# c-tree Server V8.14 Technology

# Take Advantage of Six-Byte Transaction Number Support

# COPYRIGHT NOTICE



FairCom welcomes your comments on this document and the software it describes. Send comments to:

**Documentation Comments**
**FairCom Corporation**
**6300 W. Sugar Creek Drive**
**Columbia, MO 65203**
**Phone: 573-445-6833**
**Fax: 573-445-9698**
http://www.faircom.com /contact

# Trillions of Transactions!

c-tree Plus technology has been powering applications for over 25 years. As the number of c-tree Plus installations has grown, so too has the number of transactions that are processed by some of these installations, particularly in financial markets. Because of the way c-tree assigns transaction numbers to individual transactions, some installations using Version 6 and Version 7 of c-tree needed to perform maintenance on the data files to avoid transaction number overflow errors. No more! Migrating your application to the V8 c-tree Server (or later) will avoid this situation entirely.

## Overview

The c-tree Server associates a unique transaction number with every transaction. The transaction numbers assigned by the server start with transaction number 1 and are ever-increasing during the server's operation.

When a c-tree client begins a transaction, the return value is the transaction number assigned to that transaction (or 0 in the case of an error). The server writes transaction numbers to the following persistent locations:

- Transaction logs (the start files *S*.FCS*, and the log files *L*.FCS*)
- *TRNLOG* indexes (in leaf nodes and the header)
- Superfile hosts (including the server's user ID and password file, *FAIRCOM.FCS*)

## Four Byte Transaction Number Limits

Until V8.14, c-tree Plus used four-byte transaction numbers. This allowed just over one billion transactions (0x3ffffff0 or 1,073,741,808 to be exact) before a c-tree Server had to be taken offline for transaction log maintenance. While this was indeed a huge number of transactions when FairCom's transaction processing logic was originally introduced (many moons ago), it is no longer sufficient for today's advanced high-volume transaction systems.

Consider a moderately high-volume application with 1000 transactions per second.

- V6 supports around 1 billion transactions providing 1,000 tps for **~12 days**.

Clearly, it is only a matter of time before many applications, running quietly for years, begin to exhaust this supply of four byte numbers.

## Nearly Limitless V8 Six Byte Transaction Numbers!

c-tree Plus V8.14 introduced six byte transaction numbers. Literally, trillions and trillions of numbers are now available! Six byte transaction numbers essentially eliminate any previous shortcomings. With six byte transaction numbers, 70,000,000,000,000 (that's 70 TRILLION!) transactions can flow before the c-tree Server would require any maintenance. That's 70 trillion widgets manufactured! 70 trillion phone calls logged! 70 trillion credit card swipes! **70 trillion new customer orders!**

A sustained transaction rate of 1,000 transactions per second would not exhaust these transaction numbers for over 2,000 years!

- V8 supports $70 \cdot 10^{1012}$: 1000 tps for **~2000 years**.

Call FairCom today and seamlessly upgrade your application with a V8.14 c-tree Server and gain all the benefits of this extended support!

## 1.1 How will I know when I am close to running out of numbers?

With c-tree Servers prior to Version 8, when the current transaction number is close to the transaction number limit, the server outputs the following message in the status log, *CTSTATUS.FCS*:

```
"Pending TRANSACTION # overflow"
```

This message begins with transaction number 0x3c000000 (1,006,632,960) and repeats every 10,000th transaction. You can specify the transaction number to begin issuing warnings with the `TRAN_HIGH_MARK` c-tree Server configuration keyword. Specify a long integer for the value as in the following:

```
TRAN_HIGH_MARK 1000000000
```

With this statement the c-tree Server would begin issuing warnings beginning at the one billionth transaction. Use this configuration option if you would like plenty of advance warning before you run out of transaction numbers.

> **Note:** With the extremely large number of transaction numbers available with the c-tree Server V8 and later, no warning message is currently generated. The server will shut down when it reaches the last possible number.

## 1.2 What happens when the c-tree Server runs out of numbers?

If the transaction number high-water marks exceed the four byte limit of 0x3ffffff0 (1,073,741,808), then the transaction numbers overflow, and will cause 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 fails with error **OTRN_ERR** (534).

The c-tree Server will then **shut down** to prevent any further operations until the administrator takes steps to restart the transaction numbering. The c-tree Server administrator will need to perform the simple maintenance described in the following sections before continuing operations.

## 1.3 How do I clean and reset the transaction numbers for my files?

FairCom provides the c-tree Clean Transaction Water-Mark utility, **ctclntrn**, to reset the high-water mark transaction numbers in your index files.

In the event of an impending transaction number overflow, the server administrator should follow these steps to restart transaction numbering:

1. Perform a clean shutdown of the c-tree Server.
2. Delete the transaction logs: files *S0000000.FCS*, *S0000001.FCS*, *D*.FCS*, *I*.FCS*, and *L*.FCS*.

3. Use the **ctcIntrn** utility to clean all indices (which resets the transaction numbers in the leaf nodes and index header) used by your application, including your application index files, superfiles, variable-length data, and c-tree Server files including the following if present:
   - *FAIRCOM.FCS* -- If you are not using any User IDs or passwords other than ADMIN and GUEST, you can simply delete this file. (If you do delete this file, the c-tree Server will re-create it with the single user ADMIN and password ADMIN.)
   - *CTSYSCAT.FCS* -- This is only present if you are using c-tree ODBC Drivers.
   - *SYSLOG*.FCS*
4. Restart the c-tree Server.

The server will create new transaction logs and start transaction numbering from 1 again.

It is important to run **ctcIntrn** on all files. If you miss any files and later open that file with a large transaction number in its header, the server will again increase its transaction number to that large value. You will need to repeat this procedure should that occur.

**FYI:** The **ctcIntrn** utility uses the **CleanIndexXtd()** c-tree Plus function. This function cleans any leaf nodes of exceptional transaction marks and resets the transaction high-water mark in the header to zero. This avoids rebuilding the entire index file.

You can use the c-tree High-water Mark Utility, **cthghtrn,** to verify that the transaction high-water marks in the files are back to zero, or other reasonably low number.

## 1.4 Why did I have an unexpected "jump" in transaction numbers?

The FairCom transaction processing logic used by the c-tree Server utilizes a system of transaction number high-water marks to maintain consistency between transaction controlled index files and the transaction log files. Should the transaction log files be erased, the high-water marks maintained in the index headers permit the new log files to begin with transaction numbers which are consistent with the index files.

When the server opens a transaction-controlled index, it checks the transaction high-water mark in the header of the index file. If that value exceeds the server's current transaction number, the server changes its current transaction number to the value from the index header (unless the file open is done within a transaction, in which case the open fails with error **MTRN_ERR** (533) and the server does not change its current transaction number). This behavior means that an index file whose header has somehow been damaged and now contains a very large (erroneous) transaction number results in the c-tree Server's transaction number to jump to a large value.

When a pre-Version 8 c-tree Server detects that it's making such a large jump (either the index header transaction number is greater than 0x3c000000 or the difference is greater than 0x04000000), the following message is output to the status log:

```
"If transaction logs not purged recently, then following file may have contaminated
header..."
```

Then, if the server's current transaction number is close to the transaction number limit, the server begins to output a second message in the status log:

```
"Pending TRANSACTION # overflow"
```

It is then likely that the files mentioned in the status log potentially had a damaged header with a very large transaction number and forced the server's current transaction number to jump to this large value. This situation could also potentially arise when restoring files from backups. An older index file could contain a large transaction number which will also force the server to reset it's high-water mark.

Should this occur, and to prevent a transaction number overflow (which will cause the server to shut down), reset the transaction numbers by following the steps in the proceeding section before continuing operations.

## 1.5 How do I start using six byte transaction numbers in my application today?

In c-tree Plus V8 and later, six byte transaction numbers are used by default. They will not be used on an individual file creation in any of the following cases:

5. If there is no extended create block
6. If the *ctNO6BTRAN* bit in the *x8mode* member of the extended file create block (*XCREblk*) is turned on
7. If the *ctNO_XHDRS* bit is turned on in *x8mode*.

For legacy application support, you can override this default such that four byte transaction numbers are instead used by default by adding the `COMPATIBILITY 6BTRAN_NOT_DEFAULT` keyword to the server configuration file.

### Creating and Rebuilding Extended Transaction Number Enabled Files

**Xtd8()** file create and rebuild functions must be used in order to create files with six byte transaction number support. If a non-**Xtd8()** file create function like **CreateIFileXtd()** is used to create index files, the index file is created without an extended header, so it cannot support six byte transaction numbers. Even if the specified extended file mode do not include the six byte transaction number support flag (*ct6BTRAN*), c-tree defaults to that option when a file is created with an extended header.

Here is a listing of the functions that can be used to create indexes with extended headers:

- **CreateIFileXtd8()**
- **PermIIndex8()**
- **TempIIndexXtd8()**
- **RebuildIFileXtd8()**

> **Note:** Files supporting six byte transactions are not required to be huge, however, they do require an extended header.

Ordinary data files are unaffected by this modification, and they are compatible back and forth between servers with and without six byte transaction support. Except for superfile hosts, the *ct6BTRAN* mode is ignored for data files. Index files and superfile hosts are sensitive to the *ct6BTRAN* mode:

1. An existing index file or superfile supporting only four byte transaction numbers must be converted or reconstructed to change to six byte transaction number support.
2. The superfile host and all index members of a superfile must agree on their *ct6BTRAN* mode (either all must have the *ct6BTRAN* mode on or all must have it off), or a **S6BT_ERR** (742) occurs on index member creation.

**Note:** A previously existing index will only use four byte transaction numbers and an attempt to go past the (approx.) 1,000,000,000 transaction number limit will result in an **OTRN_ERR** (534).

An attempt to open a file using six byte transactions with code that does not support six byte transactions (prior to V8.14 libraries) will result in **HDR8_ERR** (672) or possibly **FVER_ERR** (43). This is because the extended header information is unable to be properly interpreted by the older code.

The V8 c-tree Server always stores transaction numbers in the transaction logs as six byte values and stores transaction numbers in indices as either four byte or six byte numbers depending on the index creation options. All indexes that the server creates for its own internal use are created as six byte transaction number indexes, so the only possible source of four byte transaction numbers is an application that creates index files as four byte transaction number files. If the application creates all its indexes as six byte transaction number files, the server will not be subject to the four byte transaction number limit.

## 1.6 Migrate to V8 and six byte transaction numbers in five easy steps!

### Start Using Six Byte Transaction #'s in 5 Easy Steps

1. Relink all client applications with c-tree Plus V8 or newer libraries.
2. Install a c-tree V8 Server or newer using recommended best practices.
3. Ensure ALL c-tree Server Transaction Logs (**.FCS*) are removed.

**Warning:** You will lose all User IDs and passwords when *FAIRCOM.FCS* is deleted. These can be recreated with the c-tree Server Administration utility, **ctadmn**. See items below for an alternative means.

4. Ensure all data and index files have an extended header which allows huge file and six byte transaction numbers. This is required for index files and superfile hosts (where all members must agree on their six byte transaction attribute.) Data files are unaffected by the six byte transaction number attribute, however, it is generally good practice to maintain consistency between data and index files. V6 c-tree Plus data files may be converted using the c-tree Plus Conversion utility, **ctcv67**.
5. After starting the new c-tree Server, ensure ALL indices are rebuilt using the c-tree Server.

### FAIRCOM.FCS User and Group file

The c-tree conversion utility, **ctcv67**, when compiled with #define ctFeat6BT in *ctopt1.h*, can be used to rebuild the *FAIRCOM.FCS* file for six byte transaction number support. After building the utility, execute it as follows:

```
# ctcv67 FAIRCOM.FCS ./NEW PP YES 6 YES
```

### Existing Index Files

The most direct way to convert existing index files is to rebuild them with the c-tree Server. Securely back up existing index files, delete them from the server directory area, and then rebuild them such that they gain six byte transaction support. If existing indexes without an appropriate extended create block are rebuilt, they will be rebuilt without six byte transaction support.

### Standalone Applications

You can use the c-tree Plus rebuild utilities (**ctrbildif()** and **ctrbld()**) linked with c-tree V8 c-tree SDK as a c-tree single-user library, however be sure the six byte transaction number support is turned on. It is off by default in this operational model. This requires #define ctFeat6BT found in *ctopt1.h*.

Superfiles can be converted with the c-tree superfile compact utility, **ctscmp**, however, only if compiled with the #define ctFeat6BT. Note that the superfile host, and all index members must agree in support for six byte transaction numbers.

When converting from c-tree V6 to V7 files, the c-tree conversion utility, ctcv67, should be compiled with #define ctFeat6BT to gain the six byte transaction number capability in your indexes and transaction files.

### Transaction Log Conversions

Transaction log files produced with the six byte transaction number feature are not compatible with c-tree Servers or standalone applications not running with the six byte transaction number feature. The c-tree System automatically detects the old logs. To switch servers, a site must do a clean shutdown of the server. It is not absolutely necessary to remove the old logs, and the new server will detect the old logs, however no automatic recovery will be performed and the new server will skip several log sequence numbers and start with its own new logs. A more complete alternative is to remove the old logs after a clean server shutdown and have the new server start with new log files. Regardless of conversion method, it is not possible to roll back or roll forward across the boundary between the old and new logs.

## 1.7 Can the c-tree Server enforce six byte transaction number consistency?

You can prevent the c-tree Server from creating or opening files that only support four byte transaction numbers by adding the following option to the c-tree Server configuration file:

```
COMPATIBILITY EXTENDED_TRAN_ONLY
```

That option will help you identify which files do not have six byte transaction number support, because their open or create will fail with error **R6BT_ERR** (745, *6BTRAN* file required).

> **Note:** Read-only opens are not a problem, as the file can not be updated.

The following c-tree Server configuration option will cause the c-tree Server to default to creating files supporting only four byte transaction numbers:

```
COMPATIBILITY 6BTRAN_NOT_DEFAULT
```

Additionally, the following option is available to check for files that do not support extended transaction numbers:

```
DIAGNOSTICS EXTENDED_TRAN_NO
```

When the server physically opens a file that does not support extended transaction support, a message is logged to *CTSTATUS.FCS*. This applies to index files and superfile hosts. Data files are unaffected by the six byte transaction number attribute.

## 1.8 Where can I get help?

Have questions regarding transaction number overflows in your application? Require assistance upgrading or migrating to Version 8? Contact your local FairCom office for any support needed. Our support team is unsurpassed in the industry for working with our customers. We're here to help! Call us to obtain the latest V8 c-tree Server and begin upgrading your application today.