Avoiding Common Problems in Rational ClearCase

by **Mark Zukowsky**

As a member of Rational's Customer Advocacy Group (CAG), which deals with the calls that Rational Technical Support can't handle, I've heard about certain problems with Rational ClearCase over and over again from a number of different users. In this article, based on a presentation I gave at the Rational User Conference (RUC) in 2001, I'll describe some of the most common problems I hear about. I'll analyze each problem, describe the sort of data we collect in order to fix the problem, and explain how you can avoid the problem in the first place.

[Editor's Note: We've published another presentation made by Mark at RUC 2001 as an article on Rational Developer Network. Caring for Your Rational ClearCase VOBs is intended for beginning Rational ClearCase users.]

Problems Unlocking a Client VOB

You can use metadata in multiple client VOBs (versioned object bases) at the same time via hyperlinks to and from the admin VOB. Time and again, customers run into the following problem related to the admin VOB: When they try to unlock a client VOB, they get the error messages "Trouble opening VOB database," "Unable to search for process guards," and "Unable to unlock versioned object base."

To collect data about the problem, we can run `cleartool describe` on the client VOB. Listing 1 shows the results of one such query. As you'll see at the bottom, there are two errors, both returning the pathname of an admin VOB.

```
[msz_cag] zukowsky@grouse> cleartool describe vob:/var/tmp/msz_client
versioned object base "/var/tmp/msz_client" (locked)
 created 18-Apr-01.16:00:28 by Mark Zukowsky (zukowsky.user@grouse)
 VOB family feature level: 2
 VOB storage host:pathname "grouse:/var/tmp/msz_client.vbs"
 VOB storage global pathname "/net/grouse/var/tmp/msz_client.vbs"
 database schema version: 54
 VOB ownership:
  owner atria.com/zukowsky
  group atria.com/user
```

```
 Attributes:
  FeatureLevel=2
cleartool: Error: Error from VOB database: "/var/tmp/msz_admin"
cleartool: Error: Trouble opening VOB database: "/var/tmp/msz_admin"
```

Listing 1: Results of running cleartool describe on the client VOB

In this case, we ran `cleartool lsvob` on the missing admin VOB and found no matching entries.

This told us that what happened is that the client VOB was locked and then the admin VOB was removed, resulting in a situation where the client VOB couldn't be unlocked due to a hyperlink pointing to the nonexistent admin VOB. Running `checkvob` won't work in a situation like this because the client VOB is locked. This situation can also occur if you have to restore an entire set of VOBs from backup, and you decide you no longer want the admin VOB and just don't restore it.

At first we wrote a tool to solve this problem, a tool that would unlock a VOB for you forcibly. With the introduction of UCM came two environment variables that you can set to prevent client VOBs from going to the hyperlink to the admin VOB. Here they are, along with the values they should be set to:

- `CLEARCASE_PROCFLAGS = no_abort_op`
- `CG_PROCFLAGS = no_process`

If you set these environment variables, you'll be able to unlock a VOB without errors, and then you can run `checkvob` to clean up the dangling hyperlink to the admin VOB that's no longer there.

To avoid this problem in the first place, make sure when you're moving an admin VOB that the client VOBs are unlocked. If you're restoring completely from backup, be sure the admin VOB is restored as well.

Problems Relocating a Version

When running `cleartool relocate`, customers sometimes see the error messages illustrated in Listing 2.

```
#cleartool relocate-force-update ./src/server ./src/common ../callcopy
. . .
  updated version "/main/Bcallistox/0"
cleartool: Error. INTERNAL ERROR detected and logged in
"/var/adm/atria/log/error_log".
cleartool: Error: Unable to duplicate version dbid:78438.
cleartool: Error: Unable to duplicate version dbid:78438.
cleartool: Error: Unable to duplicate versions for branch dbid:41406.
cleartool: Error: Unable to duplicate versions for branch dbid:32769.
cleartool: Error: Unable to duplicate object "server/ebrt/readme.txt"
```

Listing 2: Symptoms of a problem with cleartool relocate

To troubleshoot this problem the first time a customer brought it to us, we

asked for `cleartool dump` and `cleartool describe` output for each database ID listed in the `cleartool relocate` output. This didn't tell us anything, so we requested a copy of the VOB database from the source VOB so that we could try to reproduce the problem in house. But *that* didn't work because we had problems with hyperlinks, so we requested a copy of the customer's VOB database from the admin VOB to alleviate those issues.

Once we had the copy of the VOB database and the admin VOB in house, we did reproduce the problem with the equivalent `cleartool relocate` command. We ran everything in the debugger to find the root cause of the problem. We then reproduced the test case external to the customer environment and figured out exactly what was causing the problem.

It turned out that the version that couldn't be relocated was on a branch off of a version that had been removed. Removing data for the latter version also removed essential data for the former. And the internal error was due to referencing a container incorrectly.

So we raised a defect against `cleartool relocate`, and it got fixed in ClearCase 4.2. For the customer, we created a utility to fix the container references internally, and that allowed the relocate operation to work. There's nothing you can do ahead of time to avoid this problem, but if you do see it, you can contact Technical Support. Once the patch is out, you can use that as well.

Problems with Full Database Files

ClearCase users quite often run into problems with full database files. This happens in schema 53 databases when you hit the limits on the size of individual database files and on the number of records you can have in each individual database file. There are two database files that tend to fill up - `vob_db.d01` and `vob_db.str_file`. We'll look at examples of each here and discuss how to deal with the problem. (It's very rare for files other than these two and `vista.tjf`, which I'll briefly mention at the end of this section, to fill up.)

Listing 3 shows an excerpt from a customer's database server log illustrating the kinds of error messages you might encounter if your `vob_db.d01` file fills up. "File record limit exceeded" means we've put the maximum number of records we can into that file, which happens to be 224 (about 16 million).

```
3/15/01 09:51:57 AM db_server(3577): Ok: *** db_VISTA
   database error -909 - file record limit exceeded
03/15/01 09:51:57 AM db_server(3577): Error: DBMS error
   in "../db_oplog.c" line 118
03/15/01 09:51:57 AM db_server(3577): Error: DBMS error
   in /vobstore/equinox1/equinox_ne_loadbuild.vbs.db.
3/15/01 09:51:57 AM db_server(3577): Error: db_VISTA
   error -909 (errno == "Resource temporarily
   unavailable")
```

Listing 3: Error messages in the database server log caused by a full vob_db.d01 file

To find out what's going on here, we can use a tool called `countdb` that's

included in the etc/utils subdirectory in ClearCase 4.1 and later. (If you have an earlier version of ClearCase, contact Technical Support for help.) It tells us how many there are of each individual record type in the database list. It also gives the sum and tells us how close we are to filling up the database with the 16 million records.

Listing 4 is an excerpt from countdb output showing the information it provided on the data file vob_db.d01. The total number of records in use is 16 million plus, the maximum number of records possible. Of this total, approximately 6 million are labels attached to various versions in the database and 6 million are oplogs in the database.

```
*********************************************************
Data file name : vob_db.d01
*********************************************************
Total records in use     :     16777212
Maximum records possible :     16777215
% of maximum records used:       100.00%

VERSION_LABEL_LINK               :  6289673
HLINK_TYPE                       :       10
HARROW                           :    19465

EPOCH_TABLE_ENTRY                :        3
EXPORT_TABLE_ENTRY               :     6088
OPLOG_ENTRY                      :  6688217
```

Listing 4: Excerpt from countdb output

The .d01 file usually fills up due to high numbers of VERSION_LABEL_LINK and OPLOG_ENTRY records, as well as DOT_DOT/NAMESPACE_DIR_VERSION_ENTRY records. If you encounter the problem of full database files and get large numbers for any of these three types of records when you run countdb, we recommend that you take action to reduce the number of records.

- Each one of the VERSION_LABEL_LINK records is an instance of a label attached to a version of an element. The only way to clean these up is to actually remove the label type or the individual labels attached to the versions.
- DOT_DOT/NAMESPACE_DIR_VERSION_ENTRY records are created when you check out a directory, make an element in the directory, and then check in the directory. Every entry in each version of the directory causes one of these records, so if you have five versions of the directory with five files in that directory, that's 25 of these records. The way to add multiple files to a directory version without causing these records to proliferate like that is to check out the directory, make elements for all of the files you want to add, and then check the directory back in. To reduce the number of records of this kind, you can remove older versions of the directories that you don't need anymore.
- Each oplog causes an OPLOG_ENTRY record to be placed in the database. Unfortunately, the default is never to scrub oplogs, so if you've been syncing using MultiSite on a database for three years, you have three years' worth of oplogs in there, way more than you generally need. To

clean this up, you can modify the vob_scrubber_params file and then run the VOB scrubber, and it will delete oplogs that are older than the number of days you specify. Don't scrub the oplogs too aggressively, though. If you're syncing once a week you need at least 7 days' worth of oplogs; to be on the safe side, you should make it even more than that - maybe 60 days' worth.

To avoid the problem of full .d01 files, you can upgrade to schema 54, which alleviates the limit of 16 million records per database file. (But note that that's not an excuse to let your database grow to mammoth proportions: you should still be monitoring the database for performance, disk space, and the like.) You can run countdb to monitor the file, and if you see a lot of VERSION_LABEL_LINK, OPLOG_ENTRY, or DOT_DOT/NAMESPACE_DIR_VERSION_ENTRY records, you can clean them up as detailed above.

The other file in the database subdirectory that tends to fill up is the vob_db.str_file file, which was implemented in ClearCase 3 to store "blob" information strings, configuration records, and so on. Schema 53 combined with operating system considerations limits the size of this file to 2 GB. Listing 5 shows an excerpt from a customer's database server log during a clearmake operation illustrating the kinds of error messages you might encounter if this database file fills up.

```
07/20/99 13:24:26 db_server(15545): Error: DBMS error in
  /spm_cc_storage/SPM/db.
07/20/99 13:24:26 db_server(15545): Error: db_VISTA error 2
  (errno=="Resource temporarily unavailable")
07/20/99 13:27:03 db_server(9723): Error: DBMS error in
  "../db_str.c" line 153
```

Listing 5: Error messages in the database server log caused by a full vob_db.str_file

The utility called string_report, shipped in ClearCase 4.1 and later versions in the etc/utils subdirectory, will tell you what's filling up the string file. In general, it's usually configuration records. If you have a 2 GB string file, it's a pretty good guess that at least 1.8 GB will be filled with configuration records.

We can use the scrubber to remove unused configuration records from the string file. Well, actually, it's a little more complicated than that. As shown in Figure 1, the scrubber physically removes only those configuration records with a reference count of 0, which doesn't include all the configuration records in a library.

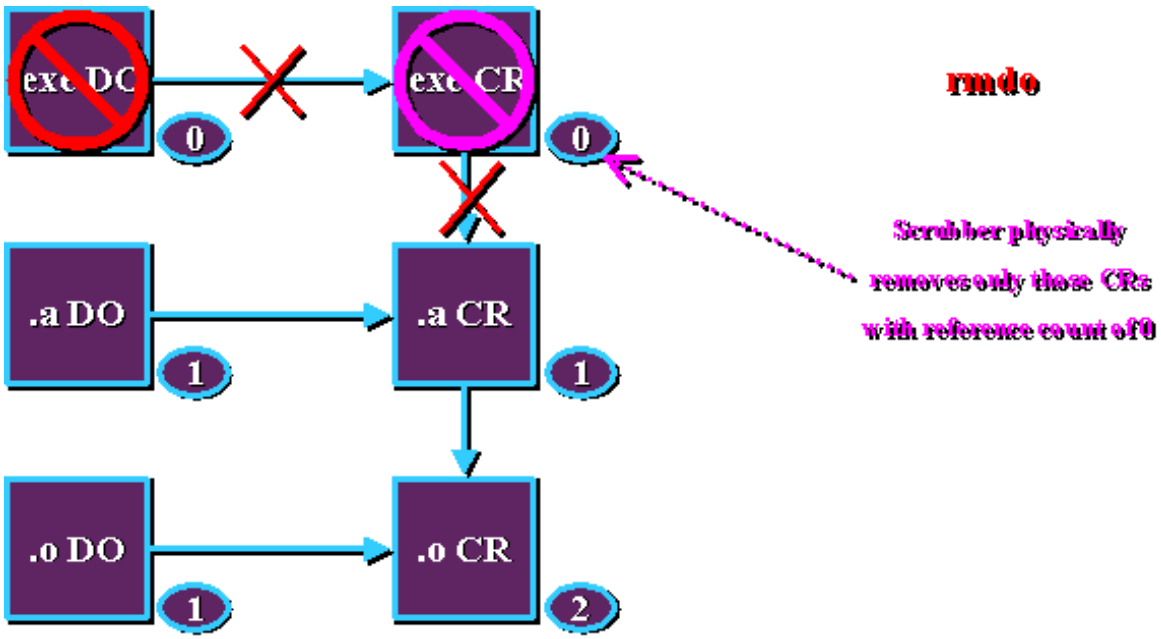

Figure 1: Versioned DOs and CRs in a library

To start out, we have a derived object (DO) with a reference count of 1, referencing a configuration record (CR) that also has a reference count of 1. We then build a library that has that DO in it, and we get a .a DO with a reference count of 1, which points to a CR for the .a DO with a reference count of 1. That .a CR also references the .o CR as a subconfiguration record, which means the reference count on that .o CR is now up to 2. At this point, if we decide our string file is full and do an `rmdo` on the .exe DO, that DO will be gone from the system and its reference count will drop to 0. That DO no longer references the CR, and the CR reference count drops to 0. The CR no longer references the .a CR; its reference count drops to 1 because it's still referenced by the .a DO.

When we run the scrubber, it removes from the string file only those configuration records with a reference count of 0. So in our example, even though we've removed the .exe DO, we still have .a CR and .o CR taking up space in the string file. To get rid of these, we need to get their reference counts to 0.

It gets even worse in MultiSite, as shown in Figure 2. When we check in a DO in MultiSite, we create an oplog for that check-in, and that oplog references the CR as well, so its reference count goes up to 3. And when we check in the .a DO and the .exe DO, we also have oplogs that reference the CRs, and their reference counts go up as well.

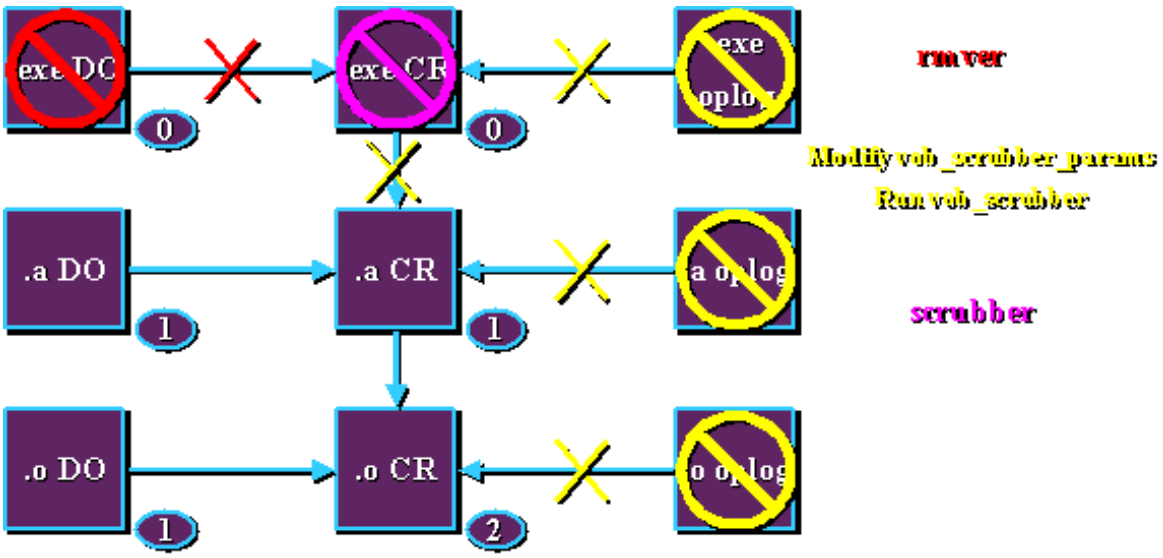

Figure 2: Versioned DOs and CRs in a MultiSite environment

In this case, when we do an rmver on the .exe DO, its reference count drops to 0. It no longer references the CR, whose reference count drops to 1. But the scrubber won't remove this CR because its reference count isn't 0. In fact, we can remove all the versions of all the DOs we have and the string file still isn't going to shrink when we run the scrubber, because we have all those oplogs still pointing at the CRs.

So what we need to do before running the scrubber is to modify the VOB's scrubber parameters, as discussed earlier for the `countdb` output. With the appropriate parameters, the scrubber will then remove each of the oplogs and its references to the CRs. The reference counts on the CRs will be decremented by 1. But the scrubber will only get rid of CRs with reference counts of 0, so we'll still have CRs left in the string file taking up space.

The thing to realize is that the string file will never actually shrink even if we've removed all our CRs religiously, but in reality there may still be plenty of space available to use. The `string_report` utility will give us that information.

In summary, then, to resolve the problem of a full string file, you must monitor DO usage and remove older ones that you don't need anymore. In addition, you need to scrub your oplogs a little more aggressively.

To avoid the problem, you can do the following:

- Monitor `string_report` output, which will tell you the total size of the file, how many gaps it has, and how much space is available.
- Monitor old view usage, which usually has references to DOs. If a reference to a DO is there, a reference to a CR is there also, taking up space.
- Find unused CRs and force their reference count to 0. The catch is that there's no way to get the reference count of a CR and there's no way to figure out what references a CR. There's a utility available from Technical Support called `cc_do_strlen_list` that will list all of the DOs

still in the VOB by database ID and tell you how much space each DO is taking up and, by implication, how much CR information that DO actually references in the string file. Removing that DO won't necessarily recover all of that space in the string file, though, because DOs are shared: a `.o` DO can be used in multiple `.a` DOs.

There's one more file that can fill up in the database subdirectory - the `vista.tjf` file, which contains transactions that were playing to the database. This file has a 2 GB limit even if you do go to schema 54. It's normally flushed every five minutes, but if you run things like the scrubber or MultiSite sync operations, it can still fill up. There are a couple of ways to avoid filling up the `vista.tjf` file:

- One of the more recent patches will prevent it from filling up in all but one case.
- A journal file limit option (found on the `config_ccase` manual page) that you can put into the `db.conf` file will, in almost every case, prevent the `vista.tjf` file from filling up, even if you're doing a lot of transactions in a five-minute period.

Problems with Imports Failing in ClearCase MultiSite

ClearCase MultiSite allows for development across multiple sites by replicating changes at one site to other sites via oplogs. An oplog is a single operation that gets replayed at each replica. Oplogs that have occurred are tracked via epoch numbers.

The main problem customers seem to have with MultiSite is that `multitool` imports fail. Depending on the version of ClearCase you're running, you may or may not see something that resembles Listing 6. Older versions will fail with some meaningless error.

```
texarkana:scm::19:multitool syncreplica -import
/usr/atria/shipping/ms_ship/incoming/sync_01-04-11.04-00-01_29413
multitool: Error: CORRUPTION DETECTED!!! Sync. packet
/usr/atria/shipping/ms_ship/incoming/sync_01-04-11.04-00-01_29413
   targeted for non-replicated VOB/vobstorage/NMIP/COTS_ARCHIVE.vws
multitool: Error: Sync. packet
/usr/atria/shipping/ms_ship/incoming/sync_01-04-11.04-00-01_29413
   is not applicable to any local VOB replicas
```

Listing 6: Error messages caused by failure of a multitool import

To collect data about such problems, CAG has a script called `multisiteinfo.pl` that runs these commands:

- `cleartool lsvob` *`vob-tag`*
- `cleartool -VerAll`
- `multitool -VerAll`
- `cleartool Isreplica -long`

- `multitool lsepoch`

It also runs the following for each replica:

- `cleartool describe replica:`*`replica`*
- `cleartool dump replica:`*`replica`*

We can send you this script, but if you want to be proactive, you can just run these commands when you have a MultiSite issue and send the output to Technical Support.

What we usually find has happened when `multitool` imports fail is that the customer has restored a replica from backup without running the `restorereplica` command as described in the admin manual. The chain of events is pictured in Figure 3. The table on the right in the figure shows how many operations replica A thinks have been replayed at replica A and at replica B, and how many operations replica B thinks have occurred at replica A and at replica B.

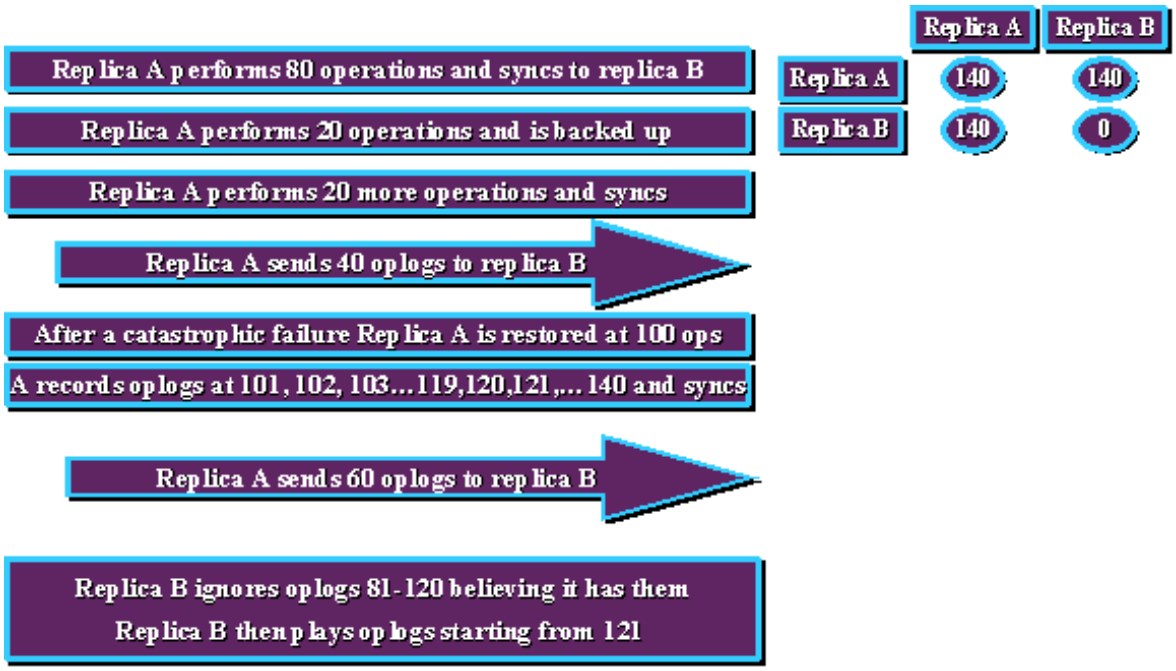

Figure 3: Restoring a replica from backup without running restorereplica

Replica A first performs 80 operations (oplogs) and syncs to replica B. At this point, replica A knows that replica A has done 80 operations and that replica B has 80 operations that it played at its site; and replica B knows about 80 operations that have been played at replica A.

Replica A then performs 20 more operations and is backed up. At this point, replica A knows it has performed 100 operations, but because it hasn't synchronized to B, it thinks B only knows about 80 of them. Replica A does 20 more operations and then syncs. It sends 40 oplogs to replica B to catch replica B up with the 120 operations it's done.

Replica A then suffers a catastrophic failure, is restored from backup without running `restorereplica`, and records oplogs up to 140. At this point, it thinks it needs to send 60 oplogs to replica B because it only knows from the backup that B had 80 of these oplogs. But replica B has already gotten 120 oplogs from replica A, so it's going to start replaying them at 121. At this point the replicas diverge and replica A is locked. The error that says "corruption detected" is based on replica B's thinking that oplog 81 is a duplicate. In reality, this error may not show up immediately; it may take a couple of months for imports to start failing, but when they do, it's usually because there was a restoration without running `restorereplica`.

To resolve the problem, we need to go back to the backup version and run `restorereplica`. When `restorereplica` is run, replica A sends a message to replica B that amounts to saying, "I don't know what's going on! I don't know how many operations I should have, or how many you have. Let me know what you've got." Replica B will respond with 20 oplogs - the 20 that A doesn't know about but B does. Replica A replays those 20 oplogs and is then unlocked and open for business after all the tables have been updated.

To avoid this problem, be sure to follow the admin manual procedures and run `restorereplica` when restoring MultiSite VOBs from backup.

Problems with UCM

UCM provides out-of-the-box processes and methodology for code development VOBs. It's implemented by way of a lot of hyperlinks between component and process VOBs. In general, the problems customers have with it are similar to admin VOB issues in that they're the result of interactions among activities, baselines, projects, and components not being cleaned up properly. For example, running `cleartool rmvob` *`component-VOB`* can cause problems when you attempt to remove a component with `rmcomp` if activities in the process VOB still reference activities in the component VOB. Defects have been raised and will be fixed, but meanwhile, such problems are part of UCM's growing pains.

To avoid these problems, be sure to back up process VOBs and component VOBs simultaneously and restore them as a unit, if it comes to that. If you want to remove components, do it via the `rmcomp` command while your component VOBs are still in existence. And if you can't deliver or rebase, contact Technical Support for help. We have numerous utilities available to help you out if you find yourself in one these situations.

Problems with the Lock Manager

The Lock Manager coordinates access to VOB databases by multiple clients. Problems occur due to various resource limitations. Listing 7 shows examples of two different types of error messages you might see as a result of a problem with the Lock Manager. In general, a `***db_VISTA database error -922` is always going to be related to the Lock Manager.

```
05/03/00 04:46:40 db_server(469): Error: db_server.exe(469): Error: Cannot
```

```
   open database in "d:\ClearCase_Storage\VOBs\gus_safeview.vbs\db"
05/03/00 04:46:40 db_server(469): Error: db_server.exe(469): Error: Too
   many open databases on host (try increasing -f argument on lockmgr
   command line)

db_server(14571): Ok: ***db_VISTA database error -922 - the lock manager
   is busy
db_server(14571): Error: DBMS error in /vobs/smi/pbn/db.
db_server(14571): Error: db_VISTA error -922(errno == "No such file or
   directory")
db_server(14571): Error: Cannot open database in "/vobs/smi/pbn/db"
```

Listing 7: Error messages caused by problems with the Lock Manager

When we collect data about a problem with the Lock Manager, we want to know how busy the server is - how many VOBs are on there and how they're being used, whether to run `clearmix` or just to sync a lot. We want to know what the current Lock Manager parameter settings are. We also want to gather some statistics that tell us how many processes are talking to the Lock Manager at any one time.

We can get these on UNIX by sending `SIGQUIT` to the `lockmgr` process. We can get them on Windows NT by first killing the Lock Manager as a service (`net stop lockmgr`) and then restarting the Lock Manager in a command window as `clearcase_albd run` -

<ATRIAHOME>\bin\lockmgr -a almd -q 1024 -u 1016 -f 1016 -nosvc

- and periodically pressing CTRL-C to get the `lockmgr` process to dump some statistics.

The Lock Manager manages lock requests from any process that needs to access a VOB database. Actually, there are only two of those - `db_server` and `vobrpc_server`. There's only one `lockmgr` process per VOB server, no matter how many VOBs you have on the server. And the Lock Manager has various limits that are defined when it's started, via the command line or via a registry value for file tables (the `-f` parameter), user tables (the `-u` parameter), or queue tables (the `-q` parameter). Let's look at each of those parameters along with their limits and what they should be set to for optimal performance (at least in theory).

- The `-f` parameter indirectly determines how many VOBs can be accessed on a system at any one time. VOB databases have 7 files each - 3 data files and 4 key files - in the VOB storage area `db` subdirectory. The default `-f` value of 256 files means that there can be 36 VOBs (256 divided by 7) on a server without modification. If you have more than 36 VOBs on a server and you haven't modified this, you might encounter problems such as poor end-user response while waiting for locks, and various error messages in the log file. Try increasing the `-f` parameter to increase the size of the `lockmgr` process. There's no practical limit to the size of the file table, but we recommend that you set the value to 7 times the number of VOBs you're going to have on

the system.

- The -u parameter determines the maximum number of db_server and vobrpc_server processes that can request locks from the Lock Manager. Again, the default value is 256. Typically, there's only going to be one active db_server process for each active client ClearCase command. This parameter essentially limits the amount of concurrent ClearCase activity, no matter how many VOBs are on the system. Again, you'll see poor end-user response and "lock manager is busy" errors if the -u parameter is set too low. If you do run into these problems, you can increase the value of this parameter. With the old Lock Manager, the maximum value allowed is 1018, based on the limit for select system calls. With the shared memory Lock Manager that came out in ClearCase 4.1 for Solaris and in ClearCase 4.2 for HP, the limit is based on virtual memory, so you can make it as large as you want. To calculate the parameter value you should use, we recommend writing a script to collect data periodically (say every half hour) for a week that counts the total number of db_server and vobrpc_server processes on the system. Take the maximum number of processes seen in any one sample and set the -u parameter value to two times that maximum number. The -u value is the most difficult to formulize; it's best to determine what your system requires by running the monitoring steps.
- The -q parameter determines how many lock requests can be queued by the Lock Manager at any one time. The default is 1024. Again, you'll see poor end-user response and "database timed out" messages in the log file if this parameter is set too low. To resolve the problem, we recommend you increase the -q parameter to up to five times the value of the -u parameter (although in actuality there's no upper bound), because the db_server process usually requests a lock for five database files in one request.

To change the value of these parameters on a UNIX machine, go to the $ATRIAHOME/etc/atria_start script and change the values in the line that looks something like this:

```
${ATRIA}/etc/lockmgr${LOCKMGR_ALIAS_OPT} -q 1024 -u 256 -f 256
   >> $ADM_DIR/log/lockmgr_log 2>$1
```

Then restart ClearCase on the machine. (Note that if you install a patch later on, these values may get overwritten unless you modify them again in the release area so that they get propagated to all the clients that may need them.)

To change the parameters on a Windows machine, use the Registry Editor to look for the LockMgrCmdLine value in the path Hkey_Local_Machine\Software\Atria\ClearCase\CurrentVersion. If the value isn't there (it won't be unless you've previously modified it), you can create it. Set the parameters in the string -a almd -u 256 -f 256 -q 1024 and make sure the type is REG_SZ. Once you've modified the parameters, restart ClearCase on the machine.

Don't Worry, Be Happy

This article has given you a place to start in addressing a number of the most common problems that Rational ClearCase users run into. If you find yourself facing one of these problems, try following the suggestions for resolving it given here. But if that doesn't work, don't hesitate to call on Rational Technical Support. They'll either help you out or spin your problem off to us in CAG. One way or the other, we'll get it handled and you can get on with your tasks.

***NOTE:** This article was originally published on Rational Developer Network, the learning and support channel for the Rational customer community. If you are a Rational customer and have not already registered for your free membership, please go to www.rational.net.

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!