn** utility to reset index file high-water marks to zero. This permits new logs to start over with small transaction numbers. To fully reset the high transaction number requires the following steps:

1. Shut down the FairCom DB Server cleanly, restart the server so it can do automatic recovery with no clients attached, and then shut it down cleanly a second time. Performing two shutdowns in a row ensures the application files are up-to-date and there are no pending recovery items so all *.FCS* files except the *FAIRCOM.FCS* file can safely be removed. Be sure not to overlook the *CTSYSCAT.FCS* file (used for ODBC) and the *SYSLOGDT.FCS* and *SYSLOGIX.FCS* files. If you are using the SERVER_DIRECTORY (which is now deprecated), LOCAL_DIRECTORY, LOG_EVEN, LOG_ODD, START_EVEN, START_ODD or similar keywords that take a directory path, be sure to check that path for any *.FCS* files that should be removed or data and index files that should be cleaned.

2. Use the **CleanIndexXtd()** function or the **ctclntrn** utility to clean all indexes used by the FairCom DB Server, including your application index files, superfiles and variable-length data files. Executing '**ctclntrn** *FAIRCOM.FCS*' will clean all the member indexes in this system superfile and the same is true of any application superfiles as well.

3. If you wish to verify the cleanup process, you can use the **cthghtrn** utility in the server utils directory to verify that the transaction high-water mark in the files you cleaned is zero or a reasonably low number. To check superfile index members, you must enter them individually, such as the following to verify the *FAIRCOM.FCS* file.

```
cthghtrn FAIRCOM.FCS!GROUP.idx
cthghtrn FAIRCOM.FCS!UG.idx
cthghtrn FAIRCOM.FCS!USER.idx
cthghtrn FAIRCOM.FCS!UVAL.dat (variable-length data file )
cthghtrn FAIRCOM.FCS!UVAL.idx (this may not exist in superfiles created with versions
of c-tree older than V6.11)
```

4. After completing the cleaning process, verify that there are no **.FCS* files other than the cleaned *FAIRCOM.FCS* (which has been verified clean as described above) in the server directory. If you are using `SERVER_DIRECTORY` (now deprecated), `LOCAL_DIRECTORY`, `LOG_EVEN, LOG_ODD, START_EVEN, START_ODD,` or a similar keyword in your *ctsrvr.cfg* file that takes a directory, be sure to check that path for any existing **.FCS* files and remove them.

5. When you are satisfied that you have completely cleaned all files, restart the c tree Server. As soon as the server is up and operational, cat or type the *CTSTATUS.FCS* file prior to attaching any clients and be sure there are no "Pending TRANSACTION # overflow" messages indicating that you have missed cleaning or removing a system (**.FCS*) file.

6. You can easily monitor the current transaction value in your application by checking the return of the **Begin()** function and verifying it against the 0x3ffffff0 or 3ffffffff0 threshold. This will ensure that you know well in advance about any impending transaction number overflow issues and will allow you to prevent any unexpected server shutdown.

> **Note:** If you perform the high-water transaction clean operation to reset your high-water mark and then perform a restore that has unclean files or transaction logs, you will need to perform the clean operation again.

1.5   Transaction Log Numbering

The limit for transaction log numbering is high enough that it is unlikely any production system will encounter it. FairCom DB allows transaction log numbers to reach values over 2 billion. Notice that this limit refers to the numbering of transaction log files, which is not related to transaction numbering (see *Transaction High-Water Marks* (page 5)). If a system should get close to the 2-billion mark, the Server Transaction Logs can be reset by safely removing the logs, following the FairCom DB Server Best Practice Upgrade Procedure, documented in the section of the *Knowledgebase* titled *Upgrading from Previous Editions* (*http://docs.faircom.com/doc/knowledgebase/28374.htm*).

1.6   FairCom DB SQL Constraints

FairCom DB SQL has its own set of constraints:

Attribute	Value
Maximum number of procedure arguments in a FairCom DB SQL CALL statement	50
Maximum length of a FairCom DB SQL statement	32MB *(prior to V10.2, the limit was 35,000 characters)*
Maximum length of a column: standard data types.	65536
Maximum length of a column: VARBINARY and VARCHAR specifying the character set designated as NATIONAL CHARACTER	32752
Maximum length of default value specification	250
Maximum length for an identifier	64
Maximum length of a connect string	100
Maximum length for a user-name in a connect string	64
Maximum length of an error message	511
Maximum number of columns in a table	2500
Maximum number of index components for a table, for all indexes on that table	500
Maximum length of a CHECK constraint clause	20,000
Maximum number of check constraints in a table	4000
Maximum number of foreign constraints in a table	4000
Maximum number of nesting levels in a FairCom DB SQL statement	25
Maximum number of table references in a FairCom DB SQL statement	250
Maximum size of input parameters for a FairCom DB SQL statement	512
Maximum number of outer references in a FairCom DB SQL statement	25
Maximum nesting level for view references	25
Maximum recursion level for stored procedures	10

1.7   c-treeRTG Constraints

The c-treeRTG products are built upon the FairCom DB database server. As such, they inherit the constraints of that server. In addition, c-treeRTG assigns an auto-incremental value associated with each record in a table. This value overflows at 4,294,967,295 (4G). Therefore, c-treeRTG has an inherent limit of 4 billion records.

2. Document History

January 9, 2019	Added *File-Related Limits* (page 1) and *FairCom DB SQL Constraints* (page 7)
November 14, 2016	Document published.

Last published Thursday, April 15, 2021.

www.faircom.com

Copyright Notice

Trademarks

Open Source Components

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

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

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

4/15/2021

3. Index