# ClearCase Support: How to Setup ClearCase Registry

# For each ClearCase environment, one server must be defined as a ClearCase Registry Server. This server must always be online so that all the other ClearCase servers and clients can attain the registry information at any time. The registry information is the list of available VOBs and Views that each computer can access along with the exact VOB and View storage locations and server names on the network.

# Typically, all registry files are located by default in this directory on UNIX, but it may be configured for somewhere else:

# /usr/adm/atria/rgy

# The Registry Server will have this file with only one word in the file.

# $ cat /usr/adm/atria/rgy/rgy\_svr.conf

# master

# All other ClearCase Servers and Clients will need to be configured with the Registry Server name.

# $ cat /usr/adm/atria/rgy/rgy\_hosts.conf

# SERVER\_NAME

# The SERVER\_NAME must be the Registry Server’s name, such as "solaris.server.com".

# Additionally, all ClearCase Servers and Clients must know the registry's name in this config file.

# $ cat /usr/adm/atria/rgy/rgy\_region.conf

# REGISTRY\_NAME

# The REGISTRY\_NAME must be the Registry’s name, such as "clearcase\_registry" or "xyz".

# Finally, to reset the Registry password, use this command as root when logged into the Registry Server:

# # /usr/atria/etc/rgy\_passwd

# ClearCase Support: Configuring ClearCase License Server

Each ClearCase server and client must be configured to know which computer is the ClearCase License Server. Here is the text file that each computer with ClearCase installed on it needs to have defined.

|  |
| --- |
| **$ *cat /usr/adm/atria/config/license\_host* SERVER\_NAME** |

The SERVER\_NAME must be the License Server's name, such as "license.server.com" or a local alias to the license server. An alias is advisable to minimize down times.

**ClearCase Support: Finding ClearCase Log and Cache Files**  
Ever try to execute a ClearCase command, but it returned an error message saying that more information is in the log files? And ClearCase only gives you the name of the log file, but not the location of the log file? Yes, well you are just like everyone else. Try these locations for ClearCase log files and cache files.

In UNIX, the default locations are:

|  |
| --- |
| **/appls/clearcase/log /appls/clearcase/cache /var/adm/atria/log /usr/adm/atria/log** |

In Windows, you will have to run this program:

|  |
| --- |
| **Start -> Programs -> Rational Software -> Rational ClearCase -> Administration -> Log Browser** |

Alternatively, you can see the log files of any computer from any computer using this command, assuming you have permissions to do so:

|  |
| --- |
| ***cleartool getlog -graphical –host* MACHINE\_NAME** |
| **ClearCase Support: Displaying ClearCase Version Numbers**  This is the ClearCase command to determine what version of ClearCase that you are running as well as all the ClearCase patches that are installed too (if any). This command also displays the MVFS version, cleartool version, db\_server version, and VOB database schema version number too. If MultiSite is installed, then this command will also display the MultiSite version and MultiSite patches that are installed (if any).   |  | | --- | | ***cleartool –ver*** |   Each line in the output will be a separate software item with the item's version number. |

# ClearCase Support: How to Display ClearCase License Info

There is a simple command to gather information about your ClearCase and MultiSite licenses. The command is:

|  |
| --- |
| ***clearlicense*** |

You may want to save the output to a file, since the output can be quite large. This output of this command will display the total number of licenses, number of available licenses, number of denied license requests, and who is using what licenses for each type of license (e.g. ClearCase or MultiSite).

**ClearCase Support: Eclipsed Warning Message**

Another common ClearCase warning message is when an element is "eclipsed". This is caused when there are two files (or directories) with the same name in a single directory. You may ask how this is possible. This problem is typically caused when a View is configured not to see or access an element, so the user creates a new file of the same name in the same directory. Therefore in ClearCase, it is possible to have a single element in the VOB, while the View can have a different element with the same name in the same directory. The View’s element always has priority, so the user would only access their View private element, yet the user would encounter this warning message: 

|  |
| --- |
| **$ *cleartool ls* test.txt test.txt@@ [eclipsed]** |

Note, that there are two files in this directory with the same name. The first file is the View private element, and the second file is an element in the VOB. Thus, the VOB’s element has been overshadowed or "eclipsed" by the higher priority of the View private file.   
  
The solution for this problem is quite easy. Just rename the View private element to a unique name. You may need to copy data over between both files, but at least no data was lost.

**ClearCase Support: How to Create a ClearCase VOB**  
Here are the procedures for creating a VOB in ClearCase on UNIX. 

1. Log in as root or your VOB administration account.
2. Check the disk space on all of your VOB storage directory partitions on all of your VOB servers. I recommend using the VOB storage directory partition with the most available disk space.
3. Go to the partition with the most available disk space. Your account must have write access to this partition.
4. Create a VOB using this command.

|  |
| --- |
| ***cleartool mkvob –tag* VOB\_NAME *-public –password* REGISTRY\_PASSWORD *-host* VOB\_SERVER\_NAME *-hpath* VOB\_PATH *–gpath* VOB\_PATH VOB\_PATH** |

1. The VOB\_NAME is a unique VOB name, such as "/vob/test". You may want to run "cleartool lsvob" to list all existing VOBs to make sure that a VOB with the same name does not already exist.   
     
   The REGISTRY\_PASSWORD is the ClearCase registry password that was set when the ClearCase environment was first configured. This ensures only administrators can create new VOBs.   
     
   The VOB\_SERVER\_NAME is the name of the VOB server that the VOB storage directory partition resides on.   
     
   Finally, the VOB\_PATH is the new VOB storage directory that you want created, such as "/vob\_partition/test" if you are creating a VOB called "/vob/test/".
2. Next, inspect the new VOB’s permissions using this command:

|  |
| --- |
| ***cleartool describe vob:*VOB\_NAME** |

|  |
| --- |
| ***cleartool describe vob:/vob/test*** |

1. Change the new VOB’s owner and group permissions. Note that most of these arguments below are optional.

|  |
| --- |
| ***cleartool protectvob -force -chown* VOB\_OWNER *-chgrp* PRIMARY\_GROUP *-delete\_group* GROUP1,GROUP2,... *-add\_group* SECONDARY\_GROUP1, SECONDARY\_GROUP2,... VOB\_PATH** |

1. The VOB\_OWNER should be your ClearCase administration account.   
     
   The PRIMARY\_GROUP is the primary UNIX group that you want to own the VOB. This is typically the same primary group that the VOB’s users belong to.   
     
   Next, GROUP# arguments are the secondary groups that you want to remove, such as any administration groups.   
     
   The SECONDARY\_GROUP# arguments are the secondary groups that you want to add to the VOB for access for secondary users’ groups.   
   Finally, the VOB\_PATH is the new VOB storage directory that you created previously.
2. Change the permissions of the data in the VOB using the "cleartool protect …" command. See this [webpage](http://www.philforhumanity.com/ClearCase_Support_5.html) for more information on this command.
3. If the VOB needs to be accessed in multiple regions, such as Windows, you will need to register the newly created UNIX VOB on Windows too. You can use the Region Synchronizer in the ClearCase Home Base to do this step for Windows.
4. Finally, setup the VOB with whatever [triggers](http://www.philforhumanity.com/ClearCase_Support_18.html) or directory structure your users may require.

Don't forget to tell your users that the new VOB is available to start using.

**ClearCase Support: How to Make a VOB**  
Here are the procedures to create a new VOB in ClearCase on UNIX.

1. The first step, before creating any VOB, is to collect this information that you will need:
   * **New VOB Tag:** You will need the new VOB's name.
   * **Data Owner:** You will need the manager's name of who will be the data owner for the VOB.
   * **Project Name:** You will need the project that this new VOB is being created for.
   * **Permissions:** Either the new VOB can have the same permissions as an existing VOB, or the (primary and secondary) groups and directory permissions need to be specified ahead of time.
   * **VOB Size:** You will probably need the estimated VOB size after a year or two of data growth.
2. Second, find a VOB storage directory that has sufficient disk space to store the estimated VOB size from the previous step. I typically just run "**cleartool lsvob**" to find all of the VOB storage directories. Then, I run "**df –k VOB\_STORAGE\_DIRECTORY**" on each VOB storage directory to find the storage directory with the most disk space available.
3. Log into the VOB Server of the chosen VOB storage directory as root using this command:

|  |
| --- |
| ***rlogin SERVER\_NAME -l root*** |

1. I typically determine the VOB Server's name by running "**cleartool describe vob:VOB\_TAG**", but this involves knowing a VOB that exists on that same VOB storage directory.
2. Go to the VOB storage directory chosen.

|  |
| --- |
| ***cd VOB\_STORAGE\_DIRECTORY*** |

1. Create or make a new VOB using this command:

|  |
| --- |
| ***/usr/atria/bin/cleartool mkvob NEW\_VOB\_TAG -public -password REGISTRY\_PASSWORD -host SERVER\_NAME -hpath NEW\_VOB\_STORAGE\_PATH -gpath NEW\_VOB\_STORAGE\_PATH NEW\_VOB\_STORAGE\_PATH*** |

As the VOB's comment, be sure to add the VOB's data owner and project information for future reference. Trust me, you will need this information in a couple of years.

1. Now that the VOB has been created, set the permissions of this new VOB. First, you can see the existing permissions of the VOB at any time with this command:

|  |
| --- |
| ***cleartool describe –vob vob:VOB\_TAG*** |

1. To change the VOB owner and primary group, run this command:

|  |
| --- |
| ***cleartool protectvob –force –chmod VOB\_OWNER -chgrp PRIMARY\_GROUP VOB\_STORAGE\_PATH*** |

1. You can also add and remove secondary groups using this command too:

|  |
| --- |
| ***cleartool protectvob –force –add\_group GROUP1,GROUP2 -delete\_group GROUP3,GROUP4 VOB\_STORAGE\_PATH*** |

1. Next, set a View, mount the VOB, enter the VOB, and change the permissions of the directories in the VOB. I recommend permission numbers of either 770 or 775.

|  |
| --- |
| ***cleartool setview VIEW\_NAME cleartool mount VOB\_TAG cd VOB\_TAG cleartool protect –chown VOB\_OWNER -chgrp PRIMARY\_GROUP -chmod PERMISSION\_NUMBER -recurse .*** |

1. Using the "Region Synchronizer" in the "ClearCase HomeBase" on any Windows computer that has ClearCase installed, synchronize or add this new VOB so any Windows computers can access it.
2. At this point the VOB is ready to be used by any user, but I recommend first installing all required ClearCase triggers that your administrators' require. This step is optional.
3. I strongly recommend mounting this new VOB on all shared servers. This step is optional too.
4. Finally, alert your users that the VOB is ready to be used once mounted on their machines.

That's it! You are done creating a ClearCase VOB. You may want to test using the newly created VOB.

# ClearCase Support: How to Copy VOB Data

Have you ever wanted to copy ClearCase data even from one VOB to another VOB? In other words, copy an element (including its entire version tree, branches, labels, merge arrows, etc.) or even a set of elements (an entire directory of ClearCase data). Here is how to do it:   
  
In this example, let’s say we want to copy a directory, at this location */vob/prototypes/ABC*, to the root directory of another VOB, called */vob/tools*, that we will call the destination VOB. 

1. First, log into the account that is the VOB owner of the destination VOB. You may need to contact your local administrators for this access.

|  |
| --- |
| ***su* root** |

1. Set a View that is owned by this account.

|  |
| --- |
| ***cleartool setview* root** |

1. Next, go to the directory where the data source resides.

|  |
| --- |
| ***cd /vob/prototypes*** |

1. Prepare the data, the *ABC* directory, to be copied over by exporting it with this command. No changes will be made to the "/vob/prototypes" VOB using this command.

|  |
| --- |
| ***clearexport\_ccase –r ABC*** |

1. The "-r" argument will recursively export all the data in the *ABC* directory and all sub-directories too. This command will create a "cvt\_data" data file in the current directory.
2. Go to the destination directory:

|  |
| --- |
| ***cd /vob/tools*** |

1. Finally, import the data file using this command.

|  |
| --- |
| ***clearimport -v /vob/prototypes/cvt\_data*** |

1. The "-v" argument will import the data in verbose mode. Again, no changes will be made to the "/vob/prototypes" VOB using this command either. You may have to un-checkout the current directory if this command failed for whatever reason, especially if you are going to try re-executing this command.

# ClearCase Support: How to Copy Data into a VOB

Ever need to copy a lot of data into a VOB? The command to add one element at a time is very tedious and could take a long time if you are trying to transfer a huge amount of data. Instead, try using these automated procedures to import data into a VOB.   
  
NOTE: ClearCase v6 or 2003 has introduced a SINGLE new command that does exactly what these steps. You can find the command [here](http://www.philforhumanity.com/ClearCase_Support_78.html).   
In this example, let’s say we want to copy a directory, at this location */home/userid/ABC*, to the root directory of a VOB, called */vob/tools*. 

1. First, log into the account that is the VOB owner of the destination VOB. You may need to contact your local administrators for this access.

|  |
| --- |
| ***su* root** |

1. If you can not attain VOB owner access, you can alternatively run this [UNIX script to recursively make elements](http://www.philforhumanity.com/ClearCase_Support_40.html) in a VOB.
2. Set a View that is owned by this account.

|  |
| --- |
| ***cleartool setview* root** |

1. Next, go to the directory where the data is located.

|  |
| --- |
| ***cd /home/userid*** |

1. Prepare the data, the *ABC* directory, to be copied over by exporting it with this command. No changes will be made to the actual data using this command.

|  |
| --- |
| ***clearexport\_ffile –r ABC*** |

1. The "-r" argument will recursively export all the data in the *ABC* directory and all sub-directories too. This command will create a "cvt\_data" data file in the current directory.
2. Go to the destination directory:

|  |
| --- |
| ***cd /vob/tools*** |

1. Finally, import the data file using this command.

|  |
| --- |
| ***clearimport -v /home/userid/cvt\_data*** |

1. The "-v" argument will import the data in verbose mode. Again, no changes will be made to the original data using this command either. You may have to un-checkout the current directory if this command failed for whatever reason, especially if you are going to try re-executing this command.

**ClearCase Support: How to Create a View in ClearCase**  
this command to create a new View is relatively straight forward.

|  |
| --- |
| ***cleartool mkview –tag* VIEW\_NAME *-host* VIEW\_SERVER\_NAME         *-hpath* VIEW\_PATH *-gpath* VIEW\_PATH VIEW PATH** |

There are only 3 arguments that need to be filled out.   
  
First, the VIEW\_NAME argument must be a unique View name. You can run this command to verify that another View does not already have this View name. 

|  |
| --- |
| ***cleartool lsview | grep* VIEW\_NAME** |

If the View name is already in use, then you will just get an error message.   
  
Second, the VIEW\_SERVER\_NAME argument must be the name of one of your View servers. You must have at least one View server. This View server must already be configured as a View server ahead of time.   
  
Third, the VIEW\_PATH argument is the View’s storage directory. This path is the most tricky argument to determine. First, you may have several View storage directories on different View servers, but a single View must reside on a single View storage directory. You can load balance all Views on different View servers and/or different View storage directories if performance or disk space is an issue. Second, you need to make sure the View storage directory is writable by all users and used by all users. Third, the View storage directory must be a shared directory that all computers and users can access. Finally, there must always have plenty of disk space available for View growth. A new View is under a megabyte in size, however compiling code or generating source code can result with gigabytes of data in the Views in a very short period of time.   
  
Otherwise, if your ClearCase Adminstrators have already defined a single default View's storage directory or View path, then you can use this command instead.

|  |
| --- |
| ***cleartool mkview –tag* VIEW\_NAME *-stgloc -auto*** |

This command is a lot shorter and removes the possibility of users making mistakes.

1. Run the **cleartool** **describe** command, and note the view's UUID from the list of views referenced by a VOB:

cleartool describe –long vob:*vob\_tag*

If the view tag still exists, you can use **lsview** –long to find the view UUID.

1. If the view tag still exists, remove it from the registry. In the Rational ClearCase Administration Console, you can use the **View Tags** node for the tag's regions to remove view tags. You can also use the following command:

cleartool rmtag –view *view\_tag*

1. Unregister the view. In the Rational ClearCase Administration Console, you can use the **View Object** node to remove a view object. You can also run the following command, using the view's UUID from Step [1](https://www.ibm.com/support/knowledgecenter/SSSH27_7.1.2/com.ibm.rational.clearcase.cc_admin.doc/topics/t_viewadm_rmview.htm?view=kc#t_viewadm_rmview__step_get_uuid):

cleartool unregister –view –uuid *uuid*

1. Remove references to this view from each VOB that holds them. In the Rational ClearCase Administration Console, you can use the **Referenced Views** subnode of a VOB storage node to remove a view's records from a VOB. You can also run the following command, using the view's UUID from Step [1](https://www.ibm.com/support/knowledgecenter/SSSH27_7.1.2/com.ibm.rational.clearcase.cc_admin.doc/topics/t_viewadm_rmview.htm?view=kc#t_viewadm_rmview__step_get_uuid):

cleartool rmview –all –uuid *uuid*

**How can you checkout (reserved) an element that is already checked out (reserved) on the same branch in another IBM® Rational® ClearCase® snapshot or dynamic view.**

Cause: One of two common scenarios is associated with this problem:

1. An element is checked out (reserved) in a view that still exists and you are unable to checkin the element because:
   1. The checkout is in a snapshot view (and the user is unavailable to perform the checkin or cancel the checkout).
   2. The view is inaccessible.
2. The view has been removed incorrectly leaving checked out references in the VOB.

## Answer

Review the [About reserved and unreserved checkouts](https://publib.boulder.ibm.com/infocenter/cchelp/v7r1m0/index.jsp?topic=/com.ibm.rational.clearcase.dev.doc/topics/cc_dev/about_res_unres_cos.htm)section of *the IBM Rational ClearCase Developing Software*manual for a detailed explanation about reserved and unreserved checkouts.

There are two solutions to this problem:

**1. Uncheckout the element(s)**

* If the view exists and you do not care about losing the changes in the view
* If the view no longer exists  
  See [**technote 1122515**](http://www-1.ibm.com/support/docview.wss?rs=984&uid=swg21122515) which provides instructions on how to remove all the checkout references from a view using the*cleartool rmview -uuid* command.

**2. Change the checkout status of the elements(s) from reserved to unreserved**  
The element that is checked out reserved can be set to unreserved.

Review the *ClearCase Command Reference Guide* on the topic of **unreserve** (cleartool man unreserve) for more information.

There are two scenarios that use slightly different methods to achieve the same results. These scenarios depend on the availability of the view and the ability of a user to start and or set into the view itself.

[View Available](http://www-01.ibm.com/support/docview.wss?uid=swg21129391#Table1) -- The view can be accessed and the checkout status can be directly changed

[View Unavailable](http://www-01.ibm.com/support/docview.wss?uid=swg21129391#Table2)--The view cannont be accessed directly.

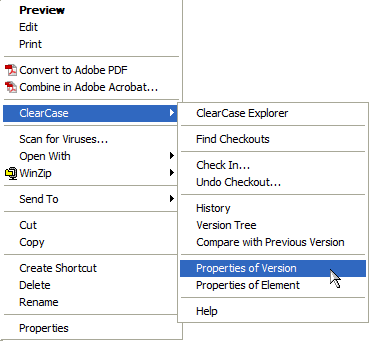
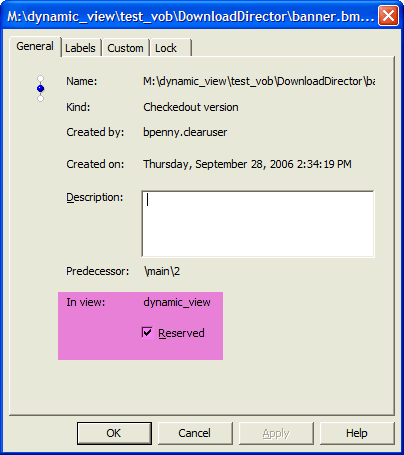
**VIEW IS AVAILABLE**

If you can start and set into the view that has the checkouts, but do not want to checkin the data and do not want to lose the changes that were made in that view, you can simply change the reserve status directly in one of two ways:

[GUI](http://www-01.ibm.com/support/docview.wss?uid=swg21129391#Table3)

[Command Line](http://www-01.ibm.com/support/docview.wss?uid=swg21129391#Table4)

**GUI**

1. From either ClearCase Explorer or Windows Explorer, right click on the element and select **Properties of Version**  
     
   Windows Explorer example:  
    
2. Uncheck the **Reserved** check box and click OK to make the version checkedout unreserved.

**Command line:**

1. Open a command prompt (**Start > Run** and type *cmd*)
2. Set into the view and cd down to the directory where the element is located.  
     
   Example:  
     
   M:\>**cd my\_view\my\_vob\docs**
3. Type **cleartool unreserve <element\_name>** to change the reserve status of the checkout.   
     
   Example:  
     
   M:\my\_view\my\_vob\docs>**cleartool unreserve bar.txt**  
   Changed checkout to unreserved for "bar.txt" branch "\main".

**VIEW UNAVAILABLE**

This example would be an option for a snapshot view that is currently disconnected from the network which has changes that can't be lost.

Once the element is changed to an unreserved checkout, another view can then check the element out reserved, make changes, and then check it back in.

**Command Line:**

1. Open a command prompt (**Start > Run** and type *cmd*)
2. CD into a view down to the directory where the element is located.  
     
   Example:  
     
   M:\>**cd my\_view\my\_vob\docs**
3. Type **cleartool lsco -long <element\_name>**. This will show which view has this element checked out reserved.   
     
   Example:  
     
   M:\my\_view\my\_vob\docs>**cleartool lsco -long bar.txt**  
   31-Jul-02.14:17:31     user\_name.group@hostname  
   checkout version "bar.txt" from \main\0 (reserved)  
   by view: user\_view\_with\_checkout  
   ("**hostname:D:\Views\user\_name\user\_view\_with\_checkout.vws**")
4. Type **cleartool unreserve -view <view\_path> <element\_name>** to change the reserve status of the checkout.   
     
   Example:  
     
   M:\my\_view\my\_vob\docs>**cleartool unreserve -view**  
   **hostname:D:\Views\user\_name\user\_view\_with\_checkout.vws bar.txt**  
   Changed checkout to unreserved for "bar.txt" branch "\main".
5. Type **cleartool lsco -long <element\_name>** again to verify the file is listed as an unreserved checkout now.   
     
   Example:  
     
   M:\my\_view\my\_vob\docs>**cleartool lsco -long bar.txt**  
   31-Jul-02.14:17:31     user\_name.group@hostname  
   checkout version "bar.txt" from \main\0 (unreserved)  
   by view: user\_view\_with\_checkout  
   ("hostname:D:\Views\user\_name\user\_view\_with\_checkout.vws")

**ClearCase Support: How to Delete a View in ClearCase**  
The command to delete a new View is also relatively straight forward.

|  |
| --- |
| ***cleartool rmview –tag* VIEW\_NAME** |

**WARNING: Deleting the View will delete all View private files including checked out files, compiled files, derived objects, config specs, and config records in the View. Use this command with caution. Check in all files that you want to keep into the VOB before deleting the View.**   
  
If a View is partially deleted or corrupted, then the above command might not work. In this case, you will have to run these three steps.

First, find the View’s storage directory with this command:

|  |
| --- |
| ***cleartool lsview | grep* VIEW\_NAME** |

Second, manually delete the entire View’s storage directory that was listed in the output of the previous command.   
Third, remove the View’s tag from the Registry Server, so the View no longer appears available or active with this command:

|  |
| --- |
| ***cleartool rmtag –view* VIEW\_NAME** |
| **ClearCase Support: Finding and Setting Views**  I'm often asked to find out Views, so here are a few ClearCase commands to help find and set Views.  First, to list all the Views from every user, run this command:   |  | | --- | | ***cleartool lsview*** |   Second, to search every View belonging to a certain user (assuming the user's id is in the View's name or path), run this command:   |  | | --- | | **In UNIX: *cleartool lsview | grep USER\_ID*  In Windows: *cleartool lsview | find "USER\_ID"*** |   Third, to find more information about any View, run this command:   |  | | --- | | ***cleartool lsview -long VIEW\_NAME*** |   Next, to set a View from the command line, run this command:   |  | | --- | | ***cleartool setview VIEW\_NAME*** |   Finally, to display the ClearCase View that the current command line is in, run this command:   |  | | --- | | ***cleartool pwv*** |   In short, this command is called "pwv" and that stands for "Path of Working View".  **ClearCase Support: Finding Old Views**  I was recently asked a question on how to determine when a ClearCase View was created and last used. This is easy to do with this command:   |  | | --- | | ***cleartool lsview -pro*VIEW\_NAME** |   For example:   |  | | --- | | **$ *cleartool lsview -pro VIEW\_NAME***  **VIEW\_NAME VIEW\_STORAGE\_DIRECTORY**  **Created 23-Jun-07.11:10:11 by USERID.GROUP@MACHINE**  **Last modified 04-May-08.14:57:04 by USERID.GROUP@MACHINE**  **Last accessed 04-May-08.17:57:34 by USERID.GROUP@MACHINE**  **Owner: SITE/USERID : rwx (all)**  **Group: SITE/GROUP : r-x (read)**  **Other: : r-x (read)**  **Additional groups: SITE1.GROUP1 SITE2.GROUP2** |   I recommend that you automate the checking of all ClearCase Views with this [script](http://www.philforhumanity.com/ClearCase_Support_59.html).  **ClearCase Support: Deleting Checkouts from Old View** Quite often a ClearCase View becomes corrupted or deleted, yet it still has checked out files in a VOB. Since the View does not exist or is no longer usable, the only solution is to delete all references of that View from a VOB. This method will uncheckout all the files for a single View in a specific VOB. This is also known as deleting ClearCase View references.   First, you must find the Unique Unix Identification Number (uuid) of that View in the VOB in question with this command:  **cleartool describe -long vob:*VOB\_NAME***  This will display a lot of information. In the section labeled "VOB holds objects from the following views:", you will see lines that look like this:   |  | | --- | | ***MACHINE\_NAME:VIEW\_PATH [uuid UUID\_NUMBER]*** |   For instance,   |  | | --- | | ***PHILS\_COMPUTER:C:\view\viewname.vws [uuid f03ee621.35da11da.95f9.00:30:6d:48:ad:10]*** |   In this example, the uuid is "f03ee621.35da11da.95f9.00:30:6d:48:ad:10".   Next, to delete all references of this View in a certain VOB, you will need to run this command. Note that there is no way to undo these changes, unless you made backups of the VOB and View.   **cleartool rmview -vob *VOB\_NAME* -uuid *UUID\_NUMBER***  For our example, the command would look like this:   |  | | --- | | ***$ cleartool rmview -vob /vob/test -uuid f03ee621.35da11da.95f9.00:30:6d:48:ad:10 Remove view "C:\view\viewname.vws" references from VOB? [no] y Removed references to view "C:\view\viewname.vws" from VOB "/vob/test".*** |   **ClearCase Support: Common ClearCase Error when Running Cleartool Commands**  Here is another common error message that I am asked about a lot.    |  | | --- | | **cleartool> lsview | grep username *cleartool: Error: No matching entries found for view tag "|". cleartool: Error: No matching entries found for view tag "grep". cleartool: Error: No matching entries found for view tag "username".*** |   This error message, "cleartool: Error: No matching entries found…", is caused because the user assumed that the cleartool prompt is a DOS or UNIX prompt and can run DOS or UNIX command from the cleartool prompt. However, this is not true. You can not use any DOS or UNIX commands, in the case the "| grep username" arguments, on the cleartool prompt.   A simple workaround is to exit out of the cleartool prompt, and run the command directly from DOS or UNIX, like this:   |  | | --- | | **$ cleartool lsview | grep username** |   That should solve the user's problem.  **ClearCase Support: Creating Elements**  Ever wonder how to add a file to source control in ClearCase from the command line. Here's how. First, check out the directory where you want to create a new file in a VOB. Second, run this command:   |  | | --- | | ***cleartool mkelem -ci -nc FILE\_NAME*** |   Finally, check in the new element’s parent directory.  If you want to create a new empty directory in the VOB, then run this command instead:   |  | | --- | | ***cleartool mkelem -ci -eltype directory -nc DIRECTORY\_NAME*** |   **ClearCase Support: How to Add an Element Type**  Ever try adding a new file under source control in ClearCase, however you can not because the element type does not exist?   Well, I have this problem whenever I add Rational Rose RealTime files into a VOB for the first time, especially with newly created VOBs. You would assume that IBM/Rational would support, as a default, their other tools in ClearCase, however they do not.   Here is the command that I use to add the Rational Rose RealTime element type to a ClearCase VOB. I think you need to be the VOB owner to run this command and in the VOB too.   |  | | --- | | **cleartool mkeltype -supertype text\_file -manager petalrt\_file\_delta -c "RoseRT files" rosert\_unit** |   Alternatively, if you have an element type in one VOB but another VOB does not have the element type, then you can copy the element type from one VOB to another VOB using this command.   |  | | --- | | **cleartool cptype eltype:ELEMENT\_TYPE\_NAME@SOURCE\_VOBeltype:ELEMENT\_TYPE\_NAME@DESTINATION\_VOB** |   Again, I think you need to be the VOB owner to run this command. ClearCase Support: Recursively Adding to Source Control This is the new ClearCase command for installing a large number of UNIX files and directories into the VOBs.   |  | | --- | | **$ clearfsimport -nsetevent [-rmname] -recurse [-mklabel LABEL\_NAME] SOURCE\_DATA DESTINATION\_DIRECTORY** |   You can even use this command to re-install a newer set of files to the same location in ClearCase.   Also, don't forget to change the [permissions](http://www.philforhumanity.com/ClearCase_Support_5.html) of the new elements that were installed, if necessary. |

**ClearCase Support: How to Delete a ClearCase Element**  
I am often asked how to remove an element (file or directory) from a ClearCase VOB. By removing an element, realize that all versions of the element will either become permanently erased or not easily accessible/viewable by default (more on that later).   
  
Let's say you wanted to delete the file element /vob/test/a/b.txt.   
  
First, you need to check out the parent directory. In this example, check out /vob/test/a manually.   
  
Second, decide if you want to permanently delete all versions of the element or if you just want the file to not be accessible from the latest version of the parent directory from now on.   
  
To permanently delete all versions of the element, run this command:

|  |
| --- |
| **cleartool rmelem /vob/test/a/b.txt** |

ClearCase will ask you for confirmation, since this is a dangerous command. **I recommend using this command to only free up disk space of elements that you will NEVER POSSIBLY need again. Use it at your own risk!** You may need to be an administrator to do this command depending on permissions, and you may need to force this command to run using the "-force" argument. You can delete multiple elements at the same time by just appending more elements to the end of the line or by using wildcards.   
  
On the other hand, I recommend to just delete the element from the latest directory listing by using this command:

|  |
| --- |
| **cleartool rmname /vob/test/a/b.txt** |

Using this method, the file is still in the VOB, but the file is only accessible using older versions of the parent directory. This is great for cleaning up the VOB without permanently deleting anything. You can also delete multiple elements at the same time using this command too.   
  
Finally with either command, you will need to check in the parent directory.

# ClearCase Support: How to Synchronize a UNIX View to Windows in UNIX

Do you create ClearCase Views on UNIX and then use the ClearCase Region Synchronizer on Windows to import the UNIX View into the Windows Region? This enables UNIX Views to be used on a Windows computer; however, this process is extremely slow and manual. It is slow because the Region Synchronizer needs to collect the entire list of all UNIX Views and exclude the UNIX Views that have already been synchronized to Windows. Then you have to manually find the UNIX View that you want to synchronize. All of this takes time. If you have a lot of UNIX Views, then this process can take a long time.   
  
Well, if you synchronize the UNIX View from the UNIX command line, then this step is much faster and even automatable. Here is how to do this.   
  
Assume that you have already created a UNIX View called "test". You will need the View UNIX storage directory, for example "/viewstore/view\_directory1/test.vws". You will also need to know the View Server’s name that hosts that View storage directory, for example "view\_server1".   
  
You will also need several pieces of information from the ClearCase Windows Region. First, you need to know the Windows Region name, such as "win\_region". Additionally, you will need the Windows path to access the UNIX Views storage directory, such as "\\view\_server\view\_directory1\test.vws". However on UNIX this Windows path would need to have each backlash with an extra backlash like this "\\\\view\_server\\view\_directory1\\test.vws".   
  
With all of this information, this is the UNIX command to synchronize a UNIX View onto the Windows region. You can even run this command on Windows if you want. 

|  |
| --- |
| **cleartool mktag -view -tag VIEW\_NAME -reg WINDOWS\_REGION\_NAME -host VIEW\_SERVER\_NAME**  **-gpath WINDOWS\_VIEW\_STORAGE\_DIRECTORY UNIX\_VIEW\_STORAGE\_DIRECTORY** |

For example, with all of our example data filled out, the final command would look like this: 

|  |
| --- |
| **cleartool mktag -view -tag test -reg win\_region -host view\_server1**  **-gpath \\\\view\_server\\view\_directory1\\test.vws /viewstore/view\_directory1/test.vws** |

You may want to re-direct the output of this command to standard error or /dev/null, since this command usually results with a lot of warning messages, since ClearCase is unable to validate the Windows View Storage directory from UNIX. You can just append "2>&1 /dev/null" to the end of the command.

**ClearCase Support: How to Duplicate a VOB**

Here are the procedures to move or duplicate a ClearCase UNIX VOB from one VOB server to another VOB server. If the VOB server is not changing, then the VOB can be [moved](http://www.philforhumanity.com/ClearCase_Support_42.html) instead, but these procedures will work too. 

1. Determine the VOB's current location and the VOB server that hosts the VOB by using this command.

|  |
| --- |
| ***cleartool describe [-long] vob:VOB\_TAG*** |

1. Determine the VOB's size by using this command.

|  |
| --- |
| ***cleartool space VOB\_TAG*** |

1. Determine the server and location that the VOB is being moved to and if the VOB has sufficient disk space in the new location by using this command. You may need the remote ClearCase administrator to do this step.

|  |
| --- |
| ***df –k NEW\_VOB\_PARTITION*** |

1. Schedule a VOB shutdown. Wait until the start of the shutdown period before continuing these procedures.
2. Verify that the VOB was recently and successfully backed up as a precaution.
3. Log into the VOB's server as root.

|  |
| --- |
| ***rlogin SERVER\_NAME -l root*** |

1. Lock the VOB.

|  |
| --- |
| ***cleartool lock vob:VOB\_TAG*** |

1. Dump the VOB. Dumping the VOB may take a very long time depending on the size of the VOB. You can skip this step if you are duplicating the VOB to a server with the identical OS version and ClearCase version. Otherwise, dumping the VOB requires a temporary disk space to store an extra copy of the entire VOB. If you do not have enough disk space, you can use the "-to" argument to reference a temporary location.

|  |
| --- |
| ***cleartool reformatvob -dump [-to DUMP\_DIRECTORY] VOB\_STORAGE\_DIRECTORY*** |

When this command finishes, you will find two files or directories called "db.reformat" and “dump.done" in the VOB's storage directory or dump directory. If these files do not exist, then the dump was not successful.

1. Compress the dumped VOB data for easy transport.

|  |
| --- |
| ***cd VOB\_PARTITION tar –cf - ./VOB\_STORAGE\_DIRECTORY | /usr/atria/etc/Gzip > TEMP\_DIRECTORY/VOB\_NAME.tar.gz*** |

1. If you created the "dump.done" file in another directory, then you will need to compress this data too.

|  |
| --- |
| ***cd DUMP\_DIRECTORY tar –cf - ./DUMP\_SUB\_DIRECTORY | /usr/atria/etc/Gzip > TEMP\_DIRECTORY/VOB\_NAME\_dump.tar.gz*** |

1. Re-load or restore the VOB Database back to normal. This command should unlock the VOB too. Skip this step if you skipped the dumping of the VOB because you are duplicating the VOB to a server with the identical OS version and ClearCase version. If you do skip this step, then you must manually unlock this VOB with the command: "**cleartool unlock vob:VOB\_TAG**". Otherwise, load the VOB with this command.

|  |
| --- |
| ***cleartool reformatvob -load VOB\_STORAGE\_DIRECTORY*** |

When done, this will create a temporary DB file with the name "db.TIME\_STAMP", such as "db.10.16". Verify this file exists in the VOB\_STORAGE\_DIRECTORY. You may delete this file after verifying the VOB is working again, but I recommend waiting a few days before deleting it.

1. Test that the VOB is working again.
2. Transfer the dumped and compressed files via fship or ftp. Ask the other ClearCase administrator what machine and how to transfer it.
3. Notify the other ClearCase administrator that the duplicated VOB was shipped.

At this point, the other ClearCase administrator (who could be you) will have to import this duplicated or dumped VOB onto their VOB server. They can follow these procedures. 

1. Log into the VOB's server as root.

|  |
| --- |
| ***rlogin SERVER\_NAME -l root*** |

1. Go to the VOB partition that will host the VOB.

|  |
| --- |
| ***cd VOB\_PARTITION*** |

1. Uncompress the VOB data files.

|  |
| --- |
| ***/usr/atria/etc/Gzip –d TEMP\_DIRECTORY/VOB\_NAME.tar.gz tar –xf VOB\_NAME.tar*** |

1. Load the VOB Database.

|  |
| --- |
| ***cleartool reformatvob –load VOB\_STORAGE\_PATH*** |

1. Delete the temporary timestamp database.

|  |
| --- |
| ***rm –rf VOB\_STORAGE\_PATH/db.TIME\_STAMP*** |

1. Register the VOB on the UNIX Region with this command.

|  |
| --- |
| ***cleartool reg –vob –host HOST\_NAME VOB\_STORAGE\_PATH VOB\_STORAGE\_PATH*** |

1. Create the VOB's UNIX tag.

|  |
| --- |
| ***cleartool mktag –vob –tag VOB\_TAG [-public] -password REGISTRY\_PASSWORD -host HOST\_NAME –hpath VOB\_STORAGE\_PATH VOB\_STORAGE\_PATH*** |

1. Mount the VOB on all UNIX servers.

|  |
| --- |
| ***cleartool mount VOB\_TAG*** |

1. Use the Windows Region Synchronizer to make the VOB available on Windows, and mount the VOB on any Windows servers too.
2. Test the VOB, change the VOB's permissions to the required local configuration, and install all required triggers.
3. Skip this step if the VOB is not MultiSited or synchronized with another VOB replica. Have the master replica of VOB change the host name of the moved VOB replica with this command.

|  |
| --- |
| ***multitool chreplica -nc -host NEW\_HOST\_NAME replica:REPLICA\_NAME@vob:VOB\_TAG*** |

1. Inform all users that the VOB is online and ready to be used.
2. After several days or weeks with the duplicated VOB in use and with no problems, you can delete the compressed files.

You are done duplicating the VOB.

**ClearCase Support: How to Move a VOB**

Here are the procedures to move a ClearCase UNIX VOB from one VOB partition to another VOB partition on the same VOB server. If the VOB server is changing, then the VOB will need to be [duplicated](http://www.philforhumanity.com/ClearCase_Support_43.html) instead of moved. 

1. Determine the VOB's current location and the VOB server that hosts the VOB by using this command.

|  |
| --- |
| ***cleartool describe vob:VOB\_TAG*** |

1. Determine the VOB's size by using this command.

|  |
| --- |
| ***cleartool space VOB\_TAG*** |

1. Determine the location that the VOB is being moved to and if the VOB has sufficient disk space in the new location by using this command.

|  |
| --- |
| ***df –k VOB\_PARTITION*** |

1. Schedule a VOB shutdown. Wait until the start of the shutdown period before continuing these procedures.
2. Verify that the VOB was recently and successfully backed up as a precaution.
3. Log into the VOB's server as root.

|  |
| --- |
| ***rlogin SERVER\_NAME -l root*** |

1. Lock the VOB.

|  |
| --- |
| ***cleartool lock vob:VOB\_TAG*** |

1. Unmount the VOB.

|  |
| --- |
| ***cleartool umount VOB\_TAG*** |

I recommend unmounting this VOB on all servers in another window. If you are unable to unmount the VOB, then kill the process of the user who is currently using the VOB. You can determine which user who is using the VOB by using this command "**fuser –c VOB\_TAG**".

1. Remove the VOB's UNIX and Windows tags using this command.

|  |
| --- |
| ***cleartool rmtag –vob –all -password REGISTRY\_PASSWORD VOB\_TAG*** |

1. Go to the VOB's storage partition. For example, if the VOB's storage directory is "/vob\_store1/vob\_name.vbs", then use this command:

|  |
| --- |
| ***cd /vob\_store1*** |

1. Copy the VOB's storage directory. In this example, we are copying it to the "/vob\_store5" directory.

|  |
| --- |
| ***find ./VOB\_STORAGE\_DIRECTORY –print | cpio –o | (cd /vob\_store5 ; cpio –idumv)*** |

1. When completed, you will have two VOB storage directories for the same VOB. Change the permissions of the old VOB storage directory to "000" with this command.

|  |
| --- |
| ***chmod 000 ./VOB\_STORAGE\_DIRECTORY*** |

1. Re-register the VOB on the UNIX Region with this command.

|  |
| --- |
| ***cleartool reg –vob –replace –host HOST\_NAME VOB\_STORAGE\_PATH VOB\_STORAGE\_PATH*** |

1. Re-create the VOB's UNIX tag.

|  |
| --- |
| ***cleartool mktag –vob –tag VOB\_TAG –replace [-public] -password REGISTRY\_PASSWORD -host HOST\_NAME –hpath VOB\_STORAGE\_PATH VOB\_STORAGE\_PATH*** |

1. Re-mount the VOB on all UNIX servers.

|  |
| --- |
| ***cleartool mount VOB\_TAG*** |

1. Use the Windows Region Synchronizer to make the VOB available on Windows, and mount the VOB on any Windows servers too.
2. Kill the old VOB Server process(es). Run "ps –ef | grep VOB\_NAME". If the old VOB's storage directory is still listed as a running process, then kill the old process.
3. Unlock the VOB.

|  |
| --- |
| ***cleartool unlock vob:VOB\_TAG*** |

1. Test that the VOB works on both UNIX and Windows, maybe even test that the VOB's synchronization is working again too.
2. Inform all users that the VOB is online and ready to be used again.
3. After several days or weeks with the moved VOB in use and with no problems, you can delete the old VOB storage directory manually. You may have to change the permission to "700" first.

You are done moving the VOB.

**ClearCase Support: How to Check ClearCase VOB Size**  
The command to query the size of a VOB is this:

|  |
| --- |
| ***cleartool space* VOB\_NAME** |

For example:

|  |
| --- |
| **$ *cleartool space /vob/test***  **Use(Mb) %Use Directory**  **0.1 0% administration data /vob\_partition/test/admin**  **100.0 0% VOB database /vob\_partition/test/db**  **0.8 0% cleartext pool /vob\_partition/test/c/cdft**  **0.0 0% derived object pool /vob\_partition/test/d/ddft**  **200.0 0% source pool /vob\_partition/test/s/sdft**  **-------- ---- ---------------------------------------------------------**  **300.9 0% Subtotal**  **100234.4 50% Filesystem VOB\_SERVER:/vob\_partition (capacity 200468.8 Mb)**    **Total usage 28-Mar-08.05:00:21 for vob "/vob/test" is 300.9 Mb** |

If you want to capture the disk space for all VOBs, use this command:

|  |
| --- |
| ***cleartool space -avobs*** |

I recommend that you save the output of this command to a file, since the output will be very large if you have a lot of VOBs.

# ClearCase Support: How to Reformat or Defragment a ClearCase VOB

Reformatting a VOB is just like defragmenting a hard drive. The VOB’s Database files will be re-sorted in a more compact and linear order to save disk space and decrease read time. On the other hand, reformatting a VOB may increase the time it takes to write to a VOB, since there will be fewer empty slots available for new data within the VOB’s Database files. Reformatting a VOB is optional unless you are running out of disk space on the partition that the VOB storage directory resides on.   
  
Here are the instructions for reformatting or defragmenting a VOB. 

1. Schedule a ClearCase downtime by asking all users to exit out of their accounts for a few hours or possibly even a day depending on the size of the VOB and the speed of the computer.
2. Log onto the VOB server that has the VOB’s storage directory as either root or the VOB admin account. I strongly recommend doing this on the console of the VOB server, because if anything interrupts the defragmenting of the VOB, then you will have to recover the VOB from backup.
3. Verify that the VOB was recently and successfully backed up just in case.
4. Make sure the VOB server’s temp directory (such as /tmp) has enough disk space to hold an entire copy of the VOB that you are about to reformat. Also make sure that the VOB’s partition also has enough disk space to hold the entire copy of the VOB’s DB. You can find out the size of a VOB and DB by using these [instructions](http://www.philforhumanity.com/ClearCase_Support_26.html). You may be required to move other VOBs from this partition or increase the size of the partition in order to reformat the VOB.
5. Reformat the VOB with this command:

|  |
| --- |
| **cleartool reformatvob –to NEW\_TEMP\_DIRECTORY VOB\_PATH** |

1. This command may make take a few minutes, a few hours, or maybe a day to complete. I typically estimate about 1 hour per gigabyte of the VOB’s entire size. Make sure the temporary directory does not exist before running the command. An example of this command would be.

|  |
| --- |
| **cleartool reformatvob –to /tmp/reformat\_temp\_directory /vob\_partition/test** |

1. When the reformatting has completed, restart ClearCase on the VOB server by executing these [instructions](http://www.philforhumanity.com/ClearCase_Support_21.html). I believe this is necessary to disconnect any ClearCase accounts that did not log off or Views that may still be active.
2. After a few days of the VOB working without any issues, it will be safe to delete the temporary directory created during reformatting.
3. After a few days of the VOB working without any issues, you will also have to manually delete the old DB pool/directory in the VOB’s storage directory. The active version of the DB pool will usually be called "db", so don’t delete this directory. The old DB pool to be deleted will be called "db\_DEFRAG\_DATE" such as "db\_03\_30".   
     
   If something goes wrong with the reformatting of the VOB, you can restore the VOB using this old DB directory or from backup.

**ClearCase Support: How to Clean, Recover, and Defragment ClearCase Views**

ClearCase Views are basically your private workspace to access and make changes to VOBs, however these Views quickly decrease in performance as they grow in size. Additionally, ClearCase Views break often. Therefore, Views need to be periodically cleaned, fixed, and even defragmented.   
  
Here are a set of instructions to help with Views. You can actually pick and choose which steps to follow, since most of these steps are optional if your View is still in working condition. You will need to be logged in as the View's owner for all of the instructions to work, and maybe even in your View. 

1. First, I recommend checking in or unchecking out all elements (files and directories) that are checkedout in your View. I do not recommend having elements checkedout for over a week. You can run these [procedures](http://www.philforhumanity.com/ClearCase_Support_9.html) to find all of your checkedout elements.
2. Second, you can clean up all View private elements in your View. You can run this command to find all View private elements in your View.

|  |
| --- |
| ***cleartool lspriv*** |

1. Copy all View private files that you want to keep out of your View and/or permanently add them to source control. Next, make sure you have no checkedout elements, using the instructions in the first step above. Finally, delete all the remaining View private elements with this command. Please note that this command will delete your checkedout elements too, so be very careful with this command.

|  |
| --- |
| ***cleartool lspriv | xargs rm -rf*** |

1. Now, you should have an empty View.
2. Next, sometimes VOBs are deleted, moved, or renamed. However, Views still think they have View private files in the old VOB location that no longer exists. So, run this command to find these lost files.

|  |
| --- |
| ***cleartool lspriv | grep "DIR-"*** |

1. If any output is produced, then you need to remove this lost View private data files from these obsolete VOB directories.

|  |
| --- |
| ***cleartool recoverview -force -dir DIRECTORY\_NUMBER -tag VIEW\_NAME*** |

1. The directory number is the information between "
2. Next, go to the View's storage directory. Then go to the source sub-directory called ".s". Then go the View's lost and found sub-directory called "lost+found". Delete all files and directories in this lost and found directory. Do not delete this lost and found directory!

|  |
| --- |
| ***rm –rf VIEW\_STORAGE\_DIRECTORY/.s/lost+found/\**** |

1. Now that you have cleaned your View, you now need to clean up the data tables with the follow command. This should drastically improve the speed of large Views that had a lot of cleaning.

|  |
| --- |
| ***cleartool recoverview -force -tag VIEW\_NAME*** |

1. This command may take a long time to run. After this command finishes running, immediately run the same command a second time. If a lot of data tables were cleaned up, then the second time that you run this command will defragment or recover internal disk space of the View. In other words, make the data tables smaller without large data gaps. This should improve performance of the View even more.
2. Finally, force a View refresh on all servers using this command.

|  |
| --- |
| ***cleartool setcs -current*** |

Your ClearCase View should now be much smaller and faster after you have finished cleaning, recovering, and defragmenting it.

**ClearCase Support: How to Scrub a ClearCase VOB**

If one of your ClearCase VOBs is too slow or stopped working because it is full, then it might be time for you to scrub your VOB. Here are the steps to determine if you need to scrub your VOB and how to scrub your VOB.

1. First, verify the VOB needs scrubbing by executing these commands as root or the VOB owner on the VOB server. First, go to the VOB’s Database (DB) storage directory like this:

|  |
| --- |
| **cd VOB\_PATH/db** |

1. If any DB files are over one gigabyte in size, then run this tool to see how full the DB String File(s) actually are.

|  |
| --- |
| **/usr/atria/etc/utils/countdb vob\_db** |

1. If you have a lot of DB files, then you may need to run the program like this:

|  |
| --- |
| **/usr/atria/etc/utils/countdb vob\_db.\*** |

1. The "maximum records used" field is the most important output. If it is at 100%, then the VOB DB String File is full and you have to scrub the entire VOB. Otherwise, no one will be able to make changes to the VOB. I recommend scrubbing the VOB if the "maximum records used" is over 90%. If all the VOB DB String Files are less than 90%, then I do not recommend scrubbing your VOB. However, scrubbing your VOBs won’t hurt them yet may significantly improve their performance.   
     
   Furthermore, you may want to periodically check how full your VOB DB String Files are on all of your VOBs to ensure minimal downtime and peak performance.
2. The next step is to delete as many Views as possible. Some ClearCase Views have derived objects (D.O.s) that are promoted to the VOB Database. Deleting these Views will free up memory in the VOB’s DB String Files. At the very least, you can delete Views from de-activated and deleted accounts. Also, you can contact all of your users to have them delete their Views that they no longer need. Read these [instructions](http://www.philforhumanity.com/ClearCase_Support_23.html) if you need help removing Views.
3. Logged in as root or the VOB owner, scrub the VOB using this command:

|  |
| --- |
| **/usr/atria/etc/vob\_scrubber VOB\_PATH** |

1. This command may take a long time to finish running.. possibly days or just a few minutes. This command will "remove event records and oplog entries from VOB database." In other words, temporary data within the database will be freed up; however disk space usage will not decrease.
2. Re-execute the first step again. If the VOB DB String Files did not noticeably and significantly decrease, then your VOB’s configuration will need to be modified to keep oplogs for a shorter period of time. Oplogs are temporary history data packets used for ClearCase MultiSite between VOB replicas. By default, oplogs are kept forever. I recommend changing this value to 60 days. This means if synchronization is broken for longer than 60 days, then it will be impossible to re-synchronize two replicas. After 60 days of no synchronization, you will have to send a new replica to continue MultiSiting this VOB. If the VOB has never been MultiSited, then you can probably skip this step.   
     
   Edit this configuration file. By default, this file may not exist.

|  |
| --- |
| **VOB\_PATH/vob\_scrubber\_params** |

1. Change or add this line in this configuration file:

|  |
| --- |
| **oplog -keep 60** |

1. WARNING: Do not lower this oplog number too low.
2. Since you’ve changed the oplog retention configuration, re-execute Step #3 to delete the old oplogs in the VOB’s DB String Files. Afterwards, you may also want to re-execute Step #1 to verify that the VOB’s DB String Files have more free space.
3. Now that there are large sections of the VOB DB String Files that have been freed up, this next command "removes data containers from VOB storage pools and removes D.O.s from VOB database". Basically, the next command will remove all derived objects and config records from the VOB’s DB tables. This is temporary data that ClearCase can automatically recover if needed, yet may cost significant processing time.

|  |
| --- |
| **/usr/atria/etc/scrubber -e VOB\_PATH** |

1. This command should only take a few minutes to execute.
2. All of the previously VOB scrubbing commands only cleaned up internal space within the VOB Database Files, however these commands did not reclaim any unused disk space. In order to free up disk space on the VOB’s partition, you must defragment or reformat the VOB. This step is optional, unless you are running out of disk space on the VOB’s partition or hard drive. Read these [instructions](http://www.philforhumanity.com/ClearCase_Support_29.html) if you need assisting defragmenting VOBs.

**ClearCase Support: How to Back up a VOB**

Backing up a ClearCase VOB is only a 3 step process.

1.) Lock the VOB so that no changes can be applied to the VOB.

Users will still have read access to the VOB, but they won't be able to make any changes at all.

|  |
| --- |
| modify View private files in the VOB directory, such as compiling code in a VOB  **cleartool lock vob:VOB\_TAG** |

2.) Copy or backup the VOB's storage directory.

I recommend tarballing and zipping the VOB's storage directory, then placing the zip file onto a blank tape or DVDs

Recommend storing these backup tapes or DVD offsite in another secure location (building) in case of fire or theft in the server room.   
  
3.) Unlock the VOB and resume work as normally with this command:

|  |
| --- |
| **cleartool unlock vob:VOB\_TAG** |

VOBs that only exist locally (not multisited) and VOBs that you own mastership of.

**ClearCase Support: List VOB Replicas**

List all VOB replicas:

|  |
| --- |
| **$ cleartool lsreplica** |

**Note:** that you have to be in the VOB for this command to work, or you can use this additional argument with the command to point to a VOB that you may or may not be in:

|  |
| --- |
| **$ cleartool lsreplica -invob *VOB\_TAG*** |

The only problem is that each replica may have an unclear name. **Replicas should have both the site identification name and the VOB tag for clarification purposes.**

**ClearCase Support: How to Synchronize 2 VOBs**  
"Fix" the synchronization or updates between VOB replicas when a change in one VOB is not replicated to other sites fast enough. This is a simple two-step task.   
  
1.) First, the ClearCase administrator at the site that has the needed data must **create and export a synchronization packet to the site that needs that data.**

**Note:** This command must be run on the VOB server that hosts the VOB, as the VOB owner, and the destination VOB replica must be known.

|  |
| --- |
| ***/usr/atria/bin/multitool syncreplica –export –nc –fship  replica:VOB\_REPLICA@vob:VOB\_TAG*** |

This command will create a synchronization packet file in the outgoing shipping bay.

By default, this directory is located here**: /usr/atria/shipping/ms\_ship/outgoing/.**

Then this synchronization packet will be shipped to the remote replica's VOB server, and then the packet file will be deleted.   
2.) On the receiving replica, a ClearCase administrator must import the synchronization packet file that was just received.

The synchronization packet will be located on the VOB replica's host server in the incoming shipping bay; by default, this directory is located here**: /usr/atria/shipping/ms\_ship/incoming/.**

Note: must be run as the VOB owner and on the VOB host server.

Here is the command to import a single file:

|  |
| --- |
| ***/usr/atria/bin/multitool syncreplica – import SYNC\_PACKET\_FILE*** |

Alternatively, you can import all the available synchronization packets for a single VOB with this command:

|  |
| --- |
| ***/usr/atria/bin/multitool syncreplica – import –vob VOB\_TAG -receive*** |

Or, you can import all the synchronization packets for every VOB with this command:

|  |
| --- |
| ***/usr/atria/bin/multitool syncreplica – import -receive*** |

That's it! You have just synchronized one VOB replica with other VOB replica.

**ClearCase Support: How to Replicate a VOB**  
Here are the procedures for replicating or MultiSiting a VOB from one ClearCase region to another ClearCase region. These ClearCase regions can be in different buildings, sites, or even countries, but they must be on the same network. These instructions are split into three parts. The first part of these instructions is for the ClearCase administrator to export a local copy of the VOB. The second part of these instructions is for another ClearCase administrator to import this VOB onto their servers. Finally, the third part is the final tasks for both administrators at both sites. 

1. First, you will need this information before replicating a VOB.
   * **VOB Tag:** You first need to know the name of the VOB that will be exported. I strongly recommend keeping the same VOB tag on all sites that have this VOB.
   * **Destination Site:** Get the location of the ClearCase where this VOB will be replicated to.
   * **Destination Site's ClearCase Administrator:** Throughout these procedures, it will be required to contact the ClearCase administrator at the remote site.
2. Second, determine the VOB's current location and the VOB server that hosts the VOB by using this command.

|  |
| --- |
| ***cleartool describe vob:VOB\_TAG*** |

1. Log into the VOB's server as root.

|  |
| --- |
| ***rlogin SERVER\_NAME -l root*** |

1. If the local VOB replica has never been replicated to another site, then the default replica name is "original" for this master replica. Therefore, we are going to first rename this replica with these commands. First, determine the local VOB replica name using this command.

|  |
| --- |
| ***cleartool lsreplica –invob VOB\_TAG*** |

1. If the replica name is "original", then rename the replica name with these commands.

|  |
| --- |
| ***multitool rename replica:original@vob:VOB\_TAG replica:NEW\_REPLICA\_NAME@vob:VOB\_TAG multitool chreplica –host HOST\_NAME replica:NEW\_REPLICA\_NAME@vob:VOB\_TAG*** |

1. I recommend the new replica name include the site name and the VOB tag name. For instance, if the VOB tag is "tools" and the region name is "1", then a good VOB replica name could be "region\_1\_tools".
2. Determine the replica name for the new copy of the VOB. For instance, if the VOB tag is "tools" and the region name that the VOB is being sent to is "5", then a good VOB replica name could be "region\_5\_tools".
3. Determine the VOB size. Keep in mind, only 3 data pools (administration data, VOB database, and source pool) will be exported to the remote site. The other 2 data pools (cleartext pool and derived object pool) are temporary local databases, thus will not be exported or count as the size of the VOB being exported. Therefore, subtract these data pools from the total VOB size.

|  |
| --- |
| ***cleartool space VOB\_TAG*** |

1. Furthermore, you will be making a copy of this entire VOB before exporting it, so you will need a temporary directory capable of storing this entire VOB. And the remote site will need the disk space to support this VOB too.
2. At this point, contact the ClearCase administrator at the remote site. Provide the VOB TAG, new replica name, and VOB size. In return, the ClearCase administrator at the remote site will verify that they do not already have this VOB and will give you the host name (not the IP address) of the VOB server that will host this new VOB that has enough disk space for this VOB.
3. Go to a temporary directory that has sufficient disk to store a copy of this VOB.

|  |
| --- |
| ***cd TEMPORARY\_DIRECTORY*** |

1. Now we are ready to export the VOB into a format that can be shipped to any other server using this command. Note that this command locks and unlocks the VOB, so users will not be able to make changes to this VOB when this command is running. This command may take a long time to run, depending on the size of the VOB and hardware speed. Therefore, it is recommended to run this command during non-business hours.

|  |
| --- |
| ***multitool mkreplica –export –work ./tmp –maxsize 100m –vob VOB\_TAG –out ./VOB\_TAG.pkt REMOTE\_HOST\_NAME:REMOTE\_REPLICA\_NAME*** |

1. If the VOB size is large, then I recommend compressing the VOB packet files before shipping them. You can use zip, tar, etc. This is the ClearCase default compression method.

|  |
| --- |
| ***tar –cf – VOB\_TAG.pkt? | /usr/atria/etc/Gzip > TEMPORARY\_DIR/VOB\_TAG\_#\_tar.gz*** |

1. Now ship the VOB packet files that were created in the previous step. You can use ftp, blank CDs, etc., but this is the ClearCase default method that can be run for each packet file. Remember to only send the compressed files if created.

|  |
| --- |
| ***/usr/atria/etc/mkorder –data PACKET\_FILENAME -nc –fship –copy REMOTE\_HOST\_NAME*** |

1. At this point, notify the ClearCase administrator at the remote site that the VOB was exported to their site.

The remote ClearCase administrator now needs to import the exported VOB using these instructions. 

1. Log into the VOB server as root.

|  |
| --- |
| ***rlogin SERVER\_NAME -l root*** |

1. Go to the directory that has the sync packets or to the directory that the you copy the packets to. Copy the files to this directory if still necessary.

|  |
| --- |
| ***cd DIRECTORY*** |

1. Import the VOB.

|  |
| --- |
| ***multitool mkreplica –import –workdir ./tmp –tag VOB\_TAG –vob VOB\_STORAGE\_DIRECTORY –public –password REGISTRY\_PASSWORD –ignoreprot –npreserve –host VOB\_SERVER –hpath VOB\_STORAGE\_DIRECTORY –gpath VOB\_STORAGE\_DIRECTORY ./FIRST\_SYNC\_PACKET\_FILE*** |

1. Now that the VOB has been imported/created, set the permissions of this new VOB. First, you can see the existing permissions of the VOB at any time with this command:

|  |
| --- |
| ***cleartool describe –vob vob:VOB\_TAG*** |

1. To change the VOB owner and primary group, run this command:

|  |
| --- |
| ***cleartool protectvob –force –chmod VOB\_OWNER -chgrp PRIMARY\_GROUP VOB\_STORAGE\_PATH*** |

1. You can also add and remove secondary groups using this command too:

|  |
| --- |
| ***cleartool protectvob –force –add\_group GROUP1,GROUP2 -delete\_group GROUP3,GROUP4 VOB\_STORAGE\_PATH*** |

1. Next, set a View, mount the VOB, enter the VOB, and change the permissions of the directories in the VOB. I recommend permission numbers of either 770 or 775.

|  |
| --- |
| ***cleartool setview VIEW\_NAME cleartool mount VOB\_TAG cd VOB\_TAG cleartool protect –chown VOB\_OWNER -chgrp PRIMARY\_GROUP -chmod PERMISSION\_NUMBER -recurse .*** |

1. Using the "Region Synchronizer" in the "ClearCase HomeBase" on any Windows computer that has ClearCase installed, synchronize or add this new VOB so any Windows computers can access it.
2. At this point the VOB is ready to be used by any user, but I recommend first installing all required ClearCase triggers that your administrators' require. This step is optional.
3. I strongly recommend mounting this new VOB on all shared servers. This step is optional too.

There are still a few more steps to follow on both sites. 

1. Start synchronization on both VOB servers so that changes to either VOB is replicated to each other. I personally like to use cron. Here is an example.

|  |
| --- |
| ***0 1,3,5,7,9,11,13,15,17,19,21,23 \* \* \* /usr/atria/bin/multitool syncreplica –export –nc –fship replica:VOB\_REPLICA@vob:VOB\_TAG  0 0,2,4,6,8,10,12,14,16,18,20,22 \* \* \* /usr/atria/bin/multitool syncreplica –import –vob VOB\_TAG -receive*** |

1. Inform all users at the new site that the VOB is online and ready to be used.
2. Finally, the ClearCase administrators at both sites can delete the synchronization packets created when the VOB was exported and later imported.  
   You are done replicating a VOB to another site.

**ClearCase Support: How to Delete a VOB Replica**  
these are the procedures for turning off a VOB replica. This is also known as VOB decommission or obsolete VOB replica. 

1. The first step is to stop all synchronization exports to the VOB replica that you are decommissioning. I recommend finding all of these replicas by looking up which replicas this obsolete VOB is exporting to, then contacting the ClearCase administrators of those replicas. Skip this step if the VOB is not MultiSited.
2. Find all checked out elements in the local VOB replica.

|  |
| --- |
| ***cleartool lsco –all VOB\_TAG*** |

1. You will have to checkin or uncheckout each checkedout element found in the previous step, and that may require root access to log in as the users that have the checkedout elements.
2. Change mastership of each local object to the master site (or the new master site). You will need to be logged in as VOB owner or root to do this step. Skip this step if the VOB is not MultiSited. First, find the master replica name, then change mastership.

|  |
| --- |
| ***cleartool describe vob:VOB\_TAG  cleartool setview VIEW\_NAME cd VOB\_TAG multitool chmaster –vob . –all –long MASTER\_REPLICA\_NAME*** |

1. Push the final synchronization export of the obsolete VOB replica. You should wait until the remote master site imports and confirms these changes. Skip this step if the VOB is not MultiSited.

|  |
| --- |
| ***multitool syncreplica -export -fship replica:MASTER\_REPLICA\_NAME@vob:VOB\_TAG*** |

1. Next, turn off all local synchronization exports and imports for this obsolete VOB replica. Skip this step if the VOB is not MultiSited.
2. Remove the local obsolete VOB replica. Skip this step if the VOB is not MultiSited.

|  |
| --- |
| ***multitool rmreplica replica:OBSOLETE\_REPLICA\_NAME@vob:VOB\_TAG*** |

1. The master site should rename the deleted replica. Skip this step if the VOB is not MultiSited.

|  |
| --- |
| ***multitool rename replica:OBSOLETE\_REPLICA\_NAME@vob:VOB\_TAG replica:OBSOLETE\_REPLICA\_NAME.deleted@vob:VOB\_TAG*** |

1. It is strongly recommended that the VOB is backed up in permanent storage, such as tape or DVDs.
2. It is strongly recommended that the VOB is backed up in permanent storage, such as tape or DVDs.
3. Remove the VOB TAG.

|  |
| --- |
| ***cleartool rmtag –vob –password REGISTRY\_PASSWORD VOB\_TAG*** |

1. Next, un-register the VOB TAG.

|  |
| --- |
| ***cleartool unreg –vob VOB\_STORAGE\_DIRECTORY*** |

1. Finally, delete the obsolete VOB storage directory.

You are done deleting a VOB replica.

**ClearCase Support: Recommended Triggers**  
As you probably already know, ClearCase VOBs are not perfect, however they are customizable. That is why I recommend that these three ClearCase triggers be installed on every production VOB.

1. First, ClearCase Administrators should not allow regular users access to permanently remove entire ClearCase elements. That is why I recommend implementing and installing a ClearCase trigger that only allows administrators to remove elements. Here is an [example of this trigger](http://www.philforhumanity.com/ClearCase_Support_18.html).
2. Second, only build engineers, administrators, and software configuration managers (SCMs) should have access to checkout on release branches. So, I recommend having a trigger that allows only certain users access to check out on any of the release branches.
3. Finally, the biggest problem that I had when I first learned ClearCase Administration was that new elements were always created by users with the wrong permissions. So, I recommend a ClearCase trigger that changes the permissions on all new elements. First, directories should be writable by all users, so directories should have permissions 660 or 666. Second, files should be writable and executable by all users, so files should have permissions 770 or 777. Furthermore, all elements should verify and possibly change to the correct *primary* group for the VOB.

These scripts are very simple to implement, if you know how to install ClearCase triggers. And the advantages of implementing these three triggers far outweigh the time used to implement them.

# ClearCase Support: Installing and Implementing ClearCase Triggers

A ClearCase Trigger is simply a script that runs whenever a ClearCase action is performed. Triggers are completely customizable by the ClearCase Admins (VOB owner). Triggers can run before (preop) or after (postop) a ClearCase action. By default, ClearCase does not have any triggers turned on.   
  
For example, if we would want to implement a trigger that will prevent users from permanently deleting ClearCase elements from the VOB using the "cleartool rmelem" command, here is one possible solution.   
  
First, this is how to install a UNIX trigger, called "rmelem\_trigger", on the VOB called "/vob/test". The trigger is actually a Perl script at this location "/clearcase/rmelem\_trigger/rmelem\_trigger.pl". 

|  |
| --- |
| **cleartool mktrtype -nc -element –all -preop rmelem -execunix "/usr/local/bin/perl /clearcase/rmelem\_trigger/rmelem\_trigger.pl" rmelem\_trigger@/vob/test** |

In an environment with multiple types of operating systems, you may need to install a second trigger script. For example, here is the same installation command while running a different executable for Windows would be: 

|  |
| --- |
| **cleartool mktrtype -nc -element –all -preop rmelem -execunix "/usr/local/bin/perl /clearcase/rmelem\_trigger/rmelem\_trigger.pl"  -execwin "\\\\network\_machine\_name\\shar\_directory\\perl\\perl.exe \\\\network\_machine\_name\\shar\_directory\\rmelem\_trigger\\rmelem\_trigger.pl" rmelem\_trigger@/vob/test** |

An example of the trigger script in Perl for UNIX would be:

|  |
| --- |
| **#!/usr/local/bin/perl my $WHOAMI = `/bin/whoami`; chomp $WHOAMI;  if ($WHOAMI eq "root") {    exit 0;  } else {    print “\n\nERROR: You are not allowed to remove elements from the VOB!\n\n\n”;     exit 1;  }** |

If the script exits with a status of "0", then the ClearCase Trigger action (cleartool rmelem) is executed. If the script exits with a status of "1", then the ClearCase Trigger action is NOT executed. Of course, the Windows Perl script needs to be written too.   
  
To un-install a ClearCase trigger, use this command:

|  |
| --- |
| **cleartool rmtype -force trtype:rmelem\_trigger@vob:/vob/test** |
| **ClearCase Support: Finding ClearCase Triggers**  If you need to find the list of ClearCase triggers installed in a single VOB, then run this command. You can also use this same command to find the location of each trigger's scripts and the configuration of each trigger too.   |  | | --- | | ***cleartool lstype -long -kind trtype -invob* VOB\_NAME** |   For instance, here is an example of this command and the command's output. This command may take a long time to execute, especially if you have a large VOB or if there are a lot of triggers installed in the VOB.   |  | | --- | | **$ *cleartool lstype -long -kind trtype -invob /vob/test***  **trigger type "checkin\_permission\_trigger"**  **14-Apr-99.14:22:22 by root (root@machine\_name)**  **owner: root**  **group: group**  **all element trigger**  **pre-operation checkin**  **action: -exec \\windows\_machine\path\script.bat || /path/script.pl**  **trigger type "stop\_rmelem\_trigger"**  **24-Jun-04.13:03:03 by root (root@machine\_name)**  **owner: root**  **group: group**  **all element trigger**  **pre-operation rmelem**  **action: -execunix /usr/local/bin/perl /path/script2.pl**  **action: -execwin \\windows\_machine\path\perl.exe \\windows\_machine\path\script2.pl** |   **ClearCase Support: Finding Labeled Versions**  I am often asked how to find all the versions that have a certain label attached to it. The command is something like this:  **cleartool find . -all -ver 'lbtype(*LABEL\_NAME*)' -print**  This command will start recursively searching in the current directory or ".", but this can be changed to a specific VOB or directory like this example:   |  |  |  | | --- | --- | --- | | ***$ cleartool find /vob/test/a/ -all -ver 'lbtype(TEST\_LABEL)' -print /vob/test/a@@/main/5 /vob/test/a/b.txt@@/main/1 /vob/test/a/c@@/main/2 /vob/test/a/c/d.txt@@/main/1 /vob/test/a/e.txt@@/main/3***  **ClearCase Support: Finding Label and Branch Types**  If you need to get a list of label types that exist in a VOB, you can run this command:   |  | | --- | | ***cleartool lstype -kind lbtype -invob* VOB\_NAME** |   You can also add the "-long" argument to the command for additional information about each label type. You can even use the "-short" argument for even less information. Keep in mind that just because a label type exists, this does not mean the label has been applied to any elements, files, or directories.  Alternatively, if you want to get a list of all branch types that exist in a VOB, you can run this similar command:   |  | | --- | | ***cleartool lstype -long -kind brtype -invob* VOB\_NAME** |   Again, just because a branch type exists, this does not mean that a branch exists with that branch type name. | |

**Issues:**

# ClearCase Support: Updating Magic Files for Changing Default Element Types

If you consistently get element "type" error messages when trying to check in a new element for the first time, then you probably manually change the element’s type to be able to check in the file using this [process](http://www.philforhumanity.com/ClearCase_Support_64.html).   
  
However, if the file name is consistent, then you can reconfigure ClearCase to do this automatically for you. For instance, you can define all files that end in the extension ".install" to be a compressed file type instead of the default text file type.   
  
Just update your magic file located in one of these locations: 

|  |
| --- |
| **UNIX path: /usr/atria/config/magic/default.magic  Windows path: C:\Program Files\Rational\ClearCase\config\magic\default.magic** |

For my example, you just need to add a line like this at the correct location of the file:

|  |
| --- |
| **install\_file compressed\_file : -name "\*.install" ;** |

You may have to stop and restart ClearCase on your computer for this change to take affect. And don’t forget to make this change on all your servers and clients.

**ClearCase Support: Checked Out But Removed Error**

One of the most common error messages in ClearCase is when an element (file or directory) is "checked out but removed". From some graphical user interfaces (GUIs), the element even appears to be missing from the directory where the element is suppose to be located.   
  
However, by executing "cleartool ls" from the command line, the element appears to exist yet removed. For example, this is what a typical user would see: 

|  |
| --- |
| **$ *cleartool ls* test.txt@@/main/CHECKEDOUT from /main/2 [checkedout but removed]** |

There are several possible reasons for this issue to happen. The first most likely reason is because the user had permission to checkout the file, yet the user did not have permission to copy the file from the VOB to the user’s View in the current directory. In other words, the user did not have write permission in the parent directory where the file is located. Therefore, uncheckout the file (using the command "cleartool unco FILENAME"), and then verify that the user has write permissions in the parent directory. Either try creating a new file in the directory or directly check the directory’s write permissions with the user’s permissions. After the [permissions have been corrected](http://www.philforhumanity.com/ClearCase_Support_5.html), then the user can checkout and modify the file correctly.   
  
Another reason for this error to happen is that the element was checked out correctly but later deleted, moved, or renamed. There is nothing that can be done for this issue, except unchecking out of the element and re-checking out the element.

**ClearCase Support: Changing Comments**  
Have you ever checked in a file and then realize that the checkin comment is incorrect? Most people are not aware that comments can be modified after they have been committed to the VOBs. For instance, to replace the comment of the version that the View is accessing, just run this command:

**cleartool chevent –replace –c "*NEW\_COMMENT*" *ELEMENT\_NAME***  
Keep in mind that this command will completely delete/replace the previous comment with the new comment, if there was a previous comment. Alternatively, you can append to the existing comment by removing the "-replace" argument without losing the previous data, if necessary.

Also, I recommend viewing the comment both before and after you run this command using these [procedures](http://www.philforhumanity.com/ClearCase_Support_1.html).

**ClearCase Support: Common ClearCase Checkin Error Message**

Here is a common error message in ClearCase when trying to check in a file. 

|  |
| --- |
| **$ cleartool ci -nc *FILENAME* text\_file\_delta: Error: "FILENAME" is not a 'text file': it contains a line exceeding 8000 bytes. Use a different type manager (such as compressed file). cleartool: Error: Unable to check in "FILENAME".** |

Basically, this "text file delta" error message is that the ClearCase element is a text file in the database, and you are trying to check in a non-text version. For instance, you may have only created a few versions (/main/0 and /main/1), and these versions were initially mostly empty files that ClearCase assumed were text files when you created the element for the first time. However, when you finally check in a completed version with all the data in it, this final version could contain non-text data.   
  
To resolve this problem, you have to change the element type before being able to check in the new version. If the file is a regular binary file, then I recommend changing the element type to a compressed file type with this command.

|  |
| --- |
| **cleartool chtype compressed\_file *FILENAME*** |

If the file is a Rational Rose RealTime file, then I recommend change the element type to a Rose Unit type with this command.

|  |
| --- |
| **cleartool chtype rosert\_unit *FILENAME*** |

You must be the VOB owner or element owner to run either of these commands. Don't forget to check in again afterwards.

**ClearCase Support: Changing File Permissions**

I am often asked how to change file permissions on elements in ClearCase, so here are some simple examples on how to do it.   
For instance, let's assume you have a ClearCase element with these permissions: 

**-r-x------ 1 user1 group2 76 Mar 16 02:25 filename.txt**  
Run this command to change the element's permissions. This is equivalent to UNIX's "chmod" command:

**$ *cleartool protect -chmod 755 filename.txt***

This will result with an element with these permissions:

**-rwxr-xr-x 1 user1 group2 76 Mar 16 02:25 filename.txt**  
To recursively change permissions for an entire directory (for this example, the /vob/test directory) and all sub-files and sub-directories in that directory, you can run:

**$ *cleartool protect -chmod 775 -r /vob/test***  
Now, let's say you want to change the element's owner and group. This is equivalent to UNIX's "chgrp" and "chown" commands:

**$ *cleartool protect -chgrp newgroup -chown newuser filename.txt***

This will result with an element with these permissions:

**-rwxr-xr-x 1 newuser newgroup 76 Mar 16 02:25 filename.txt**

**ClearCase Support: Recommended VOB Permissions**

ClearCase VOB permissions with UNIX groups have always a bit mysterious to most users, so I have decided to write what I recommend ClearCase VOB permissions to be.   
  
First, start off by creating a single UNIX group that \_ALL\_ users will have as their primary group. Let’s call this group as the DEFAULT group. Next, set this DEFAULT group as the primary group for all users on UNIX, Linux, and Windows accounts. Third, set all VOBs (that are not secure) to this group and no other group at this time. Finally, in a View, go to each of the VOBs and change all the elements’ permissions to 777 and in this group.   
  
That’s it for regular VOBs that you want all users to have access to.   
  
For secure VOBs, create a unique group or groups for these secure VOBs. Change the primary group of these VOBs to this new secure group that you just created. The VOBs secondary group must be the DEFAULT group. Next, change the root directory of the VOB to be in the new secure group and have permissions exactly 770. That means only people in this secure group can access this VOB. All other elements in the VOB should be set to the DEFAULT group and 777 permissions. Finally, add your users to these new secure groups as secondary groups on all users’ accounts for each operating system.   
  
While there are other schemes for setting up ClearCase VOB permissions and their groups, I think I have found what is most optimal to be defined in the paragraphs above. Of course, that means each VOB has only one secure group, which is strongly recommended. Otherwise, if you have different sub-directories with different permissions throughout the VOBs, then they could be easily changed without anyone noticing or remembering that they are different for a good reason.

# ClearCase Support: Unable to Mount a VOB

I often have the problem of being unable to mount a VOB in UNIX. Typically, it is because a process is running that is in the same path that I want the VOB mount to be. For example, someone is in the path "/vob/test" when the VOB "/vob/test" is not mounted. At this point, no one can mount the VOB unless the process leaves this directory.   
  
Here's how to find the offending process. First, find the all the processes currently in the directory with this UNIX command:

|  |
| --- |
| **$ *fuser /vob/test* /vob/test: 7394c** |

Next, find the owner(s) of each process number found with this command (note: this command can be slightly different on different operating systems):

|  |
| --- |
| **$ *ps -fp 7394c* UID PID PPID C STIME TTY TIME COMMAND userid 7394 2869 0 Jun 1 ttypd 0:00 ksh** |

At this point in time, you can either contact this person directly to exit the directory you wish to mount or manually kill this process with root access.

# ClearCase Support: Restarting ClearCase Services and Processes

If you are a UNIX or ClearCase administrator, then you might need to periodically restart the ClearCase services on a computer. Here is how to do it.   
First, you will need to log in as root.   
Second, this is the command to stop all ClearCase process on a computer: 

|  |
| --- |
| **# *atria\_start stop*** |

This "atria\_start" program can be located in any of these locations:

|  |
| --- |
| **/etc/ /sbin/init.d/ /etc/init.d/ /usr/atria/etc/ /CLEARCASE\_INSTALL\_DIRECTORY/etc/** |

Furthermore, this "atria\_start" program may also be listed as these equivalent executables:

|  |
| --- |
| **rc.atria atria clearcase** |

This all depends on your version of ClearCase and Operating System.   
Once ClearCase has been stopped, you can make changes to the ClearCase environment or operating system as needed.   
Finally, to restart all ClearCase processes, run this command:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **# *atria\_start start*** ClearCase Support: Error Removing Baselines This next ClearCase Administration tip is from Premal Acharya.   If you try to remove a baseline from an integration stream, and you get this error message:    |  | | --- | | **cleartool: Error: Cannot remove baseline that is the foundation for a timeline. cleartool: Error: Unable to remove baseline "baseline:XXXXXXXXXX".** |   You may have already removed all activities and development streams from the integration stream, so there is nothing left other than the baseline. Yet, you are still not able to remove the baseline, then here is the work around:   The baseline that you are trying to remove was used to seed other new projects and was the foundation baseline for them. When you create or deliver a baseline, or do any ClearCase activity that involves changing a baseline in any form, the data is recorded in the timeline folder of ClearCase project. The timeline is similar to system files and folders used in Windows that are not visible and not modifiable by users. It can only be changed by ClearCase using its internal processes. Unfortunately, there is not much information available about timelines on IBM's website.   To remove all baselines, where you are getting this error, try using the following process. This needs to be carry out on every component from where you want to remove the baseline.   1. Describe baseline using the **-l** option. 2. Make a note of **hyperlinks** shown at the end. 3. Remove all those hyperlinks using the **cleartool rmhlink** command. Keep in mind, removing hyperlinks without proper knowledge can be VERY harmful unless suggested by tech support. 4. Again, describe the baseline to see that there are no hyperlinks present. 5. Once all the hyperlinks are removed, you can remove the baseline using the **rmbl -f** command.   Good luck! ClearCase Support: How to Execute a Command in a View Here is the ClearCase command to execute another command inside of a ClearCase View that you are not currently in.   |  | | --- | | **cleartool setview -exec "COMMAND" VIEW\_NAME** |   Basically, this ClearCase command starts the designated View, then executes the given command in the View. If and when this given command finishes executing, then this given command will exit out of the designated View.   With this ClearCase command, you can be in a View, yet run a command in another View. This ClearCase command will return to the original View when it has completed executing. Or else, you do not need to be in a View at all for this command to work, and you will not be in a View afterwards either. ClearCase Support: How to Merge If you ever need to merge a text file or directory from the command line or from an automated script, then this is the command that you will need. Basically, this command will merge two versions together on a single element.   |  | | --- | | **cleartool merge -to "ELEMENT\_PATH" FROM\_VERSION** |   This command will merge to the version that your View's config spec is set to access.  **ClearCase Support: Drawing a Merge Arrow**  Have you ever wanted to just draw a merge arrow without actually merging two ClearCase versions together? If so, here is how to do it from the command line.   |  | | --- | | ***cleartool mkhlink -unidir Merge* FROM\_VERSION TO\_VERSION** |   For instance, here are three actual draw merge arrow commands.   |  | | --- | | **cleartool mkhlink -unidir Merge /vob/test/a.txt@@/main/branch/1 /vob/test/a.txt@@/main/4**  **cleartool mkhlink -unidir Merge /vob/test/a.txt@@/main/branch/1 /vob/test/a.txt@@/main/LATEST**  **cleartool mkhlink -unidir Merge /vob/test/a.txt@@/main/branch/1 /vob/test/a.txt@@/main/CHECKEDOUT** |   Don't forget to verify that the merge arrow was applied correctly. ClearCase Support: How to Find Merges Needed If you ever need to determine if you need to merge two versions together, then execute this "findmerge" command in ClearCase.   |  | | --- | | **cleartool findmerge "ELEMENT\_PATH" -fversion FROM\_VERSION -directory -print -short** |   The "MERGE TO" version is whatever version that your View's config spec is set to access. If the command outputs nothing, then no merge is necessary. Otherwise, if the command outputs a merge is necessary, then proceed with the upmerge.   Here is an example of this command in action:   |  | | --- | | **cleartool findmerge "filename.txt" -fversion /main/branch/5 -directory -print -short** |   The output would look something like this:   |  | | --- | | **Needs Merge "filename.txt" [to /main/3 from /main/branch/5 base /main/2]** |   I recommend that you play around with this command to get use to all the possible options and arguments. ClearCase Support: How to Find Checked Out Elements Ever wonder how to find all the checked out files in your ClearCase View from the command line?  Here is an example on how to do it while in a View:   |  | | --- | | **$ *cleartool lspriv -co* /vob/test/a/b.txt [checkedout] /vob/test/a/c/d.txt [checkedout] /vob/test/a/e.txt [checkedout]** |   You may have noticed that the previous command does not locate directories. It only locates files. So, to find all checked out elements (files and directories) in your View, run this command:   |  | | --- | | **$ *cleartool lsco -short -cview -me -avobs* /vob/test/a /vob/test/a/b.txt /vob/test/a/c /vob/test/a/c/d.txt /vob/test/a/e.txt** |   Note that the second command is much slower than the first command.  **ClearCase Support: How to Checkout, Edit, Checkin, and Uncheckout Files**  The most basic and common activity in ClearCase is making a change to a file. In order to edit any file in a ClearCase VOB, you must first checkout the file. This basically means no one can checkout the file when the file is already checkedout by someone else. You can check out a file with this command.   |  | | --- | | ***cleartool co -nc FILENAME*** |   ClearCase does allow multiple people to checkout a single file at a time. This can be done by creating [branches](http://www.philforhumanity.com/ClearCase_Support_8.html) or by using unreserved checkouts. To create an unreserved checkout, use this command.   |  | | --- | | ***cleartool co -unreserved -nc FILENAME*** |   Remember, if you allow multiple checkouts at a time for a single file, then you probably will have to [merge](http://www.philforhumanity.com/ClearCase_Support_70.html) your file versions later. I advise that you seek expert help with merging files if you have never done it before.   After you have checkedout your file, then you should have edit access to the file, so go ahead and make your changes. Once you have completed all of your changes, save your file and preferrably exit out of the program that you were using to modify the file. I do not recommend that you leave a file checkedout for more than a month or so.   Now you are ready to checkin the file. This basically takes the saved file that you worked on and copies the file from your private work space, called your View, and copies the file to the VOB database so that everyone can access your new version of the file. Here is the command to checkin your file.   |  | | --- | | ***cleartool ci -nc FILENAME*** |   Alternatively, if you do not want the changes to the file or you did not mean to checkout the file, you can uncheckout the file with this command.   |  | | --- | | ***cleartool unco FILENAME*** |   Keep in mind, that this uncheckout command will **DELETE** any changes that you may have made. ClearCase may even ask you if you want to create a backup of the data that you are about to uncheckout. Answer accordingly to your wishes.  **ClearCase Support: Finding Branches on Version Trees**  I am often asked how to find a certain branch from the command line. The command is something like this:   **cleartool find . -all -ver 'brtype(*BRANCH\_NAME*)' -print** This command will display each version that has this branch and start recursively searching in the current directory or ".", or you can specify only one element like this example:   |  | | --- | | ***$ cleartool find /vob/test/a/ -all -ver 'brtype(TEST\_BRANCH)' -print /vob/test/a/b.txt@@/main/TEST\_BRANCH/0 /vob/test/a/b.txt@@/main/TEST\_BRANCH/1 /vob/test/a/b.txt@@/main/TEST\_BRANCH/2 /vob/test/a/cc.pdf@@/main/TEST\_BRANCH/0 /vob/test/a/cc.pdf@@/main/TEST\_BRANCH/1*** |   If you only want to see the latest versions on the branch, then use this command instead:   |  | | --- | | ***$ cleartool find /vob/test/a/ -all -ver 'version(.../TEST\_BRANCH/LATEST)' -print /vob/test/a/b.txt@@/main/TEST\_BRANCH/2 /vob/test/a/cc.pdf@@/main/TEST\_BRANCH/1*** |   However, if you only wanted to find elements with a branch without listing each version on the branch, then you can run:    |  | | --- | | ***$ cleartool find /vob/test/a/ -all -element 'brtype(TEST\_BRANCH)' -print /vob/test/a/b.txt@@ /vob/test/a/cc.pdf@@*** |  ClearCase Support: How to Make a ClearCase Label You may be asking yourself how to create a label in ClearCase from the command line. Here's how!   First, create a label name (this is called a label type) in the VOB that you want to make the label in:    |  | | --- | | **$ *cleartool mklbtype TEST\_LABEL* Comments for "TEST\_LABEL": . Created label type "TEST\_LABEL".** |   Don't forget to enter a comment for the label type, or you can use the "-nc" argument for no comment.   Now, you are ready to apply this label to version in ClearCase. To apply it to a single file, use this command:   |  | | --- | | **$ *cleartool mklabel TEST\_LABEL filename.txt* Created label "TEST\_LABEL" on "filename.txt" version "/main/179".** |   This command will label the version that your config spec is pointing to. To specify a specific version and move the label existing label, use this command:   |  | | --- | | **$ *cleartool mklabel -replace TEST\_LABEL filename.txt@@/main/99* Moved label "TEST\_LABEL" on "filename.txt" from version "/main/179" to version "/main/99".** |   Finally, if you wanted to label all the files and directories recursively starting from a certain directory, use this command:   |  | | --- | | **$ *cleartool mklabel -r TEST\_LABEL /vob/test/a* Created label "TEST\_LABEL" on "/vob/test/a" version "/main/2". Created label "TEST\_LABEL" on "/vob/test/a/b.txt" version "/main/1". Created label "TEST\_LABEL" on "/vob/test/a/c" version "/main/3". Created label "TEST\_LABEL" on "/vob/test/a/c/d.txt" version "/main/1".** |   If you want to make sure no one deletes or moves the label or deletes any versions labeled, you may lock the label using this command:   |  | | --- | | **$ *cleartool lock lbtype:TEST\_LABEL*** |  ClearCase Support: How to Make a ClearCase Branch You may be asking yourself how to create a branch in ClearCase from the command line. Here's how!  First, create a branch name (this is called a branch type) in the VOB that you want to make the branch in:   |  | | --- | | **$ *cleartool mkbrtype TEST\_BRANCH* Comments for "TEST\_BRANCH":. Created branch type "TEST\_BRANCH".** |   Don't forget to enter a comment for the branch type, or you can use the "-nc" argument for no comment.  Now, you are ready to make a branch on an element in ClearCase, using this command:   |  | | --- | | **$ *cleartool mkbranch -nc TEST\_BRANCH filename.txt* Created branch "TEST\_BRANCH" on "filename.txt" from version "/main/179". Checked out "filename.txt" from "/main/TEST\_BRANCH/0".** |  ClearCase Support: How to Lock a Label Whenever [creating a label](http://www.philforhumanity.com/ClearCase_Support_7.html) that you want to permanently keep and prevent other users from modifying the label or deleting any labeled version, then the only option is to lock the label type. This can be easily done using this simple command:   |  | | --- | | **cleartool lock lbtype:*LABEL\_NAME*** |   Keep in mind that you have to lock the label type in each VOB that has the label. You can even specify the VOB name without going to the VOB using this command:   |  | | --- | | **cleartool lock lbtype:*LABEL\_NAME*@vob:*VOB\_TAG*** |   Furthermore, I recommend immediately locking the label type after creating the label type but before actually labeling any versions. Using this method, no other user can modify your label as your are labeling. To do so, you must lock the label type while specifying a user exception list who the lock does not apply to. Here is any example on how to do this:   |  | | --- | | **cleartool lock –nuser *USERID1*,*USERID2*,*…* lbtype:*LABEL\_NAME*** |   The USERID argument should at least include your core id. ClearCase Support: How to Find Label Lock Information Ever wanted to know who locked a ClearCase label? Or who has access to override a ClearCase label? Wonder no longer, because here is the command:    |  | | --- | | **$ *cleartool lslock lbtype:*LABEL\_NAME 04-Oct.09:59 userid\_who\_locked\_label lock label type "LABEL\_NAME" (locked) "Locked except for users: userid\_with\_override\_access\_1 userid\_with\_override\_access\_2 .."** |   **ClearCase Support: How to Lock a Branch**  Once in a while, I am asked to lock a branch, so that no user can add or delete to versions or labels on this branch. I prefer and recommend locking the entire branch type for each branch in an entire VOB. To do this, I use this command:   |  | | --- | | **cleartool lock brtype:*BRANCH\_NAME*** |   Keep in mind that you have to lock the branch type in each VOB that has the label. You can even specify the VOB name without going to the VOB using this command:   |  | | --- | | **cleartool lock brtype:*BRANCH\_NAME*@vob:*VOB\_TAG*** |   You can even add a user exception list who the lock does not apply to. Here is any example on how to do this:   |  | | --- | | **cleartool lock –nuser *USERID1*,*USERID2*,*…* brtype:*BRANCH\_NAME*** |   The USERID argument should at least include your core id.   |  |  |  | | --- | --- | --- | | ClearCase Support: Renaming Label and Branch Types Do you want to rename a label or branch on all files or elements that have it? Well, you can rename a label type or branch type for an entire VOB with a single command.   Here is an example. Let's say you want to rename a label type called "OLD" to become "NEW", then this is the command that you would use:   |  | | --- | | **$ *cleartool rename lbtype:*OLD *lbtype:*NEW** |   Alternatively, you can run this similiar command to rename branch types too:   |  | | --- | | **$ *cleartool rename brtype:*OLD *brtype:*NEW** |   If the label or branch exists in more than one VOB, then you have to run this command in each VOB that it exists in.   Also, these commands will not work if the label or branch type is locked by someone else, so you will need an administrator to unlock the label and branch types first. |  ClearCase Support: Renaming and Moving Elements Ever wonder how to rename or move an element (file or directory) in a VOB?   Here's how. First, check out the directory where the element is in, since you will be changing the directory listing and not the element itself. This directory is called the "parent directory" of the element. Second, run this command:   |  | | --- | | ***cleartool move OLD\_NAME NEW\_NAME*** |   Finally, check in the element’s parent directory.   Next, to move a file from one directory to another directory within the same VOB, checked out both parent directories of where you are moving the element from and where you are moving the element to. Next, run basically the same command:   |  | | --- | | ***cleartool move OLD\_PATH NEW\_PATH*** |   You can even rename the element at this time. Finally, don't forget to checkin both parent directories.  **ClearCase Support: How to Delete Labels and Label Types**  Here is the command to delete a label.   **cleartool rmlabel –nc LABEL\_NAME ELEMENT\_NAME**  For instance, if you wanted to delete the "label\_test" label on the file "/vob/test/a.txt", then here is the command.   |  | | --- | | **$ *cleartool rmlabel –nc label\_test /vob/test/a.txt*** |   Alternatively, here is the command to delete an entire label type. In other words, you wanted to delete all the labels with a certain label name as well as deleting the label name.   **cleartool rmtype –rmall –force lbtype:LABEL\_TYPE\_NAME**  Keep in mind, this will delete the label type in only one VOB at a time, so you have to run this command once in each VOB that you want the label type deleted.  **ClearCase Support: How to Delete Branches and Branch Types**  Here is the command to delete a branch.   **cleartool rmbranch –nc –force ELEMENT\_PATH@@FULL\_BRANCH\_PATH** For instance, if you wanted to delete the /test/ branch and the branch was branched from the /main/ branch on the file "/vob/test/a.txt", then here is the command.   |  | | --- | | **$ *cleartool rmbranch –nc –force /vob/test/a.txt@@/main/test*** |   Alternatively, here is the command to delete an entire branch type. In other words, you wanted to delete all the branches with a certain branch name as well as deleting the branch name.  **cleartool rmtype –rmall –force brtype:BRANCH\_TYPE\_NAME**  Keep in mind, this will delete the branch type in only one VOB at a time, so you have to run this command once in each VOB that you want the branch type deleted.  **ClearCase Support: Common ClearCase Labeling Error Message** Here is a common error message in ClearCase when recursively labeling.   |  | | --- | | **$ cleartool mklabel -nc -r TEST\_LABEL . *cleartool: Error: Unable to access "./FILE2.txt": No such file or directory. Created label "TEST\_LABEL" on "." version "/main/2". Created label "TEST\_LABEL" on "./FILE1.txt" version "/main/1". …*** |   This error message is because your config spec is setup to not access any version of the "FILE2.txt" element. Therefore, the ClearCase make label command knows that there is an element with that name; however, it can not access any version of element to label. Thus, this error message is produced.   Fortunately, the make label command does continue labeling everything else, so you can ignore this error message if you do not want that element labeled. On the other hand, if you need to label this element, then you can either change your config spec to access a version of the element and then label it or you can specify the exact version of the element to label with the make label command like this example below.  ClearCase Support: How to See the Version Tree of an Element  If you want to see the version tree of an element, there are several ways to do this. Th most common is to use the graphical user interface (GUI), known as [xlsvtree](http://www.philforhumanity.com/ClearCase_Support_13.html). However, this GUI is often slow and is not easy to automatically extract data from.   Here is the command to display the version tree of an element in text format.   |  | | --- | | ***cleartool lsvtree FILENAME*** |   Understanding the format of the output may take a moment to two, however it is well worth knowing. ClearCase Support: How to List Elements and their Rules Another common ClearCase command is to list all the elements (files and directories) in the current directory.   |  | | --- | | **cleartool ls** |   This command will also display what config spec rule that your View is using to select what version to access.   You can also run this command on a single element, like this:   |  | | --- | | **cleartool ls FILENAME** |   These commands may be very slow, if the directory is very large. |
| ClearCase Support: Renaming Label and Branch Types Do you want to rename a label or branch on all files or elements that have it? Well, you can rename a label type or branch type for an entire VOB with a single command.   Here is an example. Let's say you want to rename a label type called "OLD" to become "NEW", then this is the command that you would use:   |  | | --- | | **$ *cleartool rename lbtype:*OLD *lbtype:*NEW** |   Alternatively, you can run this similiar command to rename branch types too:   |  | | --- | | **$ *cleartool rename brtype:*OLD *brtype:*NEW** |   If the label or branch exists in more than one VOB, then you have to run this command in each VOB that it exists in.   Also, these commands will not work if the label or branch type is locked by someone else, so you will need an administrator to unlock the label and branch types first. |

# ClearCase Support: Renaming and Moving Elements

Ever wonder how to rename or move an element (file or directory) in a VOB?   
  
Here's how. First, check out the directory where the element is in, since you will be changing the directory listing and not the element itself. This directory is called the "parent directory" of the element. Second, run this command:

|  |
| --- |
| ***cleartool move OLD\_NAME NEW\_NAME*** |

Finally, check in the element’s parent directory.   
  
Next, to move a file from one directory to another directory within the same VOB, checked out both parent directories of where you are moving the element from and where you are moving the element to. Next, run basically the same command:

|  |
| --- |
| ***cleartool move OLD\_PATH NEW\_PATH*** |

You can even rename the element at this time. Finally, don't forget to checkin both parent directories.

# ClearCase Support: How to See the Version Tree of an Element

If you want to see the version tree of an element, there are several ways to do this. Th most common is to use the graphical user interface (GUI), known as [xlsvtree](http://www.philforhumanity.com/ClearCase_Support_13.html). However, this GUI is often slow and is not easy to automatically extract data from.   
  
Here is the command to display the version tree of an element in text format.

|  |
| --- |
| ***cleartool lsvtree FILENAME*** |

Understanding the format of the output may take a moment to two, however it is well worth knowing.

# ClearCase Support: Manipulating Config Specs

Ever wonder how to see, save, and edit your Config Spec in ClearCase?   
To see or display the Config Spec for the current View, use this command:

**cleartool catcs**

To edit the Config Spec for the current View, use this command:

**cleartool edcs**

To save or backup the Config Spec for the current View to a file, use this command:

**cleartool catcs > *filename.txt***  
To set the Config Spec for the current View from a file (this will delete your current config spec settings, so please make a backup first), use this command:

**cleartool setcs *filename.txt***  
To set the Config Spec for the current View to the default setting (this will delete your current Config Spec settings, so please make a backup first), use this command:  
**cleartool setcs -default**  
Finally, to refresh or synchronize your View with all servers without changing your Config Spec, use this command:

**cleartool setcs –current**

# ClearCase Support: Issue with Links and Config Specs

Here is a common problem when dealing with links in ClearCase config specs. This problem is true for both soft links and hard links.   
  
Assume for a moment that you have a regular file called "a.txt", and you have a link to this file called "b.txt". It does not matter if both files are in the same directory or in different directories. So, we have this situation:

|  |
| --- |
| **a.txt b.txt -> a.txt** |

Next, there exist multiple versions of a.txt, like this:

|  |
| --- |
| **a.txt@@/main/1 a.txt@@/main/2 a.txt@@/main/3** |

Now, assume that your config spec is set to this:

|  |
| --- |
| **element b.txt /main/1 element a.txt /main/2 element \* /main/LATEST** |

I have these two questions for you. What version of "b.txt" do you think your config spec is set to access? What version of "a.txt" do you think your config spec is set to access?   
  
I would have **assumed** that we would access version /main/1 of both files; however in actuality, the config spec is configured to only access /main/2 of both files. Basically, the first rule in the config spec is ignored, since it is not an element.   
  
Here is the worse case situation that I found. Assume that you have a directory and a link to the directory, such as this:

|  |
| --- |
| **/vob/test/dir1 /vob/test/dir2 -> dir1** |

If you have any of these rules in your config spec, they will be completely ignored:

|  |
| --- |
| **element /vob/test/dir2 /main/LATEST element /vob/test/dir2/file.txt /main/LATEST element /vob/test/dir2/dir3/file.txt /main/LATEST** |

The only workaround that I found for this issue is to NOT allow any links, either soft links or hard links, in any config specs…. ever. Also, I have stopped using links completely. I am even starting to remove them.

# ClearCase Support: Understanding Config Specs

I recently had a question concerning how to understand configuration specifications or config specs in ClearCase Views, so here goes.   
  
A config spec is the mechanism that a ClearCase View determines what versions of an element that the user accesses. A config spec is only editable, by default, by the account that created the View. A config spec has a single rule on each line, and the lines are interpreted by ClearCase from the top to the bottom as the order of importance. For example, when you create a new ClearCase View, the default config spec is set to this:

|  |
| --- |
| **element \* CHECKEDOUT element \* /main/LATEST** |

Each rule basically consists of three parts. First the word "element", second what element to find, and third is the version to access. In this default config spec example, the first rule says to access the checkedout version of the element if the current View has the element checkedout for each element. If the View does not have the element checkedout, then the next rule is interpreted. In this example, the next rule dictates that the View will access the latest version of the element on the /main/ branch. This rule will guarantee to find a version to access, so any further rules, if any existed, will be ignored.

Let’s take a look at this more complicated config spec with example #2: 

|  |
| --- |
| **element /vob/test/a.txt /main/3 element /vob/test/b.txt /main/4 # This is a comment. #element \* /main/LATEST # The previous line is a comment, thus completely ignored. element /vob/test/… /main/LATEST** |

The first rule states to only access the /main/3 version of the element "/vob/test/a.txt". This element may or may not exist, and ClearCase has no verification. Any other elements of a different path will ignore this rule. The second rule states to only access the /main/4 version of a different element called "/vob/test/b.txt". Note that anything after the first # symbol is a comment and is ignored. The third and fourth lines are comments, so they will be ignored even though they may have embedded rules. The fifth line says for all elements in the VOB call "/vob/test", access the latest versions on the /main/ branch, unless a previous rule already selected a version. Note that there are no rules in this config spec to access any versions in any other VOB, so all other VOBs will be inaccessible with this config spec.   
  
You may want to take a mental break now, since the next example is much more complicated. If you don’t know what are labels or branches yet, I recommend reading the other training web pages first. Here is example #3: 

|  |
| --- |
| **element /vob/training/hockey/… HOCKEY\_LABEL element /vob/training/baseball/… BASEBALL\_LABEL element /vob/training/football/… …/football\_branch/LATEST element /vobs/training/… /main/LATEST** |

The first line says to access only the versions that have a label called "HOCKEY\_LABEL" in the directory called /vobs/training/hockey. Not all files (or sub-directories) in this directory may have this rule, so these elements will not be accessed from this rule. Similarly, the second line says to access only the versions that have a label called "BASEBALL\_LABEL" in the directory called /vobs/training/baseball. The third rule says to access the latest versions in the "/vob/training/football" directory on the /football\_branch/ branch if that branch exists for each element. Otherwise the fourth rule says to access all other elements that the previous rules did not define to access by accessing the latest versions in the "/vob/training/" directory on the /main/ branch.   
  
Confused yet? Well, it gets MUCH more complicated. Here is example #4: 

|  |
| --- |
| **element /vob/test/a.txt -none element b.txt -none element \* /main/test/LATEST element –file \* /main/LATEST element -directory \* /main/LATEST** |

# ClearCase Support: Advanced Config Specs (Part 1)

Ever wanted to rollback the clock to see what was in the VOB in the past without using an old label in your Config Spec? Well you can, with the "Time Rules" feature of Config Specs. For instance, let's say you wanted to see how all the VOBs appeared as if it was still 11AM on March 5th, then you can use this config spec:

|  |
| --- |
| **element \* CHECKEDOUT element \* /main/LATEST -time 5-Mar.11:00** |

There are a lot of options with this "-time" argument. To see the complete list of these "Time Rules", please read the online man page for Config Specs using this command:

|  |
| --- |
| **cleartool man config\_spec** |

# ClearCase Support: Advanced Config Specs (Part 2)

Here is a sneaky shortcut when editing a Config Spec a lot. Don't!   
Instead, you can edit your Config Spec only once and have it point to a text file instead. 

|  |
| --- |
| **#element \* CHECKEDOUT include /tmp/config\_spec\_rules.txt #element \* /main/LATEST** |

Now you can more easily edit your Config Spec with your favorite editor, since it is now in a text file. This makes automating changes to your Config Spec much simpler too. Keep in mind that whenever editing the included file, the View will need to be alerted of a change using this refresh View command: 

|  |
| --- |
| **cleartool setcs -current** |

You can mix and match regular Config Spec rules with include rules in any order that you wish. And you can have any number of included files in your Config Spec, however you can not include a file from an include file. I assume this restriction is to prevent a possible infinite loop (i.e. an include file cannot include itself).

**ClearCase Support: Common ClearCase Config Spec Error Message**

Here is a common ClearCase error that I am asked to resolve about once a month or so. 

|  |
| --- |
| **$ cleartool edcs *Set config spec for view "VIEW\_NAME"? [yes] cleartool: Error: No registered VOB tag in path: "/VOB\_NAME/...". cleartool: Error: Config spec semantic processing failed. Errors during compilation - edit config spec? [yes]*** |

The problem is that a Windows Config Spec is being used on UNIX. Specifically, Windows VOB paths typically start with "\VOB\_NAME\" or "/VOB\_NAME/". For instance, a "tools" VOB will have a VOB path of "\tools\".   
  
Unfortunately, this is not a valid path for ClearCase in UNIX. On UNIX, we must use the VOB tag for the VOB path instead of just the VOB name. For example, the "tools" VOB will need to have the path of "/vob/tools/" or "/vobs/tools/". Therefore, to resolve the above error message, all you have to do is update each Windows VOB path in your config spec to use UNIX paths instead.   
  
Alternatively, the same error will happen if you try to import a UNIX config spec onto Windows. The solution is to convert the UNIX VOB paths to Windows VOB paths.

# ClearCase Support: How to Display Config Records

Have you ever had a derived object file that you wanted to know what files were used to create it? In other words, did you ever compile a file but wanted to know specifically what versions, elements, and View private files from the VOB and View went into creating the file?   
  
If you used clearmake to compile the file, then you can use this command to reverse engineer a list of all the files and directories that were used to compile the file in question.

|  |
| --- |
| ***cleartool catcr –flat -type fdl* DERIVED\_OBJECT** |

For example:

|  |
| --- |
| **$ *cleartool catcr –flat -type fdl stdio.exe***  **----------------------------**  **MVFS objects:**  **----------------------------**  **13 /vob/test/.@@/main/9 <13-Oct-98.12:23:00>**  **10 /vob/test/source@@/main/12 <17-Feb-00.08:46:23>**  **2 /vob/test/source/stdio.h@@/main/11 <02-Feb-00.19:37:24>**  **2 /vob/test/comm/stdio.c@@/main/5 <15-Dec-17.17:18:34>**  **...** |

I recommend that you save the output of this command to a file, since the output could be very large. Additionally, I recommend that you archive (source control) this file for future audit purposes.

# ClearCase Support: How to Diff Config Records

Comparing config records between two derived objects has always been easy. First, you run "[cleartool catcr …](http://www.philforhumanity.com/ClearCase_Support_30.html)" on both derived objects while saving the config records to two files and then just [diffing](http://www.philforhumanity.com/ClearCase_Support_13.html) both files.   
However, now this can be achieved with this single command.

|  |
| --- |
| **cleartool diffcr -element\_only -flat –short "DERVIED\_OBJECT\_1" "DERVIED\_OBJECT\_2"** |

You can even run this command on two derived objects in different Views like this example:

|  |
| --- |
| **cleartool diffcr -element\_only -flat –short "/view/VIEW1/vob/test/filename.exe" "/view/VIEW2/vob/test/filename.exe"** |

Do not forget to redirect the output to a file, since the output can be quite long.

**ClearCase Support: Displaying Comments**

I am quite often asked how to display ClearCase comments from the command line. The basic command is this: 

**cleartool describe -fmt 'Name:\t%En\nComment:%c' *PATH\_OF\_ELEMENT\_OR\_VERSION***

This command will let you see the command of any comment for any version in the version tree or whatever version your config spec is configured to access. For example, if you wanted to run this command on an element:

|  |
| --- |
| ***$ cleartool describe -fmt 'Name:\t%En\nComment:%c' filename.txt Name: filename.txt Comment:This is the comment of the version that my ClearCase View is accessing.*** |

Alternatively, you can run it on a specific version, such as: 

|  |  |
| --- | --- |
| ***$ cleartool describe -fmt '\tName:%En\nComment:%c' filename.txt@@/main/1  Name: filename.txt  Comment:This is the comment of version /main/1 of element filename.txt.***  **ClearCase Support: ClearAudit**  Ever want to auto-generate a config record on Windows yet you were not using clearmake, so you did not know how?   There is a command, called "clearaudit", that will record each and every version accesed (read or write) including parent directories.   To use this command, just run "clearaudit" from the DOS command line, and it will open a sub-shell. Now, no matter which window or program that accesses any ClaerCase data, the version list will be record in a master config record. Just type "exit" in the original DOS command line to stop collecting config record data.  If you want to run clearaudit on a command or script and then automatically exit from clearaudit when done, you can use this command:   |  | | --- | | **$ clearaudit /c [command]** |   The only problem with this command is that it is very slow, especially when exiting. |