ClearTrigger/ClearTrigger Lite
Version 13.1 Administration Guide

Audience

The **ClearTrigger Administration Guide** is intended as a reference for those responsible for configuring ClearTrigger, ClearTrigger Lite or ClearQuery or for help writing triggers or policy. These individuals are often different than those responsible for actually installing ClearTrigger. It is expected that this document is not made highly available and that copies might reside primarily with CM personnel responsible for creating and maintaining policy/triggers for ClearCase users.

For installation information, refer to the **ClearTrigger Installation Guide**.

Product Demarcation

This document covers both **ClearTrigger** and **ClearTrigger Lite**. As of Version 12.0 both products are combined in the same executable. ClearTrigger contains significantly more features the ClearTrigger Lite and they use a similar clearbits file; the license key in that clearbits file determines if the executable provides all of the ClearTrigger functionality or only the ClearTrigger Lite functionality. Within this document ClearTrigger may sometimes be used to mean both ClearTrigger and ClearTrigger Lite. When a feature applies only to ClearTrigger or ClearTrigger Lite it will be so noted.

Supports both ClearCase by IBM and VersionVault by HCL

ClearTrigger was initially created to support ClearCase 3.2 (as introduced by Atria Software) through the latest versions of ClearCase (introduced by IBM) as well as all versions of VersionVault (as introduced by HCL).

For clarity and brevity where we will say to “ClearCase” to both “ClearCase” (the IBM product) and “VersionVault” (the HCL product) collectively.

Introduction

ClearTrigger Overview

ClearTrigger is a custom add-on to IBM/Rational Software's ClearCase product. It serves to simplify the task of maintaining triggers and policy for your ClearCase installation. It is flexible in that it allows the user to enforce separate sets of triggers across different projects, teams or operating systems. ClearTrigger enables organizations to add triggers for specialized processing requests. It conveniently integrates with your current ClearCase installation and does not require moving VOBs to new machines or reorganizing your network. Upon installation and configuration of ClearTrigger, it is ready to use. ClearTrigger is an efficient way to manage current and existing triggers for the enterprise. Though, it does not stop there! ClearTrigger **also provides policy** so you do not have to create it. Functionality bits are provided to ensure that the most used and requested policy by ClearCase administrators around the world are already coded into ClearTrigger. Simply turn the bit on.

ClearTrigger makes use of basic, straightforward instructions to apply triggers. The dilemma of creating triggers using ***cleartool mktrtype*** is circumvented. Additional features were also added and some restrictions that existed with native ClearCase trigger definitions were either removed or relaxed.

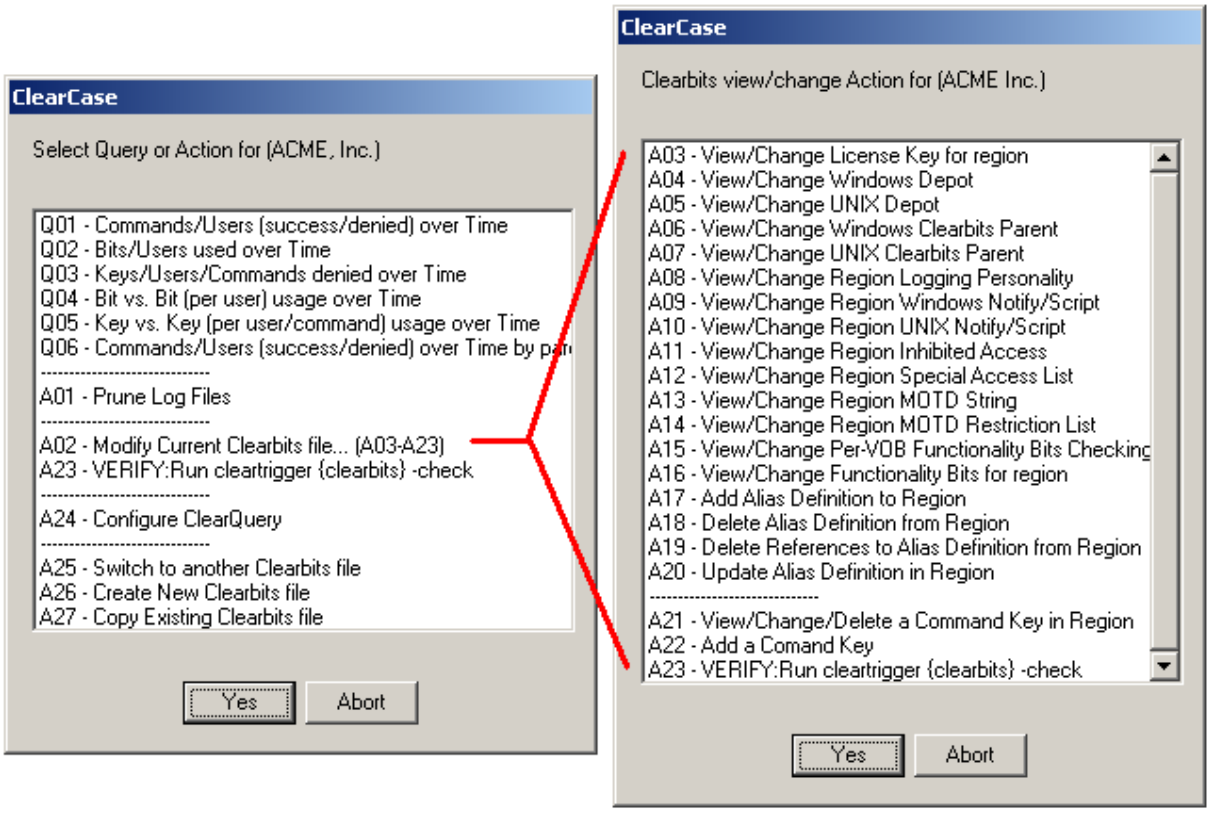
ClearTrigger's simple method of implementation is accomplished through the use of configuration files called ***clearbits_file***. The ***clearbits_file*** does not have to be named such, but for the sake of continuity within the document, all configuration files shall be referred as "***clearbits_file.***" The ***clearbits_file*** file allows you to define up to two different policy depots. The policy depot contains all of the trigger code to be managed by ClearTrigger. You may setup policy depots for UNIX and Windows if your site requires. There is no limit to the amount of *clearbits_files* that your organization can have. You may create multiple configuration files to divide your policy depot among multiple teams or departments, thereby creating ***ClearTrigger regions***. Suppose you wanted specific VOB groups to have a specialized high security policy versus the entire organization. This, too, can be easily done with ClearTrigger.

Using ClearTrigger is faster and simpler than going to each of your organizations VOBs to redefine triggers to restrict certain users or groups (as done when using the *-nusers* option on the *mktrtype* command). It also eliminates the need to update multiple script files to prohibit individual users belonging to UNIX or Windows groups from access the VOBs or VOB commands. VOBs do not have to be moved to new machines or networks reorganized. Simply install and configure ClearTrigger and get started!

ClearTrigger Lite is a custom add-on to the IBM Rational ClearCase product. It provides about 1 half of the Functionality of ClearTrigger including the highly popular functionality bits, region-wide inhibit features as well as simple CPDs which allow a region wide command/user availability matrix.

ClearQuery Overview

ClearQuery is a ClearTrigger add-on providing methods to test the usefulness of ClearTrigger policy by providing a query mechanism allowing you track successes and failures (something ClearCase does not do). You can query by commands, users, functionality bits, command keys or all of the above. The queries return easy to read graphs that allow you to quickly decide if a policy is working for your organization or if someone may need extra help/training.



You can view some sample reports below:

- Q01
  - Q01 (all commands/all users)
  - Q01 (for single command/all users)
  - Q01 (for single command/single user)
- Q02
  - Q02 (all bits/all users)
  - Q02 (for single bit/all users)
  - Q02 (for single bit/single user)
- Q03
- Q04
- Q05
- Q06

New or Changed in ClearTrigger 13.0 and 13.1 Bundles

ClearTrigger 13.0 was a major release that includes a few minor defect fixes in addition to internal changes that allows for increased ClearTrigger performance. ClearTrigger 13.1 was a minor release that added some features to 13.0, improved performance as well as addressed some defects Significant features were added or modified to address often requested features.

ClearTrigger was enhanced to add functionality, increase speed and simplify its use. The **ClearTrigger 13.0 Bundle** consists of the following: **ClearTrigger 13.0, ClearQuery 13.0** and **ClearApply 13.0.** Further, ClearTrigger is fully integrated with **ClearWeb 5.0 and higher** and **ClearReplica 2.0** and higher.

New "ClearCase Region" and "Environmental Variable" Processing

ClearTrigger was modified so that existing **Restriction List** that could execute policy based on **Users**, **Groups**, **VOBs**, **Elements**, **Views** or **Times** can now also limit base on **ClearCase Regions** or "**Environmental Variable Value**" Processing as well. This functionality was added to:

- Region Inhibited List
- Region Special Access List
- MOTD Restriction List
- Access List
    - Disallow List
    - Allow List
    - Ignore List
- Trigger List
    - Not List
    - Only List

Additional ClearTrigger Environmental Variables defined

ClearTrigger was modified so that **eight (8) additional ClearTrigger Environmental Variables** were added for use in your own trigger scripts. These were the most often requested variables and are available for your use in your triggers without you having to do anything to calculate them yourself.

There are now **30 ClearTrigger Environmental Variables** with the new ones added below:

- **CLEARTRIGGER_CLEARCASE_REGION_NAME**
- **CLEARTRIGGER_MOTD_RESTRICTIONS**
- **CLEARTRIGGER_MOTD_SHOWING**
- **CLEARTRIGGER_MOTD_STRING**
- **CLEARTRIGGER_REGION_NAME**
- **CLEARTRIGGER_VOB_ACCESS**
- **CLEARTRIGGER_VOB_FEATURE_LEVEL**
- **CLEARTRIGGER_VOB_HOST**

Added additional ClearTrigger Functionality Bit Application Aliases

ClearTrigger added **ABS_exclude_for_bit_ALL** and **ABS_only_for_bit_ALL** to the existing **ABS_exclude_for_bit_nn** and **ABS_only_for_bit_nn Functionality Bit Application Aliases**.

Added "group" option for Functionality Bit #01

ClearTrigger enhanced **Functionality Bit #1** (Auto Chown to VOB owner – or another user) to also allow for **group** ownership settings.

Added "Override" option for Functionality Bit #17

ClearTrigger enhanced **Functionality Bit #17** (Auto Add Execute Permission) to add or remove the file extensions that are affected by Functionality Bit #17.

Added "Auto Answer" option for Functionality Bit #21

ClearTrigger enhanced **Functionality Bit #21** (Auto Warn lost+found) to provide an "auto answer" option so that ClearTrigger policy makers can allow users to pre-answer the warning dialog so that this functionality can execute in a build or other automated process without human intervention.

Added "Override" alias ABS_allow_use_of_local_ccperl

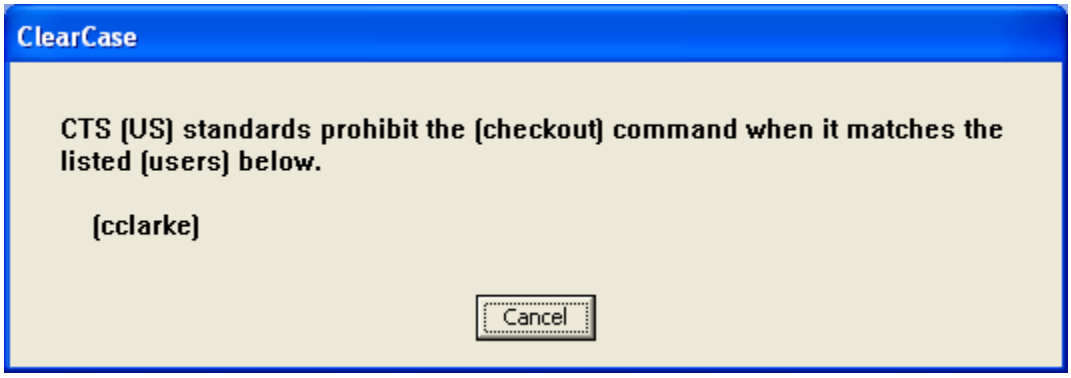
ClearTrigger added the **ClearTrigger Override Alias** **ABS_allow_use_of_local_ccperl** to improve performance. If defined in the clearbits file it allows the use of local **ccperl.exe**, **CQerl.exe** and **cleartool.exe** executables instead of ones defined in the depot for **windows script** definitions with policy key definitions..

Improved Performance (faster, smaller, shared memory improved caching)

Many ClearTrigger core functions were internally recoded to for speed and size reduction. ClearTrigger performs faster and smaller so it transfers over the network faster when needed. Additional memory and file caching were added as well to again improve its speed. A ClearTrigger server process was added to optimize data collection and hold queried data in memory between calls of ClearTrigger process for their use. Many network calls were replaced with shared memory calls resulting in improved performance for both ClearTrigger and the network.

Improved Messaging

ClearTrigger messages were examined and rewritten for clarity. Many messaged the previously referred restriction list listed all components that could match causing the denial. These denial dialogs are now more specific so that they list the matching component only.



Improved ClearQuery Reporting and Web-pages

ClearQuery was enhanced so that the generate web-pages it produces have an equivalent look as before, but the pages no longer rely on images within the images directory, this make all of its report web-pages completely re-locatable. The ClearQuery defined alias **(#!clearquery_pathing)** has been depreciated.

Additionally, all pages are smaller, load faster and have been rewritten to work for HTML5.

Improved ClearQuery Error Checking and Data Input

ClearQuery was enhanced so that additionally error checking is performed during data input to prevent errors in the clearbits file. The clearbits file is checked between the input of data items to always ensure valid relationships.

Depreciated the Use of the ABS_EDITOR Environmental Variable

As of ClearTrigger 13.1 the use of the environmental variable **ABS_EDITOR** is no longer used or supported.

Any Up-to-date Patches

ClearTrigger 13.0 was a ***major*** release consisting of ClearTrigger Performance enhancements for ClearTrigger Users. **ClearTrigger 13.1** is a ***minor*** release built on ClearTrigger 13.0 with additional features and performance improvements. All previously created patches for ClearTrigger are incorporated into ClearTrigger 13.1. The most up-to-date defect/enhancement lists for ClearTrigger may be viewed at the following link:

https://www.abs-consulting.com/_products/products_cleartrigger.shtml?patches

Upgrading from ClearTrigger Version 12.x to 13.1

Upgrading form ClearTrigger 12.x to 13.1 is painless and should only take 1-2 minutes.

Read about how to migrate from previous 12.x versions to 13.x in the section entitled Migration from 12.x versions to 13.1.

Upgrading from ClearTrigger Version 13.0 to 13.1

Upgrading form ClearTrigger 13.0 to 13.1 is also painless and should only take a few seconds.

Read about how to migrate from previous 13.0 versions to 13.1 in the section entitled Migration from 13.0 version to 13.1.

Customer Support

To obtain additional information on ClearTrigger or other services offered by A Better Solution, Inc., visit our web site at www.abs-consulting.com. To report problems with the ClearTrigger software or documentation, please send an e-mail to cleartrigger_team@abs-consulting.com.

Supported Platforms

ClearTrigger integrates with ClearCase installations of 3.2 and higher as well as all installations of VersionVault. A single installation of ClearTrigger can typically serve an entire organization. The system requirements for ClearTrigger are consistent with those used to run ClearCase versions 3.2 and higher. ClearTrigger will operate in a mixed version environment as well as a mixed OS environment. The following operating systems are supported for ClearTrigger:

- **WINDOWS:**
  - **Windows 9x**
  - **Windows NT**
  - **Windows 2000 and higher**
- **UNIX:**
  - **Solaris**
  - **SunOS**
  - **HP-UX**
  - **RedHat Linux**
  - **AIX**
  - **CentOS**
  - **Ubuntu Linux**
  - **SUSE Linux**
  - **Solaris X86**

Installing ClearTrigger

For most installation issues refer to the **ClearTrigger Installation Guide**. In that document you can read about how to install ClearTrigger and ClearQuery. Some minor client considerations are outlined in the following section.

Client setup

ClearTrigger requires the following components for setup:

UNIX & Windows:

- That ClearCase is properly installed on the client machine. Without ClearCase, there is no need for ClearTrigger as ClearTrigger relies on ClearCase to function.

- A well-known temp directory must be found or set. ClearTrigger will use the value of either the **TMP** or **TEMP** variable if set. If neither of those variables is set, it tries writing to the default temp directories for the operating system (**/tmp** or **/temp** if running on UNIX or **c:\tmp** or **c:\temp** for Windows). If neither the TMP nor TEMP environmental variable is set and writable AND the default temp directories for the operating system are not writable, ClearTrigger displays the following dialog. Because this is also a requirement of ClearCase, a proper ClearCase installation would not yield this error.

Windows Only:

The following changes are not required to run ClearTrigger unless the policy maker uses <group names> in any of the access lists in the **clearbits_file** (e.g. Region Inhibited List, Region Special Access List etc.). If you don't use this functionality, skip this section.

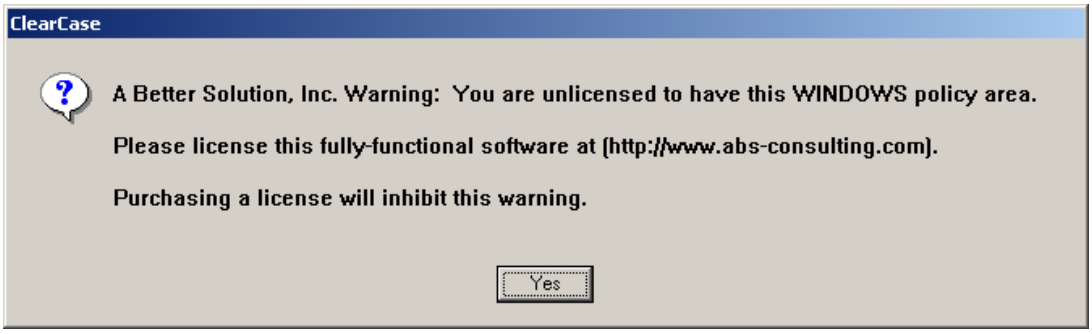
- The **CLEARCASE_PRIMARY_GROUP** variable must be set (a requirement of ClearCase checked for by ClearTrigger). Because ClearTrigger may be used to create enterprise group policy, this variable is checked against the ***creds*** information. If this function is not set and ClearTrigger reads a command_key definition requiring group checking, he user will get the error message below:

- Place a copy of the **creds.exe** distributed by Rational Software in the cleartrigger windows policy depot.

License Registration

A Better Solution, Inc. distributes user licenses for ClearTrigger on a per architecture basis. For example, to implement ClearTrigger in a UNIX and Windows InterOp environment, two licenses are required. Before contacting A Better Solution, Inc. to request a license, consider how many licenses you will need. You will receive a license request form when you purchase ClearTrigger from A Better Solution, Inc. Return the completed form to A Better Solution, Inc. via e-mail or fax. Sites may proceed with installation and configuration of ClearTrigger in the interim while a license is being obtained (ABS actually encourages this "try it first" method). ClearTrigger is distributed as fully functional software with an **evaluation license key**.

If necessary, sites may also operate using an evaluation license. Once the license key is obtained, place it in the ClearTrigger *clearbits_file*. Use of an evaluation license will cause random ***Splash Messages*** from the vendor similar to the one below. *Splash Messages* will not change the functionality of ClearTrigger.



Configuring ClearTrigger (clearbits_file)

The **clearbits_file** is the configuration file used by ClearTrigger. It holds ClearTrigger licensing and ClearTrigger region configuration information used to find, enforce and execute ClearTrigger policy for the region. The **License String** itself will determine if the ClearTrigger uses "ClearTrigger Feature Set" or the "ClearTrigger Lite Feature Set". This file must be properly populated prior to running ClearTrigger. You can populate this file manually or by using **ClearQuery**. The following information is used to configure the **clearbits_file**. Refer to the individual sections of this document for descriptions and proper formatting of this information in the configuration file. A sample **clearbits_file** file is shown in the section entitled **A sample Clearbits_file** and would contain the following ClearTrigger items for a ClearTrigger or ClearTrigger Lite Region:

Clearbits Feature	ClearTrigger Feature Set	ClearTrigger Lite Feature Set
License String(s)	✓	✓
ClearTrigger Policy Depot(s)	✓	✓
Parent Clearbits File(s)	✓	✓
Region Logging Personality	✓	✓
Mail Notification Programs	✓	N/A
Region Inhibited List	✓	✓
Region Special Access List	✓	✓
MOTD Definition : MOTD Restriction List	✓	✓
Functionality Bits	✓	✓
Command Key Definitions	✓	N/A
Command Permission Definitions	N/A	✓

*** Where ✓ denotes the feature is in the feature set.**

To view a sample **Clearbits file** for a **ClearTrigger Region** refer to the section entitled:
A sample Clearbits_file (ClearTrigger Feature Set).

To view a sample **Clearbits file** for a **ClearTrigger Lite Region** refer to the section entitled:
A sample Clearbits_file (ClearTrigger Lite Feature Set).

ClearTrigger Regions

Before examining the contents of the ***clearbits_file*** it is prudent to spend time defining ***ClearTrigger Regions*** in more detail.

A *ClearTrigger Region* is the combination of a region name and a path to the *policy depot*, both defined in the ***clearbits_file***. Any number of ClearTrigger Regions can be defined to subdivide the VOBs in your organization. *ClearTrigger Regions* are completely unrelated to *ClearCase Regions*. VOBs in different ClearCase regions can be part of a single *ClearTrigger Region* and vice-versa.

ClearTrigger adds flexibility within the ***clearbits_file*** by allowing you to define up to two different **policy depot** paths (where scripts or executables you want called when certain ClearCase actions fire reside). You may choose to establish individual paths for UNIX and Windows. This is required in a ClearCase InterOp configuration (where a VOB may be accessed from both UNIX and Windows operating Systems) because there are two path-naming conventions across the different OS partitions (UNIX/Windows). It is acceptable that these paths might point to the same physical location if you are using some NFS or Samba like software. It is also possible to create multiple clearbits.txt files that utilize the same policy depot. There is no limit to the number of clearbits.txt files that you can have in your organization. Create any number of clearbits.txt files to divide your policy across projects, teams, department, or organizations.

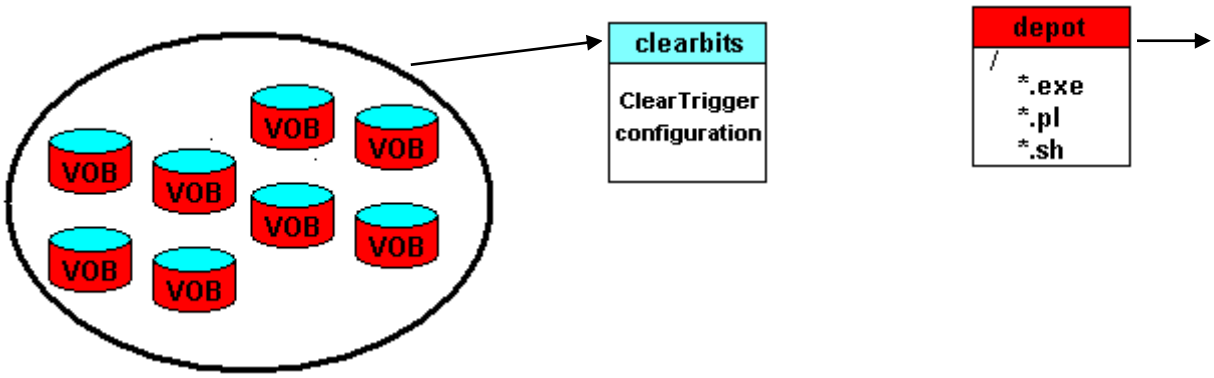
Consider a *ClearTrigger Region* as the ***region name:policy depot*** pair. Keep in mind that it is possible to have multiple ***clearbits_files*** (free of charge) and multiple policy depots (at a nominal cost). Given this, it is easy for an organization to create many *ClearTrigger Regions* that utilize the same *policy depot*.

Regardless of whether your policy depot is empty, it must be defined. ClearTrigger provides some policies that can be implemented for a *ClearTrigger Region* that does not require code in the depot to be called. ClearTrigger allows you to create ***Command Commenting Standards***, create a ***User/Command Disallow Matrix***, and set up ClearTrigger ***Region Inhibited Lists*** without having to call scripts or executables (Further information on this topic will be provided later).

Some examples of ClearTrigger Regions, associated policy depots and clearbits.txt files follow:

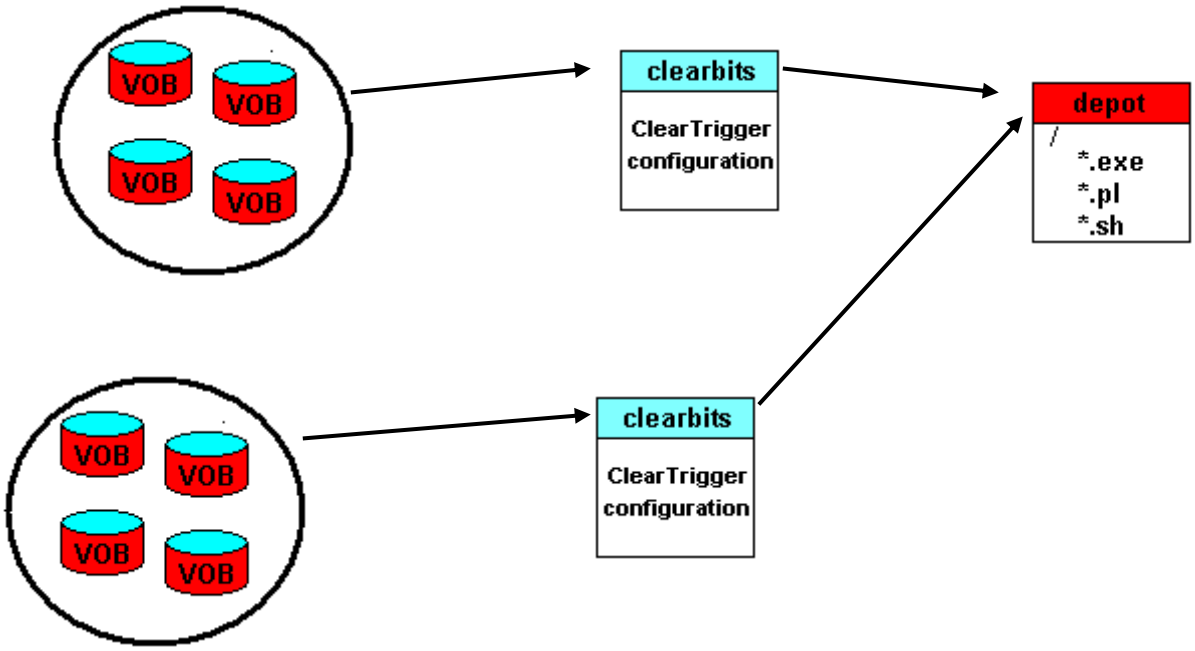
One Region – One Operating System

All of your VOBs use a single ***clearbits_file*** (which points to a *policy depot*) creating **one** ClearTrigger Region. All VOBs in this *ClearTrigger Region* can now have the same policy implemented. **This would require only one license.**



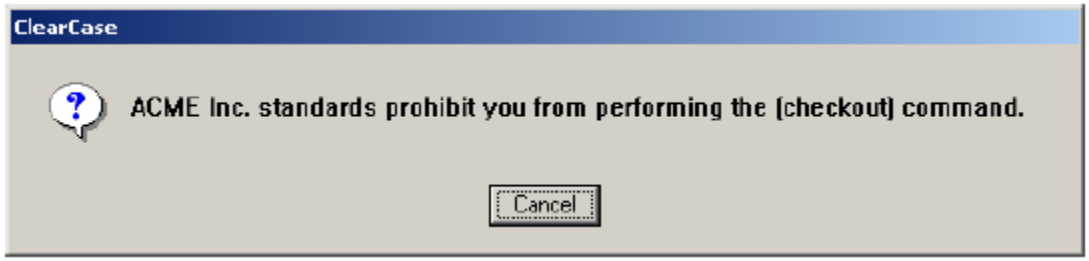
Two Regions – One Operating System

It is possible to split your VOBs between two ***clearbits_files*** (which point to a single *policy depot*), thereby creating **two** ClearTrigger Regions. All VOBs in both *ClearTrigger Regions* rely on the same policy depot for corporate label naming standards or code requiring a defect tracking number. This configuration is ideal when you need to enforce separate policies in the organization. For example, when different users are allowed access to only one region and must still enforce the global policy created for the entire organization.



This configuration requires either one or two licenses. Remember, a license key is based on a unique path to the depot and the region name (e.g. the name you want to appear in the dialog windows). You can have two regions with the same name in ClearTrigger. Therefore, if you name these regions "Acme, Inc - (Financial Division)" and "Acme, Inc - (Commerce Division)" two licenses would be required, but if both regions were named "Acme Company, Inc.," only one license is needed.

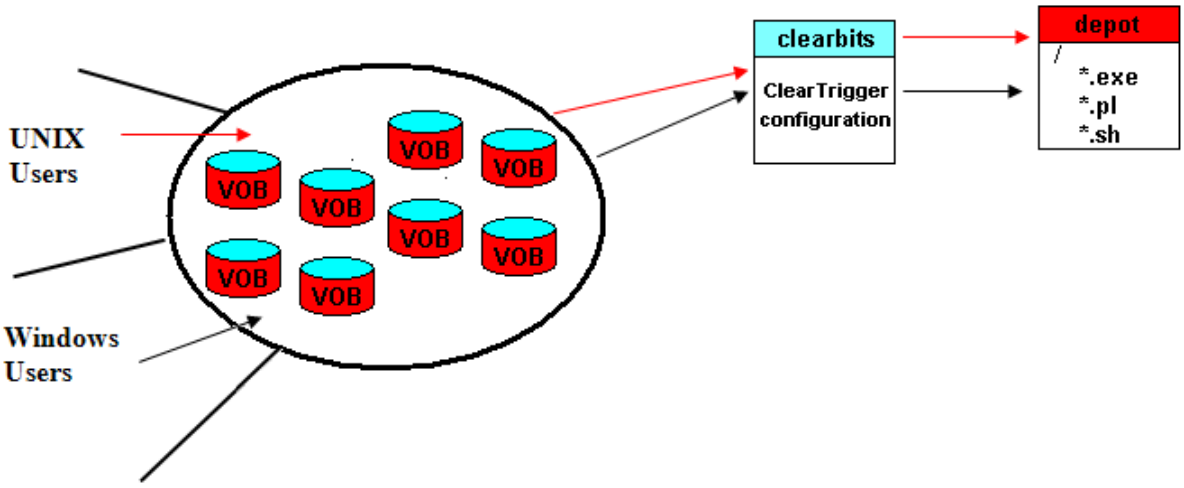
If both regions should see dialogs similar to the one below, then one license would suffice.



Defining the regions such that both were to see different dialogs similar to the dialogs below would require two licenses.

InterOp Configuration – Scenario 1

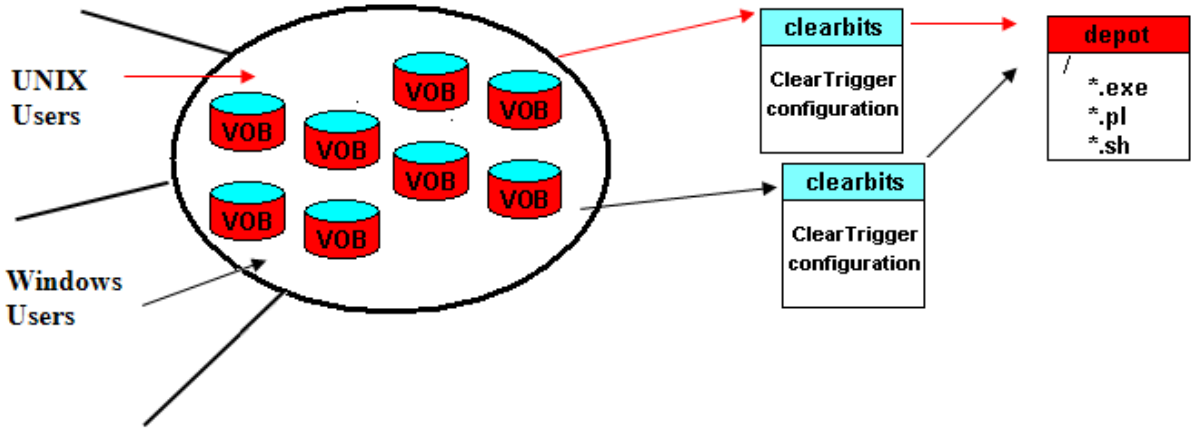
When in an InterOp configuration, it is possible that all VOBs use a single ***clearbits_file*** (through NFS or Samba), which points to a single policy depot and you have in essence created **two** ClearTrigger Regions. All VOBs in the *ClearTrigger Regions* can have the same policy implemented.



This configuration requires two licenses. Remember, a license key is based on a unique path to the depot **and** the region name (e.g. the name you want to appear in the dialog windows). Though, the depot for both regions point to the same physical location the windows path and UNIX path to that location will differ (e.g. the windows path might be **\\earth\policy\depot** while the UNIX path might be **/net/earth/policy/depot**).

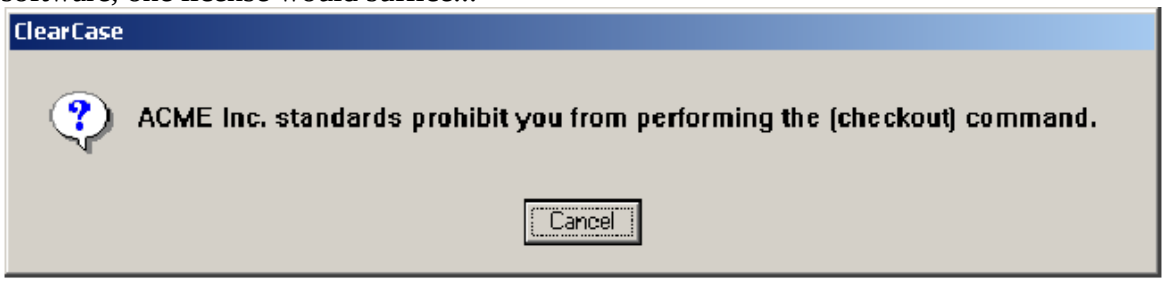
InterOp Configuration – Scenario 2

When in an InterOp configuration, it is possible to have all VOBs simultaneously using two ***clearbits_files***, (one for each OS). Each ***clearbits_file*** points to the same policy depot (through NFS or Samba) creating **two** ClearTrigger Regions. All VOBs in the *ClearTrigger Regions* can have the same policy depot created for the organization, while simultaneously enforcing slightly different policies for the Operating Systems (e.g. Documentation people can only work from the Windows side).

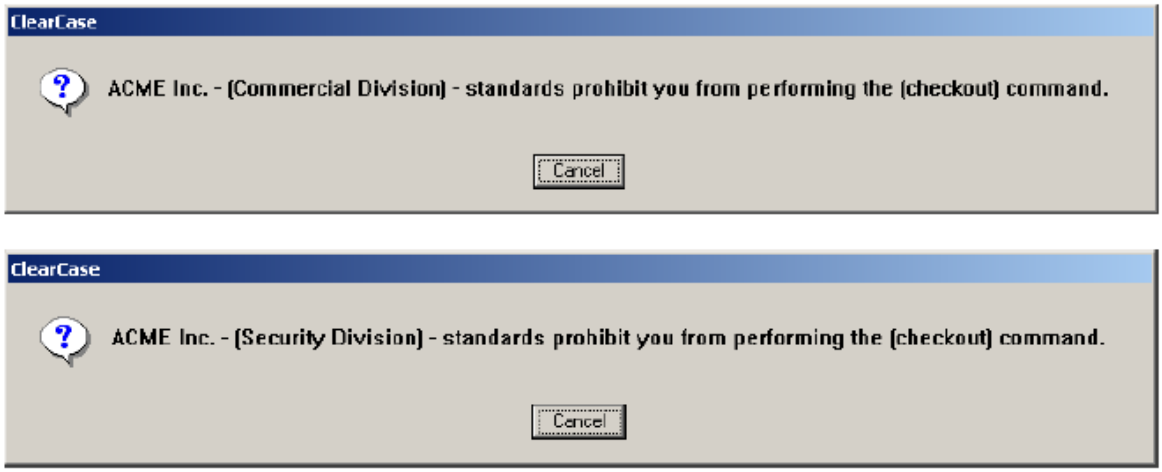


This configuration requires either one or two licenses. Remember, a license key is based on a unique path to the depot **and** region name (e.g. the name you want to appear in the dialog windows). You **can** have two regions with the same name in ClearTrigger. Therefore, if you name these regions "Acme, Inc - (UNIX development)" and "Acme, Inc - (Windows development)" you would need two licenses. If you decided to name both regions "Acme Company, Inc." two licenses would still be required because there are two paths to the single depot. If you have software that allows you to access the depot from both the UNIX side and Windows side using the same path name, only one license is required.

If you wanted both regions to see dialogs similar to the one below **and** had appropriate software, one license would suffice...

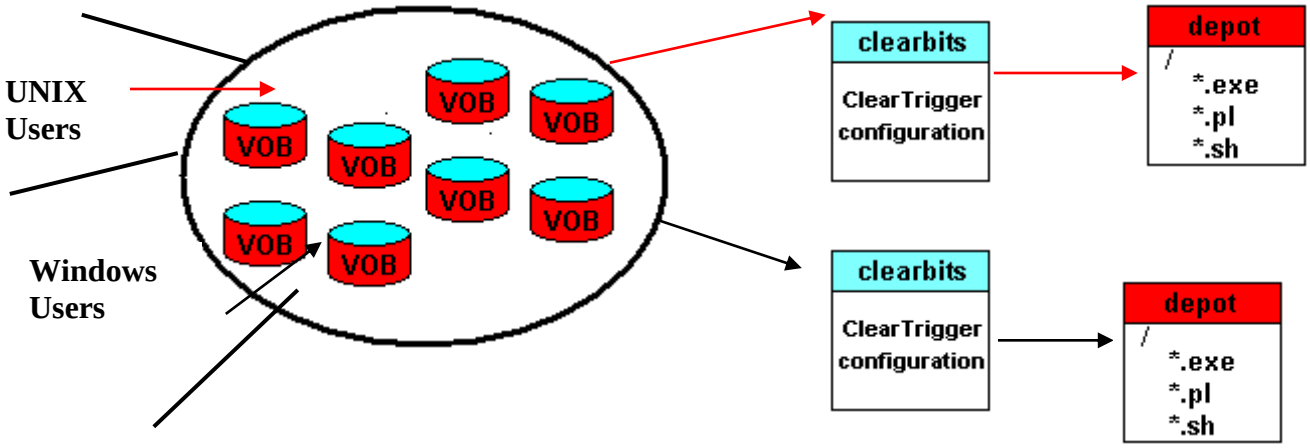


If you wanted to define regions such that they see different dialogs as shown below, two licenses are required.



InterOp Configuration – Scenario 3

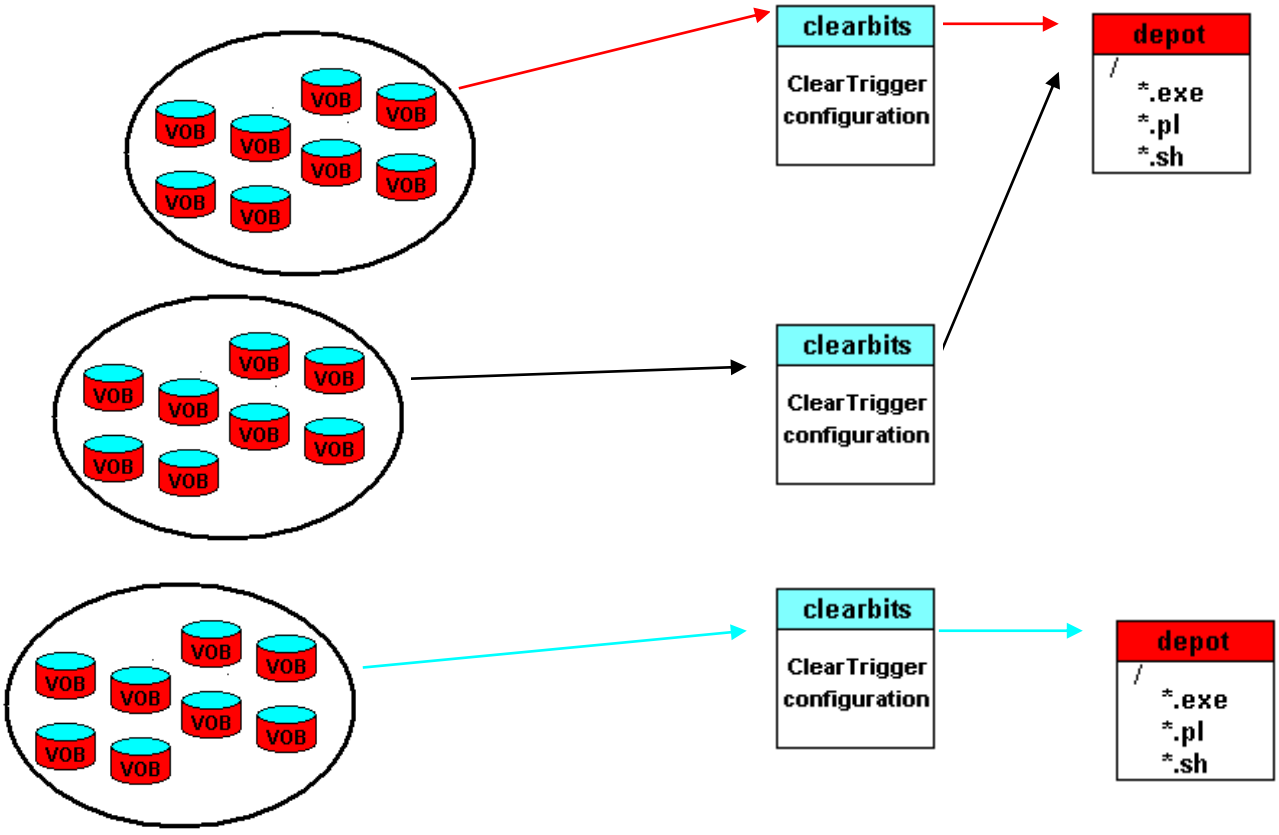
When in an InterOp configuration, it is possible to have VOBs using two *clearbits_files*, (one for each OS), with each one pointing to a different policy depot thus, creating **two** ClearTrigger Regions. This will allow the VOBs to have different trigger code executed for different OS partitions (UNIX/Windows). **This would require only two licenses.**



Multiple Regions & Operating Systems

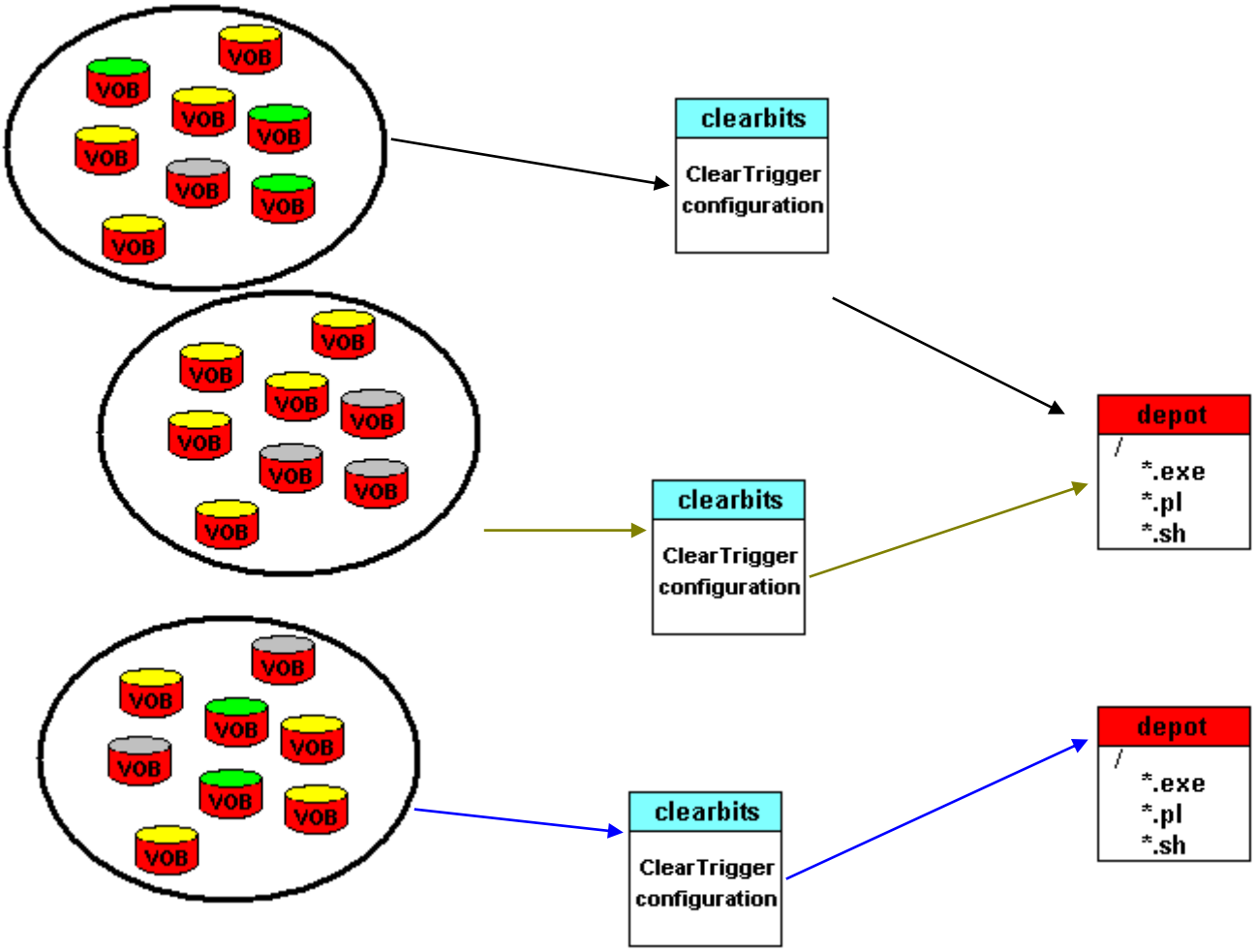
There are not many configurations that would ever require more than 2 licenses of ClearTrigger. Generally one for UNIX and one for NT is all an organization would need. However, if there are policy ownership lines in your organization (political/contractual in nature) more licenses can be purchased to fit the need.

Perhaps there is a group or individual that "handles" the "east campus" VOBs and another group or individual that handles the "west campus" VOBs, but they share a policy depot. There is also another group that must create policy for "COTS" VOBs (third-party vendor code) with totally different code branching, labeling and code promotion strategies. Perhaps it is decided that "John" (or a group of "Johns") is allowed to do anything in the "east campus" VOBs but is very limited in the "west campus" VOBs and is completely restricted on the "COTS" VOBs. Implementing many ClearTrigger Regions, clearbits_files and policy depots might create the solution. The configuration below requires **two or three licenses (depending on how regions are named).**



ClearTrigger Regions vs. ClearCase Regions

There is no association between *ClearTrigger Regions* and *ClearCase Regions*. It is acceptable for VOBs in different ClearCase regions to be part of the same *ClearTrigger Region* and vice-versa. VOBs in another ClearCase region (none of which are seen by default as a result of the cleartool **lsvob** listing) might have the same company-wide policy implemented. In the example below, VOBs in different *ClearCase Regions* (**Green**, **Yellow**, and **Grey**) exist in different *ClearTrigger Regions* (**blue**, **brown**, **black**).



A sample Clearbits_file (For ClearTrigger)

```
#####################################################################
# ClearTrigger License File
#####################################################################

############### ClearQuery Defaults ##################
#!clearquery_unix_cleartrigger_check=/net/ccserv/policy/cleartrigger.exe
#!clearquery_windows_cleartrigger_check=\\ccserv\policy\cleartrigger.exe
#!clearquery_unix_browser=/net/machine/dir/netscape
#!clearquery_windows_browser=C:\Program Files\Internet Explorer\IEXPLORE.EXE
#!clearquery_relocated_logging_dir_unix=
#!clearquery_relocated_logging_dir_windows=
#!clearquery_relocated_results_dir_unix=
#!clearquery_relocated_results_dir_windows=
#########################################################

############### ClearTrigger Aliases #################
#!cleartrigger_alias interns (user1) (user2) (user14)
#!cleartrigger_alias windows_vobs [\*]
#!cleartrigger_alias special_types brtype<main> lbtype<REL#.#*>
#!cleartrigger_alias qa_vobs [*qa] [test*] [*tutor*]
#!cleartrigger_alias dev_regions 'development' 'US_*'
#!cleartrigger_alias during_builds ^release_build^ ^QA_build^

#########################################################

#####################################################################
# ClearTrigger License String
#
# The license string contains a:
# Preamble, Windows License Key, UNIX License Key and Company Name
# when generated by the vendor.
#####################################################################
cleartrigger ABS licman;123456789;123456789;ACME Inc. - Region1;

#####################################################################
# The ClearTrigger Policy Depots;Region_Logging_Personality
#
# The Depot definition is of the form:
# {windows_depot_path};{UNIX_depot_path};{region_logging_personality};
#####################################################################
\\Win2k_01\Policy\region1_depot;/net/earth/policy/region1_depot;O;

#####################################################################
# Parent Clearbits File
#
# The Parent Clearbits File definition is of the form:
# {windows_clearbits_parent};{UNIX_clearbits_parent};
#####################################################################
\\Win2k_01\Policy\ACME_bits.txt;/net/earth/policy/ACME_bits.txt;
```

##
 # Enterprise logging/mail notification programs. Provide paths
 # for both Unix and Windows operating systems. The directory where
 # the program is expected is the policy depot(s) defined above.
 # The last optional field is the regions logging flag.
#
 # The Notify Program definition is of the form:
 # {windows_notify_program};{UNIX_notify_program};
##
NOTIFY.bat;NOTIFY.pl;

##
 # Region Inhibited List
 # These users or <groups> are quickly prohibited from performing VOB
 # modifying actions in all enterprise VOBs within a ClearTrigger
 # region. Instead, they are shown a consistent dialog that clearly
 # states that enterprise standards prohibit them from performing
 # the action. Additionally, [vobs], [vob:replicas] can be listed to prevent any data modification
 # in those VOBs for anyone; {elements} can be listed to prevent any data modification
 # for those elements; %views% can be listed to prevent any data modification
 # from within those views; and &times& can be listed to prevent any data modification
 # during those times.
##
; (user3) (user4) <mgt> <iterns> [/vobs/a_vob] [\another_vob] ;

##
 # Region Special Access List
 # These users or <groups> are quickly exempt from complying to ClearTrigger
 # policy in all enterprise VOBs within a ClearTrigger region (essentially, turning
 # off ClearTrigger for those users or groups).
 # Additionally, [vobs], [vob:replicas] can be listed to exempt policy for anyone
 # in those VOBs; {elements} can be listed to exempt policy for those elements;
 # %views% can be listed to exempt policy from within those views; and
 # &times& can be listed to exempt policy during those times.
##
; (vobadm) <cmgrp> [/vobs/b_vob] [\cm_vob] ;

##
 # MOTD : MOTD Restriction List
 # This portion of the license file contains Message-of-the-day
 # functionality. You can place a message between the ';' and that
 # message will appear on many dialogs while using ClearCase.
 # The MOTD Restriction list will limit when the message is displayed.
##
;Final build for 12.0 tonight... Please check in all code.; (qa*) [/vobs/qa*];

##
 # Per-VOB Functionality Bits Checking ; Functionality Bits
 # This portion of the license file contains the functionality bits for commonly requested
 # policy; ClearTrigger already has that functionality compiled in it for speed.
 # Simply turn it on for the region or on a per-VOB basis.
##
D;00110000000000010001110;

```
######################################################################
# Enterprise Command Key Definitions:
#
#  cmd_id;RIN_type;RIN_list;inhibit_flag;comment_req;notify_flag;
#  access_list_type;access_list;trigger_list_type;trigger_list;
#  Series_begin_end;Windows_script;Windows_script_params;
#  UNIX_script;UNIX_script_params;inhibit_cleartrigger_GUI_bit;
#  dialog_type;dialog_prompt;dialog_mask|default_string|dialog_selection;
#
#  NOTE: Keys are evaluated in order of appearance;  cleartrigger
#        will process all matching keys or until the command
#        is prohibited.
######################################################################
pre_rmver;R;;X;
pre_chevent;R;;X;
pre_mkproject;R;S;
pre_mkelem;R;;0;5;
pre_rmelem;R;;0;0;x;
pre_checkout;R;;0;0;0;D; (fred) <usr> [/vobs/cm_vob]  ;
pre_mv;R;;0;0;0;A; (charles) [/vobs/documentation] <dev> ;
pre_checkin;R; brtype<main> brtype<*_int_*> ;0;20;
pre_checkin;R; lbtype<REL#.#*> brtype<rel#.#*> ;0;10;
pre_mkattr;I; attype<QA_State> ;0;20;
pre_mktype;N; brtype<test> ;x;
pre_mkbl;R;;0;0;0;D; ;O;<intern_*>;0;Inform.pl;;Inform.pl;
post_checkin;R;;0;0;0;D; ;O;^QA_build^ ;0;METRIX.pl;;METRIX.pl;
pre_checkin;R;;0;0;0;D; ;N; ;0;BUILD.pl;-D386 debug;BUILD.pl;-Dhp10;
pre_checkin;R;;0;0;0;D; ;O; {*.c} ~dev_regions ;0;DEV_STANDARDS.pl;;DEV_STANDARDS.pl;
pre_checkin;R;;0;0;0;D; ;N; ;1;GET_BUG.pl;;GET_BUG.pl;
pre_chtype;R;;0;0;0;D; ~special_types ;N; ~interns ;0;APPROVE.pl;;APPROVE.pl;;1;
pre_lock;R;;0;0;0;D; ;N; ;0;;;;;0;Q;Are you sure?;YQ;
pre_mktype pre_rmtype;R;;0;0;0;D; (inter*) <usr> [\p#] ;
post_MODIFY_TYPE;R;;0;10;0;D; (inter*) <usr> [\p#] ;
```

A sample Clearbits_file (For ClearTrigger Lite)

##
 # ClearTrigger Lite License File
##

################ ClearTrigger Aliases ##################
#!cleartrigger_alias interns (user1) (user2) (user14)
#!cleartrigger_alias windows_vobs [*]
#!cleartrigger_alias special_types brtype<main> lbtype<REL#.#*>
#!cleartrigger_alias qa_vobs [*qa] [test*] [*tutor*]
##

################ ClearQuery Defaults ###################
#!clearquery_unix_cleartrigger_check=/net/ccserv/policy/cleartrigger.exe
#!clearquery_windows_cleartrigger_check=\\ccserv\policy\cleartrigger.exe
#!clearquery_unix_browser=/net/machine/dir/netscape
#!clearquery_windows_browser=C:\Program Files\Internet Explorer\IEXPLORE.EXE
##

##
ClearTrigger Lite **License String**
#
The license string contains a:
Preamble, **Windows License Key**, **UNIX License Key** and **Company Name**
when generated by the vendor.
##
cleartrigger_lite ABS licman;123456789;123456789;ACME Inc. - Region1;

##
 # The ClearTrigger Policy Depots;Region_Logging_Personality
#
 # The Depot definition is of the form:
 # {windows_depot_path};{UNIX_depot_path};{region_logging_personality};
##
\\Win2k_01\Policy\region1_depot;/net/earth/policy/region1_depot;O;

##
 # Parent Clearbits File
#
 # The Parent Clearbits File definition is of the form:
 # {windows_clearbits_parent};{UNIX_clearbits_parent};
##
\\Win2k_01\Policy\ACME_bits.txt;/net/earth/policy/ACME_bits.txt;

```
######################################################################
# Enterprise logging/mail notification programs.  Provide paths
# for both Unix and Windows operating systems.  The directory where
# the program is expected is the policy depot(s) defined above.
# The last optional field is the regions logging flag.
#
# The Notify Program definition is of the form:
# {windows_notify_program};{UNIX_notify_program};
# NOTE: This field is ignored by ClearTrigger Lite
######################################################################
;;

######################################################################
# Region Inhibited List
# These users  or <groups> are quickly prohibited from performing VOB
# modifying actions in all enterprise VOBs within a ClearTrigger
# region. Instead, they are shown a consistent dialog that clearly
# states that enterprise standards prohibit them from performing
# the action.   Additionally,  [vobs], [vob:replicas] can be listed to prevent any data modification
# in those VOBs for anyone;   {elements} can be listed to prevent any data modification
# for those elements;  %views% can be listed to prevent any data modification
# from within those views; and   &times& can be listed to prevent any data modification
# during those times.
######################################################################
; (user3) (user4) <mgt> <iterns> [ /vobs/a_vob] [ \another_vob] ;

######################################################################
# Region Special Access List
# These users  or <groups> are quickly exempt from complying to ClearTrigger
# policy in all enterprise VOBs within a ClearTrigger  region (essentially, turning
# off ClearTrigger for those users or groups).
# Additionally,  [vobs], [vob:replicas] can be listed to exempt policy for anyone
# in those VOBs;   {elements} can be listed to exempt policy for those elements;
# %views% can be listed to exempt policy from within those views; and
# &times& can be listed to exempt policy during those times.
######################################################################
; (vobadm) <cmgrp> [ /vobs/b_vob] [ \cm_vob]  ;

######################################################################
# MOTD : MOTD Restriction List
# This portion of the license file contains Message-of-the-day
# functionality.  You can place a message between the ';' and that
# message will appear on many dialogs while using ClearCase.
# The MOTD Restriction list will limit when the message is displayed.
######################################################################
;Final build for 12.0 tonight... Please check in all code.; (qa*) [ /vobs/qa*];

######################################################################
# Per-VOB Functionality Bits Checking ; Functionality Bits
# This portion of the license file contains the functionality bits for commonly requested
# policy; ClearTrigger already has that functionality compiled in it for speed.
# Simply turn it on for the region or on a per-VOB basis.
######################################################################
D;00110000000000010001110;
```

```
#####################################################################
# Enterprise Command Permission Definitions (CPDs):
#
#  Definition_Type Command_list User_list
#
#  NOTE: Keys are evaluated in order of appearance.  Cleartrigger Lite
#        will process all matching keys until prohibited.
#
#####################################################################
 D: .* ramesh fred jaun                 # Prevents Fred and Juan from doing anything in the region
 D: .checkout sally tom                 # Prevents Sally and Tom from performing checkout in the region
 D: .checkout .checkin tom bob          # tom and bob can't do checkouts or checkins in the region
 D: .chevent *                          # No one can perform a chevent in the region
 A: .rmelem charles scott               # Only Charles and Scott can perform rmelem in the region
 A: .mktype .rmtype robert david        # Only Robert and David can make/remove types in the region
```


Checking your clearbits file

While it is always possible to use **ClearQuery** to create or modify a ***clearbits_file***, when users create or changed the clearbits files by hand, this increases the amount of errors in the file. ClearTrigger allows users to check the file at any time by using the –check option on ClearTrigger. The checking option was added in order for the policy maker to find incorrect keys, bits or fields before users do. It is also possible to modify an existing clearbits file using the new **ClearQuery** application.

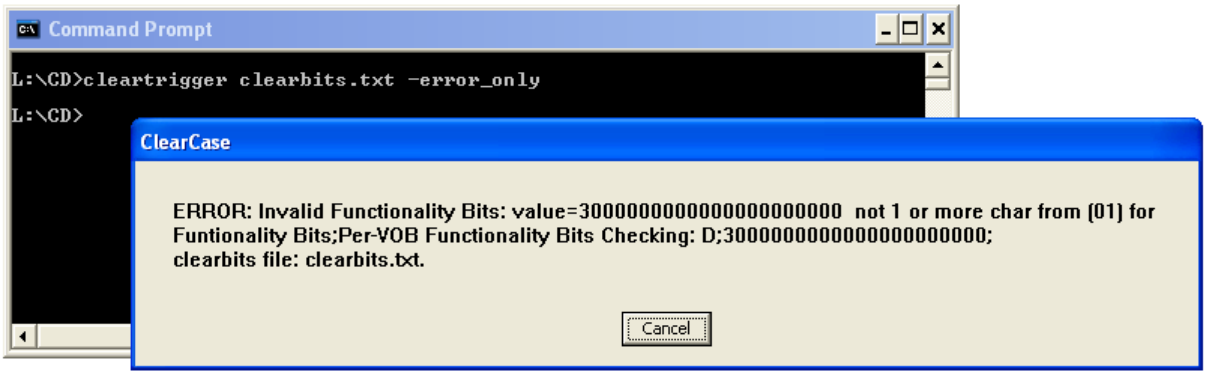
Invoking **cleartrigger.exe {*path_to_clearbits_file*} -check** will perform the sanity checking on the clearbits file and display the checking output in a clearprompt file display dialog. The return code returned is 0 if the file has no "syntax" and non-zero if the file has any "syntax" errors.

Invoking **cleartrigger.exe {*path_to_clearbits_file*} -dump** will perform the sanity checking on the clearbits file and display the checking output to stdout. The return code returned is 0 if the file has no "syntax" and non-zero if the file has any "syntax" errors.

Invoking **cleartrigger.exe {*path_to_clearbits_file*} –status_only** will perform the sanity checking on the clearbits file and return a calculated return code. The return code returned is 0 if the file has no "syntax" and non-zero if the file has any "syntax" errors.

Invoking **cleartrigger.exe {*path_to_clearbits_file*} -error_only** will perform the sanity checking on the clearbits file and display an error dialog for the first syntax error it encounters (if any) and also calculated return code. The return code returned is 0 if the file has no "syntax" and non-zero if the file has any "syntax" errors.

For example, if the clearbits file had an error with its functionality bits definition, invoking the command with the –error_only switch will result in a dialog similar to the one below:



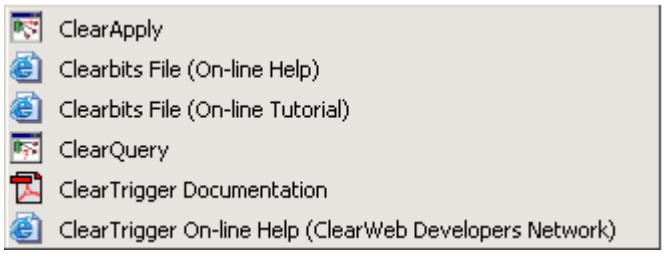
When using the "**-check**" or "**-dump**" switches the output displayed would look similar to the output that follows:

```
Read and assigned alias: contractors=(fred) (remesh) (bill)
Read and assigned alias: QA_vobs=[*_qa] [/pra_mock]
Read and assigned alias: protected_types=brtype<main> brtype<*_int_*> brtype<rel#.#*>
Read and assigned alias: ABS_bit_19_protection_override=a-r -chown administrator
Read and assigned alias: ABS_bit_01_owner_override=cclarke3
###
INFORMATION: ### No Alias definitions after this point - only references to them
INFORMATION: ### No clearquery variables defined after this point
###

Read license key: cleartrigger ABS licman;000000000;000000000;ACME, Inc.;
Read policy_dirs (windows;unix;): l:\ClearTrigger\depot;;A;
Read Logging Value: A
Read any parent Clearbits files (windows;unix;): ;;
Calc license: License is invalid

    VERIFIED: (l:\ClearTrigger\depot) Windows depot is accessible and is a directory
    WARNING:UNVERIFIED Any defined depot on the Unix OS is UNVERIFIED...

    OK!!: EMPTY Windows parent clearbits file . This is normal and (PERFECTLY OK)
    WARNING:UNVERIFIED Any defined parent on the Unix OS is UNVERIFIED...

    VERIFIED: (l:\ClearTrigger\depot\log) Windows logging directory is an accessible DIRECTORY as
needed with LOGGING = (A)
    WARNING:UNVERIFIED Any logging on the Unix OS is UNVERIFIED...

Read Notify_programs (windows;unix;): ;;
Read Region Limited Users/Groups/VOBs/Elements: ;  ;
Read Region Special Users/Groups/VOBs/Elements: ;    ;
Message Of The Day - MOTD: Final build for 10.5 tonight... Please check in all code.
Message Of The Day - MOTD Restriction List: (cclarke) (administrator)
Read Per-VOB Functionality Bits Checking: A; 01000000000000000001111110;
  Per-VOB Functionality Bit Checking read A : Enabled
  Functionality Bit(00) read BIT__AUTO_BRANCH_REMOVAL           : Disabled
  Functionality Bit(01) read BIT__CHOWN_VOB_OWNER               : Enabled
  Functionality Bit(02) read BIT__EVIL_TWIN                     : Disabled
  Functionality Bit(03) read BIT__ONE_CO_PER_BRANCH             : Disabled
  Functionality Bit(04) read BIT__CHECK_BRANCH_LABEL_NAMES      : Disabled
  Functionality Bit(05) read BIT__VOB_OWNER_CI_MAIN             : Disabled
  Functionality Bit(06) read BIT__RECURSIVE_CI_CO_UNCO          : Disabled
  Functionality Bit(07) read BIT__CHECK_MAGIC_PATH              : Disabled
  Functionality Bit(08) read BIT__NO_SPACES_IN_ELEMENT_NAMES    : Disabled
  Functionality Bit(09) read BIT__NO_MULTIPLE_MERGES            : Disabled
  Functionality Bit(10) read BIT__SMART_CHECKIN                 : Disabled
  Functionality Bit(11) read BIT__CASE_CHECKING                 : Disabled
  Functionality Bit(12) read BIT__ATOMIC_CHANGE                 : Disabled
  Functionality Bit(13) read BIT__REQUIRE_EXTENSION             : Disabled
  Functionality Bit(14) read BIT__NOTIFY_PARENT_CHANGE_ON_CO    : Disabled
  Functionality Bit(15) read BIT__NOTIFY_PARENT_CHANGE_ON_CI    : Disabled
  Functionality Bit(16) read BIT__PREVENT_DATA_LOSS             : Disabled
  Functionality Bit(17) read BIT__AUTO_EXEC_BIT                 : Enabled
  Functionality Bit(18) read BIT__PREVENT_CORE_ELEMENTS         : Enabled
  Functionality Bit(19) read BIT__AUTO_DIRECTORY_PROTECT        : Enabled
  Functionality Bit(20) read BIT__AUTO_REMOVE_CONTRIB           : Enabled
  Functionality Bit(21) read BIT__WARN_LOST_AND_FOUND           : Enabled
  Functionality Bit(22) read BIT__OWN_LIKE_PARENT_DIR           : Enabled
  Functionality Bit(23) read BIT__FORCE_CHECKOUTS_UNRESERVED    : Disabled
Calc Functionality Bits: 01000000000000000001111110
---------------keys follow----------
```

```
**** File (l:\ClearTrigger\depot\clearbits.txt) looks good.
**** All fields and keys have valid syntax.
**** but some keys might serve no purpose.
**** WARNING: Paths for keys for fields for the Unix side are UNVERIFIED.
```

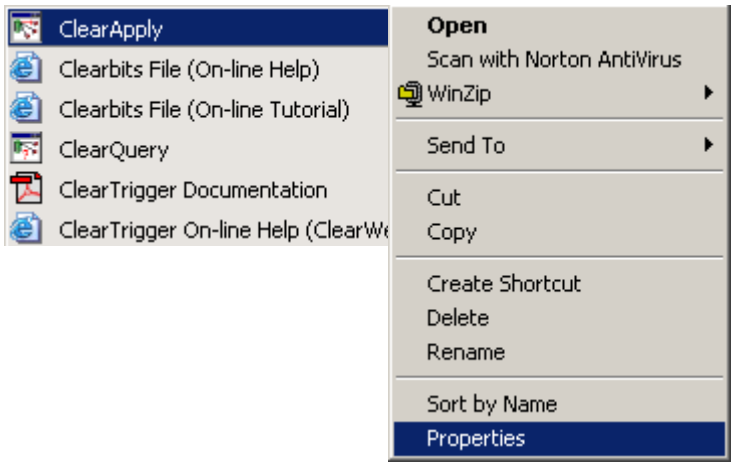

Windows Menu Shortcuts and on-line help tools

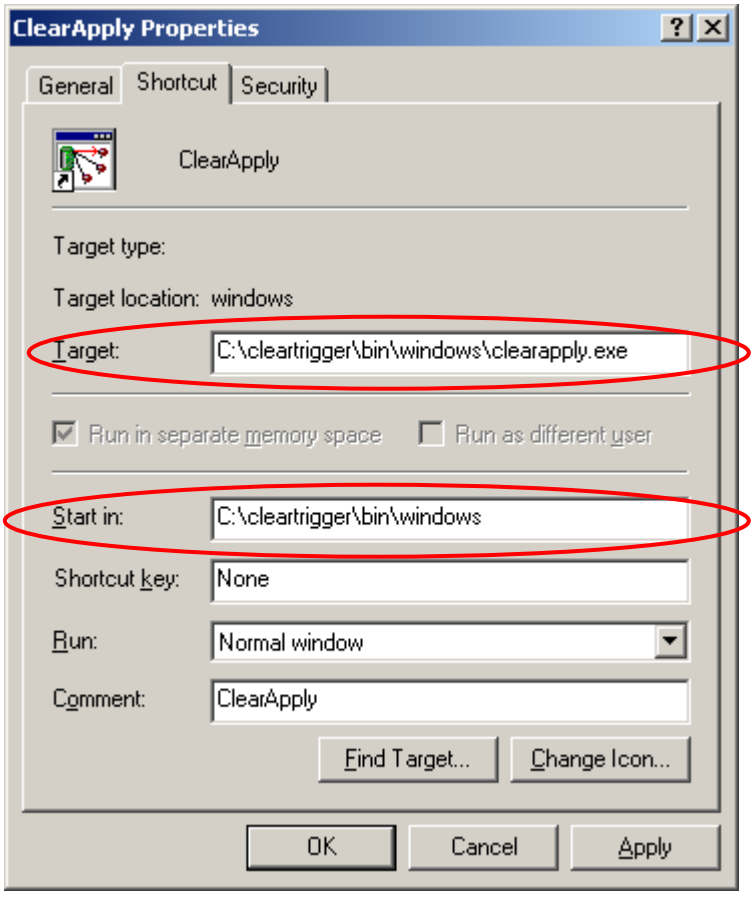
When ClearTrigger is installed on Windows architectures a menu pane is created under **Start>Programs>ClearTrigger** similar to the one below. They are place here for convenience so it is easier to access these web pages or applications.



Windows	Purpose
ClearApply	**Starts** ClearApply
Clearbits File (On-line Help)	Opens a browser window set to the Clearbits File Interactive Help Application (Requires internet access).
Clearbits File (On-line Tutorial)	Opens a browser window set to the Clearbits File Interactive Tutorial Application (Requires internet access).
ClearQuery	**Starts** ClearQuery
ClearTrigger Documentation	**Opens the** ClearTrigger Administration Guide **(this document)**
ClearTrigger Online Help (ClearWeb Developers Network)	Opens a browser window set to the ClearWeb Developer Network (Requires internet access). (http://forums.abs-consulting.com)

Note: If you installed ClearTrigger using a network path you might want to change the ***UNC path*** to the **ClearApply** and **ClearQuery** programs to a "***drive resident***" path. When displayed, right mouse button lick on the item to display its properties menu item then select "properties" to display the item's properties tab. Map a drive (if needed) and change the "**Target**" and "**Start in**" fields to a drive-resident path.





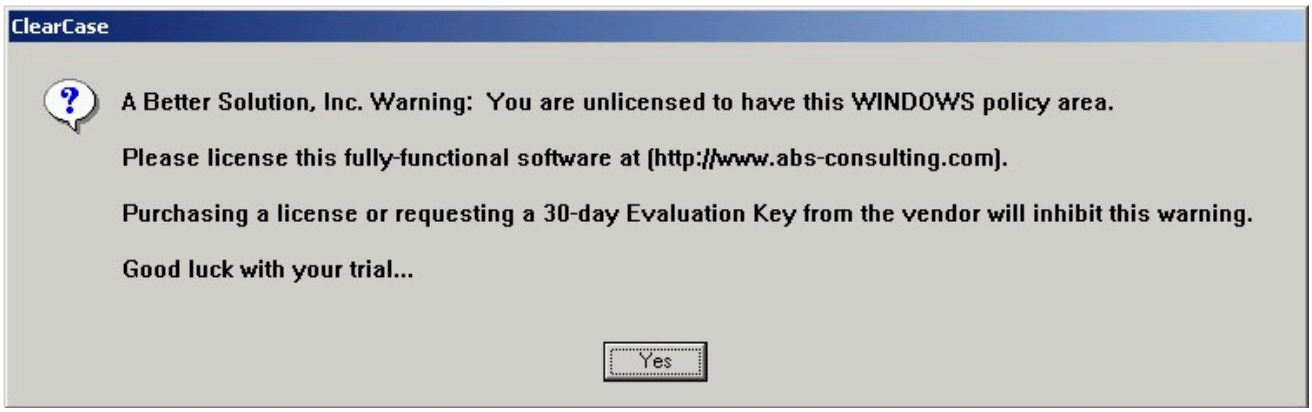
License Strings

The first section of clearbits.txt is the license section. It contains the Product Name, Vendor Name, License Key Generator ID, Windows License Key, UNIX License Key, and your Organization Name. Here, this section will replace the default license entry with your license key for each architecture license purchased from A Better Solution, Inc. or a licensed reseller. If an evaluation license key was acquired, place it here. It should look similar to the three (3) examples below:

cleartrigger ABS licman;123456789;000000000;ACME Inc – East Campus;
cleartrigger ABS licman;000000000;123456789;ACME Inc.;
cleartrigger_lite ABS licman;123456789;123456789;ACME Inc. – West Campus;

Note: The product word (first word in the license string) determines if the regions uses "ClearTrigger Features Set" or the "ClearTrigger Lite Feature Set".

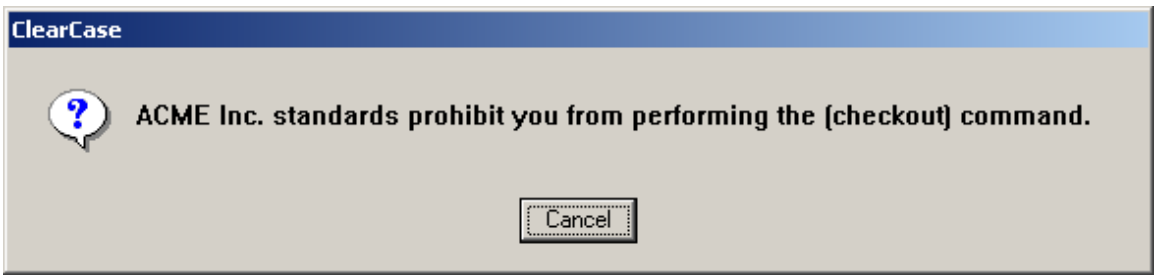
Note: Only a vendor-generated license will allow ClearTrigger to operate without randomly displaying the *ABS Splash Message*.



Carefully choose the Organization Name, as it will appear in error dialogs for the policy you create. For example for the license key:

cleartrigger ABS licman;123456789;000000000;ACME Inc.;

Error dialogs like this may appear when users violate the regions policy:



Policy Depots

The policy depot is the storage directory for the enterprise trigger code that will be managed by ClearTrigger. The section of the ***clearbits_file*** entitled 'ClearTrigger Region Information' contains the network path to the policy depot for Windows, UNIX, or both. It is also used to calculate the vendor generated license string. The policy depot(s) can be an enterprise wide area or project specific. When defining policy depots, use the semicolon (;) as a separator between the ***Windows Policy Depot*** and the ***UNIX Policy Depot*** and end the line with a semicolon. If you choose not to define a policy depot, leave that portion blank (ClearTrigger policy can be created without a defined depot). ClearTrigger will assume that no policy depot exists and any action causing the policy depot to be examined will generate an error message. Additionally and optionally, you can define the Region's Logging personality (discussed in the next section). The following are examples of formatting the policy depot:

Examples:

Windows Only:

\\win2k_01\Policy\region1_depot;;

UNIX Only:

;/net/earth/policy/region1_depot;

InterOp Configuration:

\\win2k_01\Policy\region1_depot;/net/earth/policy/region1_depot;

Undefined:

;;

The policy depot should be in a valid architecture specific structure that terminates to a well-known location from your ClearCase clients. The following tables show acceptable conventions for path naming:

Windows	Allowed path naming
UNC Naming	**\\SERVER_NAME\SHARE_NAME\{PATH_TO_POLICY_DEPOT}**
Drive Resident	**D:\{PATH_TO_POLICY_DEPOT}**

UNIX	Allowed path naming
Network Naming	**/NET/SERVER_NAME/{PATH_TO_POLICY_DEPOT}**
Mounted or Absolute local	**/{PATH_TO_POLICY_DEPOT}**

Parent Clearbits files

Any clearbits file can refer to or point to a "parent" clearbits file. For VOBs within the ClearTrigger region, policy defined in its parent clearbits file must be satisfied before policy in the region's clearbits file. This allows one to build or take advantage of a clearbits file hierarchy. Policy can be defined at a company level that applies to all VOBs within the company while a department might have additional policy that applies to a subset of the company VOBs. Each region/clearbits file could have different personnel that are responsible for its modification; each region/clearbits file has its own depot and logging. The section of the **clearbits_file** entitled "Parent Clearbits file" contains the network path to the parent clearbits file (if any) for Windows, UNIX, or both. It is also used to calculate the vendor generated license string. When defining parent clearbits files, use the semicolon (;) as a separator between the ***Windows Parent Clearbits File*** and the ***UNIX Parent Clearbits File*** and end the key with a semicolon. If you choose not to define a clearbits parent, leave that portion blank (e.g. "**;;**"). ClearTrigger will assume that no parent clearbits file is requested. The following are examples of formatting the parent clearbits file:

Examples:

Windows Only:

\\win2k_01\Policy\ACME_clearbits.txt;;

UNIX Only:

;/net/earth/policy/ACME_clearbits.txt;

InterOp Configuration:

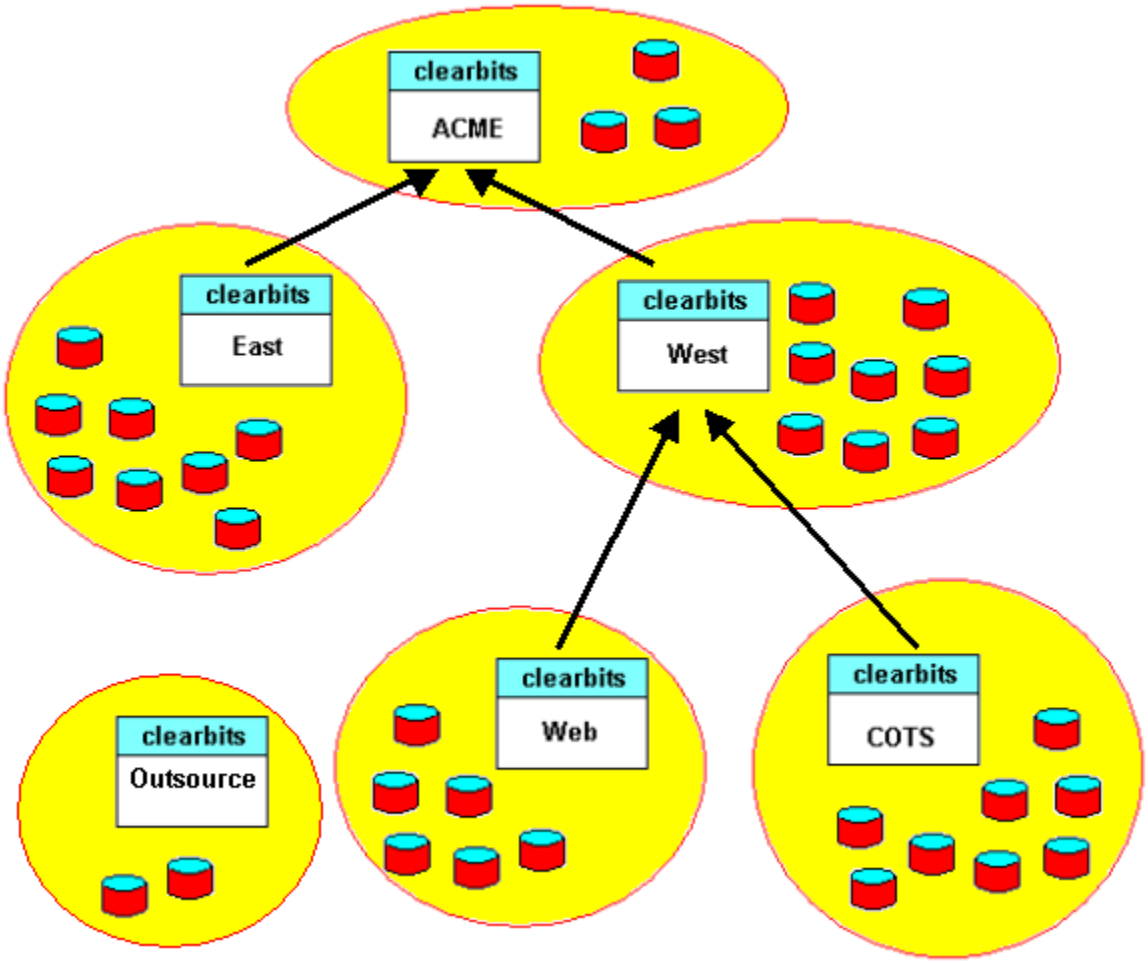
\\win2k_01\Policy\ ACME_clearbits.txt;/net/earth/policy/ ACME_clearbits.txt;

Undefined (empty):

;;

When in an InterOp configuration, you could have all of your VOBs using two *parent **clearbits_files***, (one for each OS), with each one pointing to a different parent clearbits file. This will allow the VOBs to have different trigger code executed for the different OS partitions (UNIX/Windows).

Configurations are flexible. Several examples are illustrated below:



As depicted above:

- Policy defined in the "ACME" clearbits file applies to all VOBs except those in the "Outsource" ClearTrigger region.
- Policy defined in the "East" clearbits file applies only to the VOBs in the "East" ClearTrigger region
- Policy defined in the "West" clearbits file applies to the VOBs in the "West" ClearTrigger region and the VOBs in the "Web" and "COTS" ClearTrigger regions.
- Policy defined in the "Web", "COTS" and "Outsource" clearbits files applies only to the VOBs in those respective ClearTrigger regions.

Region Logging Personality

Each ClearTrigger region can define a logging personality for that region. This allows the policy maker the ability to log all triggerable ClearCase commands. Both successful commands as well as attempts that were "denied" by defined ClearTrigger policy can be logged. All logging data is stored in files within the "log" directory in the depot(s). That information can be queried upon and maintained using the ClearQuery application, delivered with ClearTrigger.
Denied attempts are logged if requested because such information is vital when ascertaining whether turned or defined policy is being properly used or of value. To see the value of generated reports, read the section entitled ClearQuery Reports.

To avoid resource blocking, the logging information is spread over multiple files such that 20 or more may be seen at any time in the "log" directory. The Region Logging personality value is optionally located directly after the Policy Depots definitions in the clearbits file.
e.g.
\\win2k_01\Policy\region1_depot;/net/earth/policy/region1_depot;O;

Note: The web interface pull-down choices for Region Logging Personality are limited to only valid values. They are defined as follows:

Region Logging Value	Meaning
'O'	Logging is 'O'ff for the cleartrigger region.
'E'	Logging of 'E'rrors (denied commands) is turned on – Inhibited commands are written to the logs.
'S'	Logging of 'S'uccessful command is turned on – Successfully completed commands are written to the logs .
'A'	Logging of 'A'll command attempts is turned on – Denied and successfully completed commands are written to the logs.

***Note:** If Logging is turned on, the appropriate logging directory ("log") **must be created in the depot** with appropriate write permissions (Write permission for everyone).

***Hint:** To avoid possible permission errors, make sure that the directory if fully writable for everyone. It is also helpful to pre-create the files that logging activities would create ("logging_0.txt" – "logging_19.txt") and ensure the files are writable for everyone.

Mail Notification Programs

This portion of the *clearbits_file* contains the ClearTrigger region defined mail notification programs for Windows, UNIX or both. This program could be a general mail or logging program that your organization wrote or purchased. A mail program is readily available to use with ClearTrigger (placed in the depot directory) at www.abs-consulting.com/abs_consulting_triggers.html by A Better Solution, Inc. The script or program can be of any executable type (e.g. *.bat, *.pl, *.exe, *.etc). The script is expected by default to reside in the Depot for the Operating System (UNIX or Windows), but can also be described with an absolute path or a path relative to the Depot.

When defining a region notification program, use the semicolon (;) as a separator between the ***Windows Notification Program*** and the ***UNIX Notification Program*** and end the line with a semicolon. If you choose not to define a *Notification Program*, leave that portion blank. ClearTrigger will assume that no *Notification Program* exists and any action causing a *Notification Program* to be called for that OS will generate an error message. For information on when this program is executed, read the section entitled 'Formatted Command Keys'. **Any defined *Notification Program* is expected to reside by default in the depot for the OS in which it will be executed, but can also be described with an absolute path or a path relative to the Depot.** The following are examples of formatting the *Notification Program*:

List Description	Notification Program Example
Undefined	;;
Windows Only	Notify.pl;;
Unix Only	;Notify.pl;
InterOp Configuration	Notify.pl;Notify.pl;
Calling program not in depot by absolute path	\\machine\share\program.exe;/net/machine/policy/program;
Calling program not in depot by relative path to depot	..\directory\script.pl;.../directory/program;

Pattern Matching

Many portions of the license file can contain *patterns*. These patterns are useful when matching one or more items. An item might consist of a **(user)**, **<group>**, **[VOB]**, **[VOB:replica], {element}, %view%, @time@**, **'ClearCase_region'** or ABS_TRIGGER_POLITY environmental variable **^value^** as found in:

- Region Inhibited List
- Region Special Access List
- MOTD Restriction List
- Command Key Allow List
- Command Key Disallow List
- Command Key Ignore List
- Command Key Trigger Not List
- Command Key Trigger Only List

or they may be a **type item** as found in **RIN List**:

- Restriction List
- Inclusion List
- Negation List

There are three (3) recognized pattern characters that are described below:

Pattern Character	Description
*	matches any string
?	matches any single character
#	matches any single digit

These **pattern characters** can be used to match characters within an item's pattern area. The pattern area consists of the characters between an item's initial and terminating characters as described below:

Item	Examples
(User)	(bob) (intern_*) (sec_#_*) (temp_??_service)
<Group>	<dev> <dev_#> <pra##> <dev?_service>
{Element}	{/vobs/pra/bar.c} {*/src/*} {*.c} {#.???}
[VOB]	[\vob_a] [/vob/pra*] [*tut*] [*] [/vob_?]
[VOB:Replica]	[\vob_a:origional] [/vob/pra*:atl*] [*core:site#] [*:*]
%view%	%some_view% %*_build_views%
@time@	@h1#@ @D######15@ @d*@
'ClearCase Region'	'development' 'US_*'

NOTES:

- Adjacent '*' characters are treated as a single '*' character
- A '#' or '?' that ***directly*** follows a '*' characters is ignored

HINT:

VOB replica names can be changed even if ClearCase Multisite is not being used. This serves as an easy way to make VOB groupings for VOBs that don't have a naming pattern that one can easily group with. For example

you could change the replica name of some VOBs to “cm_vobs” and other to “dev_vobs” then select policy on **[*:cm_vobs]** to distinguish between the two types of VOBs.

ClearTrigger Aliases

Many portions of the license file can contain ***aliases***. These aliases are useful when grouping several objects together by a single name, which represents that grouping throughout the clearbits file. This is useful for creating fictitious "groupings" (the term "grouping" instead of "group" is used because ClearTrigger aliases can be used to group both users into groups as well as to "group" numbers of <groups>, [VOBs], [VOB:replica], {elements} or any combination of each of the types. Additionally, aliases can be used to "group" types as in brtype<integration> lbtype<main>. An alias "grouping" might consist of **(users)**, **<groups>**, **[VOBs], [VOB:Replica]**, **{elements}, %views%, @times@** or **'ClearCase Region'** as found in:

- Region Inhibited List
- Region Special Access List
- Command Key Allow List
- Command Key Disallow List
- Command Key Trigger Not List
- Command Key Trigger Only List

or they may be in a **type item** as found in **RIN List**:

- Restriction List
- Inclusion List
- Negation List

You may define up to **80** aliases in the ClearTrigger Region such that:

- Alias names may contain up to 50 characters from [a..z,A..Z,0..9,_]
- Alias values may contain up to 256 characters not to include ',' or ';'
- Alias values may contain other aliases (so a hierarchy can be built) up to **5 levels deep**
- Aliases must fit under 2048 characters when fully expanded
- Alias definitions are only valid **before the license key** in the clearbits file. All other alias definitions after the license key are treated as comments.

Alias definitions are "hidden" in the upper portion (above the license) of the clearbits file and are of this form:

#!cleartrigger_alias {alias_name} {alias_value}

Dynamic Variables and **Patterns** may also be used in the alias value and are evaluated at run-time.

Example (each on single line)
#!cleartrigger_alias **good_users** (cclarke) (slewand) (rcarter)
#!cleartrigger_alias **dev_vobs** [*cell] [*proj_##_?] [*planet:atl] [*hr:site#] [*:original]
#!cleartrigger_alias **mgt_access** [*mgt*] {*/corp/*} <mgt>
#!cleartrigger_alias **limit_types** brtype<main> lbtype<REL#.#*>
#!cleartrigger_alias **corp** ~good_users ~dev_vobs ~mgt_access
#!cleartrigger_alias **downtime** @h23@ @D6@
#!cleartrigger_alias **runtime_views** %web_view% %release_#_view%
#!cleartrigger_alias **dev_regions** 'development' 'US_*'

Once defined in the clearbits file, that alias can be used instead of the "grouping" in Region Lists and Command Keys. Refer to an alias as **~alias** when you want ClearTrigger to expand the alias at runtime. There are examples below:

Example
; vobadm **~good_users** <*adm> ;
; jgardner **~bad_users ~interns** ;
pre_MODIFY_ELEM;R; ;0;0;0;D;**~students** ;N; **~admins** ;0;tag.pl;
pre_MODIFY_ELEM;R; **~limit_types** ;x;10;

NOTES:

- The ~alias **must** have a trailing ' ' (space) character after it when referenced
- Alias references are expanded before error messages are built so "real" users, groups, VOBs etc. are displayed (not the ~alias name)

ClearTrigger Override aliases

Some reserved aliases are used by ClearTrigger to override at the region level the default behavior of some core ClearTrigger functionality or of some ClearTrigger **functionality bits**. For example, when ClearTrigger Functionality bit #01 is enabled in the region the ownership of newly created elements is changed to the current VOB owner; however the alias **ABS_bit_01_owner_override** may be used to modify this behavior such that the ownership is instead changed to another user when new elements are created.

***Override Aliases added or modified in version 13.0 are so indicated by highlight.**

Override Alias	Purpose
#!cleartrigger_alias **ABS_bit_01_owner_override**	Sets owner to a user other than the current VOB owner when new elements are created. Set to ***user*** to change the user or set to ***user:group*** to change user and group. **Dynamic Variables** (i.e. (&):<&>) can also be used. (Refer to **functionality bit# 01**)
#!cleartrigger_alias **ABS_bit_05_main_override**	Sets some other user (other than the current VOB owner) to be allowed to checkin element on the /main branch (Refer to **functionality bit# 05**)
#!cleartrigger_alias **ABS_bit_07_personality_unix**	Require a Unix Magic Path other than the default magic path (path to region depot) before new elements can be created. (Refer to **functionality bit# 07**)
#!cleartrigger_alias **ABS_bit_07_personality_windows**	Require a Windows Magic Path other than the default magic path (path to region depot) before new elements can be created. (Refer to **functionality bit# 07**)
#!cleartrigger_alias **ABS_bit_17_exec_override**	Set to additional element extension types for new elements that should automatically have their execution bit set when new elements of this patter are created. Multiple patters should be separated by a space (e.g. "***.mod .cmd .rpy***"). (Refer to **functionality bit# 17**)
#!cleartrigger_alias **ABS_bit_12_personalilty**	Set to "choice" or "force" and determines if atomic checkouts are "suggested" or "forced". (Refer to **functionality bit# 12**)
#!cleartrigger_alias **ABS_bit_19_protection_override**	Override default behavior of bit 19 – Set owner and protections of newly created directory elements. (Refer to **functionality bit# 19**)
#!cleartrigger_alias **ABS_bit_21_auto_answer_allow**	Defines the Users, Groups, VOBs, Elements, Views, Times or even ClearCase Regions that can take advantage of auto answer for functionality bit #21 by setting their user environmental CLEARTRIGGER_BIT_21_AUTO_INTERACTIVE_ANSWER to "yes" or "no". (Refer to **functionality bit# 21**)
#!cleartrigger_alias **ABS_perl_security_override**	If defined in the clearbits file it allows Perl Debug variables (**PERLDB_OPTS** or **PERL5DB**) to be set when triggers fire.
#!cleartrigger_alias **ABS_allow_use_of_local_ccperl**	If defined in the clearbits file it allows the use of local **ccperl.exe**, **CQerl.exe** and **cleartool.exe** executables instead of ones defined in the depot for **windows script** definitions with policy key definitions.

ClearTrigger Functionality Bit Application Aliases

Some reserved aliases are used by ClearTrigger to limit what a ClearTrigger **functionality bit** applies to. The special aliases are referred to as **Functionality Bit Application Aliases**. These special aliases are of form:

ABS_exclude_for_bit_*nn* or **ABS_only_for_bit_*nn***
as well as
ABS_exclude_for_bit_ALL or **ABS_only_for_bit_ALL**

Where "***nn***" is from "00"..."23" (the lowest numbered bit to the highest numbered bit). If the associated **Functionality bit** is on in the clearbits file then these aliases can be used limit under what condition the bit will fire. Only use **Functionality Bit Application Aliases** when you wish to limit when a Functionality Bit fires.

Where "**ALL**" designates that this would apply to all bits "00"… ""23".

Functionality Bit Application Aliases	Purpose
#!cleartrigger_alias **ABS_exclude_for_bit_*nn***	If the associated **functionality bit** is on, this serves to limit (by exclusion) under what conditions this bit will fire. Excludes processing for the matching conditions.
#!cleartrigger_alias **ABS_only_for_bit_*nn***	If the associated **functionality bit** is on, this serves to limit (by inclusion) under what conditions this bit will fire. The bit will fire only for the matching conditions.
#!cleartrigger_alias **ABS_exclude_for_bit_ALL**	For all of the associated **functionality bitS** that are on, this serves to limit (by exclusion) under what conditions those bits will fire. Excludes processing for the matching conditions.
#!cleartrigger_alias **ABS_only_for_bit_ALL**	For all of the associated **functionality bitS** that are on, this serves to limit (by inclusion) under what conditions those bits will fire. The bits will fire only for the matching conditions.

At run-time **ABS_exclude_for_bit_nn** aliases are evaluated first so if ***possible*** conflicting associated aliases (e.g. **ABS_exclude_for_bit_05** and **ABS_only_for_bit_05**) are both placed in the clearbits file, then the "'exclude" alias will take precedence.

For example, if in the clearbits file **Functionality bit 5** is enabled and that file also contains the definitions below:

#!cleartrigger_alias **CM_users (cclarke3) (rcarter) (slewand) (syang)**
#!cleartrigger_alias **ABS_only_for_bit_05 ~CM_users**
#!cleartrigger_alias **ABS_exclude_for_bit_05 (rcarter)**

The **Functionality bit 5** (checkin on /man limited to VOB owner) would fire for all of the CM_users ***except*** the user **rcarter** who is excluded.

At run-time, if both a specific **ABS_exclude_for_bit_nn** alias and the specific **ABS_exclude_for_bit_ALL** alias are defined then they are combined.

For example, if in the clearbits file **Functionality bit 5** is enabled and that file also contains the definitions below:

#!cleartrigger_alias **ABS_exclude_for_bit_ALL (rcarter) (cclarke3)**
#!cleartrigger_alias **ABS_exclude_for_bit_05 (slewand)**

Any enabled **Functionality bit** will NOT fire for the users **rcarter** and **cclarke3**. Additionally, **Functionality bit 5** will NOT fire for the user **slewand** as well as **rcarter** and **cclarke3**.

At run-time, if both a specific **ABS_only_for_bit_nn** alias and the specific **ABS_only_for_bit_ALL** alias are defined then they are combined.

For example, if in the clearbits file **Functionality bit 5** is enabled and that file also contains the definitions below:

#!cleartrigger_alias **ABS_only_for_bit_ALL (rcarter) (cclarke3)**
#!cleartrigger_alias **ABS_only_for_bit_05 (slewand)**

Any enabled **Functionality bits** will ONLY fire for the users **rcarter** and **cclarke3**. Additionally, **Functionality bit 5** will ONLY fire for the user **slewand** as well as **rcarter** and **cclarke3**.

ClearTrigger Performance Alias Enhancers

In order to improve ClearTrigger performance on commands that have no policy defined region-wide you can tell ClearTrigger to ignore those commands. ClearTrigger will not perform it's preprocessing of those commands, exception is to check for parent clearbits files so that any parent clearbits file policy is not circumvented. For example, the proper use of these aliases will significantly speed the up the performance the recursive **mklabel** command for regions that have no policy on the **cleartool mklabel** command.

Performance Alias	Purpose
#!cleartrigger_alias **ABS_cleartrigger_preop_active**	Enables ClearTrigger preop clearcase commands only for the included commands
#!cleartrigger_alias **ABS_cleartrigger_preop_inactive**	Disables ClearTrigger preop clearcase commands only for the included commands
#!cleartrigger_alias **ABS_cleartrigger_postop_active**	Enables ClearTrigger postop clearcase commands only for the included commands
#!cleartrigger_alias **ABS_cleartrigger_postop_inactive**	Disables ClearTrigger postop clearcase commands only for the included commands

If associated preop aliases (e.g. **ABS_cleartrigger_preop_active** and **ABS_cleartrigger_preop_inactive)** are both placed in the clearbits file, only the first one encountered in the file is used.

If associated postop aliases (e.g. **ABS_cleartrigger_postop_active** and **ABS_cleartrigger_postop_inactive)** are both placed in the clearbits file, only the first one encountered in the file is used.

Examples of appropriate values are in the table below:

Performance Alias Example	Purpose
#!cleartrigger_alias **ABS_cleartrigger_preop_active checkin checkout uncheckout**	Enables ClearTrigger preop processing for ***only*** the checkin, checkout and uncheckout commands.
#!cleartrigger_alias **ABS_cleartrigger_preop_active NONE**	Disables ClearTrigger preop processing for ***all*** commands.
#!cleartrigger_alias **ABS_cleartrigger_postop_active checkin checkout uncheckout**	Enables ClearTrigger postop processing ***only*** for the checkin, checkout and uncheckout commands.
#!cleartrigger_alias **ABS_cleartrigger_preop_inactive mklabel mkhlink**	Disables ClearTrigger preop processing for the mklabel and mkhlink commands.
#!cleartrigger_alias **ABS_cleartrigger_postop_inactive mklabel mkhlink**	Disables ClearTrigger postop processing for the mklabel and mkhlink commands.

ClearTrigger Relocation Alias Enhancers

Relocation Aliases allow for ClearTrigger logging information and ClearReplica shipout information to be written to any network location. Relocation Aliases allow for the ClearTrigger depot itself to be located on a ***read-only*** share, device or media if needed.

The four (4) Relocation Aliases are listed below:

ClearTrigger Relocation Alias	Purpose
#!cleartrigger_alias **cleartrigger_relocated_logging_dir_windows**	Change the Windows logging directory from the default "***log***" directory in the "depot" to ***this*** directory.
#!cleartrigger_alias **cleartrigger_relocated_logging_dir_unix**	Change the Unix logging directory from the default "***log***" directory in the "depot" to ***this*** directory.
#!cleartrigger_alias **cleartrigger_relocated_shipout_dir_windows**	Change the Windows shipout directory from the default "***_ship_out***" directory in the **ClearReplica** "***m_bay***" directory to ***this*** directory.
#!cleartrigger_alias **cleartrigger_relocated_shipout_dir_unix**	Change the Unix shipout directory from the default "***_ship_out***" directory in the **ClearReplica** "***m_bay***" directory to ***this*** directory.

Examples of appropriate values are in the table below:

ClearTrigger Relocation Alias Examples	Meaning
#!cleartrigger_alias **cleartrigger_relocated_logging_dir_windows** \\vobsvr1\policy\cleartrigger\log	The logging directory for Windows is changed to: **\\vobsvr1\policy\cleartrigger\log**
#!cleartrigger_alias **cleartrigger_relocated_logging_dir_unix** /net/vobsvr1/policy/cleartrigger/log	The logging directory for Unix is changed to: **/net/policy/cleartrigger/log**
#!cleartrigger_alias **cleartrigger_relocated_shipout_dir_windows** \\vobsvr1\policy\CR\shipout	The shipout directory for Windows is changed to: **\\vobsvr1\policy\CR\shipout**
#!cleartrigger_alias **cleartrigger_relocated_shipout_dir_unix** /net/vobsvr1/policy/CR/shipout	The logging directory for Unix is changed to: **/net/policy/CR/shipout**

NOTE: When using the "relocated logging directory" aliases the ClearQuery Configuration Aliases should be set equivalently.

.

If ClearTrigger Relocation Alias is set	it should match the ClearQuery configuration alias
#!cleartrigger_alias **cleartrigger_relocated_logging_dir_windows**	#!clearquery_relocated_logging_dir_windows
#!cleartrigger_alias **cleartrigger_relocated_logging_dir_unix**	#!clearquery_relocated_logging_dir_unix

NOTE: Only user the "relocated shipout directory" aliases when VOBs are replicated using **ClearReplica 9.2** or higher.

ClearReplica Performance Alias Enhancers

In order to improve ClearTrigger performance for non-ClearReplica replicated VOBs in a ClearTrigger Region that has ClearReplica enabled you can prevent ClearTrigger from checking to see if "replication" is needed by taking advantage of special **ClearReplica Performance Alias Enhancers**. The special aliases will limit which VOBs attempt to read the **ClearReplica shipper_conf file** or inform ClearReplica of data updates during modification of ClearTrigger data within certain VOBs; this improves ClearTrigger performance for those VOBs.

Only VOB entries (e.g. [/a_vob] [*cm]) or VOB:replica (e.g. [/a_vob:earth] [*:atlanta]) are valid or read in the alias, all other entry types are ignored.

NOTE: This only affects "outbound" changes from the region. Any ClearReplica "receiver" processes for the region are unaffected.

ClearReplica Performance Alias	Purpose
#!cleartrigger_alias **ABS_replication_active**	Enables ClearReplica shipping processing only for the included VOBs.
#!cleartrigger_alias **ABS_replication_inactive**	Disables ClearReplica shipping processing for the included VOBs.

If both aliases (e.g. **ABS_replication_active** and **ABS_replication_inactive)** are placed in the clearbits file and a VOB is contained in both list then replication "inactive" for that VOB.

Examples of appropriate values are in the table below:

ClearReplica Performance Alias Examples	Meaning
#!cleartrigger_alias **ABS_replication_active** [*qa] [\vobs\dev*]	VOBs matching the patterns are the only ones with outbound replication considered.
#!cleartrigger_alias **ABS_replication_inactive** [*cm] [*hr]	VOBs matching the patterns are not considered for outbound replication.

Enable ClearReplica replication of rmtype -brtype commands

By default **ClearTrigger** does not inform the **ClearReplica** "shipper" of destructive command so ClearReplica shipper does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. However, ClearTrigger can be configured to retain and inform ClearReplica of successful **cleartool rmtype -brtype** command information.

Note: This feature requires the use of ClearReplica 9.5 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmtype -brtype" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.8 or higher encounters this directory it will place encrypted 'rmtype -brtype information" there for the rmtype -brtype commands that successfully completed within the scope of a replication key.

Enable ClearReplica replication of rmtype -lbtype commands

By default **ClearTrigger** does not inform the **ClearReplica** "shipper" of destructive command so ClearReplica shipper does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. However, ClearTrigger can be configured to retain and inform ClearReplica of successful **cleartool rmtype -lbtype** command information.

Note: This feature requires the use of ClearReplica 9.4 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmtype -lbtype" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.7 or higher encounters this directory it will place encrypted 'rmtype -lbtype information" there for the rmtype -lbtype commands that successfully completed within the scope of a replication key.

Enable ClearReplica replication of rmelem commands

By default **ClearTrigger** does not inform the **ClearReplica** "shipper" of destructive command so ClearReplica shipper does not replicate destructive commands to remote replicas as that remote replica is often a backup of the local replica and this default behavior prevents the replication of inadvertent destruction in one replica to other replicas. However, ClearTrigger can be configured to retain and inform ClearReplica of successful cleartool rmelem command information.

Note: This feature requires the use of ClearReplica 9.3 or higher on the "shipping" locations VOB.

To inform the ClearReplica shipper that you wish it to package "rmelem" information in the packets it sends to remote locations, just create a fully writable directory named "**_dhist**" in the "**_ship_out**" directory (or that directory's **relocated location**). If ClearTrigger 12.6 or higher encounters this directory it will place encrypted 'rmelem information" there for the rmelem commands that successfully completed within the scope of a replication key.

ClearTrigger Dynamic Variables

Many portions of the license file can contain ***dynamic variables***. These variables are useful to refer to certain entities are not known at policy creation time and might vary or change within the ClearTrigger Region. For example, consider that one might wish to have a trigger fire for everyone except the current VOB owner, but many VOBs in your region are owned by different individuals. Normally you would have to apply the trigger to all users and then include code in your script that checks for the current VOB owner to except that user or apply the trigger with a different –nuser for each VOB. With a ***dynamic variable*** you can refer to the VOB owner and it is calculated when needed and always correct even if the VOB owner is changed at a later date.

Currently there are three (3) ***dynamic variables*** defined in ClearTrigger:

Dynamic Variable	Character String	Such that…
VOB Owner	(&	(&) expands to (cclarke) at runtime in VOBs that are owned by the user cclarke. (&_#) expands to (cma_#) at runtime in VOBs that are owned by the user cma and therefore matches the users "cma_0" – "cma_9".
VOB Primary Group	<&	<&> expands to the <cm_team> at runtime in VOBs with a primary group of cm_team. <&_#> expands to <cma_group_#> at runtime in VOB with a primary group of cma_group and therefore matches the users "cma_group _0" – "cma_group _9".
Top of VOB	{&	{&} expands to {/vobs/some_vob} at runtime in the VOB /vobs/some_vob for UNIX) and to {\some_vob} for Windows. {&/src} expands to the {/vobs/some_vob/src} at runtime for UNIX and {\ some_vob\src} for Windows.

Dynamic Variables are used in list and can be found in:

- Region Inhibited List
- Region Special Access List
- Command Key Allow List
- Command Key Disallow List
- Command Key Ignore List
- Command Key Trigger Not List
- Command Key Trigger Only List

NOTE: Dynamic Variables are evaluated at runtime after any **aliases** are expanded and before any **Pattern Matching** is applied.

Region Inhibited List - Users/Group/VOBs/Elements/Views/Times/Regions

The next field is the **Region Inhibited List.** In this field/list you can add Users, Groups, VOBs, Elements, Views, Times or even ClearCase Regions. Additionally, **dynamic variables**, **pattern matching** and user defined ClearTrigger **aliases** can be used in these fields. Matching items in the list have restricted access within the ClearTrigger region. For each type of match the reaction is described below:

Any listed users or user groups cannot perform ClearCase commands that change the data inside any VOB in the ClearTrigger region. These users or user groups will get a uniform error dialog when attempting data modifying actions in the region. This method is much better than updating the ***license.db*** file for ClearCase. Doing that would disallow a ClearCase license and also prohibit many ClearCase read-only commands such as **cleartool lsvob** from being executed. This has an advantage over locking the VOB by allowing the user to perform a build, but prohibiting "development" actions. The ***Region Inhibited List*** can prevent users or user groups from performing VOB data modification commands for a group of VOBs in a *ClearTrigger Region* and will continue to allow modification in another group of VOBs.

Any listed VOBs or elements are designated such that no data modifying commands for anyone are allowed. Again, this has an advantage over locking the VOB for the user in that the users are still allowed to perform builds, but all "development" actions are prohibited. The UNIX VOBTAG style or Windows VOBTAG style or both can be used. The ***Region Inhibited List*** can prevent VOBs from being changed by anyone.

Any listed Views are designated such that no data modifying commands for anyone are allowed through the matching View. Again, this has an advantage over locking the VOB for the user in that the users are still allowed to perform builds, but all "development" actions are prohibited. The ***Region Inhibited List*** can prevent VOB modification through the view by anyone.

Any listed Times are designated such that no data modifying commands for anyone are allowed during a matching Time (e.g. hour(s), day(s)-of-week or specific date). Again, this has an advantage over locking the VOB for the user in that the users are still allowed to perform builds, but all "development" actions are prohibited. The ***Region Inhibited List*** can prevent VOB modification at a known time by anyone.

Any listed ClearCase Regions are designated such that no data modifying commands for anyone are allowed within a matching ClearCase Region. This could be used to prevent modification in the ClearCase Region "Production" while still allowing modification in the "Development" ClearCase Region. Again, this has an advantage over locking the VOB for the user in that the users are still allowed to perform builds, but all "development" actions are prohibited. The ***Region Inhibited List*** can prevent VOB modification within a known ClearCase Region by anyone. **This functionality was added in ClearTrigger 13.0**.

Any matching Environmental Variable Value are designated such that no data modifying commands for anyone are allowed when the user's environment variable **ABS_TRIGGER_POLICY** is set to a matching value. This could be used to prevent modification in the ClearCase Region during easily determined times during a development cycle. **This functionality was added in ClearTrigger 13.1**

The **Region_Inhibited_List** is specified in the *clearbits_file* file below the policy directory path. It takes precedence over the Special_Access_List. The format for this section is a space separated list of zero or more user-ids, groups, VOBs, elements, views or times enclosed by semicolons (;). Users ids are surrounded with parentheses (e.g. (user1)). Group names are surrounded with brackets (e.g. <groupname>). VOB tags are surrounded with square brackets (e.g. [/vobs/a_vob] or [\a_vob]); VOB:Replica can also be designated (e.g. [/vobs/a_vob:original] or [\a_vob:atl]). Elements are surrounded with curly braces (e.g. {\vobs\a_vob\src\foo.c}). }). Views are surrounded with percent characters (e.g. %a_view%). Times are surrounded with at characters (e.g. @time@). ClearCase Regions are surrounded with single quote characters (e.g. 'development'). The value(s) that the environmental variable ABS_TRIGGER_POLICY are matched against are surrounded with the carat characters (e.g. ^development^).

Valid time formats are any of the three (3) defined below:

Hour definition:	'@h##@' (where ## represent an hour from 0-23 "Military Hours")
Day definition:	'@d#@' (where # represent a day-of-week from 0-6 "Sun-Sat")
Date definition	'@D########@' (where ######## represent a date from YYYYMMDD)

All users can be represented by a (*). All groups can be represented as <*>; all VOBs can be represented as [*] or [*:*]; all elements can be represented by {*}; all Views can be represented as %*%; all Times can be represented as @h*@, @d*@ or @D*@; all ClearCase Regions can be represented as '*'; all values that the ABS_TRIGGER_POLICY environmental variable will match against as ^*^.

For more examples of pattern characters that can be used, refer to the sections on **Dynamic Variables**, **Pattern Matching** and **aliases**.

For example, if you wanted to prevent administrative personnel and new-hires of certain UNIX or Windows groups from performing changes in any VOB until they've received ClearCase training you could do so by adding their user-id to the ***Region Inhibited List*** in the *clearbits_file*.

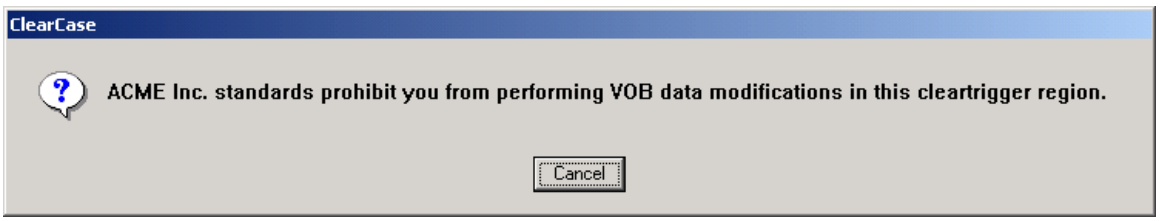
Remember that when using group names, any windows clients that access the VOB must have the CLEARCASE_PRIMARY_GROUP Environmental Variable set.

Here are some examples:

Example Type	Example
Empty List	; ;
Single user entry	; (user1) ;
Single group entry	; <janitors> ;
Single VOB entry	; [/vobs/a_vob] ;
Single Element entry	; {/vobs/a_vob/some_file.c} ;
Single View entry	; %night_build_view% ;
Single Hour entry (11pm – midnight)	; @h23@ ;
Single day entry (Sunday)	; @d0@ ;
Single Date entry (Feb. 19, 2005)	; @D20050219@ ;
Multiple VOB entries (UNIX/Windows)	; [\a_vob] [/vobs/a_vob] [\b_vob] ;
Multiple VOB:Replica patterns	; [\a_vob:original] [*:atl] [*:site#] ;
Multiply Populated List	; (jit) (qa_#*) <dev1> ~interns [/vobs/a_vob] ;
all users	; (*) ;
all elements	; {*} ;
particular elements	; {/vobs/a/src/foo.c} {\vob_a\foo.?} ;
select elements	; {*.c} {/vobs/*/src/*} ~core_code ;
all Windows VOBs	; [*] ;
all UNIX VOBs	; [/*] ;
All times	; @d*@ ;
Particular ClearCase Regions	; 'development' 'webdoc' 'US_*' ;
All ClearCase Regions	; '*' ;
Current VOB owner	; (&) ;
Current VOB's Primary Group	; <&> ;
Top of VOB	;{&} ;
"src" directory at top of VOB	;{&/src} ;
During "Code Freeze"	; ^code_freeze^ ;
In any testing phase	; ^test_*^

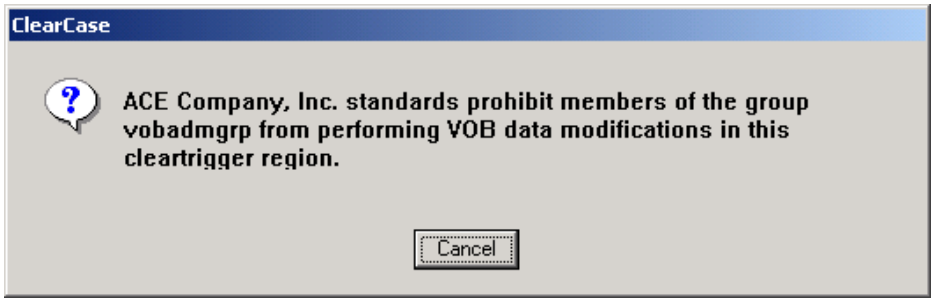
> **Note: When using both users and groups: the user is granted denied access if they are a current user in the list or they are currently representing a group within the list.**

If any ***Inhibited Users*** attempted any VOB-modifying command in your *ClearTrigger Region*, they would receive a uniform dialog box containing a consistent message similar to this:



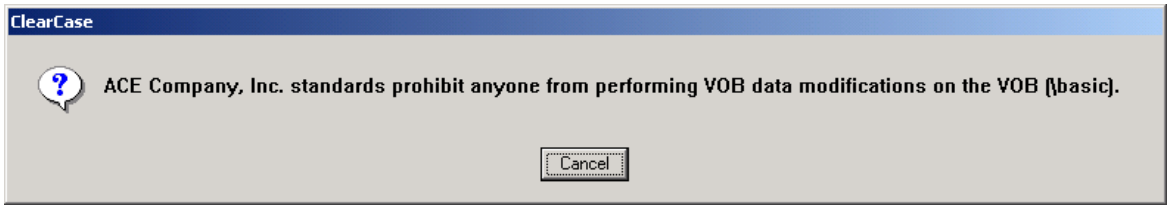
Users can be stopped on a command-by-command basis as well. You will see examples of this in the section entitled 'Policy: Prevent Use of Certain Commands'.

If any ***Inhibited Group*** member attempted any VOB-modifying command in your *ClearTrigger Region* they would receive a uniform dialog box containing a consistent message similar to this:



Groups can be stopped on a command-by-command basis as well. You will see examples of this in the section entitled 'Policy: Prevent Use of Certain Commands'.

If anyone attempts a VOB-modifying command for an ***Inhibited VOB*** or on an ***Inhibited Element*** they would receive a uniform dialog box containing a consistent message similar to this:



VOBs can be limited on a command-by-command basis as well. You will see examples of this in the section entitled 'Policy: Prevent Use of Certain Commands'.

Region Special Access List - Users/Groups/VOBs/Elements/Views/Times/Regions

The next field is the **Region Special Access List.** In this field/list you can add Users, Groups, VOBs, Elements, Views or Times. Additionally, **dynamic variables**, **pattern matching** and user defined ClearTrigger **aliases** can be used in these fields. For Users, Groups, VOBs, elements, views or times in the list, none of the policy defined in the clearbits file is considered (You have, in essence, turned off ClearTrigger for those Users, Groups, VOBs, elements, views, ClearCase regions, or times – this is good to do temporarily for ClearCase imports of for special/short periods of time).

Any listed users or user groups are designated such that ClearTrigger will immediately stop its processing of commands for that user or for members for group to prevent the policy defined in the clearbits file from being applied for that user or group.

Any listed VOBs or elements are designated such that ClearTrigger will immediately stop its processing of commands in a particular VOB to prevent the policy defined in the clearbits file from being applied to those VOBs or elements. The UNIX VOBTAG style or Windows VOBTAG style or both can be used.

Any listed Views are designated such that ClearTrigger will immediately stop its processing of commands in a particular view to prevent the policy defined in the clearbits file from being applied while in the view.

Any listed Times are designated such that ClearTrigger will immediately stop its processing of commands at a particular time to prevent the policy defined in the clearbits file from being applied at that time.

Any listed ClearCase Regions are designated such that ClearTrigger will immediately stop its processing of commands in a particular ClearCase Region to prevent the policy defined in the clearbits file from being applied while in that ClearCase Region. **This functionality was added in ClearTrigger 13.0**.

Any matching Environmental Variable Value are designated such that ClearTrigger will stop its processing when the user's environment variable **ABS_TRIGGER_POLICY** is set to a matching value. This could be used to prevent modification in the ClearCase Region during easily determined times during a development cycle. **This functionality was added in ClearTrigger 13.1**

The **Special_Access_List** is specified in the *clearbits_file* file below the Region_Inhibited_List. The format for this section is a space separated list of zero or more user-ids, groups, VOBs, elements, views or times enclosed by semicolons (;). Users ids are surrounded with parentheses (e.g. (user1)). Group names are surrounded with brackets (e.g. <groupname>). VOB tags are surrounded with square brackets (e.g. [/vobs/a_vob] or [\a_vob]); VOB:Replica can also be designated (e.g. [/vobs/a_vob:original] or [\a_vob:atl]). Elements are surrounded with curly braces (e.g. {\vobs\a_vob\src\foo.c}). }). Views are surrounded with percent characters (e.g. %a_view%). Times are surrounded with at characters (e.g. @time@). ClearCase Regions are surrounded with single quote characters (e.g. 'development'). The value(s) that the environmental variable ABS_TRIGGER_POLICY are matched against are surrounded with the carat characters (e.g. ^development^).

Valid time formats are any of the three (3) defined below:

Hour definition:	'@h##@' (where ## represent an hour from 0-23 "Military Hours")
Day definition:	'@d#@' (where # represent a day-of-week from 0-6 "Sun-Sat")
Date definition	'@D########@' (where ######## represent a date from YYYYMMDD)

All users can be represented by a (*). All groups can be represented as <*>; all VOBs can be represented as [*] or [*:*]; all elements can be represented by {*}; all Views can be represented as %*%; all Times can be represented as @h*@, @d*@ or @D*@; all ClearCase Regions can be represented as '*' ; all values that the ABS_TRIGGER_POLICY environmental variable will match against as ^*^.

For more examples of pattern characters that can be used, refer to the sections on **Dynamic Variables**, **Pattern Matching** and **aliases**.

For example, if one needed to disable all policies in order to perform a large code import, one could do any of the methods below:

- Place one or more users in this list to turn off policy for those users.
- Place a Windows or Unix group in the list to turn off policy for that group.
- Place a VOB in the list to turn off policy for that VOB.
- Place a View in the list access so the view that does not invoke the policy.
- Place a time in the list to provide a time period where no policy will be invoked.
- Place some well-known value for the ABS_TRIGGER_POLITY env. Variable to match against

Here are some examples:

Example Type	Example
Empty List	; ;
Single user entry	; (user1) ;
Single group entry	; <janitors> ;
Single VOB entry	; [/vobs/a_vob] ;
Single Element entry	; {/vobs/a_vob/some_file.c} ;
Single View entry	; %night_build_view% ;
Single Hour entry (11pm – midnight)	; @h23@ ;
Single day entry (Sunday)	; @d0@ ;
Single Date entry (Feb. 19, 2005)	; @D20050219@ ;
Multiple VOB entries (UNIX/Windows)	; [\a_vob] [/vobs/a_vob] [\b_vob] ;
Multiple VOB:Replica patterns	; [\a_vob:original] [*:atl] [*:site#] ;
Multiply Populated List	; (jit) (qa_#*) <dev1> ~interns [/vobs/a_vob] ;
all users	; (*) ;
all elements	; {*} ;
particular elements	; {/vobs/a/src/foo.c} {\vob_a\foo.?} ;
select elements	; {*.c} {/vobs/*/src/*} ~core_code ;
all Windows VOBs	; [*] ;
all UNIX VOBs	; [/*] ;
All times	; @d*@ ;
Particular ClearCase Regions	; ‘development’ ‘webdoc’ ‘US_*’ ;
All ClearCase Regions	; ‘*’ ;
Current VOB owner	; (&) ;
Current VOB’s Primary Group	; <&> ;
Top of VOB	;{&} ;
“src” directory at top of VOB	;{&/src} ;
During “Code Freeze”	; ^code_freeze^ ;
In any testing phase	; ^test_*^

Note: When using both users and groups: the user is granted special access if they are a current user in the list or they are currently representing a group within the list.

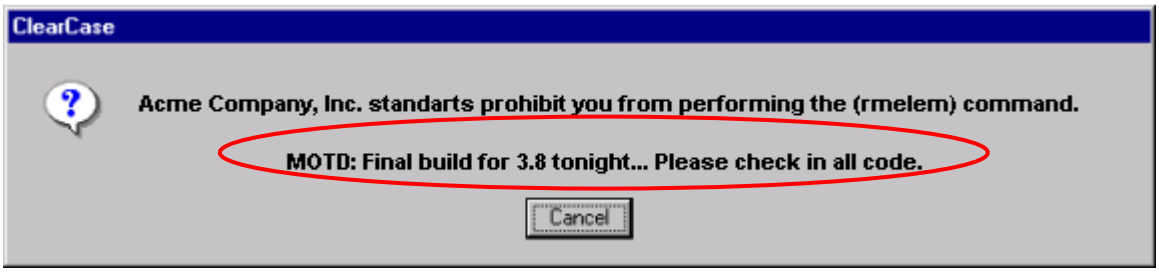
Remember that when using group names, any windows client that accesses the VOB must have the CLEARCASE_PRIMARY_GROUP Environmental Variable set.

MOTD (Message-of-the-day)

The next field in the clearbits_file is the **MOTD field**. The MOTD (Message Of The Day) can be blank or populated and provides a method for the policy maker to communicate important messages to ClearCase users. The text placed in the field is displayed on many question or advise dialog boxes. An example of a MOTD definition and an associated dialog box follows. Given the MOTD definition below:

;Final build for 3.8 tonight… Please check in all code.;;

Many dialogs that would normally be displayed to a ClearCase user would also have the additional MOTD displayed.



To restrict the conditions where the MOTD is displayed, populate the MOTD Restriction List.

MOTD Restriction List

Populating the MOTD (Message of the Day) causes the MOTD to appear on many dialogs that ClearCase users see during their use of ClearCase. For example:

;Final QA build tonight...;;

The capability to restrict when the MOTD is appended to dialogs exists. Display of the MOTD can be limited to certain to Users, Groups, particular VOBs or elements. For example with the MOTD and the MOTD Restriction defined below:

;Final QA build tonight...; (qa_test*) [/vobs/qa_*] [*:atl] ;

ClearTrigger displays the MOTD, but only to "qa_test*" users while working in any "qa*" VOB.

The **MOTD Restriction List** section contains a list of zero or more user-ids, group names, VOB tags, element names, view names or times enclosed by semicolons (;). User names are surrounded with Parenthesis (e.g. (cclarke)), group names are surrounded with brackets (e.g. <dev>), VOB tags are surrounded with square brackets (e.g. [/vobs/a_vob] or [\a_vob]), VOB:Replica can also be designated (e.g. [/vobs/a_vob:original] or [\a_vob:atl]), element names are surrounded with curly braces (e.g. {/vobs/a_vob/foo.c} or {\a_vob\foo.c}, view names are surrounded with percent characters (e.g. %night_build% or %import_view%) and times are surrounded with at characters (e.g. @time@). ClearCase Regions are surrounded with single quote characters (e.g. ‘development’). The value(s) that the environmental variable

ABS_TRIGGER_POLICY are matched against are surrounded with the carat characters (e.g. ^development^).

Valid time formats are any of the three (3) defined below:

Hour definition:	'@h##@' (where ## represent an hour from 0-23 "Military Hours")
Day definition:	'@d#@' (where # represent a day-of-week from 0-6 "Sun-Sat")
Date definition	'@D########@' (where ######## represent a date from YYYYMMDD)

All users can be represented by a (*). All groups can be represented as <*>; all VOBs can be represented as [*] or [*:*]; all elements can be represented by {*}; all Views can be represented as %*%; all Times can be represented as @h*@, @d*@ or @D*; all ClearCase Regions can be represented as '*' ; all values that the ABS_TRIGGER_POLICY environmental variable will match against as ^*^.

For more examples of pattern characters that can be used, refer to the sections on **Dynamic Variables**, **Pattern Matching** and **aliases**.

Example Type	Example
Empty List	; ;
Single user entry	; (user1) ;
Single group entry	; <janitors> ;
Single VOB entry	; [/vobs/a_vob] ;
Single Element entry	; {/vobs/a_vob/some_file.c} ;
Single View entry	; %night_build_view% ;
Single Hour entry (11pm – midnight)	; @h23@ ;
Single day entry (Sunday)	; @d0@ ;
Single Date entry (Feb. 19, 2005)	; @D20050219@ ;
Multiple VOB entries (UNIX/Windows)	; [\a_vob] [/vobs/a_vob] [\b_vob] ;
Multiple VOB:Replica patterns	; [\a_vob:original] [*:atl] [*:site#] ;
Multiply Populated List	; (jit) (qa_#*) <dev1> ~interns [/vobs/a_vob] ;
all users	; (*) ;
all elements	; {*} ;
particular elements	; {/vobs/a/src/foo.c} {\vob_a\foo.?} ;
select elements	; {*.c} {/vobs/*/src/*} ~core_code ;
all Windows VOBs	; [*] ;
all UNIX VOBs	; [/*] ;
All times	; @d*@ ;
Particular ClearCase Regions	; 'development' 'webdoc' 'US_*' ;
All ClearCase Regions	; '*' ;
Current VOB owner	; (&) ;
Current VOB's Primary Group	; <&> ;
Top of VOB	;{&} ;
"src" directory at top of VOB	;{&/src} ;
During "Code Freeze"	; ^code_freeze^ ;
In any testing phase	; ^test_*^

Note: When using both users and groups: the user is shown the MOTD if they are a current user in the list or they are currently representing a group within the list.

Remember that when using group names, any windows client that accesses the VOB must have the CLEARCASE_PRIMARY_GROUP Environmental Variable set.

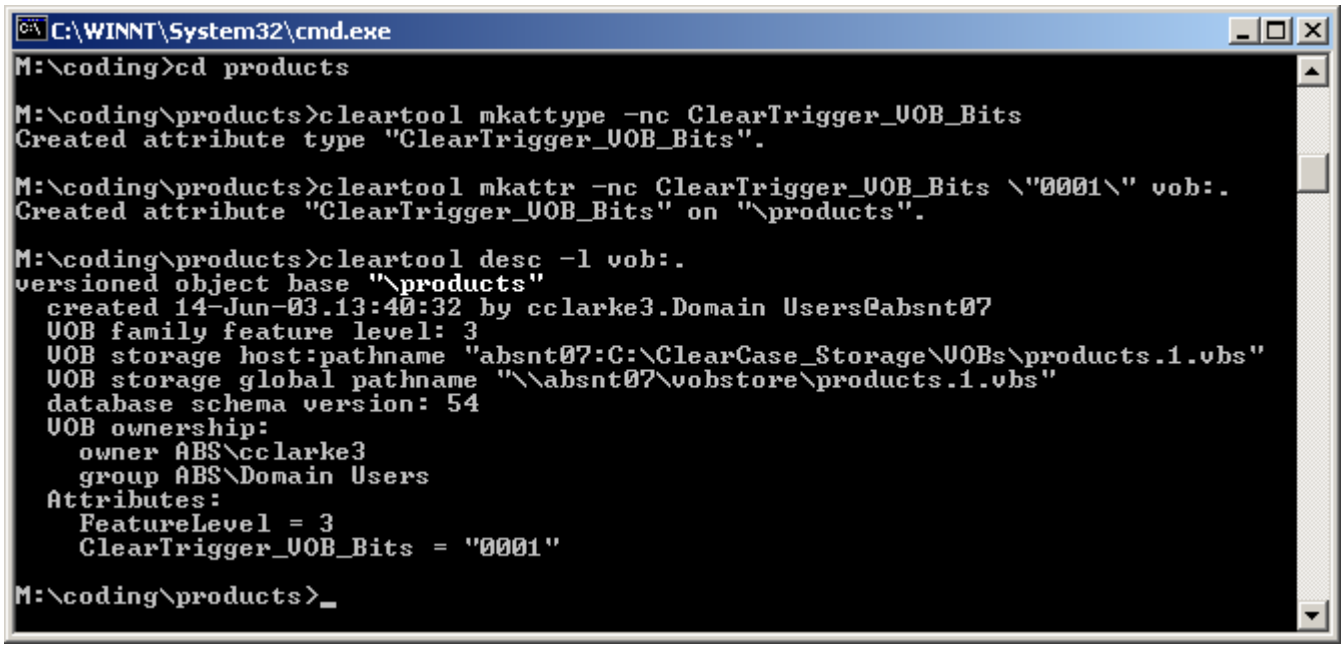
Per-VOB Functionality Bits Checking

This portion of the license file determines if Functionality Bits are evaluated for just the region or for the region and on a per-VOB basis. When enabled (The value is set to 'A' for allowed) then if a particular functionality bit is not enabled for the region then the VOB itself is checked to see if the bit is enabled for the VOB. The Functionality Bits for the region are a series of '0' and '1' values indicated in the next field in the clearbits file. The Per-VOB Functionality Bits for the VOB are a series of '0' and '1' values indicated in an "optional" attribute ("ClearTrigger_VOB_Bits") attached to the VOB.

For example if your Region Functionality Bits were such that Single Checkout Per Branch was off for the region as per the Functionality Bits defined in the clearbits file:

1010000100010100000001110

You could in turn enable this feature for a particular VOB by enabling per-VOB checking for the region and setting the ClearTrigger_VOB_Bits attribute on the VOB like the example below:



Having done so will tell ClearTrigger to turn on bit #3 (Single Checkout per Branch) for the \products VOB though it is not enabled for the region.

The ClearTrigger_VOB_Bits attribute is only checked if the Per-VOB Checking is enabled for the region AND the bit if interest is not enabled for the region.

Note: You need only define the bits necessary to properly define the *highest* bit you wish to enable.

Per-VOB Functionality Bits Checking – Contains a valid value from below:

Value	Defines this feature as...
'D'	Per-VOB Functionality Bits checking is Disallowed
'A'	Per-VOB Functionality Bits checking is Allowed

Functionality Bits

This portion of the license file contains the functionality bits for ClearCase policy commonly requested ClearCase users. This common ClearCase functionality is already compiled into ClearTrigger for speed. **There are Twenty-Four (24) bits evaluated (Functionality bit 0 – 23).**

Bit #	To include functionality for …
0	Automatic removal of empty branches created by uncheckout, rmbranch and rmver commands.
1	Automatically change ownership to the VOB-owner for all newly created elements. The default user can be changed from the VOB_owner to another userid using the ClearTrigger Override Alias **ABS_bit_01_owner_override** within the clearbits file.
2	Prevent duplicate elements on hidden branches (**classic Evil Twin avoidance**).
3	Only allow 1 checkout of an element per branch.
4	Ensure that all newly created Label type names are fully **UPPERCASE** and Branch type names are fully **lowercase**.
5	Only allow the current VOB owner to perform checkins on the main branch. The default user can be changed from the VOB_owner to another userid using the ClearTrigger Override Alias **ABS_bit_05_main_override** within the clearbits file.
6	User is queried for recursive checkins, checkouts and uncheckouts when directories are encountered.
7	Enforcement of an enterprise ClearCase Magic Path.
8	Prevent embedded spaces in element names
9	Checkins not allowed when the element is the destination of multiple merges.
10	Smart Checkins – unchanged text_files and directories are automatically unchecked out when checkin on that element is attempted.
11	Element Name Case Checking – Prevents the creation of elements in the same directory with the same case insensitive name. (e.g. it will not allow "foo.c" to be created in "Foo.c" or "foo.C" or some such exist in the directory).
12	Atomic change bit – Elements can be grouped together such that they must be checked out, checked-in, or unchecked out together as a group.
13	Require that new element names have an extension. Inhibits the creation of "foo" and requires "foo.*" (e.g. foo.c, foo.h, foo.bat etc).
14	Require that the user is notified when they checkout on a branch if the same element on that branch's parent branch has advanced and needs merging or rebasing.
15	Require that the user is notified when they checkin on a branch if the same element on that branch's parent branch has advanced and needs merging or rebasing.
16	Prevent users (other than the VOB owner) from performing certain destructive commands from the list: chmaster chtype (element type) rmbranch rmelem rmtype rmver.

17	Automatically add execute permissions to new elements (programs and scripts) when those elements are created. When new file elements are added to the VOB that are known by name (e.g. *.pl, *.sh, .login, *.exe, etc) or are typed to a known element type (e.g. scripts, perl_scripts, executables, etc) then owner, group and other execute permissions are added to the new file. The default extensions can be changed by using the ClearTrigger Override Alias **ABS_bit_17_exec_override** within the clearbits file.
18	Prevent files named "core" or files typed as “core” files from being added as new elements to the VOB.
19	Automatically protect newly created directory elements as **775** in the VOB. The default for the ClearTrigger region can be changed from 775 to any value by using the ClearTrigger Override Alias **ABS_bit_19_protection_override** within the clearbits file.
20	Automatically remove “.contrib.” files that are created when merges are performed. The files are removed after a checkin or uncheckout of the associated file element. The user can set their environmental variable ABS_CONTRIB_KEEP to ignore the value of the region defined functionality bit.
21	Automatically test to see if an uncheckout of a directory will cause possible movement of child elements to the VOB’s lost+found directory. If there are such elements then they are listed and the user is given the choice to proceed or cancel the directory uncheckout.
22	Automatically change ownership of newly created elements to match the ownership of the elements parent directory.
23	Automatically force all checkouts to be ‘unreserved’.

Note: The web-interface pull down choices for all functionality bits are limited to only valid values. They are defined as follows:

Value	Defines this feature as …
0	disabled
1	enabled

The ***Per-VOB Functionality Bits Checking Flag*** and ***Functionality Bits*** definitions exist just below the 'Region Special Access List' in the clearbits_file. The Functionality Bits definition consist of a list of values (either 0 or 1) for each the associated Functionality Bits. For example, to enable Functionality Bits **1**, **2** and **19** for the region while not considering **per-VOB Functionality Bits** one would create a Functionality Bits definition in the clearbits_file like the one below:

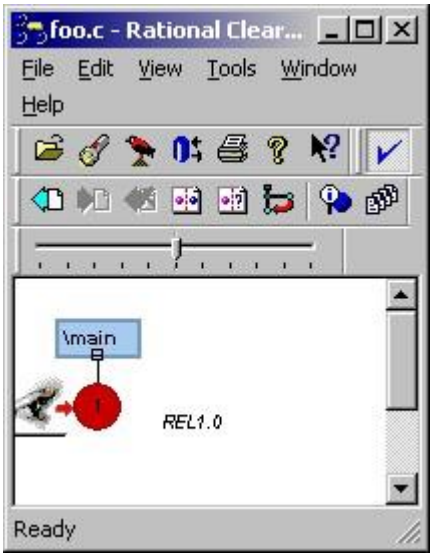
D;01100000000000000000100000;

While to enable Functionality Bits **1**, **2** and **19** for the region and also consider **per-VOB Functionality Bits** one would create a Functionality Bits definition in the clearbits_file like the one below:

A;01100000000000000000100000;

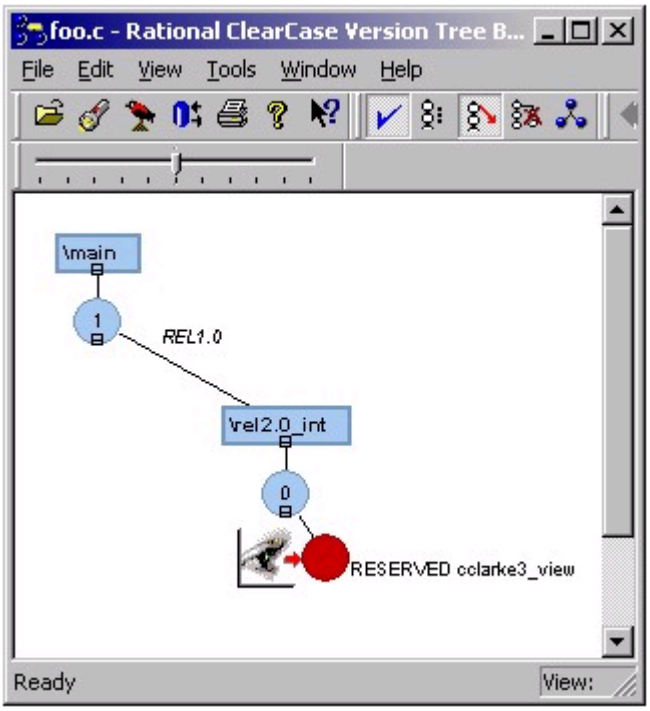
Functionality bit 0 - Auto Remove Empty Branch

This bit causes the automatic removal of empty branches caused by cleartool uncheckout, rmbranch or rmver commands. When such a removal would itself cause an empty branch then that branch is also removed. An example of this bit's functionality is displayed below. Given the version tree and the configuration specification below…

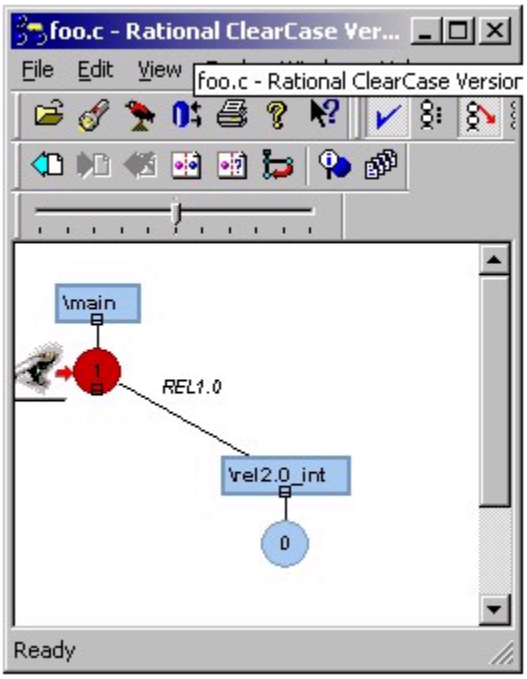


element * CHECKEDOUT
element * /main/rel2.0_int/LATEST
element * REL1.0 -mkbranch rel2.0_int
element * /main/0 -mkbranch rel2.0_int

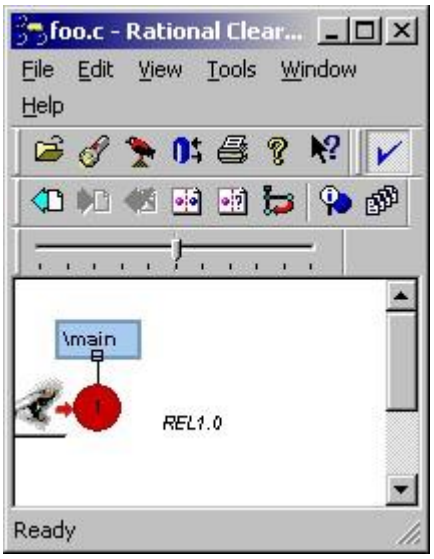
Checking out foo.c would create the rel2.0_int branch and the zeroth version on that branch and perform the checkout there (as depicted below and left).



Without the bit enabled, if the user performs a cleartool uncheckout on the element the checked out version is removed resulting in an empty branch (as depicted below).



Most development shops would prefer that this "needless" version (foo.c@@/main/rel2.0_int/0) and branch (/foo/c@@/main/rel2.0_int) were automatically removed. When this bit is enabled, the branch is automatically removed resulting in the version tree state equivalent to the starting state:



Functionality bit 1 - Auto Chown to VOB Owner (or another user:group)

When enabled, this bit automatically transfers the ownership of newly created elements to the current VOB owner. The group ownership is unchanged by default, but can also be changed. This is a common request of development shops as it limits who can remove the element to the VOB owner and provides a consistent element ownership throughout the VOB.

To change the personality of this bit to change the owner to another desired other than the current VOB owner or to change the group, just make sure to set the special ClearTrigger Override Alias **ABS_bit_01_owner_override** in the clearbits file. The alias can be set to any userid or group. For example, the alias setting below:

#!cleartrigger_alias ABS_bit_01_owner_override cclarke:sd_dev

will change the ownership of newly created files to the user id “cclarke” and the group “sd_dev”.

While, the alias setting below:

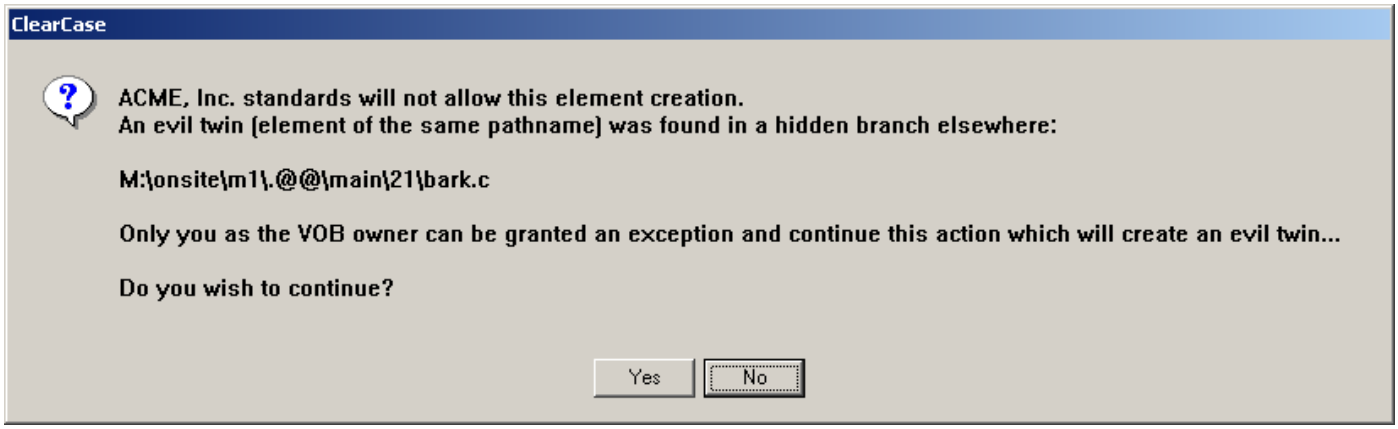
#!cleartrigger_alias ABS_bit_01_owner_override (&):sd_dev

will change the ownership of newly created file to the user id of the current VOB owner and the group “sd_dev”.

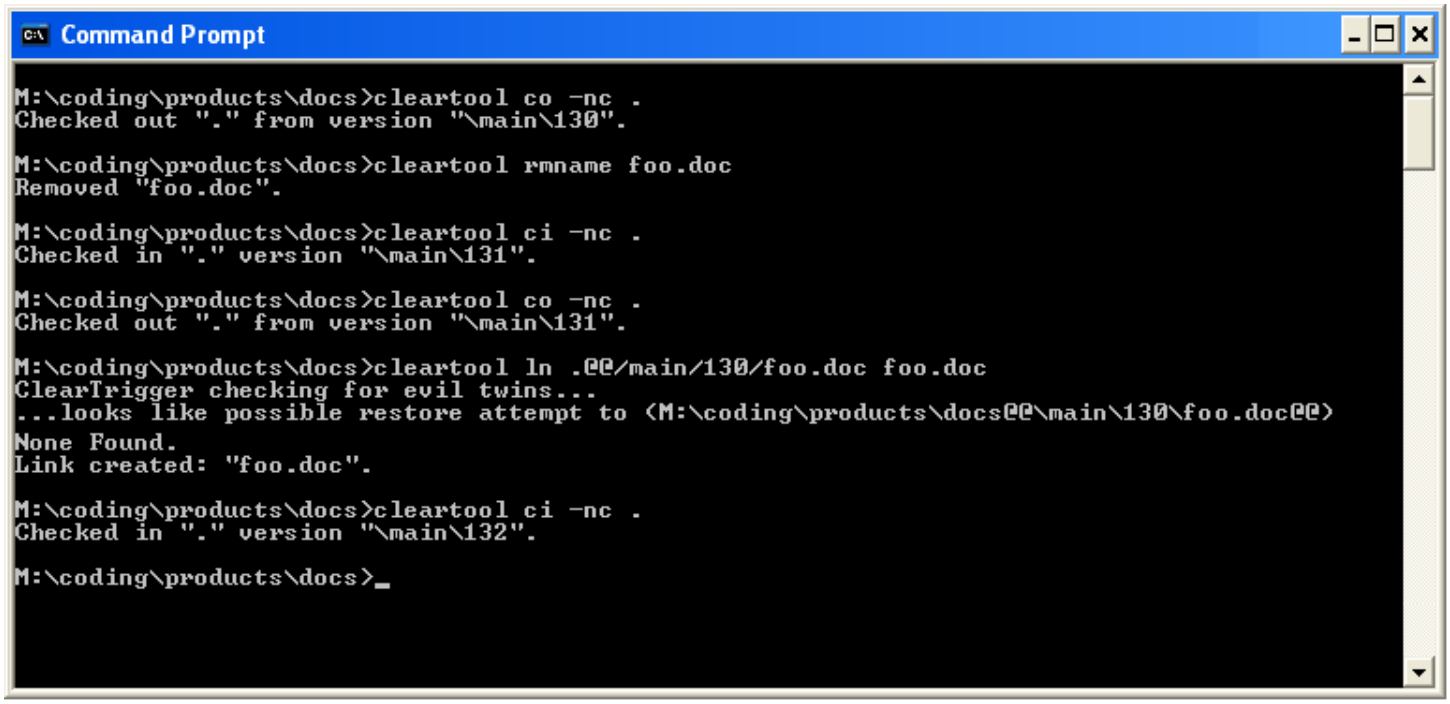
Functionality bit 2 - Prevent Evil Twins

This bit prevents duplicate elements on hidden branches (classic Evil Twin avoidance). When this bit is enabled, any attempt to make an element that previously exists on a hidden branch is detected and the user is prevented from making the element and is shown where the hidden element is (via version extended pathname). This gives the user the opportunity to make a cleartool link to the existing element instead and prevents future merge abnormalities and future data loss. An attempt to make "**aagha.c**" that already exists on some hidden branch results in:

If the current user is also the current VOB-owner then they will receive a warning and get an opportunity to override the warning and make the Evil Twin.



Additionally if the potential "Evil Twin" is recognized as an "***attempt to repair***" from a previous element **rmname** command it is noted and allowed is depicted below.



Equivalent processing to the "mkelem" command was added to the "ln", "ln -s" and "mv" commands.

Functionality bit 3 - Single Checkout per Branch

This bit will only allow a single checkout per branch per element. Some development organizations prefer to limit checkouts on a branch to a single checkout rather than allow the developer to perform an "unreserved" checkout. These shops prefer the second engineer communicate with the first engineer to gain write access to the element or that the second engineer work on a separate branch.

If a second checkout for an element is attempted for a branch then a dialog similar to the one below is displayed:

Functionality bit 4 - Branch/Label Naming Standards

This bit enforces the industry-wide minimal standards for naming branches and label types. **All label names are fully UPPERCASE and all branch names are fully lowercase**.
One of the most common triggers that many development shops put in place is one that insures that this rule is enforced. This is because ClearCase uses the branches and labels to implement MVFS (ClearCase's proprietary **M**ulti **V**ersioned **F**ile **S**ystem). As Labels and Branches share the same namespace within MVFS it is imperative that you can distinguish between the two types when scripting or debugging a problem. It is the most common convention that ClearCase users have. When enabled, this bit enforces that convention/rule.

If a user attempts to create a **label** name that is not fully **UPPERCASE** then this dialog is displayed:

If a user attempts to create a **branch** name that is not fully **lowercase** then this dialog is displayed:

Functionality bit 5 - Checkin on /main limited to VOB Owner (or another user)

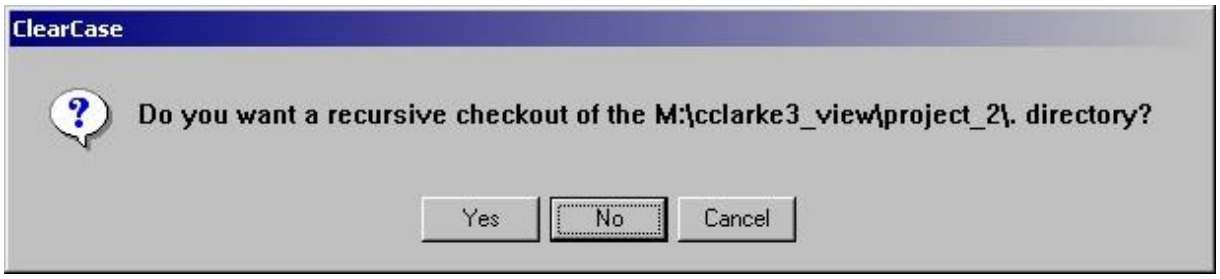
This bit helps to minimize who can perform checkins on the /main branch. The most popular restriction of this sort is that only the VOB owner can perform this checkin. This is usually better than selecting a single user as a global schema in that there might be several VOBs in your organization and it can usually be assumed that the most trusted user that has access to a VOB is the user that owns that particular VOB.

If an attempt is made to checkin on the main branch by anyone but the current VOB owner then a dialog similar to the one below is displayed:

To change the personality of this bit to limit who can checkin on /main to another user other than the current VOB owner, just make sure to set the special ClearTrigger Override Alias **ABS_bit_05_main_override** in the clearbits file. The alias can be set to any userid.

Functionality bit 6 - Recursive CI/CO/UNCO

This bit allows for recursive abilities to the checkin, checkout, and uncheckout commands. When a directory is encountered during one of these commands, the user is given the opportunity to perform that operation on the directories' children as well as the directory. An example dialog that would be encountered during attempts to checkout a directory follows:

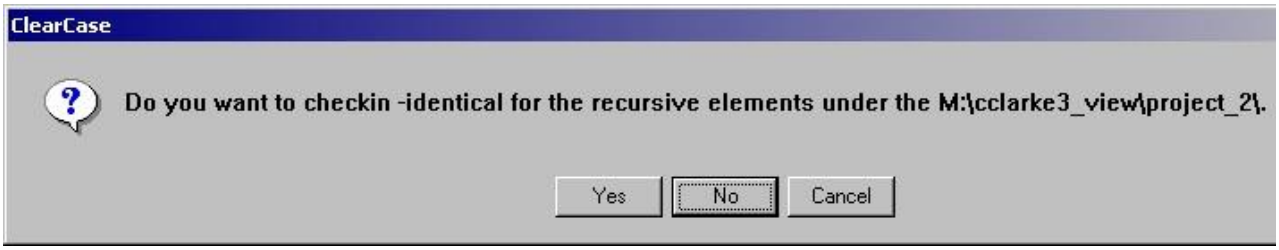


If the user requests recursion then the command is repeated recursively for any children that can have the action applied to it at that time (e.g. for checkout - all children are checked out that are currently checked in).

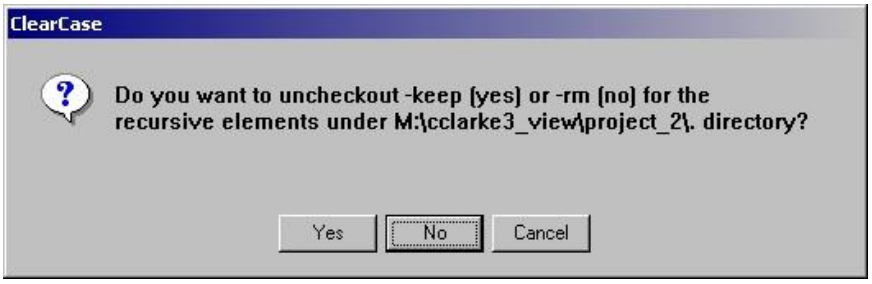
Additionally,

- When the user is performing checkout or checkin, the comment for the directory is automatically applied to the children.

- When the user is performing a checkin, they are prompted whether or not to use the -identical option for the children.



- When the user is performing an uncheckout then they are prompted whether or not to use the -keep or -rm option for the children.



Functionality bit 7 - Enterprise MAGIC_PATH

This bit enforces an enterprise controlled ClearCase MAGIC_PATH for all users. Some development shops use element types to create or control policy (e.g. "all *c_source* must compile before it is checked in" or "only lead developers can modify *c_include* files after the system test phase"). In this case, this policy itself is only as good as the method used to make sure that all "*.c" files are stored as the c_source element type and all "*.h" files are stored as c_include element type. Unfortunately, in base ClearCase, any user can change their MAGIC_PATH environmental path such that the intended and approved magic_file is not used.

This bit ensures that no elements are created unless the user's MAGIC_PATH environmental variable is set to the development group's standard. This ensures that all users use only ONE MAGIC FILE.

The company magic path must be set by default to the path in the ClearTrigger Depot (e.g. Magic files are placed in the ClearTrigger Depot) or it can be set to another directory location by adding the special ClearTrigger Override Alias **ABS_bit_07_personality_windows** or **ABS_bit_07_personality_unix** in the clearbits file. The alias can be set to any valid directory for the associated operating system.

If an attempt is made to create an element and the user's MAGIC PATH environmental variable is not set to the expected company standard then a dialog box is displayed like the one below:

Functionality bit 8 - Prevent Embedded Spaces in Element Names

This bit prevents users from creating new elements with embedded spaces in the name (e.g. "this file is not valid.c" would not be allowed and "this_file_is_valid.c" would be allowed). Some development shops prefer to prevent embedded spaces in element names because it makes scripting much more complicated and will often cause user issues in InterOp (Windows and UNIX) configurations. Enabling this bit prevents their creation unless the user is the current VOB owner. In this case creation would be allowed, but the user would be notified that it is only allowed for them.

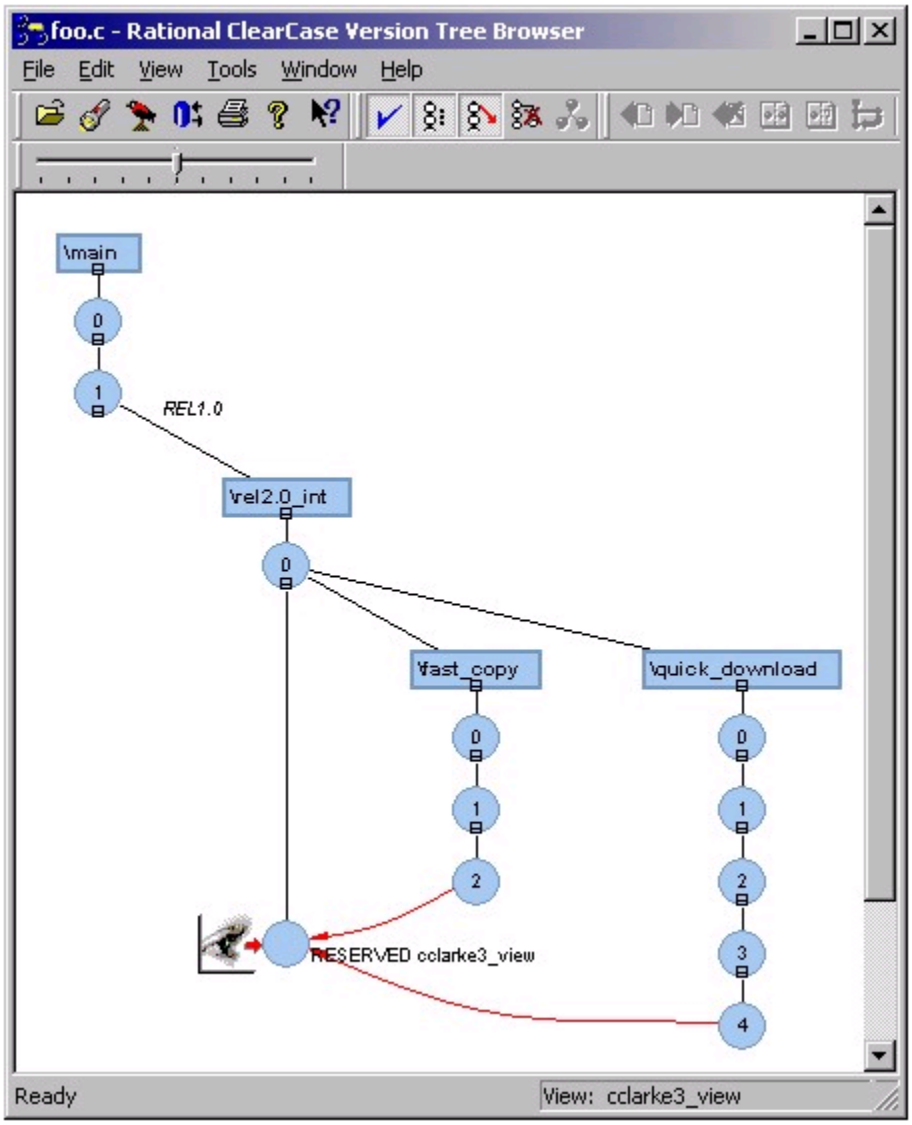
Equivalent processing to the "mkelem" command was added to the "ln", "ln -s" and "mv" commands.

This bit prevents their creation unless the user is the current VOB owner. When the user is not the current VOB owner then this dialog is displayed:

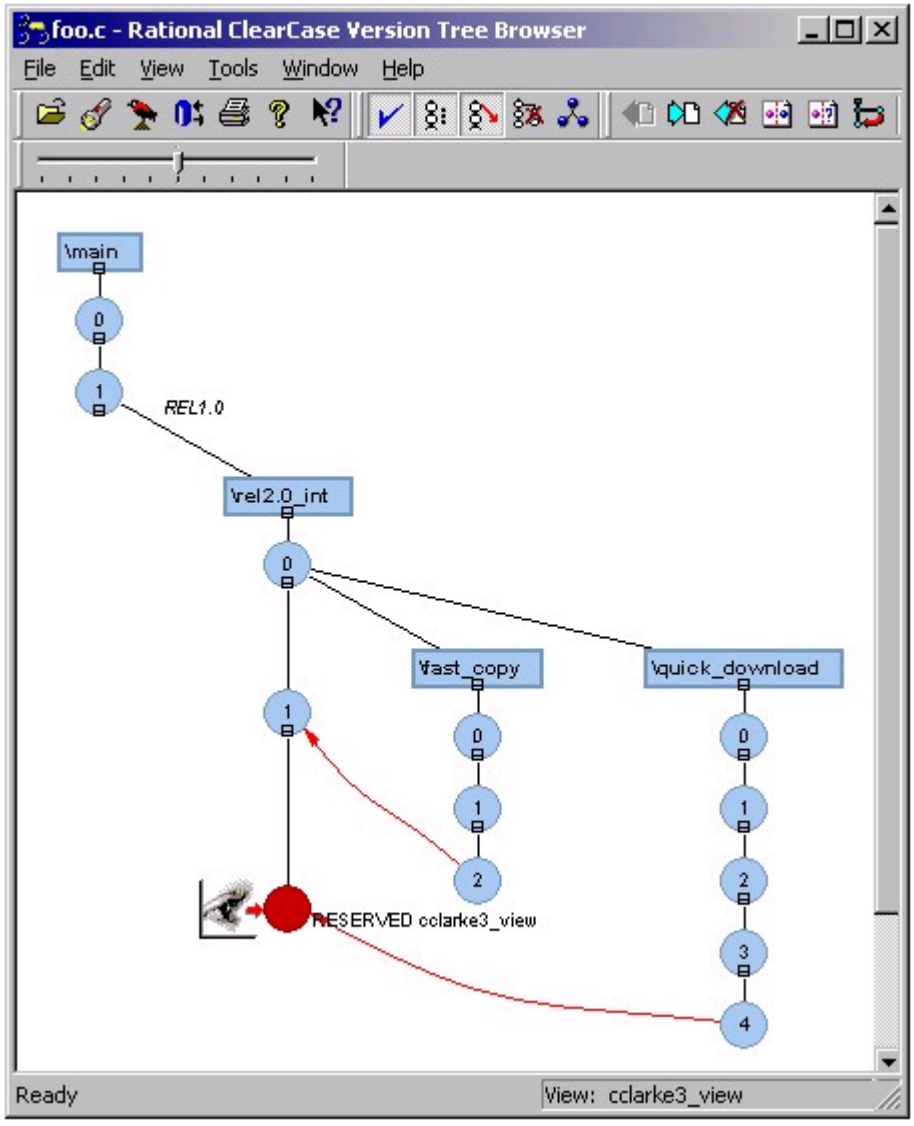
If the user were the current VOB owner then user would be notified that it is only allowed for them and the following dialog is displayed:

Functionality bit 9 - Multi-Merge Checkin not Allowed

This bit prevents checkins when the version is the destination of multiple merges.
Some development shops frown upon multiple merge checkins because it becomes difficult to know which changes are associated to which feature within the resulting version (e.g. the checkin below would not be allowed):

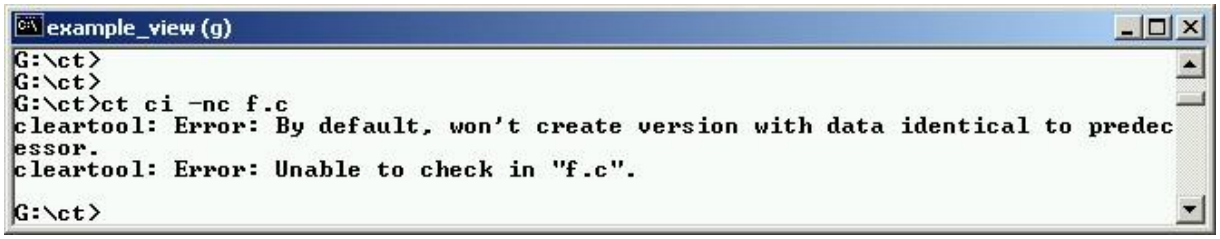


Because the following result is more desirable to the company and thus encouraged.

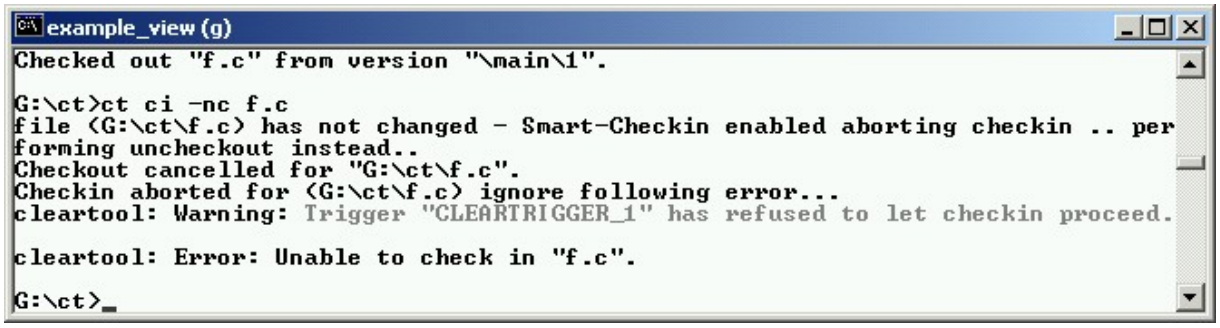


Functionality bit 10 – Smart Checkins

This bit will enforce that unchanged files or directories are automatically unchecked out when a checkin command is attempted. Many times there is no need to checkin elements because no change was made to the element. Usually the user does not know the element has not changed until they receive an error when a checking in the unchanged element as in the example that follows:



When this bit is enabled, ClearTrigger checks appropriate element types against its predecessor to determine if a change was made, if no change was made then the element is unchecked out instead.

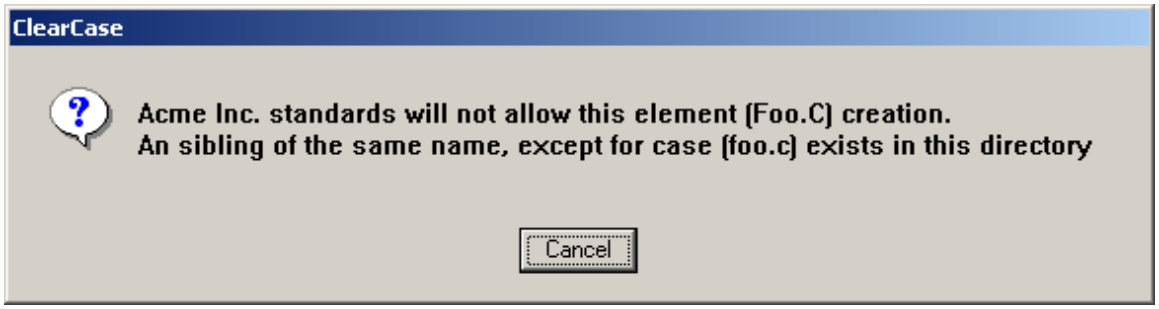


NOTE: An exception is made when the predecessor version is /main/0 or the current version and predecessor version are of zero length to allow for new elements to be created from the "add to source control" GUI without providing warnings.

NOTE: This functionality bit is ignored for ClearCase Web users.

Functionality bit 11 – Element Name Case Checking

This bit prevents the creation of elements in the same directory with the same case insensitive name. (e.g. it will not allow "foo.c" to be created if "Foo.c", "foo.C" or some such exists in the directory). An example of a dialog that would be displayed if one attempted to create such an element follows.



Equivalent processing was added to the "ln", "ln -s" and "mv" commands.

Functionality bit 12 – Atomic change hyperlink support

Within ClearTrigger there exists a method to link elements together such that they can be checked in, checked out or unchecked out together. The policy maker can then determine if the linked elements must be modified together or if it is just "suggested" that they are

Users themselves can create the appropriate links between the elements, as the ability to "bind" elements is usually a developer function. The hyperlink type "atomic_change" should be created in a VOB with ClearTrigger applied and then anyone can link elements together.

To create the hyperlink type once in the VOB, type the command below:

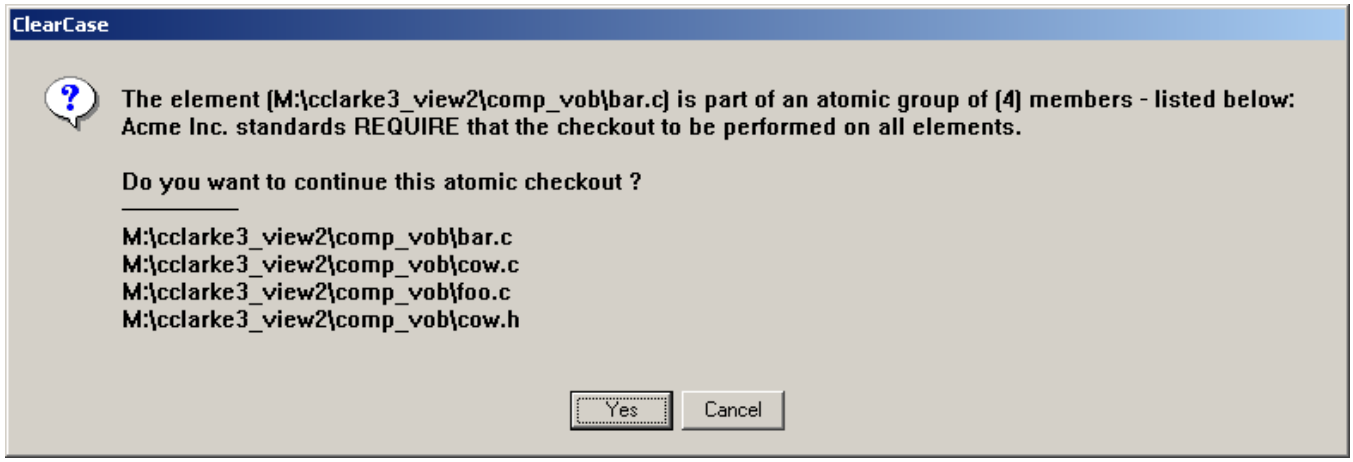
cleartool mkhltype –nc atomic_change

Then link elements together by using the native ClearCase mkhlink command like so:

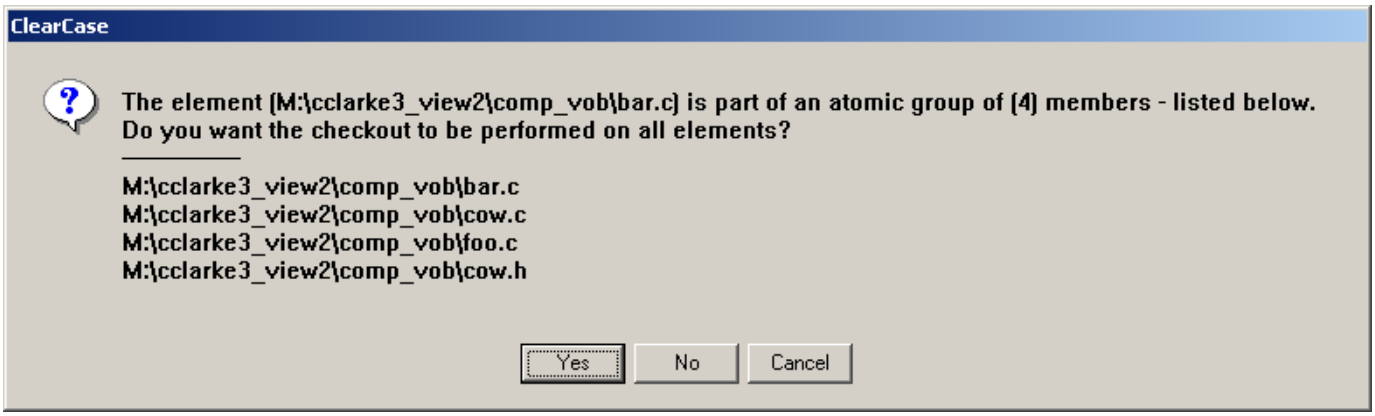
cleartool mkhlink –nc atomic_change foo.c@@ bar.c@@
cleartool mkhlink –nc atomic_change bar.c@@ cow.c@@
cleartool mkhlink –nc atomic_change cow.h@@ cow.c@@

It does not matter when the elements are linked, how many are linked, or the direction of the links. The linked family is "headless" such that if John knows of an association between bar.c and cow.c creates a link while Robert knows of a association between cow.h and cow.c and creates a link while Kathy creates a link between foo.c and bar.c, all elements are "equally" linked such that if Tomas checks out any of the files, they are all checked out.

By default if the bit is enabled, when a user checks out a linked file they may see a dialog similar to the one below that "forces" the association and provides them the opportunity checkout all of the elements or cancel the attempt. The default selection is the "Yes" button – to check them all out. If the region policy maker has turned on the Atomic Change" bit (bit 12) then only the "yes" and "cancel " option are given. The elements are checked out together by "force" or the checkout is canceled.



To change the personality of this bit such that the user is given a choice between “yes”, “no” or “cancel,” set the special ClearTrigger Override Alias **ABS_bit_12_personality** in the clearbits file to “choice”. The alias can be set to “choice” or “force” (the default is “force”). For example, if a user checked out “bar.c” when the cleartrigger_alias **ABS_bit_12_personality** is set to "choice" they may see a dialog like that below:

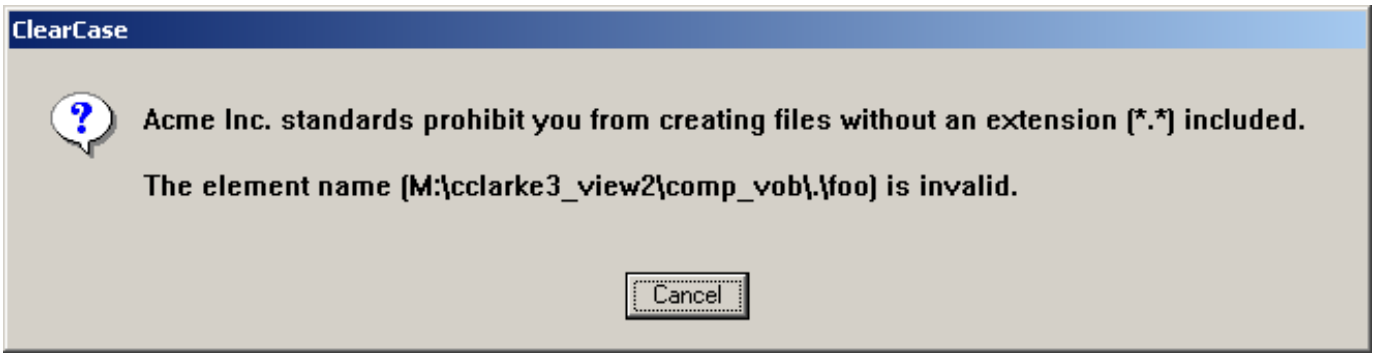


The default selection is the “Yes” button – to check them all out.

The elements do not have to be in the same directory and are not limited to files.

Functionality bit 13 – Require Element name Extensions

This bit prevents the creation of new file elements that do not have a file extension within the filename. When enabled the bit would not allow the file element “foo” to be created, but would allow “foo.c” or “foo.h” to be created. If a user attempts to create a new file element without a filename extension then a dialog box similar to the one below is displayed:



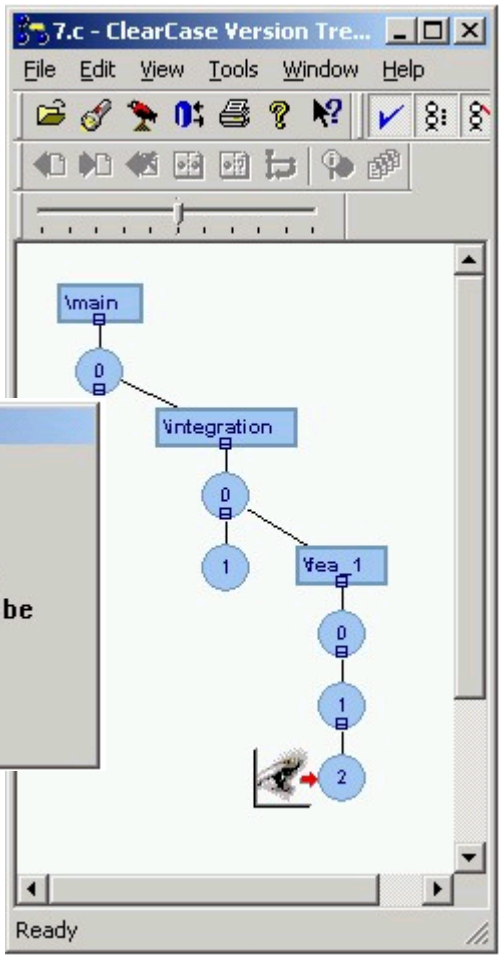
Equivalent processing was added to the "ln", "ln -s" and "mv" commands.

Functionality bit 14 – Notify Parent Change on Checkout

This bit ensures that a user is notified when they perform a checkout on a branch if the same element on that branch's parent branch (or default delivery branch if the branch is a UCM stream) has advanced and needs merging or rebasing.

For example, if the user performs a checkout on the element 7.c (the version selected \main\integration\fea_1\2) as depicted they would receive a warning dialog similar to the one below:

.

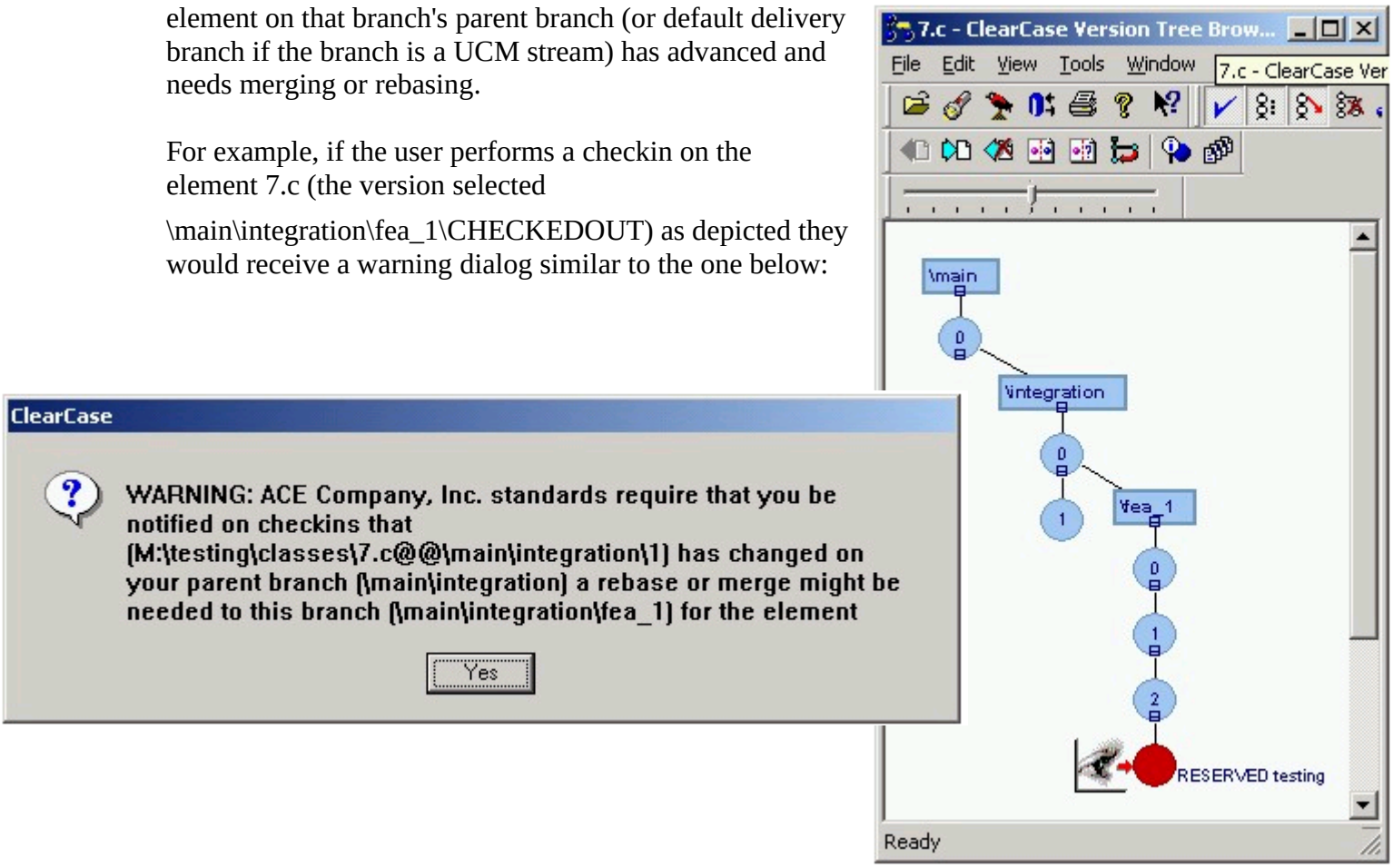


Functionality bit 15 – Notify Parent Change on Checkin

This bit ensures that a user is notified when they perform a checkin on a branch if the same element on that branch's parent branch (or default delivery branch if the branch is a UCM stream) has advanced and needs merging or rebasing.

For example, if the user performs a checkin on the element 7.c (the version selected

\main\integration\fea_1\CHECKEDOUT) as depicted they would receive a warning dialog similar to the one below:



Functionality bit 16 – Prevent Data Loss

This bit prevents all users, other than the VOB owner, from performing certain destructive cleartool commands. The cleartool sub-commands that are disallowed are:

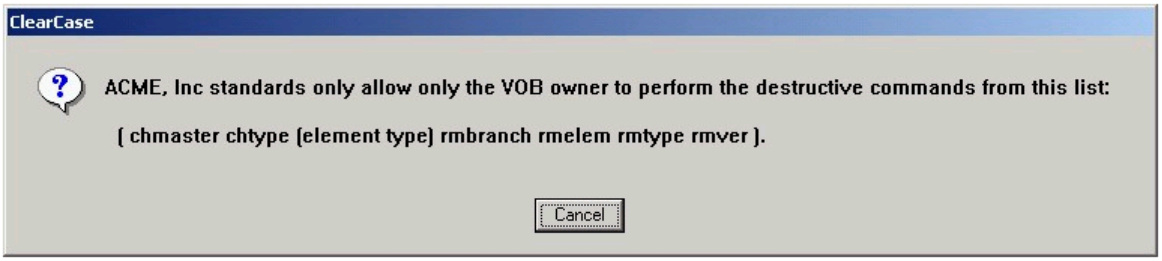
- chmaster
- chtype (element type)
- rmbranch
- rmelem
- rmtype
- rmver

as these are commands that can remove data permanently from the VOB and are frequently the cause of accidental data deletion that can only be repaired by recovering from backup tape or other sources.

At ABS we live by this rule:

"If your user gets into trouble that you cannot get them out of without having to go to last night's VOB backup or requesting data from another replica then it is not their fault... it is *yours*" - ABS

For example, if a user attempted one of the commands they would receive a dialog like the one below:

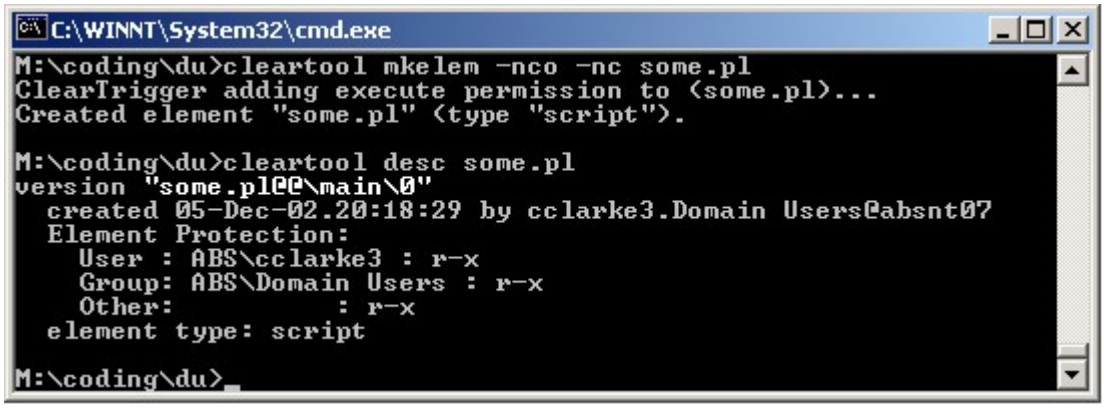


Functionality bit 17 – Auto Add Execute Permissions

This bit automatically adds execute permissions to newly created elements (programs and scripts) when those elements are created. When new file elements are added to the VOB known to commonly need execute permission either by name (e.g. *.pl, *.sh, .login, *.dll, *.exe, etc) or known element type (e.g. scripts, perl_scripts, executables, etc) then owner, group and other execute permissions are added to the new file element.

The elements that will have the execution bit enabled when created are those that end in: ".sh .ksh .profile .login .cshrc .kshrc .csh .exe .pl .dll .bat " or those where their clearcase element type contains the string “program or “script”

For example, if a user created new elements (**some.pl**, **some.h** and **some.c**), execute permission would be added to **some.pl** as depicted below:



To change the personality of this bit to add or remove extensions to/from the default extensions just make sure to set the special purpose special ClearTrigger Override Alias **ABS_bit_17_exec_override** in the clearbits file. Set the alias to a space separated list of extensions to add or remove from the default behavior.

For example, the alias setting below:

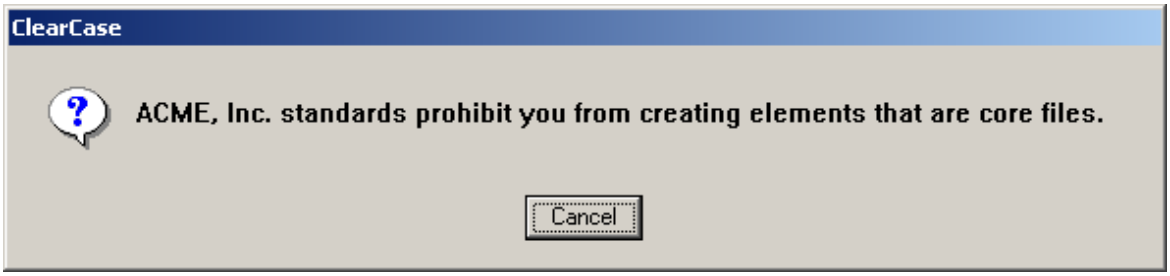
#!cleartrigger_alias ABS_bit_17_exec_override .glb .ffs !.pl

will add ***.glb** and ***.ffs** files to the default files processed for auto exec processing and will remove the ***.pl** from auto exec processing.

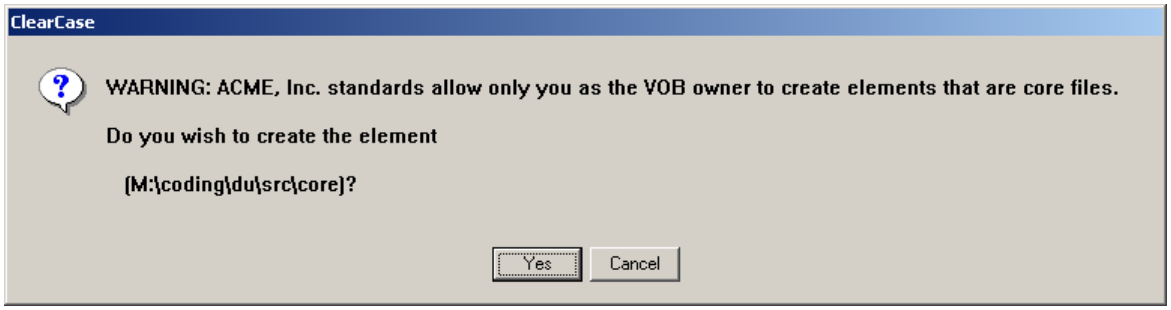
Functionality bit 18 – Auto Prevent "core" Files

This bit automatically prevents files named "core" or files of element type "core" from being added to the VOB.

For example, a user would be prevented from creating a new element named "core" to the VOB like so:



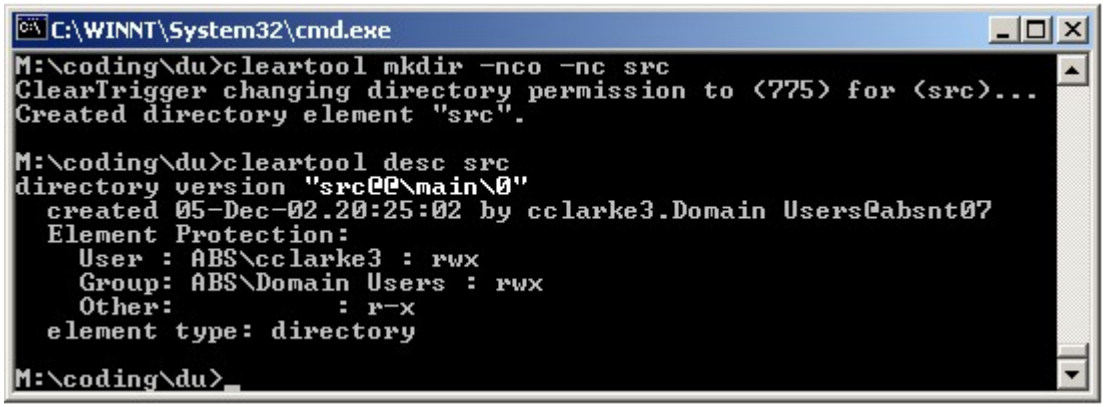
If the user is the current VOB owner, they would be notified that only they, as the VOB owner, can perform this action and given a choice to proceed with the element creation or not:



Functionality bit 19 – Auto Protect "directory" Elements

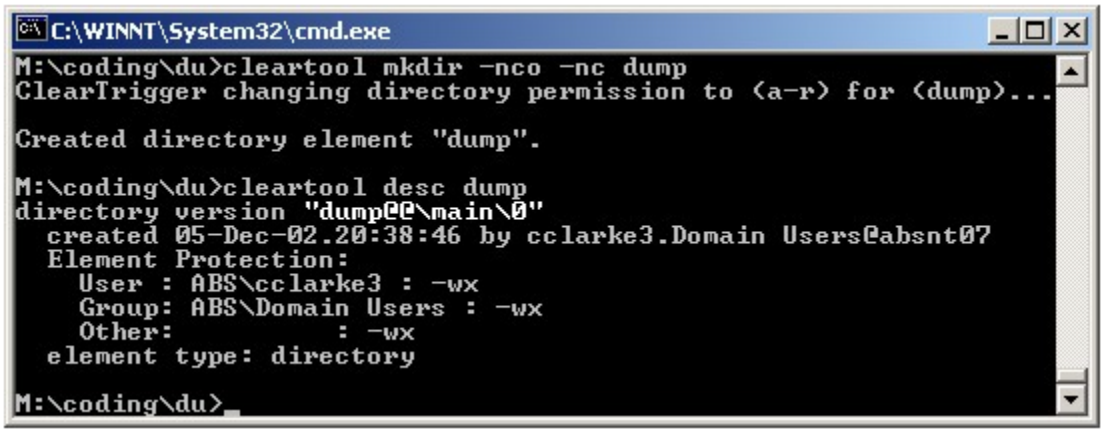
This bit automatically protects newly created directory elements as **775** (or some other default - including changing ownership) in the VOB.

A user creating a new directory might see output similar to the output example below:



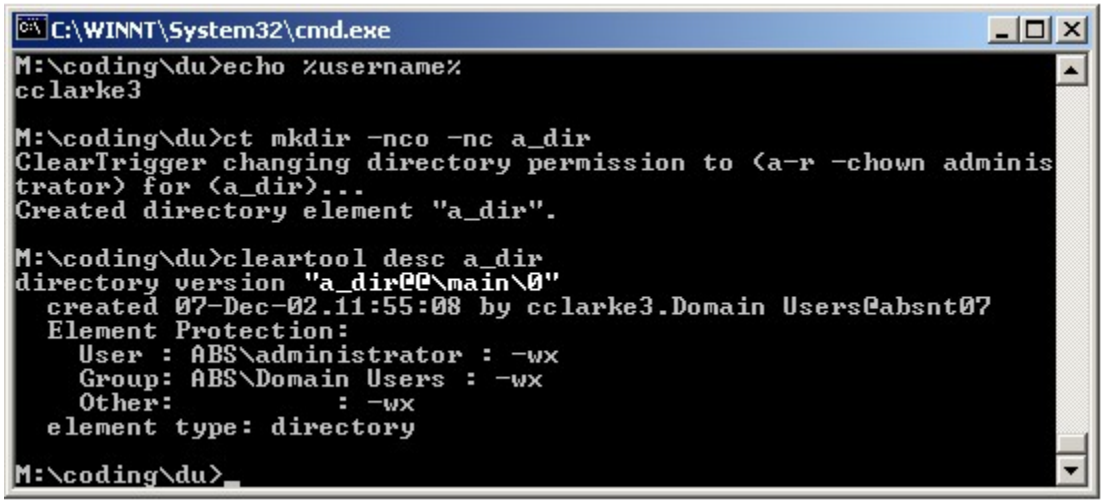
To change the personality of this bit to make the default of directories other than 775, just make sure to set the special purpose special ClearTrigger Override Alias **ABS_bit_19_protection_override** in the clearbits file. The alias can be set to any valid assignment as per the cleartool man page for cleartool protect. Both absolute values (e.g. "770") and symbolic values (e.g. "a-r") can be used.

A user creating a new directory when the cleartrigger_alias **ABS_bit_19_protection_override** is set to "a-r" would see output like the similar to the example below:



One may also assign the directory "ownership" at this time by adding additional information to the special purpose cleartrigger_alias **ABS_bit_19_protection_override** in the clearbits file. After assigning the permissions you can add -chown or -chgrp options to the value. For example, a user (cclarke) creating a new directory when the cleartrigger_alias

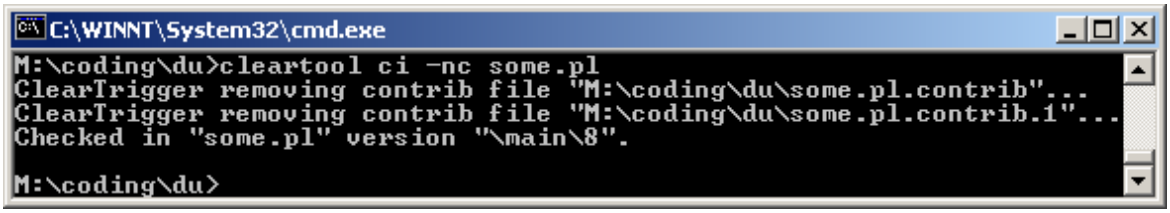
ABS_bit_19_protection_override is set to "a-r -chown administrator" would see output similar to that in the following example:



Functionality bit 20 – Auto Remove ".contrib" Files

This bit automatically removes ".contrib." files that are created when merges are performed. The files are removed after a checkin or uncheckout of the associated file element.

A user that previously performed a merge (or several merges) to some.pl file and checked in the file may see output similar to that in the following example:

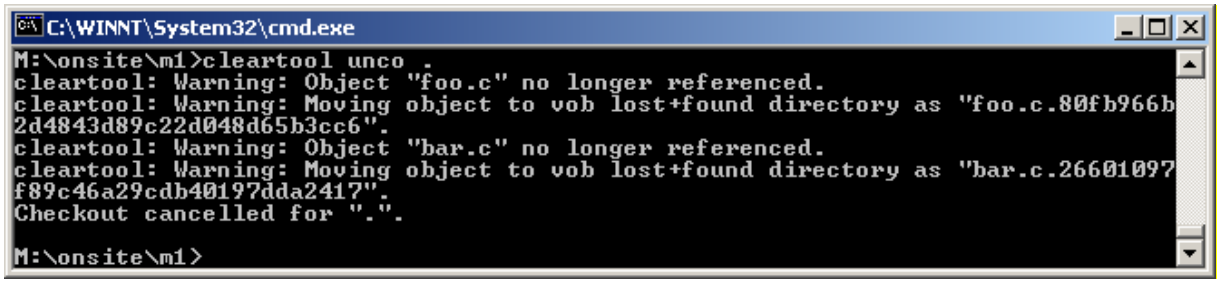


Note: The user can use the environmental variable ABS_CONTRIB_KEEP to ignore the value of the region defined functionality bit. If the variable is set then any .contrib files are not removed during checkin or uncheckout for that user.

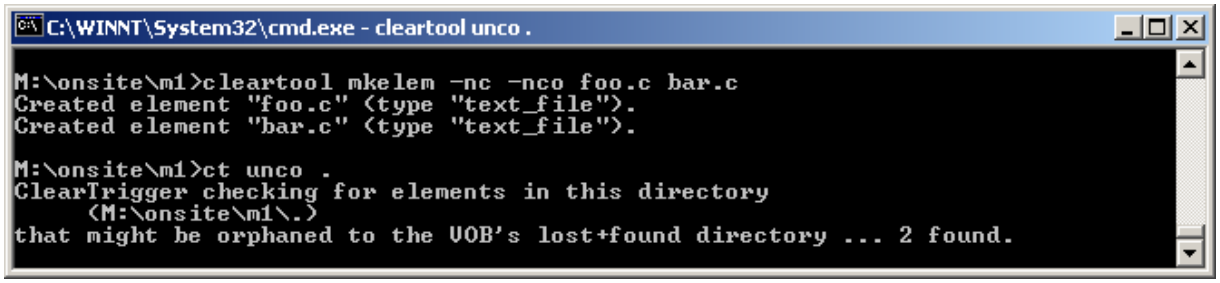
Functionality bit 21 – Auto Warn lost+found

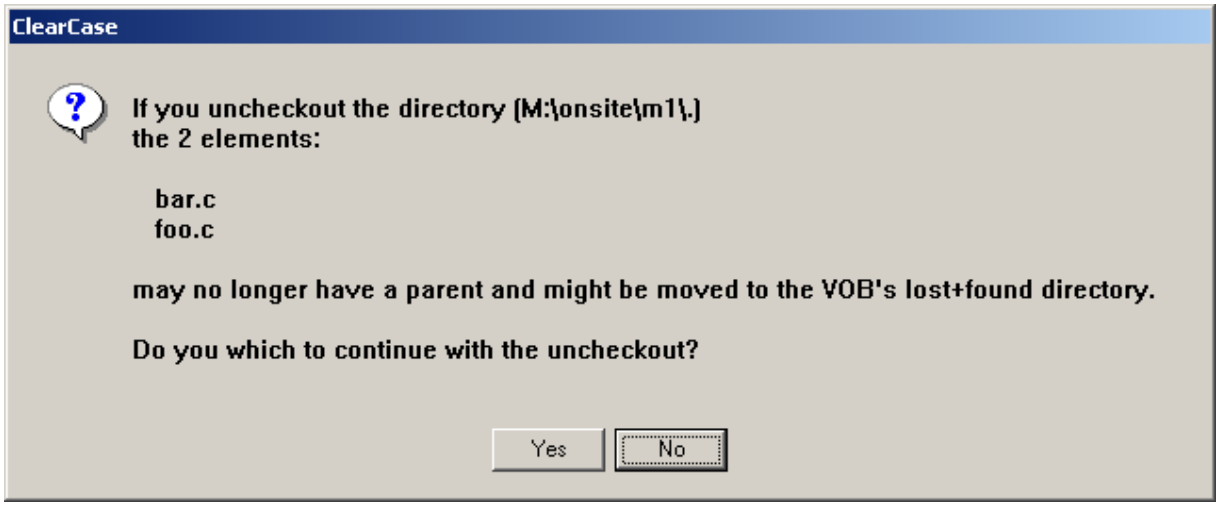
This bit automatically warns of possible elements that may be moved to the VOB's lost+found directory as a result of a directory uncheckout, ***before*** the directory uncheckout is attempted.

A user that previously performed a directory uncheckout might have had several element moved to lost+found. They were only notified of this "after" the successful uncheckout of the directory resulting in notification (a bit late) that elements were moved to lost+found as in the following example:



When the bit is enabled the same uncheckout of the directory would warn of the possible movement of elements to lost+found first and give the user the option to continue or cancel the uncheckout operation.





You can change the personality of this bit so that the user is allowed to auto answer this dialog with "yes" or "no" so no human intervention is needed. This is handy for builds or other automated processes that perform directory uncheckouts. To allow this "auto answer" functionality the policy maker can appropriately set the special purpose ClearTrigger Override Alias **ABS_bit_21_auto_answer_allow** in the clearbits file to match the appropriate Users, Groups, VOBs, Elements, Views, Times or even ClearCase Regions.

If a directory uncheckout might cause an element to be placed in the VOB's Lost+Found directory and that current user has set their environmental variable CLEARTRIGGER_BIT_21_AUTO_INTERACTIVE_ANSWER to "yes" or "no" then that dialog would not appear, but the warning would appear within SDTOUT and that question would be auto answered according to the environmental variable.

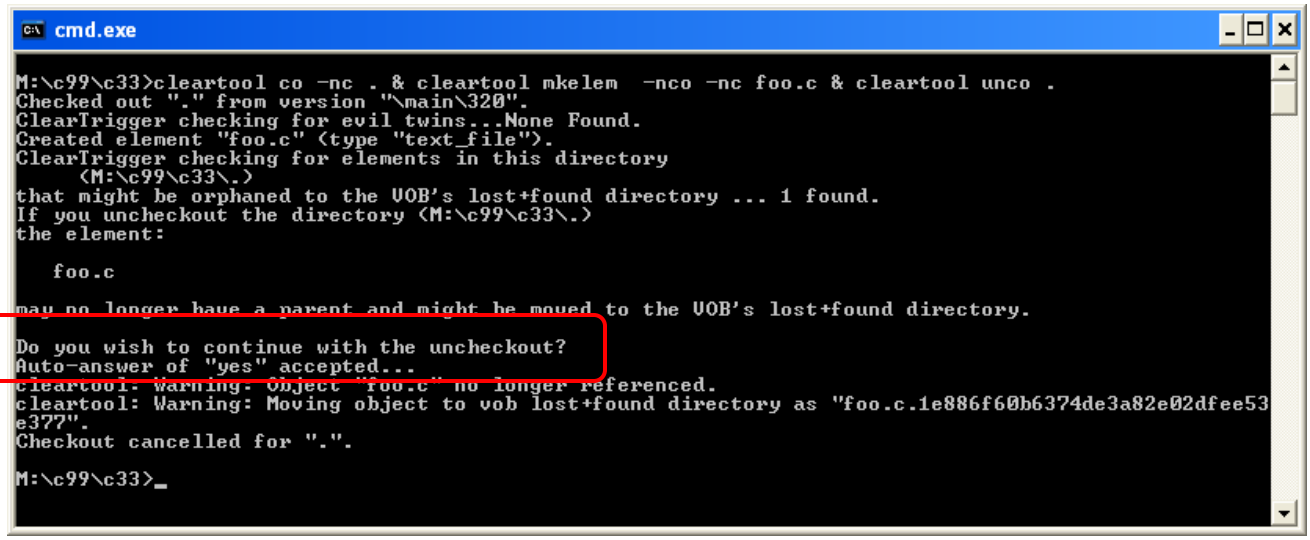
An example of such a scenario where **ABS_bit_21_auto_answer_allow** is set in the clearbits file like below:

```
#!cleartrigger_alias ABS_bit_21_auto_answer_allow (cclarke)(cc_build)
```

and the user "cclarke" (or the build id "cc_build") has set their environmental variable

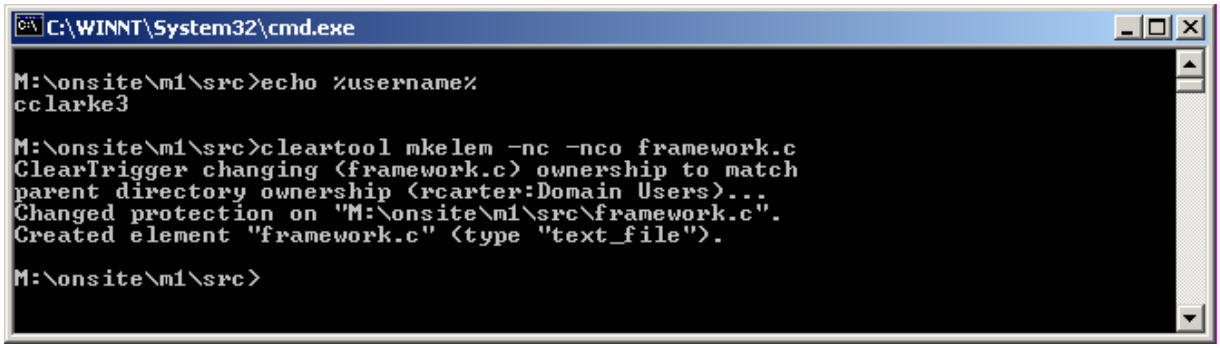
CLEARTRIGGER_BIT_21_AUTO_INTERACTIVE_ANSWER=yes

In this scenario, checking out a directory, adding an element (foo.c) in that directory and then unchecking out that directory will cause **foo.c** to go to **Lost+Found**. Rather than a **warning dialog** being displayed for the user to confirm the uncheckout, the user shown the warning at STDOUT and the auto answer of "yes" (the value of the cclarke's **CLEARTRIGGER_BIT_21_AUTO_INTERACTIVE_ANSWER** environmental variable) is provided allowing the uncheckout to continue without human intervention. This is depicted below:



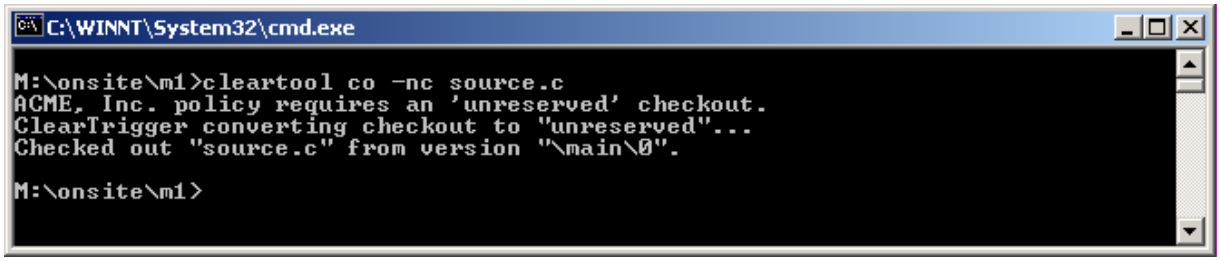
Functionality bit 22 – Own Like Parent Dir

This bit will automatically change the ownership of newly created elements to match the ownership (user and group) of the parent directory of the new element. This allows for a consistent ownership of elements in a directory as in the following example:



Functionality bit 23 – Force Checkouts to Unreserved (new in 12.3)

This bit will automatically change the state of newly created checkout to the 'unreserved' state. This allows for developers to work with their checkouts for longer than expected without preventing other developers from performing checkins. Refer to the following example:



Command Key Definitions (ClearTrigger Feature Set)

In a ClearTrigger region, the final section of clearbits_file allows you to specify your region-wide or enterprise-wide policy. This is probably the most important section of the file. Defining policy can be as simple as adding new command keys. The keys can be duplicated if one wishes to implement a schema that indicates "severity" or "priority" (e.g. perhaps you want a data-base updated twice for some actions and once for others). Each key definition must begin on a new line. Matching keys are invoked in order of appearance and may match multiple keys (e.g. a *checkin* command would match both the "MODIFY_ELEM" and "checkin" keys).

Correct formatting of the command key definition is essential to successfully implement ClearTrigger. Descriptions of the individual keys, including proper formatting, are outlined in this document under 'Key Descriptions'. The syntax for the command key definition is as follows:

cmd_id;RIN_type;RIN_list;inhibit_flag;comment_req;notify_flag;cmd_list_type;
cmd_list;cmd_trigger_list_type;trigger_list;Series_begin_end_bit;Windows_script;
Windows_script_params;UNIX _script;UNIX_script_params;inhibit_cleartrigger_GUI_bit;
{dialog_type};{dialog_prompt};{dialog_mask or default_string or dialog_choices};

The **cmd_id** of the key definition determines which cleartool command and/or command groups are to be executed by ClearTrigger. It also determines when ClearTrigger will execute--as a preop or postop. Here you can enforce policy on individual cleartool commands (e.g. post_checkin) or entire command groups (e.g. pre_MODIFY_DATA). Command groups can appear as *element type* operations or *trigger type* operations.

Operations for '-Element -all' Trigger Types

The following list shows the operation keywords and the functions they encompass for use in definitions of -element -all trigger types (**-element –all**).

Operation Keyword	Restrictions Checked
MODIFY_ELEM	
checkout	Element type, branch type
reserve	Element type, branch type
uncheckout	Element type, branch type
unreserve	Element type, branch type
mv	Element type - See **NOTE #2** at end of tables

Operation Keyword	Restrictions Checked
MODIFY_DATA	
checkin	Element type, branch type
chtype	All type objects
lnname	Element type, branch type
lock	See ***NOTE #1*** at end of tables
mkbranch	Element type, branch type
mkelem	Element type - See ***NOTE #3*** at end of tables
mkslink	N/A - See *NOTE #4* at end of tables
protect	Element type
rmbranch	Element type, branch type
rmelem	Element type
rmname	N/A
rmver	Element type, branch type
unlock	See ***NOTE #1*** at end of tables

Operation Keyword	Restrictions Checked
MODIFY_MD	
chevent	See ***NOTE #1*** at end of tables
chmaster	See ***NOTE #1*** at end of tables
mkattr	Element type, attribute type, branch type
mkhlink	Element type, hyperlink type, branch type
mklabel	Element type, label type, branch type
mktrigger	Element type, trigger type
rmattr	Element type, attribute type, branch type
rmhlink	Element type, hyperlink type, branch type
rmlabel	Element type, label type
rmtrigger	Element type, trigger type

NOTE #1: These commands fire for both element and type triggers. To distinguish between the two types use a command modifier ("_elem" or "_type") to distinguish them, (e.g. chevent_elem" or "chevent_type"). If you do not distinguish a type (e.g. "chevent"), then it applies to both "chevent_elem" and "chevent_type" commands.

NOTE #2: ClearCase itself has no "mv" command within its list. The command is implemented by the two commands lnname and rmname. ClearTrigger allows you to set a trigger on "mv" that does not interfere with the separately called lnname or rmname commands. For more on this refer to the section entitled Additional "mv" command.

NOTE #3: In native ClearCase, the "mkelem" command was actually implemented by calling a cleartool lnname command followed by a cleartool mkelem command. ClearTrigger allows you to set a trigger on "mkelem" that does not interfere with the separately called lnname commands. For more on this refer to the section entitled: Sophisticated mkelem command processing.

NOTE #4: In native ClearCase, the "mkelem" command was actually implemented by calling a cleartool lnname command followed by a cleartool mkelem command. ClearTrigger allows you to set a trigger on "mkelem" that does not interfere with the separately called lnname commands. For more on this refer to the section entitled: Sophisticated ln –s (mkslink) command processing.

Operations for 'Type' Trigger Types

The following list shows the operation keyword(s) and the functions they encompass for use in definitions of type trigger types (**-type**).

Operation Keyword
MODIFY_TYPE
chevent
chmaster
lock
mkattr
modtype
rmattr
rmhlink
rmtype
rntype
unlock

Operations for 'UCM' Trigger Types)

The following list shows the operation keyword(s) and the functions they encompass for use in definitions of UCM trigger types (**-ucm**).

Operation Keyword
MODIFY_UCM
deliver_cancel
deliver_complete
deliver_start
rebase_cancel
rebase_complete
rebase_start
mkactivity
chactivity
rmactivity
setactivity
mkstream
chstream
rmstream
mkbl
chbl
rmbl
mkproject
chproject
rmproject
mkcomp
rmcomp
mkfolder
chfolder
rmfolder

Additional Command Information

To simplify policy creation, additional processing is performed on some particular ClearCase command and some commands were ***added***. This section describes the additional processing.

More on UCM commands

In ClearCase many UCM commands cause other UCM and non-UCM commands to fire. Measures were taken so that triggers placed on a command that causes multiple commands to fire will not cause triggers on those additional commands to fire unless you want it to. For example, perhaps you have restrictions on mkbrtype that you don't want for mkstream (which also makes a branch type and thus causes mkbrtype to fire), in ClearTrigger, the commands will not interfere with one another. Or perhaps you prevent developers from making label types using the mkbrtype command, but need them to be able to perform a UCM mkbl command which might cause a label type to be copied from the process VOB; again ClearTrigger handles these cases seamlessly.

Additional UCM/Base ClearCase Command Distinctions

On occasion it is critical to perform the same script or provide the same action on a Base ClearCase command (e.g. checkin) that one would perform on a command if it called as part of the implementation of a UCM command (e.g. deliver). Measures were taken in ClearTrigger to ensure that a trigger attached to checkin did NOT fire when that checkin was part of a UCM series (as explained in the ***previous*** section). When command keys are defined (by default) they refer to either a UCM command or a Base ClearCase command as described in the previous section. When you specifically require that the command key match only if the command is indeed part of a UCM series (e.g. a **checkin** command that is part of a UCM **deliver** command) you may add a **sub-UCM** ("**_u**") designator to the command key definition.

Therefore to start a command key that is to apply to Base ClearCase checkin commands one would start the key similar to the one below:

pre_checkin;

While to start a command key that is to apply to checkins that are the result of UCM commands one would start the key similar to the one below:

pre_checkin_u;

Finally, to start a command key that is to apply to base checkins and checkins that are the result of a UCM command one would start the key similar to the one below:

pre_checkin pre_checkin_u;

Additional "mv" command

Native ClearCase has no trigger that fires on the cleartool mv command. The cleartool mv command is actually implemented by calling a cleartool lnname command followed by a

cleartool rmname command. Because of this the only way to properly place a trigger on a move was to actually place a trigger on lnname for pre-op triggers and rmname for post-op triggers. This is further complicated in that one would have to place code in any written trigger script to prevent its execution when an actual cleartool lnname or cleartool rmname command was singularly called.

With ClearTrigger you have the ability to set trigger on the cleartool mv command itself. This trigger will fire only on lnname for the pre-op situation and the trailing rmname on the post-op condition and only when these commands are fires as a result of an actual "mv" command. This prevents incomplete or unwanted processing that results when one of the two commands fails.

When a "mv" command key (e.g. *pre_mv* … or *post_mv*…) is created it will not fire when the user executes the *lnname* or *rmname* command. When command keys are placed on the *lnname* or *rmname* command, it does not fire for the "mv" command even though that command is built from the *lnname* and *rmname* commands.

Sophisticated mkelem command processing

In native ClearCase, the **mkelem** command was actually implemented by calling a cleartool **lnname** command followed by a cleartool **mkelem** command. Because of this it was possible to place a pre-op trigger on mkelem that prevented elements from being created, but the preceding lnname was already successfully executed. This was the cause of many "checked out but removed" errors. The most proper way to place a pre-op trigger on mkelem was to actually place it on lnname with code added to ascertain if the trigger was actually called for a mkelem command or a separately called lnname command.

This was further complicated in one had to place code in any written trigger script to prevent its execution when actual cleartool lnname command was separately called.

In ClearTrigger when you set a trigger on the cleartool mkelem command the trigger will fire only on lnname for the preop situation and the trailing mkelem on the post-op condition. This prevents incomplete or unwanted processing that results when one of the two commands fails.

When a "mkelem" command key (e.g. *pre_mkelem* … or *post_mkelem*…) is created it will not fire when the user executes the *lnname* command. When command keys are placed on the lnname command the trigger will not fire for the mkelem command even though that command is built from the *lnname* and *mkelem* commands.

Sophisticated ln –s (mkslink) command processing

In native ClearCase, the **mkslink** command was actually implemented by calling a cleartool **lnname** command followed by a cleartool **mkslink** command. Because of this it was possible to place a pre-op trigger on mkslink that prevented the symbolic link from being created, but the preceding lnname was already successfully executed. This was the cause of many "checked out but removed" errors. The most proper way to place a pre-op trigger on mkslink was to actually place it on lnname with code added to ascertain if the trigger was actually called for a mkslink command or a separately called lnname command.

This was further complicated in that one had to place code in any written trigger script to prevent execution when actual cleartool lnname command was separately called.

In ClearTrigger, when you set a trigger on the cleartool mkslink command the trigger will fire only on lnname for the preop situation and the trailing mkslink on the post-op condition. This prevents incomplete or unwanted processing that results when one of the two commands fails.

When a "mkslink" command key (e.g. ***pre_mkslink*** … or ***post_ mkslink*** …) is created it will not fire when the user executes the *lnname* command. When command keys are placed on the lnname command the trigger will not fire for the move command even though that command is built from the *lnname* and *mkslink* commands.

Key Descriptions

The section outlines command keys used in the command key definition to help create policy for an organization.

Cmd_id

- **cmd_id** – any valid operation as per the *cleartool mktrtype* command. Valid operations are listed under Command Key Definitions for both elements and trigger types. They can also be listed as per the *cleartool man mktrtype* command. Included are the command operation groups MODIFY_ELEM, MODIFY_DATA, MODIFY_TYPE and MODIFY_MD.

  This value is combination of the **command_key_type** and **command_key_op** to build the **Command Key Operation** token.

  For Example:

Command key_type	command key_op	command key_operation
pre_	checkout	pre_checkout
pre_	MODIFY_ELEM	pre_MODIFY_ELEM
post_	MODIFY_ELEM	post_MODIFY_ELEM
post_	checkout	post_checkout

  Note: It is perfectly acceptable to have multiple Command Key Operation Tokens on a single line like so:

  pre_checkout post_rmver

RIN_type

- **RIN_type -** identifies which type of RIN list is defined in next key, **RIN_list**. It's specified by an [Rr], [Ii], or [Nn].

RIN Type Value	Type Value Description
'R'	Indicates use of a *restriction list*. (Fully Matching)
'I'	Indicates use of an *inclusion list*. (Fully Matching)
'N'	Indicates use of a *negation list*. (Fully Matching)
'r'	Indicates use of a *restriction list*. (Single Matching)
'i'	Indicates use of an *inclusion list*. (Single Matching)
'n'	Indicates use of a *negation list*. (Single Matching)

RIN_list

- **RIN_list** - either restricts or expands what the **cmd_id** applies to. The list may be empty or contain a *restriction list*, *inclusion list* or *negation list*. The list may be designated as **"fully"** matching (where all different types in the list must match) or **"single"** matching (where only a single type must match). Lists can contain references to base ClearCase data types and UCM data types (**component**, **project** and **stream** as in native ClearCase as well as **activity** and **folder** which only ClearTrigger supports). A description of each list follows:

  - **Restriction List**
    When **RIN_type** is defined to be **'R' or 'r',** then **RIN_list** represents a *restriction list*. When there is no restriction list (the list is unpopulated) then there are no restrictions and all commands that match the associated operation will apply; when the list is populated, only instances that match the restriction will apply.

    Any number of restrictions may be placed in the restriction. There can be a restriction on any of the metadata types: **attributes**, **branches**, **elements**, **hyperlinks**, **labels** and **triggers** as well as certain UCM Data Types: **activities**, **components**, **folders**, **projects** and **streams** (ClearTrigger supports matching on **activities** and **folders** even though native ClearCase does not). If the RIN_type is 'R', ***"full matching"*** applies and restrictions of the same type are ***OR***ed together and restrictions of different types are ***AND***ed together before the restriction is tested. If the RIN_type is 'r', ***"single matching"*** applies and all restrictions of any type are ***OR***ed together before the restriction is tested.

    A ***restriction list*** contains zero or more base metadata type selections or UCM type selections of the form type<name> where type is from:

Base type	UCM type
• attype	• activity
• brtype	• component
• eltype	• folder
• hltype	• project
• lbtype	• stream
• trtype	

    and **Type Name** which is an implicit Base or UCM type name like "REL1.9", "integration" or a **pattern** to match several or a few type instances. You may use **~alias** names to identify large or often used series of type selections.

Example Type	example
attribute	; attype<QA_State> ;
branch	; brtype<integration> ;
element	; eltype<c_source> ;
hyperlink	; hltype<merge> ;
label	; lbtype<BETA> ;
trigger	; trtype<no_co> ;
UCM activity	;activity<bug_152>;
UCM component	;component<\comp1>;
UCM folder	; folder<a_folder>;
UCM project	;project<project_z>;
UCM stream	;stream<feature_x>;
patterns	; lbtype<REL#.#_*> brtype<*_int_*> ;
patterns	; attype<CR##?_*> trtype<*> ;
aliases	; ~protected_labels ~restricted_braches ;

Note: Rather than implement restriction lists for element –all triggers and inclusion lists for –type triggers as ClearCase has done, ClearTrigger has implemented restriction lists for both -type and element -all triggers.

- **Inclusion List**
  When the **RIN_type** is defined to **'I' or 'i'** then the **RIN_list** represents an *inclusion list*. Inclusion lists are used when defining 'type' trigger types. When there is no inclusion list (it is unpopulated) then there are no inclusions and all commands that match the associated operation are still ignored; when the list is populated, only instances that match the inclusion will apply.

  Any number of inclusions may be placed in the inclusion list. There can be an inclusion on any of the metadata types: **attributes**, **branches**, **elements**, **hyperlinks**, **labels** and **triggers** as well as certain UCM Data Types: **activities**, **components**, **folders**, **projects** and **streams** (ClearTrigger supports matching on **activities** and **folders** even though native ClearCase does not). If the RIN_type is 'I', ***"full matching"*** applies and inclusions of the same type are ***OR***ed together and inclusions of different types are ***AND***ed together before the inclusion is tested. If the RIN_type is 'i', ***"single matching"*** applies and all inclusions of any type are ***OR***ed together before the inclusion is tested.

An ***inclusion list*** contains zero or more base metadata type selections or UCM type selections of the form type<name> where type is from:

Base type	UCM type
• attype	• activity
• brtype	• component
• eltype	• folder
• hltype	• project
• lbtype	• stream
• trtype	

and **Type Name** which is an implicit Base or UCM type name like "REL1.9", "integration" or a **pattern** to match several or a few type instances. You may use **~alias** names to identify large or often used series of type selections.

Example Type	example
attribute	; attype<QA_State> ;
branch	; brtype<integration> ;
element	; eltype<c_source> ;
hyperlink	; hltype<merge> ;
label	; lbtype<BETA> ;
trigger	; trtype<no_co> ;
UCM activity	;activity<bug_152>;
UCM component	;component<\comp1>;
UCM folder	; folder<a_folder>;
UCM project	;project<project_z>;
UCM stream	;stream<feature_x>;
patterns	; lbtype<REL#.#_*> brtype<*_int_*> ;
patterns	; attype<CR##?_*> trtype<*> ;
aliases	; ~protected_labels ~restricted_braches ;

Note: Rather than implement restriction lists for element –all triggers and inclusion lists for –type triggers as ClearCase has done, ClearTrigger has implemented inclusion lists for both -type and element -all triggers.

- **Negation List**
  Sometimes it is easier to define what one does ***not*** want than it is to define what one does want. This is why ClearTrigger has implemented the *negation list*. When the **RIN_type** is defined to **'N' or 'n',** then the **RIN_list** represents a negation list. When there is no negation list (the list is unpopulated) then there are no negations and all commands that match the associated operation apply; when the list is populated, only instances that do not match the negation will apply.

  Any number of negations may be placed in the negation list. There can be a negation on any of the metadata types: **attributes**, **branches**, **elements**, **hyperlinks**, **labels** and **triggers** as well as certain UCM Data Types: **activities**, **components**, **folders**, **projects** and **streams** (ClearTrigger supports matching on **activities** and **folders** even though native ClearCase does not). If the RIN_type is 'N', ***"full matching"*** applies and negations of the same type are ***OR***ed together and negations of different types are ***AND***ed together before the negation is tested. If the RIN_type is 'n', ***"single matching"*** applies and all negations of any type are ***OR***ed together before the negation is tested.

A ***negation list*** contains zero or more base metadata type selections or UCM type selections of the form type<name> where type is from:

Base type	UCM type
• attype • brtype • eltype • hltype • lbtype • trtype	• activity • component • folder • project • stream

and **Type Name** which is an implicit Base or UCM type name like "REL1.9", "integration" or a **pattern** to match several or a few type instances. You may use **~alias** names to identify large or often used series of type selections.

Example Type	example
attribute	; attype<QA_State> ;
branch	; brtype<integration> ;
element	; eltype<c_source> ;
hyperlink	; hltype<merge> ;
label	; lbtype<BETA> ;
trigger	; trtype<no_co> ;
UCM activity	;activity<bug_152>;
UCM component	;component<\comp1>;
UCM folder	; folder<a_folder>;
UCM project	;project<project_z>;
UCM stream	;stream<feature_x>;
patterns	; lbtype<REL#.#_*> brtype<*_int_*> ;
patterns	; attype<CR##?_*> trtype<*> ;
aliases	; ~protected_labels ~restricted_braches ;

Inhibit_flag

- **inhibit_flag** – This flag is used to partially or globally inhibit a command for VOBs in a ClearTrigger Region. It can also be used to stop further ClearTrigger processing on a particular event (Stop Key Processing). The flag may be set to any of the following valid values:

Inhibit_Flag Value	Inhibit_flag Description
'0'	Do NOT globally inhibit the command or command group.
'x'	Globally inhibits the command or command group for all users ***except*** the ***Current VOB Owner***.
'X'	Globally inhibits the command or command group for all users ***including*** the ***Current VOB Owner***.
's'	Stop ClearTrigger Key processing for the matching command or command group for all users ***except*** the ***Current VOB Owner***.
'S'	Stop ClearTrigger Key processing for the matching command or command group for all users ***including*** the ***Current VOB Owner***.
'V'	Stop ClearTrigger Key processing for the matching command or command

	group for ***only*** the ***Current VOB Owner***.

Comment_restriction

- **comment_restriction –** This field is used to enforce minimum commenting requirements on commands executed within VOBs in a ClearTrigger Region. When used in a **pre_op** key these users will be shown a comment dialog box when attempting this command without satisfying the comment restriction in the region that "requires" a matching comment in order to perform this command; this allows a policy maker to quickly enforce a commenting standard within a group of VOBs without creating new triggers. When used in a **post_op** key the users will see a "suggestion" dialog if they complete a matching command that "suggest" that they provide a matching comment. It is specified by an integer number as described below:
-

Comment Value	Comment Restriction Description
'0'	Indicates that there are NO comment restrictions for this command or command group.
Negative Integer	Requires a minimum number (-#) of words comment for this command or command group for all users ***except*** the ***Current VOB Owner***.
Positive Integer	Requires a minimum number (#) of words comment for this command or command group for all users ***including*** the ***Current VOB Owner***.

Notify_flag

- **notify_flag** – This flag is used to invoke the ClearTrigger region notification or logging program to be called *prior to* and *after* executing a command. These programs are pre-defined for the Windows or UNIX operating systems and exist in either the Windows policy depot or UNIX policy depot. The flag may be set to any of the following valid values:

Notify_Flag Value	Notify_Flag Description
'0'	Do NOT run the system NOTIFY script(s) as a pre-op/post-op for the command or command group.
'x'	Run the system NOTIFY script(s) as a pre-op/post-op for the command or command group for all users ***except*** the ***Current VOB Owner***.
'X'	Run the system NOTIFY script(s) as a pre-op/post-op for the command or command group for all users ***including*** the ***Current VOB Owner***.

Access_list_type

- **Access_List_type** – This list designates which type of Access_List list is defined in next key, **Access_List**. It's specified by a 'D' or a 'd' to designate **Disallow_List**, 'A' or an 'a' to designate an **Allow_List** or specified by and 'I' or an 'i' to designate an **Ignore_List**. The list may be designated as **"fully"** matching (where all different types in the list must match) or **"single"** matching (where only a single type must match). A description of each list follows:

Command List Type Value	Type Value Description
'D'	**Disallow** – Scenarios that "Fully" match the ***Access_List*** are prohibited from performing the command set. **Dynamic Variables**, **Patterns** and **~alias** names may be used in the list.
'A'	**Allow**- Only scenarios that "Fully" match the ***Access_List*** are allowed to perform the command set. **Dynamic Variables**, **patterns** and **~alias** names may be used in the list.
'I'	**Ignore**- Scenarios that "Fully" match the ***Access_List*** cause the reminder of the key to be ignored. **Dynamic Variables**, **patterns** and **~alias** names may be used in the list.
'd'	**Disallow** - Scenarios that match the ***Access_List*** are prohibited from performing the command set. **"Single Matching"** applies. **Dynamic Variables**, **patterns** and **~alias** names may be used in the list.
'a'	**Allow**- Only scenarios that match the ***Access_List*** are allowed to perform the command set. **"Single Matching"** applies. **Dynamic Variables**, **patterns** and **~alias** names may be used in the list.
'i'	**Ignore**- Scenarios that match the ***Access_List*** cause the reminder of the key to be ignored. **Dynamic Variables**, **patterns** and **~alias** names may be used in the list.

Access_list

- **access_list** – This list defines what users or user group may perform the matching command or command group and for which VOBs. The list may be empty or contain a *Disallow list*, *Allow list* or *Ignore_list*. The list may be designated as **"fully"** matching (where all different types in the list must match) or **"single"** matching (where only a single type must match). A description of each list follows:

  - **Disallow List**
    When **Access_List_Type** is equal to **'D' or 'd',** then the **Access_list** represents a ***disallow list***. When there are no entries in the list then all users are allowed to perform the matching command operation. When populated the list describes the conditions where the matching command would be prohibited. If the Access_List_Type is 'D', ***"full matching"*** applies and list items of the same type are ***OR***ed together and list items of different types are ***AND***ed together before the list is tested. If the Access_List_Type is 'd', ***"single matching"*** applies and all list items of any type are ***OR***ed together before the list is tested.

    The ***Disallow List*** contains zero or more user-ids, group names, VOB tags, element names, view names, times or ClearCase Regions enclosed by semicolons (;).Users ids are surrounded with parentheses (e.g. (user1)). Group names are surrounded with brackets (e.g. <groupname>). VOB tags are surrounded with square brackets (e.g. [/vobs/a_vob] or [\a_vob]); VOB:Replica can also be designated (e.g. [/vobs/a_vob:original] or [\a_vob:atl]). Elements are surrounded with curly braces (e.g. {\vobs\a_vob\src\foo.c}). }). Views are surrounded with percent characters (e.g. %a_view%). Times are surrounded with at characters (e.g. @time@). ClearCase Regions are surrounded with single quote characters (e.g. 'development'). The value(s) that the environmental variable

ABS_TRIGGER_POLICY are matched against are surrounded with the carat characters (e.g. ^development^).

Valid time formats are any of the three (3) defined below:

Hour definition: '@h##@' (where ## represent an hour from 0-23 “Military Hours”)
Day definition: '@d#@' (where # represent a day-of-week from 0-6 "Sun-Sat")
Date definition'@D########@' (where ######## represent a date from YYYYMMDD)

All users can be represented by a (*). All groups can be represented as <*>; all VOBs can be represented as [*] or [*:*]; all elements can be represented by {*}; all Views can be represented as %*%; all Times can be represented as @h*@, @d*@ or @D*@; all ClearCase Regions can be represented as ‘*’; all values that the ABS_TRIGGER_POLICY environmental variable will match against as ^*^.

For more examples of pattern characters that can be used, refer to the sections on **Dynamic Variables**, **Pattern Matching** and **aliases**.

Example Type	Example
Empty List	; ;
Single user entry	; (user1) ;
Single group entry	; <janitors> ;
Single VOB entry	; [/vobs/a_vob] ;
Single Element entry	; {/vobs/a_vob/some_file.c} ;
Single View entry	; %night_build_view% ;
Single Hour entry (11pm – midnight)	; @h23@ ;
Single day entry (Sunday)	; @d0@ ;
Single Date entry (Feb. 19, 2005)	; @D20050219@ ;
Multiple VOB entries (UNIX/Windows)	; [\a_vob] [/vobs/a_vob] [\b_vob] ;
Multiple VOB:Replica patterns	; [\a_vob:original] [*:atl] [*:site#] ;
Multiply Populated List	; (jit) (qa_#*) <dev1> ~interns [/vobs/a_vob] ;
all users	; (*) ;
all elements	; {*} ;
particular elements	; {/vobs/a/src/foo.c} {\vob_a\foo.?} ;
select elements	; {*.c} {/vobs/*/src/*} ~core_code ;
all Windows VOBs	; [*] ;
all UNIX VOBs	; [/*] ;
All times	; @d*@ ;
Particular ClearCase Regions	; ‘development’ ‘webdoc’ ‘US_*’ ;
All ClearCase Regions	; ‘*’ ;
Current VOB owner	; (&) ;
Current VOB’s Primary Group	; <&> ;
Top of VOB	;{&} ;
“src” directory at top of VOB	;{&/src} ;
During “Code Freeze”	; ^code_freeze^ ;
In any testing phase	; ^test_*^

Remember that when using group names any windows clients that access the VOB must have the CLEARCASE_PRIMARY_GROUP Environmental Variable set.

- **Allow List**
  When **Access_List_Type** is equal to **'A',** then the **Access_list** represents an ***allow list***. When there are no entries in the list then no users are allowed to perform the matching command operation. When populated the list describes the conditions where the matching command would be allowed. If the Access_List_Type is 'A', ***"full matching"*** applies and list items of the same type are ***OR***ed together and list items of different types are ***AND***ed together before the list is tested. If the Access_List_Type is 'a', ***"single matching"*** applies and all list items of any type are ***OR***ed together before the list is tested.

  The ***Allow List*** contains zero or more user-ids, group names, VOB tags, element names, view names or times enclosed by semicolons (;).Users ids are surrounded with parentheses (e.g. (user1)). Group names are surrounded with brackets (e.g. <groupname>). VOB tags are surrounded with square brackets (e.g. [/vobs/a_vob] or [\a_vob]); VOB:Replica can also be designated (e.g. [/vobs/a_vob:original] or [\a_vob:atl]). Elements are surrounded with curly braces (e.g. {\vobs\a_vob\src\foo.c}). }). Views are surrounded with percent characters (e.g. %a_view%). Times are surrounded with at characters (e.g. @time@). ClearCase Regions are surrounded with single quote characters (e.g. 'development'). The value(s) that the environmental variable ABS_TRIGGER_POLICY are matched against are surrounded with the carat characters (e.g. ^development^).

  Valid time formats are any of the three (3) defined below:

  Hour definition: '@h##@' (where ## represent an hour from 0-23 "Military Hours")
  Day definition: '@d#@' (where # represent a day-of-week from 0-6 "Sun-Sat")
  Date definition'@D########@' (where ######## represent a date from YYYYMMDD)

  All users can be represented by a (*). All groups can be represented as <*>; all VOBs can be represented as [*] or [*:*]; all elements can be represented by {*}; all Views can be represented as %*%; all Times can be represented as @h*@, @d*@ or @D*@; all ClearCase Regions can be represented as '*'; all values that the ABS_TRIGGER_POLICY environmental variable will match against as ^*^.

  For more examples of pattern characters that can be used, refer to the sections on **Dynamic Variables**, **Pattern Matching** and **aliases**.

Example Type	Example
Empty List	; ;
Single user entry	; (user1) ;
Single group entry	; <janitors> ;
Single VOB entry	; [/vobs/a_vob] ;
Single Element entry	; {/vobs/a_vob/some_file.c} ;
Single View entry	; %night_build_view% ;
Single Hour entry (11pm – midnight)	; @h23@ ;
Single day entry (Sunday)	; @d0@ ;
Single Date entry (Feb. 19, 2005)	; @D20050219@ ;
Multiple VOB entries (UNIX/Windows)	; [\a_vob] [/vobs/a_vob] [\b_vob] ;

Multiple VOB:Replica patterns	; [\a_vob:original] [*:atl] [*:site#] ;
Multiply Populated List	; (jit) (qa_#*) <dev1> ~interns [/vobs/a_vob] ;
all users	; (*) ;
all elements	; {*} ;
particular elements	; {/vobs/a/src/foo.c} {\vob_a\foo.?} ;
select elements	; {*.c} {/vobs/*/src/*} ~core_code ;
all Windows VOBs	; [*] ;
all UNIX VOBs	; [/*] ;
All times	; @d*@ ;
Particular ClearCase Regions	; 'development' 'webdoc' 'US_*' ;
All ClearCase Regions	; '*' ;
Current VOB owner	; (&) ;
Current VOB's Primary Group	; <&> ;
Top of VOB	;{&} ;
"src" directory at top of VOB	;{&/src} ;
During "Code Freeze"	; ^code_freeze^ ;
In any testing phase	; ^test_*^

Remember that when using group names any windows clients that access the VOB must have the CLEARCASE_PRIMARY_GROUP Environmental Variable set.

- **Ignore List**
  When **Access_List_Type** is equal to **'I' or 'i',** then the **Access_list** represents an ***ignore list***. When there are no entries in the list then no users are ignored and the rest of the key is evaluated if needed. When populated the list describes the conditions where the rest of the key (any portion to the right of the list) is ignored or skipped. If the Access_List_Type is 'I', ***"full matching"*** applies and list items of the same type are ***OR***ed together and list items of different types are ***AND***ed together before the list is tested. If the Access_List_Type is 'i', ***"single matching"*** applies and all list items of any type are ***OR***ed together before the list is tested.

  The ***Ignore List*** contains zero or more user-ids, group names, VOB tags, element names, view names or times enclosed by semicolons (;).Users ids are surrounded with parentheses (e.g. (user1)). Group names are surrounded with brackets (e.g. <groupname>). VOB tags are surrounded with square brackets (e.g. [/vobs/a_vob] or [\a_vob]); VOB:Replica can also be designated (e.g. [/vobs/a_vob:original] or [\a_vob:atl]). Elements are surrounded with curly braces (e.g. {\vobs\a_vob\src\foo.c}). }). Views are surrounded with percent characters (e.g. %a_view%). Times are surrounded with at characters (e.g. @time@). ClearCase Regions are surrounded with single quote characters (e.g. 'development'). The value(s) that the environmental variable ABS_TRIGGER_POLICY are matched against are surrounded with the carat characters (e.g. ^development^).

  Valid time formats are any of the three (3) defined below:

  Hour definition: '@h##@' (where ## represent an hour from 0-23 "Military Hours")
  Day definition: '@d#@' (where # represent a day-of-week from 0-6"Su -Sat")
  Date definition'@D########@' (where ######## represent a date from YYYYMMDD)

All users can be represented by a (*). All groups can be represented as <*>; all VOBs can be represented as [*] or [*:*]; all elements can be represented by {*}; all Views can be represented as %*%; all Times can be represented as @h*@, @d*@ or @D*@; all ClearCase Regions can be represented as '*'; all values that the ABS_TRIGGER_POLICY environmental variable will match against as ^*^.

For more examples of pattern characters that can be used, refer to the sections on **Dynamic Variables**, **Pattern Matching** and **aliases**.

Example Type	Example
Empty List	; ;
Single user entry	; (user1) ;
Single group entry	; <janitors> ;
Single VOB entry	; [/vobs/a_vob] ;
Single Element entry	; {/vobs/a_vob/some_file.c} ;
Single View entry	; %night_build_view% ;
Single Hour entry (11pm – midnight)	; @h23@ ;
Single day entry (Sunday)	; @d0@ ;
Single Date entry (Feb. 19, 2005)	; @D20050219@ ;
Multiple VOB entries (UNIX/Windows)	; [\a_vob] [/vobs/a_vob] [\b_vob] ;
Multiple VOB:Replica patterns	; [\a_vob:original] [*:atl] [*:site#] ;
Multiply Populated List	; (jit) (qa_#*) <dev1> ~interns [/vobs/a_vob] ;
all users	; (*) ;
all elements	; {*} ;
particular elements	; {/vobs/a/src/foo.c} {\vob_a\foo.?} ;
select elements	; {*.c} {/vobs/*/src/*} ~core_code ;
all Windows VOBs	; [*] ;
all UNIX VOBs	; [/*] ;
All times	; @d*@ ;
Particular ClearCase Regions	; 'development' 'webdoc' 'US_*' ;
All ClearCase Regions	; '*' ;
Current VOB owner	; (&) ;
Current VOB's Primary Group	; <&> ;
Top of VOB	;{&} ;
"src" directory at top of VOB	;{&/src} ;
During "Code Freeze"	; ^code_freeze^ ;
In any testing phase	; ^test_*^

Remember that when using group names any windows clients that access the VOB must have the CLEARCASE_PRIMARY_GROUP Environmental Variable set.

Trigger_list_type

Trigger_List_type – This list designates which type of Trigger_List list is defined in next key, **Trigger_List**. It's specified by a 'N' or a 'n' to designate **Not_List** or specified by and 'O' or an 'o' to designate an **Only_List**. The list may be designated as **"fully"** matching (where all different types in the list must match) or **"single"** matching (where only a single type must match).

Trigger List Type Value	Type Value Description
'N'	**Not_List** – Scenarios that "Fully" match the list **do not** cause any defined custom triggers to fire or cause the display of user defined GUI dialogs. **Dynamic Variables**, **Patterns** and **~alias** names may be used in the list.
'O'	**Only_List**- Scenarios that "Fully" match the list are the **only** ones that cause any defined custom triggers to fire or cause the display of user defined GUI dialogs. **Dynamic Variables**, **Patterns** and **~alias** names may be used in the list.
'n'	**Not_List** – Scenarios that match the list **do not** cause any defined custom triggers to fire or cause the display of user defined GUI dialogs. **"Single Matching"** applies. **Dynamic Variables**, **Patterns** and **~alias** names may be used in the list.
'o'	**Only_List**- Scenarios that match the list are the **only** ones that cause any defined custom triggers to fire or cause the display of user defined GUI dialogs. **"Single Matching"** applies. **Dynamic Variables**, **Patterns** and **~alias** names may be used in the list.

trigger_list

- **Trigger_list** – This list defines what users or user groups may cause **can** cause any defined custom triggers to fire or cause the display of user defined GUI dialogs. The list may be empty or contain a *Not_List* or Only_List. The list may be designated as **"fully"** matching (where all different types in the list must match) or **"single"** matching (where only a single type must match). A description of each list follows:

  - **Not_List**
    When **Trigger_List_Type** is equal to **'N' or 'n',** the **Trigger_list** represents a ***Not_List***. When there are no entries in the list then all users that are allowed to perform the command action will also cause any defined custom triggers scripts or cause the display of user defined GUI dialogs. When populated the list describes the conditions where the matching command will NOT cause any defined custom triggers scripts or cause the display of user defined GUI dialogs. If the Trigger_List_Type is 'N', ***"full matching"*** applies and list items of the same type are ***OR***ed together and list items of different types

are ***AND***ed together before the list is tested. If the Access_List_Type is 'n', ***"single matching"*** applies and all list items of any type are ***OR***ed together before the list is tested.

The *Not List* contains zero or more user-ids, group names, VOB tags, element names, view names or times enclosed by semicolons (;).Users ids are surrounded with parentheses (e.g. (user1)). Group names are surrounded with brackets (e.g. <groupname>). VOB tags are surrounded with square brackets (e.g. [/vobs/a_vob] or [\a_vob]); VOB:Replica can also be designated (e.g. [/vobs/a_vob:original] or [\a_vob:atl]). Elements are surrounded with curly braces (e.g. {\vobs\a_vob\src\foo.c}). }). Views are surrounded with percent characters (e.g. %a_view%). Times are surrounded with at characters (e.g. @time@). ClearCase Regions are surrounded with single quote characters (e.g. 'development'). The value(s) that the environmental variable ABS_TRIGGER_POLICY are matched against are surrounded with the carat characters (e.g. ^development^).

Valid time formats are any of the three (3) defined below:

Hour definition: '@h##@' (where ## represent an hour from 0-23 "Military Hours")
Day definition: '@d#@' (where # represent a day-of-week from 0-6 "Sun-Sat")
Date definition'@D########@' (where ######## represent a date from YYYYMMDD)

All users can be represented by a (*). All groups can be represented as <*>; all VOBs can be represented as [*]; all elements can be represented by {*}; all Views can be represented as %*%; all Times can be represented as @h*@, @d*@ or @D*@; all ClearCase Regions can be represented as '*'; all values that the ABS_TRIGGER_POLICY environmental variable will match against as ^*^.

For more examples of pattern characters that can be used, refer to the sections on **Dynamic Variables**, **Pattern Matching** and **aliases**.

Example Type	Example
Empty List	; ;
Single user entry	; (user1) ;
Single group entry	; <janitors> ;
Single VOB entry	; [/vobs/a_vob] ;
Single Element entry	; {/vobs/a_vob/some_file.c} ;
Single View entry	; %night_build_view% ;
Single Hour entry (11pm – midnight)	; @h23@ ;
Single day entry (Sunday)	; @d0@ ;
Single Date entry (Feb. 19, 2005)	; @D20050219@ ;
Multiple VOB entries (UNIX/Windows)	; [\a_vob] [/vobs/a_vob] [\b_vob] ;
Multiply Populated List	; (jit) (qa_#*) <dev1> ~interns [/vobs/a_vob] ;
all users	; (*) ;
all elements	; {*} ;
particular elements	; {/vobs/a/src/foo.c} {\vob_a\foo.?} ;
select elements	; {*.c} {/vobs/*/src/*} ~core_code ;
all Windows VOBs	; [*] ;
all UNIX VOBs	; [/*] ;

All times	; @d*@ ;
Particular ClearCase Regions	; 'development' 'webdoc' 'US_*' ;
All ClearCase Regions	; '*' ;
Current VOB owner	; (&) ;
Current VOB's Primary Group	; <&> ;
Top of VOB	;{&} ;
"src" directory at top of VOB	;{&/src} ;
During "Code Freeze"	; ^code_freeze^ ;
In any testing phase	; ^test_*^

Remember that when using group names any windows clients that access the VOB must have the CLEARCASE_PRIMARY_GROUP Environmental Variable set.

- **Only_List**
  When **Trigger_List_Type** is equal to **'O' or 'o',** then the **Trigger_list** represents an ***Only_List***. When there are no entries in the list, no users that are allowed to perform the command action will also cause any defined custom triggers scripts or cause the display of user defined GUI dialogs. When populated the list describes the conditions where the matching command will cause any defined custom triggers scripts or cause the display of user defined GUI dialogs. If the Trigger_List_Type is 'O', ***"full matching"*** applies and list items of the same type are ***OR***ed together and list items of different types are ***AND***ed together before the list is tested. If the Access_List_Type is 'o', ***"single matching"*** applies and all list items of any type are ***OR***ed together before the list is tested.

  The ***Only List*** contains zero or more user-ids, group names, VOB tags, element names, view names or times enclosed by semicolons (;).Users ids are surrounded with parentheses (e.g. (user1)). Group names are surrounded with brackets (e.g. <groupname>). VOB tags are surrounded with square brackets (e.g. [/vobs/a_vob] or [\a_vob]); VOB:Replica can also be designated (e.g. [/vobs/a_vob:original] or [\a_vob:atl]). Elements are surrounded with curly braces (e.g. {\vobs\a_vob\src\foo.c}). }). Views are surrounded with percent characters (e.g. %a_view%). Times are surrounded with at characters (e.g. @time@). ClearCase Regions are surrounded with single quote characters (e.g. 'development'). The value(s) that the environmental variable ABS_TRIGGER_POLICY are matched against are surrounded with the carat characters (e.g. ^development^).

  Valid time formats are any of the three (3) defined below:

  Hour definition: '@h##@' (where ## represent an hour from 0-23 "Military Hours")
  Day definition: '@d#@' (where # represent a day-of-week from 0-6"Su -Sat")
  Date definition'@D########@' (where ######## represent a date from YYYYMMDD)

  All users can be represented by a (*). All groups can be represented as <*>; all VOBs can be represented as [*] or [*:*]; all elements can be represented by {*}; all Views can be represented as %*%; all Times can be represented as @h*@, @d*@ or @D*@; all

ClearCase Regions can be represented as '*'; all values that the ABS_TRIGGER_POLICY environmental variable will match against as ^*^.

For more examples of pattern characters that can be used, refer to the sections on **Dynamic Variables**, **Pattern Matching** and **aliases**.

Example Type	Example
Empty List	; ;
Single user entry	; (user1) ;
Single group entry	; <janitors> ;
Single VOB entry	; [/vobs/a_vob] ;
Single Element entry	; {/vobs/a_vob/some_file.c} ;
Single View entry	; %night_build_view% ;
Single Hour entry (11pm – midnight)	; @h23@ ;
Single day entry (Sunday)	; @d0@ ;
Single Date entry (Feb. 19, 2005)	; @D20050219@ ;
Multiple VOB entries (UNIX/Windows)	; [\a_vob] [/vobs/a_vob] [\b_vob] ;
Multiply Populated List	; (jit) (qa_#*) <dev1> ~interns [/vobs/a_vob] ;
all users	; (*) ;
all elements	; {*} ;
particular elements	; {/vobs/a/src/foo.c} {\vob_a\foo.?} ;
select elements	; {*.c} {/vobs/*/src/*} ~core_code ;
all Windows VOBs	; [*] ;
all UNIX VOBs	; [/*] ;
All times	; @d*@ ;
Particular ClearCase Regions	; 'development' 'webdoc' 'US_*' ;
All ClearCase Regions	; '*' ;
Current VOB owner	; (&) ;
Current VOB's Primary Group	; <&> ;
Top of VOB	;{&} ;
"src" directory at top of VOB	;{&/src} ;
During "Code Freeze"	; ^code_freeze^ ;
In any testing phase	; ^test_*^

Remember that when using group names any windows clients that access the VOB must have the CLEARCASE_PRIMARY_GROUP Environmental Variable set.

Series_begin_end_only_bit

- **Series_begin_end_only_bit** – This is the control bit that determines if any defined trigger script or user defined GUI should execute for all elements in a series or just the first or last element in a series. A series is defined by the ClearCase environmental variables CLEARCASE_START_SERIES (set to "1" at the beginning of a series) and CLEARCASE_END_SERIES (set to "1" at the end of a series). The variables are set for the ClearCase checkin, checkout and uncheckout commands as of ClearCase 4.2. When this bit is set then any trigger scripts or user defined GUIs will execute only once as a pre-op for the first element in the series or only once as a post-op for the last element in the series.

  Examples:

  - If there was a **pre_checkout** command key defined that caused a trigger (BUG_NUM.pl) to fire and the bit was unset then the BUG_NUM.pl script would fire as a pre-op for all matching files from the command "**cleartool co –nc *.c**" while it would only fire once as a pre-op for the **first** matching c file if the bit was set.

  - If there was a **post_checkin** command key defined that caused a trigger (CLOSE_BUG.pl) to fire and the bit was unset then the CLOSE_BUG.pl script would fire as a post-op for all matching files from the command "**cleartool ci –nc *.c**" while it would only fire once as a post-op for the **last** matching c file if the bit was set.

Bit Value	Series_begin_end_only_flag Description
'0'	Custom Triggers or user defined GUI only active for all elements in a series for Pre-op triggers and Post-op triggers.
'1'	Custom Triggers or user defined GUI only active for the first in the series for Pre-op triggers and only active for the last in the series for Post-op triggers.

Windows_script

- **Windows_script** – This field defines a custom pre-op/post-op script or executable that will fire in addition to the enterprise pre-op code called by ClearTrigger when the calling activity is invoked from a Windows operating system. The enterprise trigger will always fire after ClearTrigger insuring that enterprise definitions take precedence. The script can be of any executable type. (e.g. *.bat, *.exe, *.pl, etc). The script is expected by default to reside in the Windows Depot, but can also be described with an absolute path or a path relative to the Windows_Depot.

  NOTE: If there is a Windows script defined then the user cannot have the **PERLDB_OPTS** or **PERL5DB** environmental variables set when the trigger fires or ClearTrigger will fail the command to prevent users from entering the Perl debugger and circumventing custom Perl policy code written by the administrator. The ClearTrigger Administrator can set the special **ClearTrigger Override Alias** **ABS_perl_security_override** to change this behavior.

  NOTE: If there is a Windows script defined as exactly "**ccperl.exe**" as highlighted in the table below then the ccperl.exe executable is expected to be in the depot as previously explained, but to save network bandwidth the ClearTrigger Administrator may elect to allow the use of local well-known ccperl.exe executables by setting the special **ClearTrigger Override Alias** **ABS_allow_use_of_local_ccperl** to change this behavior. When set the client local instance of **ccperl.exe or cleartool.exe** that resides in the client local directory **(%RATIONAL_HOME%\ClearCase\bin**) or CQerl.exe that resides in the client local directory **(%RATIONAL_HOME%\ClearQuest**) if it exists. This improves end-user trigger performance, but should only be used in an organization values this increase in performance over the slight security concern.

List Description	Windows Script Example
Empty List	; ;
Calling interpreted program (uses Windows_script_params field)	;ccperl.exe; \\machine\share\depot\some_script.pl;
Calling stand alone program in depot	; some_script.bat ;
Calling program not in depot by absolute path	;\\machine\share\some_program;
Calling program not in depot by relative path to Windows depot	;..\..\directory\script;

Windows_script_params

- **Windows_script_params –** This field contains the Windows script parameters (if any) that would be passed to the Windows script or executable (if any) when fired. This can include any ClearCase and ClearTrigger defined environmental variables as well as hard-coded variables. Any environmental defined in this manner are evaluated before they are sent to the called script or executable. Multiple parameters should be space separated.

Windows param field	Example
Empty List	; ;
single hard-coded parameter	;development;
single environmental parameter	;%CLEARCASE_USER%;
multiple parameters	;sys_test development;
multiple parameters	;;inform_unit %CLEARCASE_USER%;
Passing program to interpreter (uses Windows_script field)	;Perl.exe; \\machine\share\depot\some_script.pl;

UNIX_script

- **UNIX_script** – This field defines a custom pre-op/post-op script or executable that will fire in addition to the enterprise code called by ClearTrigger when the calling activity is invoked from a UNIX operating system. The enterprise trigger will always fire after ClearTrigger insuring that enterprise definitions take precedence. The script can be of any executable type. (e.g. *.ksh, *.sh *.pl, etc). The script is expected by default to reside in the UNIX Depot, can also be described with an absolute path or a path relative to the UNIX_Depot.

  NOTE: If there is a UNIX script defined then the user cannot have the **PERLDB_OPTS** or **PERL5DB** environmental variables set when the trigger fires or ClearTrigger will fail the command to prevent users from entering the Perl debugger and circumventing custom Perl policy code written by the administrator. The ClearTrigger Administrator can set the special **ClearTrigger Override Alias** **ABS_perl_security_override** to change this behavior.

List Description	Inhibited Users List Example
Empty List	; ;
Calling interpreted program (uses Unix_script_params field)	;Perl; /net/machine/location/some_code.pl ;
Calling stand alone program in depot	; some_script.sh ;
Calling program not in depot by absolute path	;/net/machine/location/some_script;
Calling program not in depot by relative path to UNIX depot	;../../directory/script;

UNIX_script_params

- **UNIX_script_params –** This field contains the UNIX script parameters (if any) that would be passed to the UNIX script or executable (if any) when fired. This can include any ClearCase and ClearTrigger defined environmental variables as well as hard-coded variables. Any environmental defined in this manner are evaluated before they are sent to the called script or executable. Multiple parameters should be space separated.

UNIX param field	Example
Empty List	; ;
single hard-coded parameter	;development;
single environmental parameter	;$CLEARCASE_USER;
multiple parameters	;design_phase $CLEARCASE_USER;
multiple parameters	;sys_test development;
Passing program to interpreter (uses Unix _script field)	;Perl; /net/machine/some_script.pl;

Inhibit_cleartrigger_GUI_bit

- **Inhibit_cleartrigger_GUI_bit** – This is the control bit that defines whether or not ClearTrigger denied by custom triggers dialogs are displayed for explicit use within a trigger key. For example, when one creates code for a pre-op trigger and that code is executed and produces its own GUI inhibit dialog windows. Your organization might prefer not to also see the ClearTrigger dialog window that says the command failed because of the call to the trigger code. If this is the case, then set the bit value to "1" to disable the additional ClearTrigger GUI dialog. To use the ClearTrigger GUI dialog, set the bit value to "0". If you set the bit value to "1" and do not specify a script to be executed, the ClearTrigger dialogs will be used by default.

Bit Value	Inhibit_cleartrigger_GUI_bit Description
'0'	Enable ClearTrigger dialogs.
'1'	Disable additional ClearTrigger dialogs – rely only on custom written dialogs

User defined GUI capability

In ClearTrigger one may create their own dialog windows that ask questions or ask for input without having to write code that uses clearprompt, worry about temporary file creation or file cleanup. Three fields are added near the end of each command key that allows you to define a prompt dialog, question dialog, selection dialog or alert dialog; the text display as well at the allowed and default buttons can also be defined.

;{dialog_type};{dialog_prompt};{dialog_mask or default_string or dialog_choices};

Dialog_type

Dialog_type – Contains a dialog_type value where:

Value	Defines this feature as...
'E'	No dialog is requested
'Q'	A **Question dialog** is requested
'P'	A **Prompt Dialog** is requested
'S'	A **Selection Dialog** is requested
'M'	A **Multi-Selection Dialog** is requested
'A'	An **Advise Dialog** "modeless message" is requested

The value of the first field determines what fields afterwards are required and how they are interpreted. Examples of each Dialog_type are below:

;E;
;Q;{dialog_prompt};{dialog_mask};
;P;{dialog_prompt};{default_string};
;S;{dialog_prompt};{dialog_choices};
;M;{dialog_prompt};{dialog_choices};
;A;{dialog_prompt};

Dialog_prompt

Dialog_prompt – Contains a dialog_prompt value where (prompt, question or message) is any string without semi-colons ';' in it. This is the prompt, question or request that is displayed. For all dialog _types except the Advise Dialog the dialog can wait for user input.

Dialog_mask or dialog_string or dialog_choices

This is a multi-purpose field whose function depends on which dialog_type is selected. The field is either a **dialog_mask**, **default_string** or **dialog_choices**.

Dialog_mask – If the dialog_type is a **Question Dialog** type then this contains a mask string containing zero or more button characters in any order from the characters ('C','N','Y'):

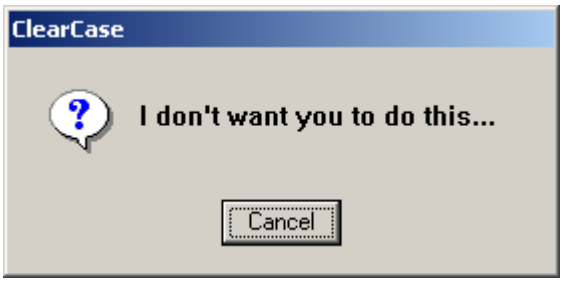
Value	Displays this button
'C'	Cancel
'N'	No
'Y'	Yes

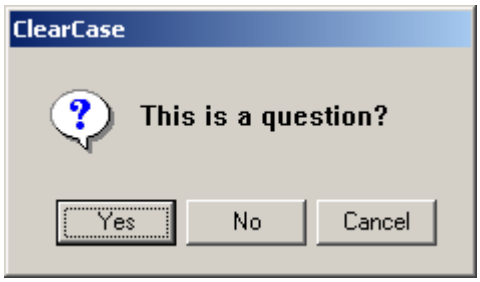
Only buttons in the mask are displayed and always in this order Yes, No, Cancel when they exist. The **first** button requested is the default button and is pre-selected in the displayed dialog.

The command key with GUI definition fields like:

";Q;I don't want you to do this…;C;" or ";Q;This is a question?;YNC;"

Produces dialogs like the ones below respectively:





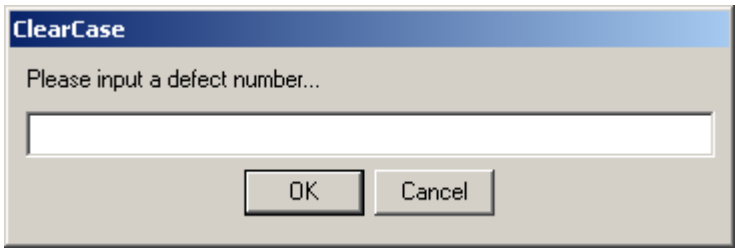
Where the environmental variable **CLEARTRIGGER_DIALOG_REPLY** would be set to 0 for Yes or 1 for No and is available for use in triggers defined within the command key. When Cancel is selected it always stops the processing.

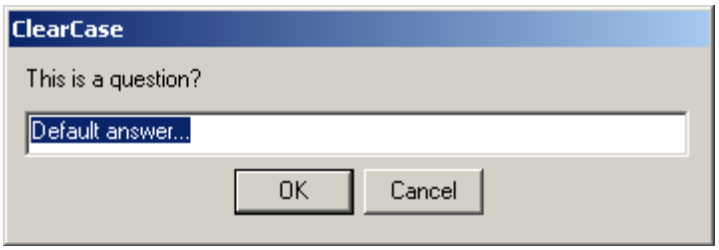
Default_string – If the dialog_type is a **Prompt Dialog** type then this contains a string (possibly empty) that constitutes the displayed default string in a text dialog box.

The command key with GUI definition fields like:

";P;Please input a defect number...;;" or ";P;This is a question?;Default answer...;"

Produces dialogs like the ones below respectively:





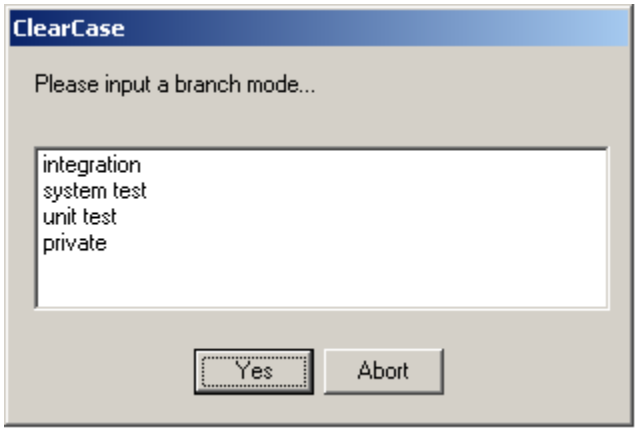
The OK and Cancel buttons always apply. The environmental variable **CLEARTRIGGER_DIALOG_REPLY** would be set to 0 for OK and is available for use in triggers defined within the command key. When Cancel is selected it always stops the processing.

The environmental variable **CLEARTRIGGER_REPLY_STRING** would be set to the input string and is available for use in triggers defined within the command key. Where the environmental variable **CLEARTRIGGER_DIALOG_REPLY** would be set to 0 for Yes or 1 for No and is available for use in triggers defined within the command key.

Dialog_choices – If the dialog_type is a **Selection Dialog** type then this contains a comma ',' separated string that constitutes the displayed choices that may be selected in the dialog box. The command key with GUI definition fields like:

";S;Please input a branch mode...;integration,system test, unit test, private;"

Produces dialogs like the one below:



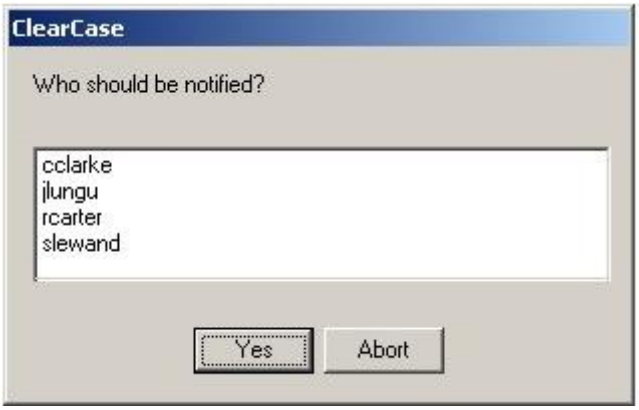
The Yes and Abort buttons always apply. Only single selection is supported. The environmental variable **CLEARTRIGGER_DIALOG_REPLY** would be set to 0 for Yes and is available for use in triggers defined within the command key. When Abort is selected it always stops the processing.

The environmental variable **CLEARTRIGGER_RESULT_STRING** would be set to the selected choice and is available for use in triggers defined within the command key.

Dialog_choices – If the dialog_type is a **Multi-Selection Dialog** type then this contains a comma ‘,’ separated string that constitutes the displayed choices that may be selected in the dialog box.
The command key with GUI definition fields like:

";M;Who should be notified?;cclarke,jlungu,rcarter,slewand;"

Produces dialogs like the one below:



The Yes and Abort buttons always apply. Multi-selection is supported. The environmental variable **CLEARTRIGGER_DIALOG_REPLY** would be set to 0 for Yes and is available for use in triggers defined within the command key. When Abort is selected it always stops the processing.

The environmental variable **CLEARTRIGGER_RESULT_STRING** would be set to a comma-separated list of the selected choices and is available for use in triggers defined within the command key.

Modeless Dialogs

The **Advise Dialog** type is a "modeless" dialog. It does not wait for user input, it displays a message that the policy maker deems appropriate and continues processing, it has no bearing on determining if a trigger completes or not.

For example, the command key with a GUI definition field like the one below:

";A;Don't forget to inform QA...;"

Produces a dialog like the one below:

The Yes or OK button is displayed, but the dialog is modeless (does not wait for an answer to continue), does not affect whether or not the command continues or not and is strictly informational in nature.

The environmental variables **CLEARTRIGGER_DIALOG_REPLY** and **CLEARTRIGGER_RESULT_STRING** are undetermined in this case.

Partial Key processing

To increase both speed of execution and speed of user key development partial command key definitions are implemented.

So, to inhibit checkouts for all users except special users one might use the key below that has all key fields defined:

pre_checkin;R;;x;0;0;D;;N;;0;;;;;0;E;;;

This key will of course work, but the key below will suffice:

pre_checkin;R;;x;

Since nothing past the **region_inhibit** field provided any value to the key nothing past that field is required.

You can change keys in the clearbits file manually or use the **ClearQuery** application.

Command Permission Definitions (ClearTrigger Lite Feature Set)

In a ClearTrigger Lite region, the final section of **clearbits_files** allows you to specify your region-wide or enterprise-wide user/command permissions matrix. ***Command Permission Definitions*** (CPD) define the matrix. *Command Permission Definitions* are evaluated in order of appearance and all matching definitions are evaluated until a matching definition results in a restriction or all definitions are evaluated.

Correct formatting of the CPD is essential to the successful implementation of ClearTrigger Lite. There are three parts to any valid ***Command Permission Definition***:

- Definition_Type
- Command_List
- User_List

The parts can come in any order and be split within the key. A CPD must exist on a single line. White space within the CPD is ignored. CPD formatting is significantly *forgiving*.

Definition_Type

The **definition_type** identifies which type of CPD is defined. The **definition_type** can exist anywhere within the CPD.

Definition_Type Value	Type Value Description
"A:"	Indicates that the CPD is a, "Allow" command definition
"D:"	Indicates that the CPD is a "Disallow" command definition

Note: The web interface pulldown choices for Definition_Type is limited to valid values only.

Command_List

The **command_list** - identifies which command (or commands) the CPD applies to. The list is any number of commands strings where each command is preceded with a '.' (period). The **command_list** can exist anywhere within the CPD and be split within the CPD. White space within the CPD is ignored.

List Description	Command List Example
Singular list	**.CHECKOUT**
Multiple list	**.checkout.checkin.chevent**
All commands	**.***

Split list	**.checkout .checkin** A: **.chevent**

User_List

The **user_list** identifies the user (or users) for which the CPD applies. The list is any number of **user_id** strings where each user is surrounded by spaces ' '. The **user_list** can exist anywhere within the CPD and can even be split within the CPD.

List Description	Users List Example
Singular list	**ROBERT**
Multiple list	**robert scott charles**
All users	*****
Split list	**robert john** A: .checkout **sara james**

Example CPDs

CPD can be used to define either who can perform a command within the region or who cannot. CPDs with an "A:" (for allow) within the definition define who can perform the command. CPDs with a "D:" (for disallow) within the definition define who cannot perform the command. Both definitions may be used. Keys are evaluated in order of appearance until the user is disallowed or no more matching keys exist in the **clearbits_file**. If a CPD has both an "A:" and "D:," the "D:" takes precedence.

(Disallow Users)

Any CPD in the **clearbits_file** that results in a user being prevented from performing a command will result in a dialog similar to the one below:

Policy: Prevent specific users from executing a command on all VOBs in a ClearTrigger Lite region.

It is possible to prevent users "Raman and Fred" from performing checkins in any VOB without having to visit each VOB. Simply add a Disallow type CPD to the clearbits_file. Three equivalent keys are shown below; any one of them would suffice.

D:.checkin raman fred

.checkin raman fred D:
D: .checkin raman fred

Policy: Prevent specific users from executing several commands on all VOBs in a ClearTrigger Lite region.

It is possible to prevent "Josh" from performing checkins, chevent or rmelem commands in any VOB without having to visit each VOB. Simply add a Disallow type CPD to the **clearbits_file**. Any of the three equivalent keys are shown below.

D:.checkin.chevent.rmelem josh
josh .chevent.checkin.rmelem D:
.chevent .checkin D: josh .rmelem

Policy: Prevent specific users from executing any command on all VOBs in a ClearTrigger Lite region.

It is possible to prevent both Sam and Paul from performing any data modifying command in any VOB without having to visit each VOB. Simply add a Disallow type CPD to the **clearbits_file**. Three equivalent keys are shown below; any one of them would suffice.

D:.* sam paul
sam .* D: paul
***. D: paul sam**

Policy: Prevent all users from executing a command on all VOBs in a ClearTrigger Lite region.

It is possible to prevent anyone from performing the rmver command in any VOB without having to visit each VOB. Simply add a Disallow type CPD to the clearbits_file. Three equivalent keys are shown below; any one of them would suffice.

D:.rmver *
*** .rmver D:**
.rmver D: *

(Allowing Users)

Any CPD in the **clearbits_file** that results in a user being prevented from performing a command will result in a dialog similar to the one below:

Policy: Allow only specific users to execute a command on all VOBs in a ClearTrigger Lite region.

It is possible to only allow the users "charles and scott" to perform mkelem in any VOB without having to visit each VOB. Simply add an Allow type CPD to the clearbits_file. Three equivalent keys are shown below; any one of them would suffice.

A:.mkelem charles scott
.mkelem charles scott A:
A: .mkelem scott charles

Policy: Allow only specific user to execute certain commands on all VOBs in a ClearTrigger Lite region.

It is possible to only allow the user "robert" to perform the mv or protect command in any VOB without having to visit each VOB. Simply add an Allow type CPD to the clearbits_file. Three equivalent keys are shown below; any one of them would suffice.

A:.mv.protect robert
.protect .mv robert A:
.mv.protect A: robert

Orderless processing within the CPD

To facilitate CPD definition building, processing within the definition is order less. The operator can quickly build definitions if it is intuitive and easily "translated" to the spoken word. Users think differently such that one policy maker would say "I want to allow only Robert to perform rmelem" (**A: robert .rmelem**). Another might think "The rmelem command is only allowed to be performed by Robert" **(.rmelem A: robert**). Still another might think "for the rmelem command Robert is the only allowed user" (**.rmelem robert A:).**

All CPD are equivalent in each of these cases, thereby making it easier for policy makers to create them.

Ordered processing within the clearbits_file

Within the **clearbits_file,** the CPDs are evaluated from top to bottom until a "denial" is processed or all matching CPDs are evaluated. This allows you to make the **clearbits_file** readable by streamlining the CPD or developing a hierarchy.

A:.mv.protect robert
A:.mv charles scott
D:mv.rmelem *
A:.rmelem david

In the above example, the "mv" command can only be performed by Robert, Charles and Scott. No user can perform the rmelem command, even David included, as he would be stopped by the 3rd CPD.

Migration from 12.x versions to 13.1

The transition from 12.x versions to 13.1 for ClearTrigger is rather painless. The upgrade itself should take no longer than a few minutes once you have started. You should pick some time when users are not accessing ClearCase and then:

- **No changes are required to upgrade a 12.x clearbits file to a 13.1 clearbits file. All 13.1 implemented clearbits file changes are additive to ClearTrigger.**

- Add the required **cleartrigger_server** binary in the same directory where the existing **cleartrigger** binary exists.

- *Optionally*, add a **cleartrigger_server.conf** file. By default (if cleartrigger_server.conf does not exists), the **cleartrigger_server** binary will create a directory named **_server_bucket** in the same directory where the **cleartrigger** and **cleartrigger_server** binary exist. The server processes write their logs there. This directory should be available to all cleartrigger clients (clearcase clients that access the associated ClearTrigger Regions). If you wish to have this logging directory designated as a different network location, then create this file and place that location as the *only line* in the file:

  \\some_machine\some_share_server_bucket
  or
  /net/some_machine/some_share/_server_bucket

  Actually, the first line in the file is used as the new location and any other line is ignored and treated as a comment.

  Use of the **cleartrigger_server.conf** file is helpful when you don't want a user writable directory so close to the binaries as many companies protect the directories where the binaries are as read-only.

- **Replace** the ClearTrigger 12.x binaries with the **13.1** binaries (**cleartrigger**, **clearquery** and **clearapply**).

- Optionally, add a **cleartrigger_server.conf** file

- Inform your users to start working again.

Migration from 13.0 versions to 13.1

The transition from 13.0 versions to 13.1 for ClearTrigger is rather painless. The upgrade itself should take no longer than a few seconds once you have started. You should pick some time when users are not accessing ClearCase and then:

- ***Stop any currently running cleartrigger_server process.***

  cleartrigger_server -stop_all

  This will stop the cleartrigger_server process on the local machine and any other remote machines that are using the cleartrigger_server executable.

- **Replace** the ClearTrigger **13.0** binaries with the **13.1** binaries (**cleartrigger**, **cleartrigger_server**, **clearquery** and **clearapply**).

- Inform your users to start working again.

ClearTrigger defined Environment Variables

When executing any user defined **Windows_script** and **UNIX_script** commands in ClearTrigger, the following environment variables are defined in addition to the ClearCase variables. Many oft-requested values (e.g. "VOB_owner") that would normally require scripting to calculate are available for you in your scripts.

***Environmental variables added in version 13.0 are so indicated by highlight.**

Environment Variable	Values
CLEARTRIGGER_BITS	The state of the calculated functionality bits as read from the clearbits_file.
CLEARTRIGGER_CLEARCASE_REGION_NAME	Set to the current ClearCase Region Name
CLEARTRIGGER_CURRENT_VOB_OWNER	Set to the current VOB owner user ID.
CLEARTRIGGER_DEPOT_PATH	The current path to the ClearTrigger Depot.
CLEARTRIGGER_DEPOT_TIME	The current Depot Time: YYYYMMDD:HHMMSS
CLEARTRIGGER_DEPOT_WDAY	Set to ["Monday"..."Sunday"].
CLEARTRIGGER_DEPOT_WDAY_NUM	Set to [1..7] for (Monday- Sunday)
CLEARTRIGGER_DIALOG_REPLY	Return of the ClearTrigger Question/prompt/selection. (dialog 0=yes, 1=no, 2=cancel/abort)
CLEARTRIGGER_KEY_NUMBER	The 1-based positive integer representation of clearbits_file command key currently associated with this call (e.g. "1" would be the first key in the file).
CLEARTRIGGER_KEY_USED	The actual command key used as read from the clearbits_file.
CLEARTRIGGER_NUMBER	The ClearTrigger trigger type is currently associated with the call. [1 ... 4] for non-process VOBS (UCM) and [1...6] for process VOBs.
CLEARTRIGGER_MOTD_RESTRICTIONS	If any MOTD restrictions are defined in the clearbits file this is defined and set to that value
CLEARTRIGGER_MOTD_SHOWING	If a **MOTD** string is defined in the clearbits file this is defined and set to either "true" or "false" to indicate if any MOTD restrictions are matched.
CLEARTRIGGER_MOTD_STRING	If a **MOTD** string is defined in the clearbits file this is defined and set to that value
CLEARTRIGGER_OPTYPE	The "pre-op" and "post-op" value of the associated command.
CLEARTRIGGER_REGION_NAME	Current ClearTrigger Region Name
CLEARTRIGGER_RESULT_STRING	User string input when ClearTrigger Prompt dialog is used or the selected string if the ClearTrigger Selection dialog is used.
CLEARTRIGGER_USER_IS_VOB_OWNER	Set to "true" if the current user is the owner of the associated VOB; set to "false" if not.
CLEARTRIGGER_USER_IS_WEB_USER	Set to "true" if user is currently accessing from the web interface, set to "false" otherwise.
CLEARTRIGGER_VERSION	Set to ClearTrigger Version Number
CLEARTRIGGER_VOB_ACCESS	Set to the ClearCase VOBs access value (i.e. "public" or "private')
CLEARTRIGGER_VOB_FEATURE_LEVEL	Set to the current VOBs ClearCase feature Level

CLEARTRIGGER_VOB_HOST	Set to the ClearCase VOBs local host machine name.
CLEARTRIGGER_VOB_IS_REPLICATED	Set to "true" if the current VOB is replicated, set to "false" otherwise.
CLEARTRIGGER_VOB_IS_PROJECT_VOB	Set to "true" if the current VOB is a UCM project VOB, set to "false" otherwise.
CLEARTRIGGER_VOB_VOB_FAMILY_UUID	Set to the current VOBs family UUID value
CLEARTRIGGER_VOB_PRIMARY_GROUP	Set to the current VOBs primary group value
CLEARTRIGGER_VOB_REPLICA_NAME	The current VOBs replica name.
CLEARTRIGGER_ USER_IS_REMOTE_CLIENT_USER	Set to "true" if the current user is using the ClearCase Remote Client, set to "false" otherwise*. * The ccweb executable must be renamed to ccweb_remote to distinguish ClearCase Web Users from ClearCase Remote Client Users.

User Defined Environment Variables

These are user defined environmental variables that ClearTrigger is aware of that the user can change to modify ClearTrigger behavior.

***Environmental variables added in version 13.0 are so indicated by highlight.**

Environment Variable	Values
CLEARTRIGGER_BIT_21_AUTO_INTERACTIVE_ANSWER	If the Regions Policy manager set has allowed auto interactive answers for Functionality bit #21 via a matching ABS_bit_21_auto_answer_allow Override alias then setting this environmental variable to either "**yes**" or "**no**" will auto answer Lost+found warning dialog with this answer.
ABS_CLEARTRIGGER_VERBOSE	If the user sets this variable to "high" then clearbits_file key number appears in many dialogs that would be displayed. Read more on the **CLEARTRIGGER_VERBOSE** feature.
ABS_CONTRIB_KEEP	If the user sets this variable to any value then the value of the region defined **Functionality bit #20** (Auto Remove ".contrib." files) is ignored and .contrib. files are not removed during checkin or uncheckout for that user.
ABS_PREFER_NO_GUI	If the user sets this variable to "YES" then most of the error dialogs that would appear when company policy is violated that provide the user only one answer from Yes/No/Cancel would display to **stdout** instead and require no input. The single default answer would be automatically selected. Useful for scripts that run unattended.

Formatted Command Keys (examples)

Adding one very powerful command key definition in the *clearbits_file* can provide quite a bit of enterprise policy. The following examples will add several elements of policy to a fictitious site and in doing so explain about formatting command keys, which can be added and/or modified in the ***clearbits_file***.

cmd_list (Inhibiting Users)

Policy: Prevent specific users from executing a command on all enterprise VOBs or all VOBs in a ClearTrigger region.

To inhibit these users (raman and fred) from performing checkins in any VOB, one can do so without having to visit each VOB. One needs only to add or modify the **cmd_users_list** command_key line in the clearbits.txt file to include Raman and Fred and set the **cmd_list_type** to 'D'issallow.:

pre_checkin;R;;0;0;0;D; (raman) (fred) ;

If users fred or raman attempted the checkin command in any of the VOBs then this dialog or a similar one would appear with a consistent Enterprise message:

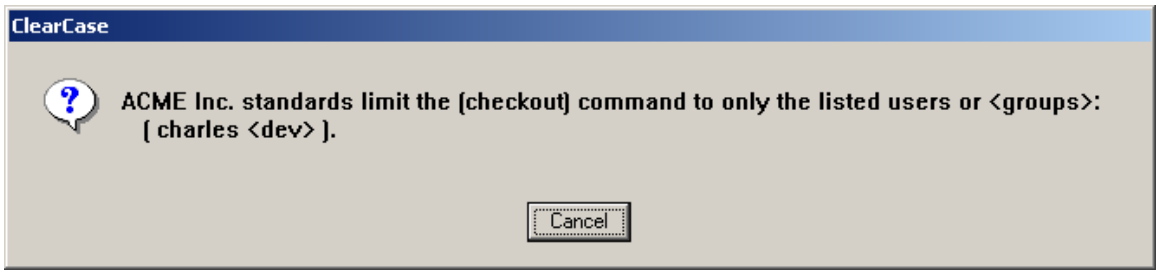
cmd_list (Allowing users)

Policy: Allow only specific users to execute a command on all enterprise VOBs or all VOBs in a ClearTrigger region.

To allow only users "charles and members of the dev group" to perform checkins in any VOB, one can do so without having to visit each VOB. One needs only to add or modify the **cmd_users_list** command_key line in the clearbits.txt file to include Charles and Scott and set the **cmd_list_type** to 'A'llow.

pre_checkin;R;;0;0;0;A; (charles) <dev> ;

If users **other** than Charles or members of the dev group attempted the checkin command in any of the VOBs then this dialog or a similar one would appear with a consistent Enterprise message:



trigger_list (Fire custom trigger for all but some users)

Policy: Allow members of the "dev" group to checkin code, additionally run the code_metrix program for everyone that checks in except "Charles" and "Scott" on all enterprise VOBs or all VOBs in a ClearTrigger region.

To allow only developers to performing checkins in any VOB, change the **cmd_user_list** and **cmd_list_type** fields. Then to run code metrics for everyone except Charles and Scott, add or modify the **cmd_trigger_list** and **trigger_list_type** command_key line in the clearbits.txt file. Include Charles and Scott in the **cmd_trigger_list** and set the **trigger_list_type** to 'N'ot_user.

pre_checkin;R;;0;0;0;A; <dev> ;N; (charles) (scott) ;0;Metrix.pl;

Only developers can perform checkins in the region. Additionally, they will all cause the Metrix.pl program (located in the depot) to execute successfully before the checkin can process, however Charles and Scott will **not** cause the Metric.pl program to execute.

trigger_list (Fire custom trigger for only a few users)

Policy: Allow developers to checkin on the bugfix branch, but when "Fred" or "Ramen" do so, start a build for all enterprise VOBs or all VOBs in a ClearTrigger region.

To allow only developers to perform checkins in any VOB, change the **cmd_user_list** and **cmd_list_type** fields. Additionally, to force successful compile for Fred or Ramen, add or modify the **cmd_trigger_list** and **trigger_list_type** command_key line in the clearbits.txt file. Include Fred and Ramen in the **cmd_trigger_list** and set the **trigger_list_type** to 'O'nly_user.

pre_checkin;R; brtype<bugfix>;0;0;0;A; <dev> ;O; (fred) (ramen) ;0;Build.pl;

Only developers can perform checkins in the bugfix branch in the region. Additionally, developers Fred and Ramen cannot checkin to that branch until they have a successful invocation of build.pl (located in the depot).

Sending parameters to triggers

Policy: After locking labels update website, send the label_type and an 'L' to the script for all enterprise VOBs or all VOBs in a ClearTrigger region.

To add parameters to called scripts or executables place them in the parameter list filed for each OS. Use the appropriate characters for the OS when passing environmental variables. :

pre_lock;R; lbtype<-all>;0;0;0;D;;N;;0;Update_web;%CLEARCASE_LBTYPE% L; Update_web; $CLEARCASE_LBTYPE L;0;E;;;

Note: You don't actually have to pass the environmental variables (or the ClearTrigger defined environmental variables) as they are available to scripts you write. You might need to pass them though to executables where you might not have access to the source code (e.g. third party software).

pre_lock;R; lbtype<-all>;0;0;0;D;;N;;0;Update_web.pl;L;Update_web.pl;L;

inhibit_flag

Policy: Prevent users from removing elements across all enterprise VOBs or all VOBs in a ClearTrigger region.

To ensure that no one could remove elements in all VOBs within the region, add or modify the **inhibit_flag** command key in your clearbits.txt file:

pre_rmelem;R;;X;

To restrict all users ***except*** the **Current VOB owner** from removing elements, change the **inhibit_flag** command key to a lowercase 'x'.

pre_rmelem;R;;x;

comment_restriction

Policy: Require commenting with a minimum number of words on a specific command within all enterprise VOBs or VOBs in a ClearTrigger region.

If you wanted to ensure that all checkouts to any VOBS contained at least a 10-word comment, add or modify the **cmd_comment_restriction** command_key line in your clearbits.txt file:

pre_checkout;R;;0;10;

If anyone attempted a *cleartool checkout* with comments containing less than 10 words in any of the VOBs then this dialog or similar would appear:

If you wanted this restriction to apply to everyone ***except*** the **Current VOB owner** then change the **cmd_comment_restriction** to negative 10.

pre_checkin;R;;0;-10;

notify_flag

Policy: Send e-mail using the Enterprise mail notification when any user attempts to perform a certain command in any enterprise VOB or VOB in a ClearTrigger region.

To ensure that every attempt to perform a 'rmelem' command in any VOB sends e-mail to the enterprise designated addresses, add or modify the **cmd_notify_flag** command_key line in your clearbits.txt file:

pre_rmelem;R;;0;0;X;

For anyone attempting the rmelem command, the enterprise-defined e-mail will automatically be sent. The **NOTIFY.xx** program you've specified in the policy depot is invoked. This command can be easily modified to send to multiple addresses, display dialog boxes or be customized in any site-specific manner.

If you wanted this e-mail notification to apply to everyone ***except*** the *Current VOB owner*, change the **notify_flag** to a lowercase 'x' instead of 'X'.

pre_rmelem;R;;0;0;x;

RIN_type – Restriction List

Policy: Do not allow any users to checkout elements with attribute type "QA_PASSED" in all VOBs within the region.

To ensure this action, add or modify the **RIN_type and RIN_list** command keys in your clearbits.txt file:

pre_checkout;R;attype<QA_PASSED>;

RIN_type - Inclusion List

Add or modify the **RIN_type and RIN_list** command keys in your clearbits.txt file to enforce the following policy:

Policy: Only allow testers to apply labels on branch "int_test" within a VOB.

pre_mklabel;I;brtype <int_test>;X;0;0;D; (tester1) (tester2) ;

RIN_type - Negation List

Policy: Deny all rmelem command for all elements except documentation and core files for all VOBs within the region.

To ensure this action you could add or modify the **RIN_type and RIN_list** command keys in your clearbits.txt file:

pre_checkin;N;eltype<word_docs> eltype<core>;;X;

Users attempting to perform this command would receive a consistent Enterprise dialog window similar to the one below:

Command_List_type – Disallowed List

Policy: Do not allow "Ramesh" or members of the "Documentation" group to perform rmelem commands in all VOBs within the region.

To ensure this action, add or modify the **Command_list_type and Command_list** command keys in your clearbits.txt file:

pre_rmelem;R; ;0;0;0;D; (ramesh) <doc> ;

"Ramesh" and members of the "doc" group attempting to perform this command or anyone attempting to perform this command in the \classes VOB would receive a consistent Enterprise Dialog window similar to the one below:

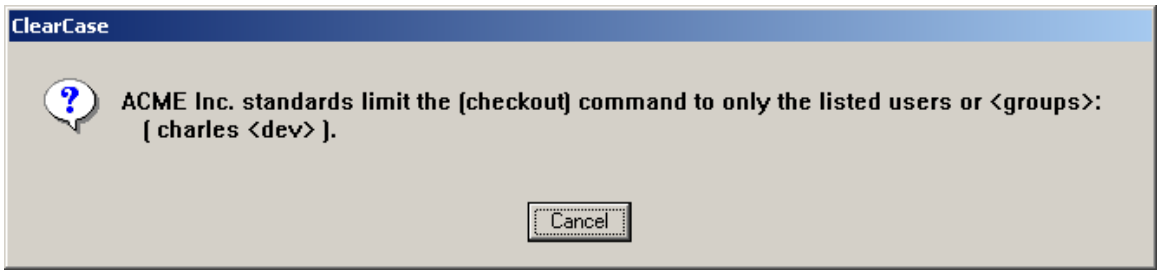
Command_List_type – Allowed List

Policy: Only allow "Charles" and members of the "dev" group to perform the checkout commands in all VOBs within the region.

To ensure this action you could add or modify the **Command_list_type and Command_list** command keys in your clearbits.txt file:

pre_checkout;R; ;0;0;0;A; (charles) <dev> ;

Users other than "Charles" and members of the "dev" group attempting to perform this command would receive a consistent Enterprise dialog window similar to the one below:

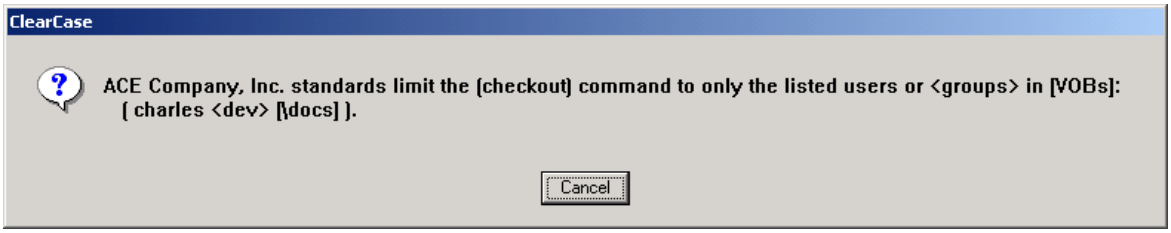


Policy: Only allow Charles and members of the "dev" group to perform the checkout commands in the VOB \docs.

To ensure this action, add or modify the **Command_list_type and Command_list** command keys in your clearbits.txt file:

pre_checkout;R;;0;0;0;A; (charles) <dev> [\docs] ;

Users other than "Charles" and members of the "dev" group attempting to perform this command would receive a consistent Enterprise dialog window similar to the one below:



Command_List_type – Ignore List

Policy: Provide a confirmation on checkins for all "interns" except the user "Erik" (who is in "intern", do not confirm for non-interns.

To ensure this action you could add or modify the **Command_list_type and Command_list** command keys in your clearbits.txt file:

pre_checkin;R;;0;0;0;I; (erik) ;O; ~interns ;;;;;0;Q;Are You sure?;YQ;

Only users that are interns (except Eric) attempting to perform this command would receive a consistent Enterprise dialog window similar to the one below:

This will allow the user to answer "yes" to continue or "Cancel" to abort.

Use of Command Groups

Wherever you can place a command key you can also place command groups (e.g. MODIFY_DATA).

For Example:
To implement that user-id "david" cannot perform any element modifying action in ALL VOBs then apply this rule or similar:

pre_MODIFY_ELEM;R;;0;0;0;D; (david) ;

To implement that all data modifying actions must be commented with at least 10 words by all users ***except*** for the **Current VOB owner** apply the following rule:

pre_MODIFY_DATA;R;;x;-10;

Remember! **ClearQuery** makes this even easier.

Use of Multiple Commands

In addition to using a single command key to define several policy points, you can also use multiple command keys. The commands simply need to be space separated. All matching keys are read unless a key is read which inhibits the command for the user. The keys are evaluated in order of appearance. This allows for creation of a hierarchy or repetition of keys.

For Example:
To implement that only user-id "ccadmin" can remove elements or versions across the enterprise then apply this rule or similar:

pre_rmelem pre_rmver;R;;X;0;0;D; (ccadmin) ;

To implement policy such that only user-ids "charles" and "scott" can remove elements, with e-mail notification when any user attempts this command and an additional e-mail notification if any user succeeds. Let's assume you also want the action commented with at least 10 words unless the user is also a *Special User.*

pre_rmelem post_rmelem;R;;X;-10;X;D; (charles) (scott) ;

Use of Compound Commands

Command groups and single commands can be compounded to create complex command keys. This powerful feature of ClearTrigger allows a multitude of combinations of command keys to be defined to fit your organizations specific needs.

For Example:
Allow all user ids ***except*** "Fred" to modify elements and remove types and require comments with at least 15 words. Also send e-mail for all successful attempts, unless the user is a *Special User*.

pre_MODIFY_ELEM pre_rmtype;R;;0;15;x;D, (fred) ;

or

pre_MODIFY_ELEM;R;;0;15;x;D; (fred) ;
pre_rmtype;R;;0;15;x; D; (fred) ;
or

pre_MODIFY_ELEM pre_rmtype;R;;0;0;x; D; (fred) ;
pre_rmtype;R;;0;15;
pre_MODIFY_ELEM;R;;0;15;0; D; (fred) ;

To implement using severity…

"I want e-mail when elements are modified and I want two when someone removes an element." Perhaps two e-mails get your attention!

post_MODIFY_ELEM;R;;0;0;X;
post_rmelem;R;;0;0;X;

This command key will send the Enterprise notification e-mail each time that an element is modified and send an additional e-mail every time that the modification is a removal of an element.

Policy without Triggers

ClearTrigger allows you to get your organization up and running with policies in place with minimal development time and effort. In many cases organizations can quickly realize benefits such as the ones listed below:

- **Create simple policy without writing code.**
- **Add or change enterprise policy without defining additional triggers.**
- **Automatically apply enterprise policy to existing and new VOBs.**
- **Implement different policies for different regions.**
- **Save time on policy implementation.**

This section describes brief scenarios that demonstrate how to use ClearTrigger to implement and enforce common development policies. **Remember! ClearQuery** makes this even easier.

Policy: Prevent Use of Certain Commands for all Users

Use ClearTrigger to prevent users from executing various commands. In the Commands section of clearbits.txt add the commands or command groups you want to restrict:

- **pre_unreserve;R;;X;**
- **pre_unlock pre_rmver;R;;X;**
- **pre_rmelem;R;;X;**
- **pre_rmbranch;R;;X;**
- **pre_MODIFY_MD;R;;X;**

Policy: Limit Use of Certain Commands to only the VOB Owner

Use ClearTrigger to prevent users (except the VOB owner) from executing various commands. In the Commands section of clearbits.txt add the commands or command groups you want to restrict or you could use the (&) dynamic variable in allowed list:

Making use of "Inhibit Flag"	Making use of "Allowed Lists"
pre_unreserve;R;;x;	**pre_unreserve;R;;0;0;0;A;(&);**
pre_unlock pre_rmver;R;;x;	**pre_unlock pre_rmver;R;; 0;0;0;A;(&);**
pre_rmelem;R;;x;	**pre_rmelem;R;; 0;0;0;A;(&);**
pre_rmbranch;R;;x;	**pre_rmbranch;R;; 0;0;0;A;(&);**
pre_MODIFY_MD;R;;x;	**pre_MODIFY_MD;R;; 0;0;0;A;(&);**

Policy: Prevent Specific Users from use of Certain Commands

Use ClearTrigger to prevent users from executing various commands. In the Commands section of clearbits_file add to command inhibited users lists or add **region_inhibited** users to the Enterprise Section of the **clearbits.txt**.

Stop certain users from executing certain commands:

- **pre_unreserve;R;;0;0;0;D; (rsmith) (jdoe) <cmteam> ;**
- **pre_unlock pre_rmver;R;;0;0;0;D; (rramesh) (tsmith) <dev> ;**
- **pre_rmbranch;R;brtype<main>0;0;0;D; (rramesh) (jdoe) ;**
- **pre_MODIFY_MD;R;;0;0;0;D; (rmoore) (pherman) <int_test> ;**
- **pre_MODIFY_MD;R;;0;0;0;D; (*) ;**

Allow only certain users/<groups> to execute certain commands:

- **pre_unreserve;R;;0;0;0;A; (rsmith) (jdoe) <cmteam> ;**
- **pre_unlock pre_rmver;R;;0;0;0;A; (rramesh) (tsmith) <dev> ;**
- **pre_rmbranch;R;brtype<main>0;0;0;A; (rramesh) (jdoe) ;**
- **pre_MODIFY_MD;R;;0;0;0;A; (rmoore) (pherman) <inttest> ;**
- **pre_MODIFY_MD;R;;0;0;0;A; (&) ; /* (&) = the current VOB owner */**

Allow commands only in certain VOBs:

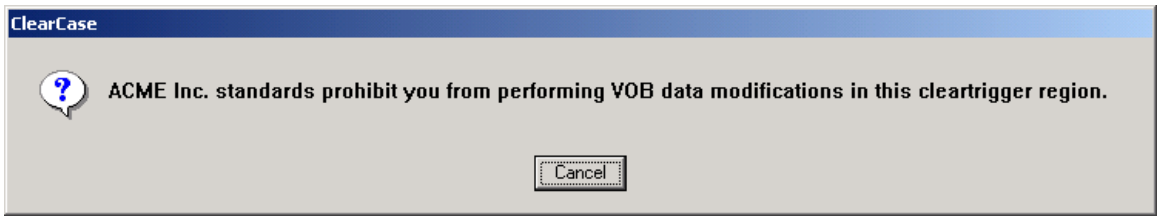
- **pre_unreserve;R;;0;0;0;A; [\a_vob] [/vobs/_a_vob] [\b_vob] ;**
- **pre_unlock pre_rmver;R;;0;0;0;A; [\a_vob] ;**
- **pre_rmbranch;R;brtype<main>0;0;0;A; [\a_vob] ;**
- **pre_MODIFY_MD;R;;0;0;0;A; [\a_vob] ;**

Restricting users/groups/VOB by region (enterprise section):

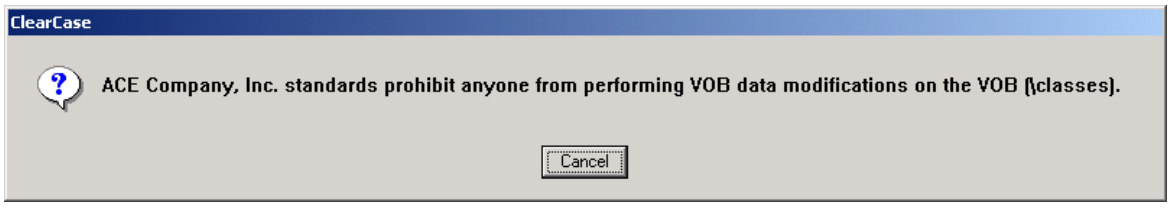
Populate the **region inhibited users** field like the one below in the enterprise section of the clearbits file:

- **; (fred) <janitors> (jgardner) [\classes] ;**

Adding an entry like the one above to the region inhibited field would product a dialog box like below for any data modifying action that might be attempted by "fred", "jgardner" or members of the “janitors” group.



Additionally, it would prevent ANYONE from performing actions in the "classes" VOB..



Policy: Prevent Specific Commands in certain ClearCase Regions

Use ClearTrigger to prevent users from executing various commands in specific ClearCase Regions. This allows you to use one clearbits file for multiple ClearCase regions that have different. In the Commands section of clearbits_file add to command inhibited users lists or add region_inhibited users to the Enterprise Section of the clearbits.txt

Stop users from executing commands in certain ClearCase regions:

- **pre_unlock pre_rmver;R;;0;0;0;D; (rramesh) (tsmith) 'Development' ;**
- **pre_rmbranch;R;brtype<main>0;0;0;D; (rramesh) (jdoe) 'QA_*';**
- **pre_MODIFY_MD;R;;0;0;0;D; (rmoore) (pherman) [\prod*] 'QA_*';**

Allow only certain users/<groups> to execute certain commands:

- **pre_unreserve;R;;0;0;0;A; (rsmith) (jdoe) 'QA_*';**
- **pre_unlock pre_rmver;R;;0;0;0;A; (rramesh) (tsmith) 'Development';**
- **pre_MODIFY_MD;R;;0;0;0;A; (&)'QA_*'; /* (&) = the current VOB owner */**

Policy: Require Comments for all Changes

If one wants all developers, but **not** the current VOB owners, to add a 10-word comment to changes made to the source code.

- In the Enterprise Commands section of clearbits_file add the checkin operation: **pre_MODIFY_MD;R;;x;-10;X;**

- If you want all users, ***including*** the VOB owner to add comments, then change the **-10** to a **10** in the key definition.

  pre_MODIFY_MD;R;;x;10;X;

Policy: User Question Dialogs

If one wants to prevent all users from performing checkins and you have a message that you want to display to them.

- In the Enterprise Commands section of clearbits_file add the checkin operation: **pre_checkin;R;;0;0;0;D; ;N; ;;;;;0;Q;Coding is over;Q;**

Which will always cause the command to fail after the user is warned why.

OR

- In the Enterprise Commands section of clearbits_file add the checkin operation: **pre_checkin;R;;0;0;0;D; ;N; ;;;;;0;Q;Are You sure?;YQ;**

This will allow the user to answer "yes" to continue or "Cancel" to abort.

Policy – Atomic Changes

Within ClearTrigger there exists a method to link elements together such that they can be checked in, checked out or unchecked out together. The policy maker can then determine if the linked elements must be modified together or if it is just "suggested" that they are

Users themselves can create the appropriate links between the elements, as the ability to "bind" element is usually a developer function. The hyperlink type "atomic_change" should be created in a VOB with ClearTrigger applied and then anyone can link element together.

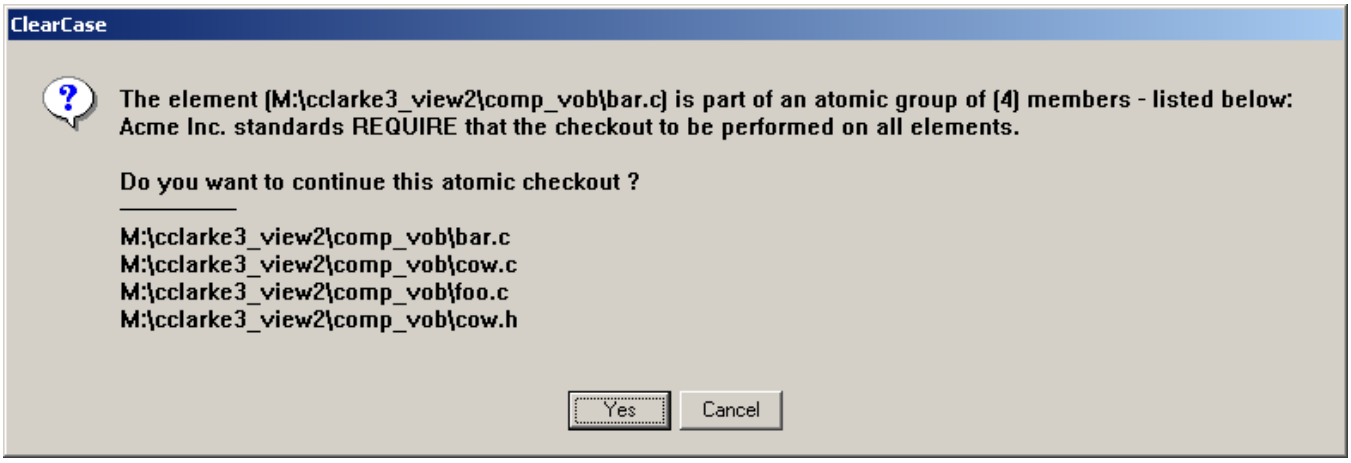
To create the hyperlink type once in the VOB, type the command below:

cleartool mkhltype –nc atomic_change

Then link elements together by using the native ClearCase mkhlink command like so:

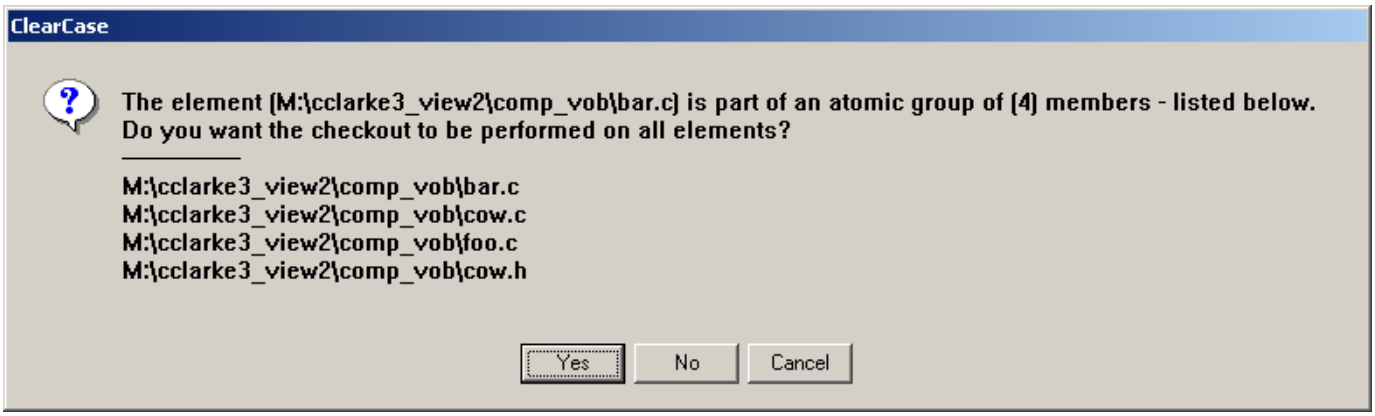
cleartool mkhlink –nc atomic_change foo.c@@ bar.c@@
cleartool mkhlink –nc atomic_change bar.c@@ cow.c@@
cleartool mkhlink –nc atomic_change cow.h@@ cow.c@@

It does not matter when the elements are linked, how many are linked, or the direction of the links. The linked family is "headless" such that if John knows of an association between bar.c and cow.c and creates a link while Robert knows of an association between cow.h and cow.c and creates a link while Kathy creates a link between foo.c and bar.c, all elements are "equally" linked such that if Tomas checks out any of the files, they can all be checked out.



By default if the bit is enabled, when a user checks out a linked file they may see a dialog similar to the one below that "forces" the association and provides them the opportunity checkout all of the elements or cancel the attempt. The default selection is the "Yes" button – to check them all out. If the region policy maker has turned on the Atomic Change" bit (bit 12) then only the "yes" and "cancel " options are given. The elements are checked out together by "force" or the checkout is canceled.

To change the personality of this bit such that the user is given a choice between “yes”, “no” or “cancel” just set the special purpose cleartrigger_alias ABS_bit_12_personality in the clearbits file to “choice”. The alias can be set to “choice” or “force” (default is “force”). For example, when the cleartrigger_alias ABS_bit_12_personality is set to "choice" a user creating a new directory would see output like that below:



The default selection is the “Yes” button – to check them all out.

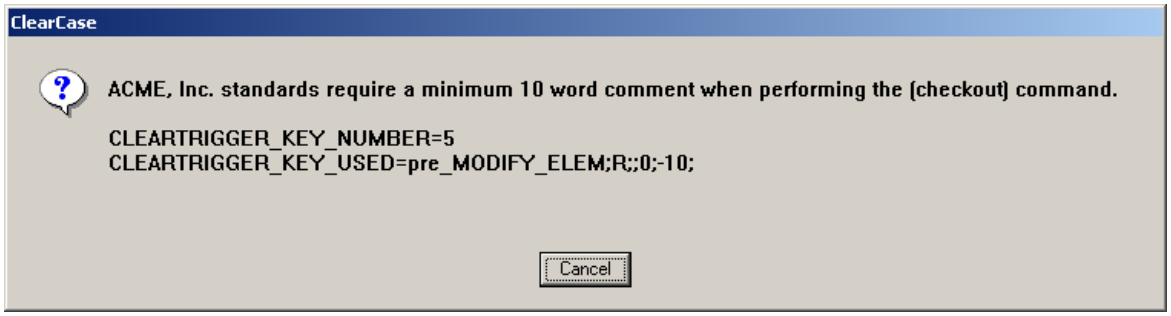
The elements do not have to be in the same directory and are not limited to files.

ClearTrigger Verbose Mode

You might have many keys in your clearbits_file that can by design prevent one or more users from performing an action in your ClearTrigger client VOB. When they are prevented from performing an action they are usually shown a dialog that explains why. Sometimes you as the administrator might want to know or confirm exactly which key produced the dialog. To get the key (and key number) to display in the message dialog you may set the **ABS_CLEARTRIGGER_VERBOSE** environmental variable to the "high" value.

Without **ABS_CLEARTRIGGER_VERBOSE** set to the "high" value a dialog like this might appear for a checkout:

With **ABS_CLEARTRIGGER_VERBOSE** set to the "high" value the same circumstance would produce a dialog like this:



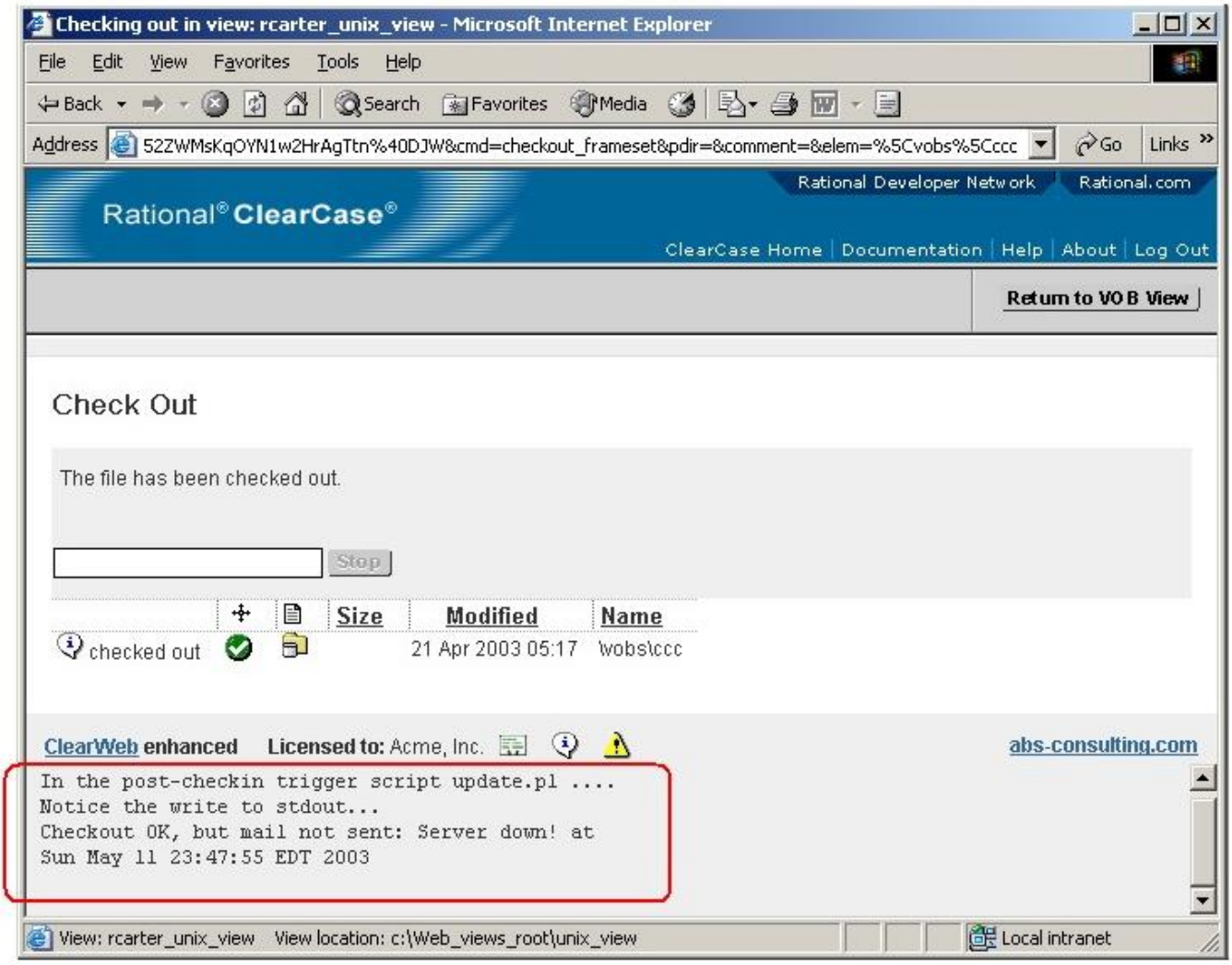
Full Integration with ClearWeb

ClearCase itself does not support interactive triggers for ClearCase web-users. ClearCase defines interactive triggers as any that provides GUI notification (e.g. use clearprompt) or that write to stdout or stderr. Given that definition, many user written triggers and many ClearTrigger features are interactive. To allow web-users to efficiently use VOBs that have ClearTrigger applied, ClearTrigger works seamlessly with ClearWeb so that dialogs like the one depicted below:

would appear to a ClearCase web user as:

All ClearTrigger dialogs have a web equivalent and all web-equivalents provide the same wait for response functionality as their non-web counterparts.

The integration also seamlessly writes stdout and stderr message to the terminal if a standard ClearCase user and to the web-client if a ClearCase web user as depicted on the next page.



Most ClearTrigger written features and functionality bits are web-enabled. Any triggers you write and apply with ClearTrigger can also be web-enabled so that your writes to stdout and stderr are displayed to the web users. You can also use ClearWeb tools to enable any of your custom triggers to have web-enabled dialogs that request for and waits for user responses on the web just as clearprompt would locally.

All ClearTrigger included functionality is by default Web-enabled (except Smart Checkin Functionality). For more on ClearWeb Functionality, refer to the ClearWeb documentation.

Applying ClearTrigger

It is easy to apply ClearTrigger using the **clearapply** application.

Pre-Existing Triggers

Prior to applying ClearTrigger, check to see if there are existing triggers in the VOB. If you have existing triggers then we recommend that your administrator:

- **Record pre-existing triggers**
- **Remove triggers from the VOB**
- **Implement triggers using ClearTrigger**

ClearApply

ClearApply registers and applies the ClearTrigger executable to a VOB. It verifies the location of the clearbits_file file and the cleartrigger.exe executable for the current operating system. You will most likely use the Graphical User Interface (GUI), but there is a Command Line Interface (CLI) as well.

- To learn how to use **clearapply** from the GUI to apply ClearTrigger to a VOB read the section entitled Applying ClearTrigger to a VOB (using the GUI).
- To learn how to use **clearapply** from the CLI to apply ClearTrigger to a VOB read the section entitled Applying ClearTrigger to a VOB (using the CLI).
- To learn how to use **clearapply** from the GUI to remove ClearTrigger to a VOB read the section entitled Removing ClearTrigger to a VOB (using the GUI).
- To learn how to use **clearapply** from the CLI to remove ClearTrigger to a VOB read the section entitled Removing ClearTrigger to a VOB (using the CLI).

Applying ClearTrigger to a VOB (using the CLI)

Use ***clearapply*** to apply ClearTrigger to a VOB or group of VOBs. Specify the path to the **cleartrigger** executable(s) as well as **clearbits_file(s)**. The command syntax is:

clearapply {-ver | -help | {remove | { [cleartrigger_path config_dir [cleartrigger_dir config_dir]} vob_tag {no_mklabel}

where:

`-ver` displays the ClearApply version pneumonic
`-help` displays the usage statement
`remove vob_tag` removes cleartrigger from VOB vob_tag
`cleartrigger_path` path to cleartrigger executable (e.g. /net/policy/cleartrigger.exe)
`config_dir` path to clearbits_file (e.g. /net/policy/clearbits.txt)
`no_mklabel` applies ClearTrigger in no_mklabel mode – ClearTrigger is not consulted and does not fire for ClearCase mklabel commands.

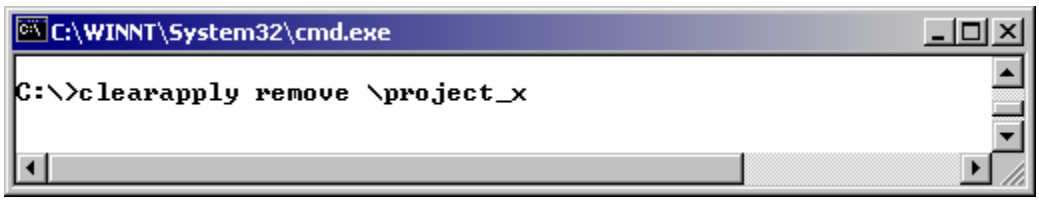
If one **cleartrigger_path/config_dir** pair is given then the trigger is applied with the **`-exec`** option for the mktrtype command (single OS), if two cleartrigger_path/config_dir pairs are given then both the **`-execwin`** and **`-execunix`** options are used (InterOp environment).

Upon execution, ClearTrigger will search for existing ClearTrigger types, remove them if found, create new trigger types and lock them. The output one would see should be consistent with the example below:

Removing ClearTrigger from a VOB (using the CLI)

TO REMOVE CLEARTRIGGER FROM A VOB EXECUTE CLEARAPPLY WITH THE REMOVE OPTION.

Example:



Clearapply remove will search for existing Cleartrigger trigger types, unlock them, and remove them from the VOB. The output one would see should be consistent with the example below:

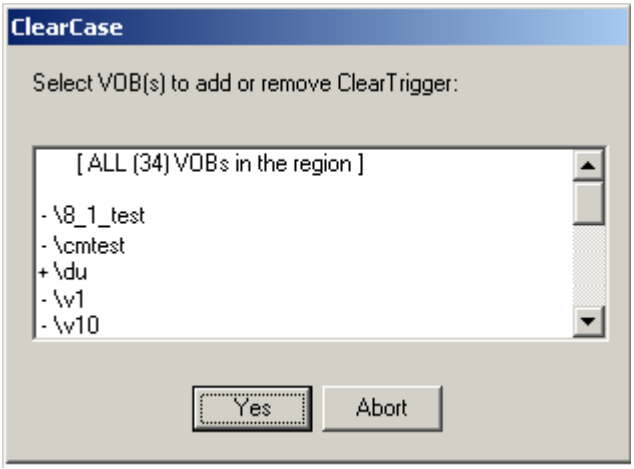
Applying ClearTrigger to a VOB using the GUI

Use ***clearapply*** to apply ClearTrigger to a VOB or group of VOBs. Specify the path to the **cleartrigger** executable(s) as well as **clearbits_file(s)**. To use the GUI version type:

```
clearapply
```

You will be asked to select any number of VOBs in the current ClearCase region or to abort the application. You can select "ALL VOBs in the region" to tell clearapply to act on all VOBs. If you have more than 150 VOBs in the region then clearapply will display them in 150 VOB groupings; if there is more than one grouping then you will have the option to "select all vobs in the displayed grouping" as well. You can select the "ALL VOBs in the region" item at any time. The VOBs in the region are shown preceded by a ClearTrigger state identifier:

ClearTrigger Identifier	Meaning
-	ClearTrigger is **not** applied to this VOB
+	ClearTrigger is applied to this VOB in single OS mode (e.g. with **`-exec`** type definitions)
*	ClearTrigger is applied to this VOB in Inter-Op mode (e.g. with both **`-execwin`** and **`-execunix`** type definitions)

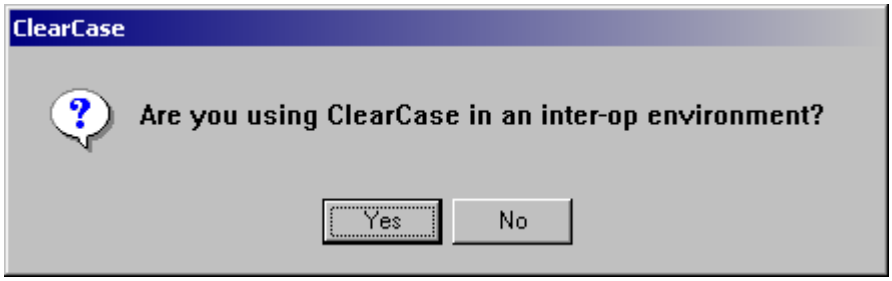


This allows the operator to know in advance the current state of the VOB as it relates to ClearTrigger. One may choose to add or remove ClearTrigger while selecting any subset. Removing ClearTrigger from a VOB that does not already have ClearTrigger will not cause any harm. Adding ClearTrigger to a VOB that already has ClearTrigger is helpful when you wish to change the **clearbits_file** location, **cleartrigger.exe** location or alternate between Inter-Op mode to single OS mode.

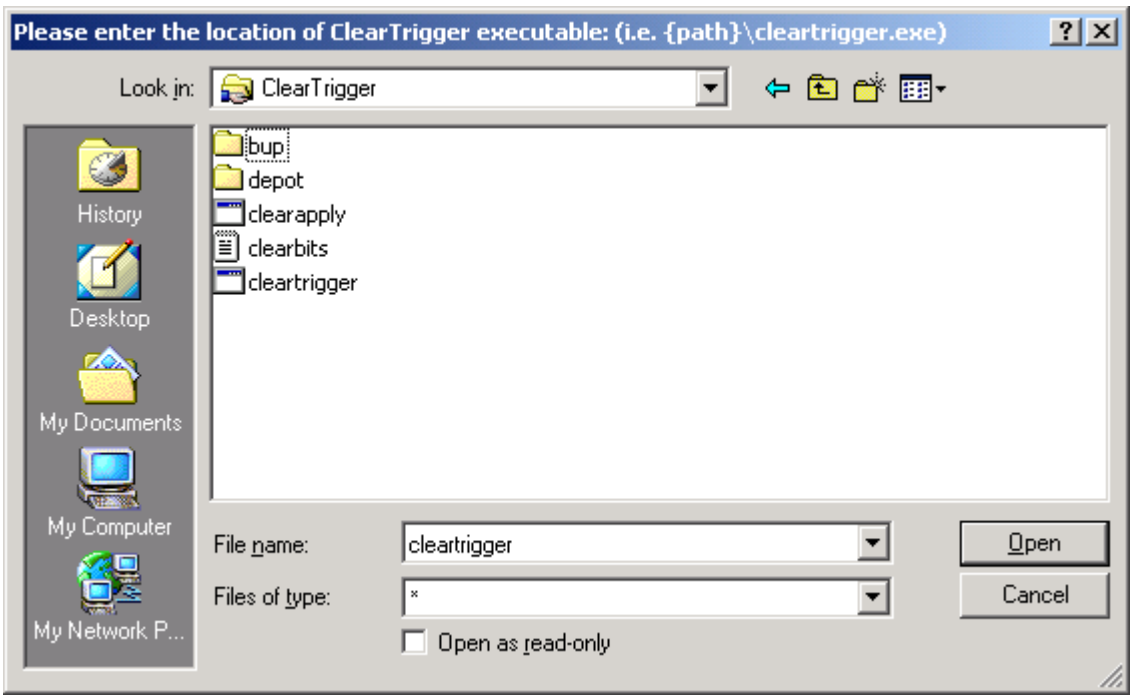
The operator is then asked if they want to add or remove ClearTrigger on the selected VOB(s).

Select "Yes" to add ClearTrigger to the selected VOB(s). You are then asked of you want to include **mklabel** ClearTrigger processing for this VOB.

Select "Yes" (the default) to configure ClearTrigger to control all triggerable commands for this VOB or select "No" if to configure ClearTrigger to control all triggerable commands *except* ***mklabel*** for this VOB – ClearTrigger is not consulted and does not fire for ClearCase mklabel commands. Once the selection is made, the dialog below is displayed.



If "No" is selected from this box the operator will be prompted for the location of the ClearTrigger executable and the location of the clearbits_file, as illustrated by the next screen shot.

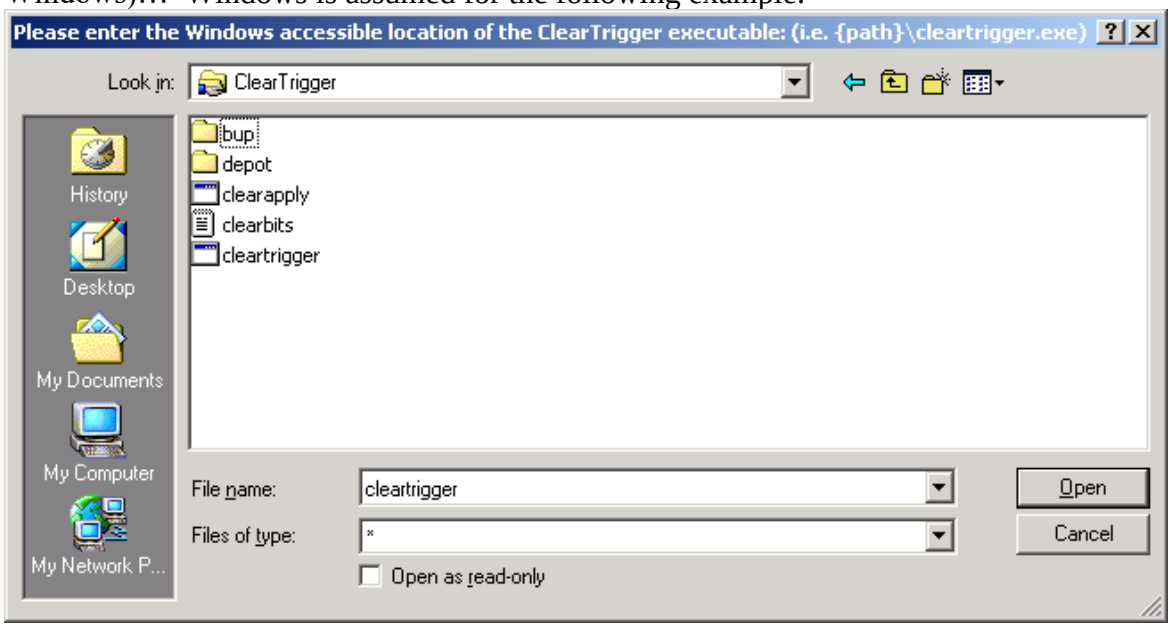


One can navigate to the cleartrigger executable to limit any typing errors then select "open" to confirm the selection. If the selected executable is not a distributed cleartrigger executable then the operator is asked to repeat the selection. The path should be such that it is valid in the desired operating system or systems if you have an environment where one path can support both Windows and UNIX locations.

A similar dialog is displayed to select the **clearbits file**. The clearbits file selected must currently exist and be readable; if not they are asked to repeat the selection.

Answering Yes to the inter-op questions yields two sets of dialogs boxes, which ask for OS specific information for ClearTrigger. You will be prompted for information for the "current" OS environment and information for the "other" OS environment separately.

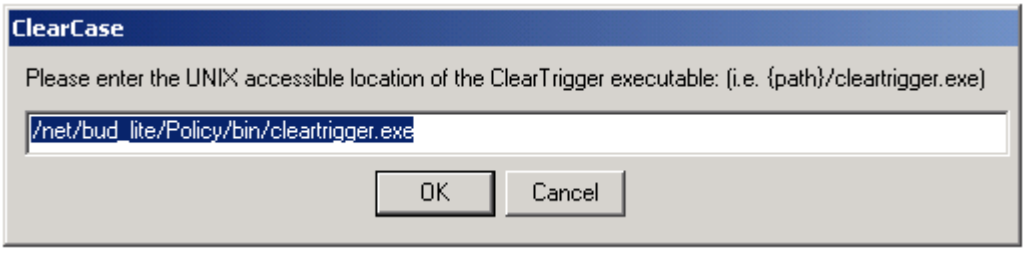
First the information for the current OS that you are running clearapply from (UNIX or Windows)… Windows is assumed for the following example:

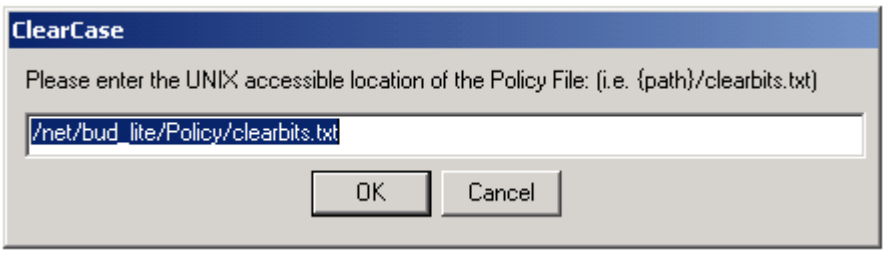


One can navigate to the cleartrigger executable to minimize typing errors then select "open" to confirm the selection. If the selected executable is not a distributed cleartrigger executable then the operator is asked to repeat the selection. The path should be such that it is valid in the current operating system.

A similar dialog is displayed to select the **clearbits file**. The clearbits file selected must currently exist and be readable; if not, they are asked to repeat the selection.

The operator is then asked to input the information for the "other" operating system. For these fields the operator is asked to type the information in because it is assumed that one may not be able to navigate to the desired location using the file selection dialog.





Upon completion of entering the appropriate information, ClearApply will add ClearTrigger to the selected VOBs.

Environmental Variables and Defaults using ClearApply

When executing ClearApply, if any of these environmental variables are set they are used for prompt dialog defaults.

Variable Name	Default
ABS_CA_SINGLE_DEPOT	Default path of clearbits_file in single OS mode
ABS_CA_UNIX_DEPOT	Default path of clearbits_file for UNIX in InterOp mode
ABS_CA_WINDOWS_DEPOT	Default path of clearbits_file for Windows in InterOp mode
ABS_CA_SINGLE_PATH	Default path of ClearTrigger in single OS mode
ABS_CA_UNIX_PATH	Default path of ClearTrigger for UNIX in InterOp mode
ABS_CA_WINDOWS_PATH	Default path of ClearTrigger for Windows in InterOp mode
ABS_CA_OMIT_MKLABEL	If defined and equal to **yes** it applies ClearTrigger in **no_mklabel** mode else if defined and equal to **no** it applies ClearTrigger in ***classic*** mode. If undefined or not equal to yes or no then the operator is queried for a mode.

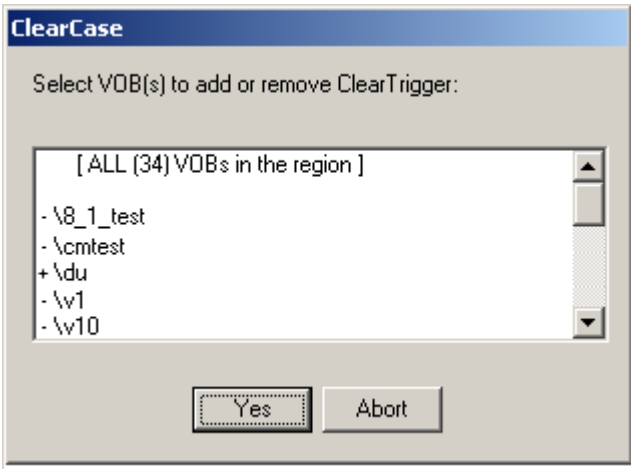
Removing ClearTrigger from a VOB using the GUI

Use ***clearapply*** to remove ClearTrigger from a VOB or group of VOBs. To use the GUI version type:

```
clearapply
```

You will be asked to select any number of VOBs in the current ClearCase region or to abort the application. You can select "ALL VOBs in the region" to tell clearapply to act on all VOBs. If you have more than 150 VOBs in the region then clearapply will display them in 150 VOB groupings; if there is more than one grouping then you will have the option to "select all VOBs in the displayed grouping" as well. You can select the "ALL VOBs in the region" item at any time. The VOBs in the region are shown preceded by a ClearTrigger state identifier:

ClearTrigger Identifier	Meaning
-	ClearTrigger is **not** applied to this VOB
+	ClearTrigger is applied to this VOB in single OS mode (e.g. with **`-exec`** type definitions)
*	ClearTrigger is applied to this VOB in Inter-OP mode (e.g. with both **`-execwin`** and **`-execunix`** type definitions)



This allows the operator to know in advance the current state of the VOB as it relates to ClearTrigger. One may choose to add or remove ClearTrigger while selecting any subset. Removing ClearTrigger from a VOB that does not have ClearTrigger will not cause any harm and adding ClearTrigger to a VOB that already has it is helpful when you wish to change the clearbits_file location or ClearTrigger location or change back and forth between Inter-Op mode and single OS mode.

The operator is then asked if you want to add or remove ClearTrigger on the selected VOB(s).

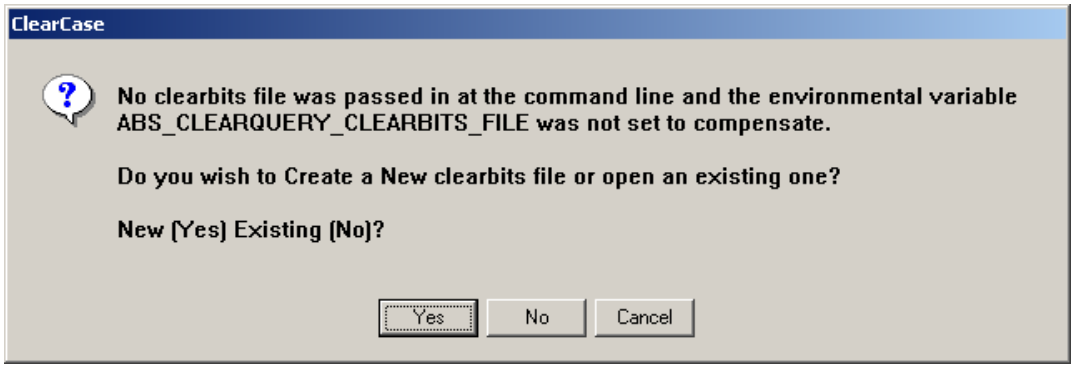
Selecting "No" will remove ClearTrigger from the selected VOB(s).

ClearQuery

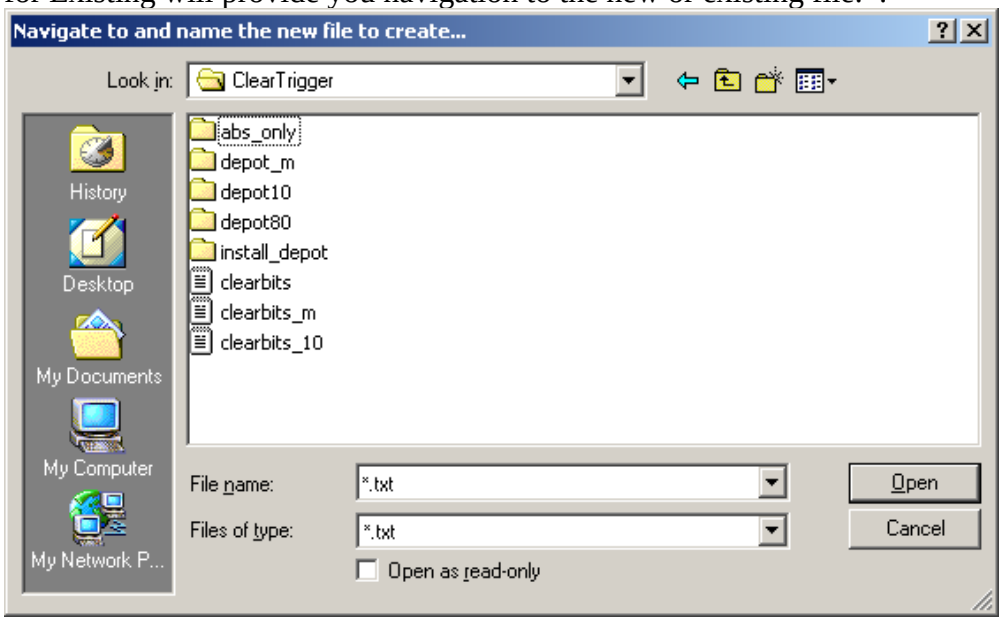
ClearQuery is bundled with ClearTrigger. ClearQuery provides an easier way to create or **change existing clearbits files** as well as provide a **reporting facility** for ClearTrigger.

Running ClearQuery – Creating a clearbits file

Creating a new clearbits file is easier with ClearQuery. If you start ClearQuery without providing an existing Clearbits file at the command line it will prompt you to create a new one or load an existing one:



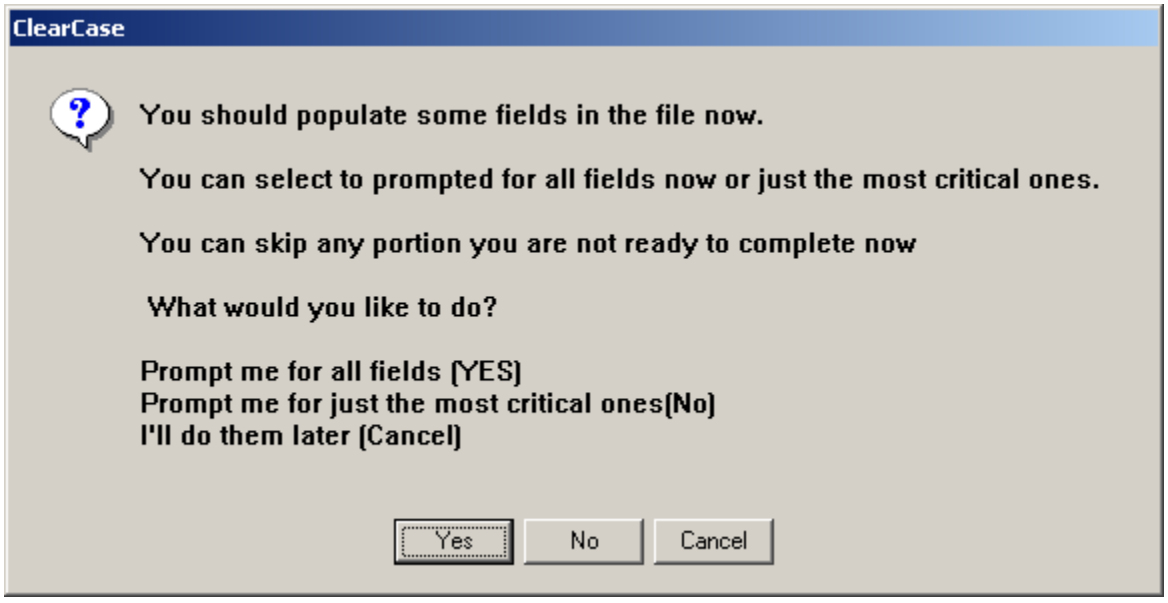
Selecting "Cancel" will abort and close ClearQuery. Selecting either "Yes" for New or "No" for Existing will provide you navigation to the new or existing file. :



If opening an existing file the selected file is loaded and you are allowed to modify the existing file (refer to the section entitled: **Running ClearQuery – Modifying a clearbits file**). If creating a new file you would navigate to the expected parent directory and then input

the new file name to create the new clearbits file. You must have "write" access to the parent directory.

The new clearbits file is created with minimal information in it initially, but you are provided an initial opportunity to populate the file with more specific information (though it can you can do this at any time).



Selecting "Yes" will guide you through all of the fields in the clearbits file as if you selected each of the fields in the **Modify Clearbits Pane.** You will be guided through the fields of this pane in this order.

A03, A24, A17, A04, A05, A06, A07, A08, A09 A10, A11, A12, A13, A14, A15, A16, A22, A23

Selecting "No" will guide you through all of the most critical fields in the clearbits file as if you selected each of the fields in the **Modify Clearbits Pane.** You will be guided through the fields of this pane in this order.

A03, A24, A17, A04, A05, A08, A10, A11, A12, A13, A16, A22, A23

Selecting "Cancel" will allow you to complete the file at your leisure by just taking you directly to the **Modify Clearbits Pane.**

Running ClearQuery - Configuring ClearQuery

The configuration of ClearQuery is minimal, ClearQuery gets most of the information it needs from any existing ***clearbits_file***. However, since it does call two other executables (cleartrigger and an internet browser), you must inform ClearQuery where the executables are. This is done once per region as ClearQuery stores the information in the clearbits file.

ClearQuery hides the information within comments in the clearbits file so as not to affect ClearTrigger itself. All ClearQuery variables start with the string "**#!clearquery_**" at the beginning of a line in the clearbits files. ClearQuery currently recognizes nine (9) variables:

- **#!clearquery_unix_cleartrigger_check=**
- **#!clearquery_windows_cleartrigger_check=**
- **#!clearquery_unix_browser=**
- **#!clearquery_windows_browser=**
- **#!clearquery_relocated_logging_dir_unix=**
- **#!clearquery_relocated_logging_dir_windows=**
- **#!clearquery_relocated_results_dir_unix=**
- **#!clearquery_relocated_results_dir_windows=**

Everything on the line "after" the '=' is the "value" of the variable.

Example:

#!clearquery_unix_cleartrigger_check=/net/ccserv/policy/cleartrigger.exe
#!clearquery_windows_cleartrigger_check=\\ccserv\policy\cleartrigger.exe
#!clearquery_unix_browser=/net/machine/dir/netscape
#!clearquery_windows_browser=C:\Program Files\Internet Explorer\IEXPLORE.EXE
#!clearquery_relocated_logging_dir_unix=
#!clearquery_relocated_logging_dir_windows=
#!clearquery_relocated_results_dir_unix=
#!clearquery_relocated_results_dir_windows=

- **#!clearquery_unix_cleartrigger_check=**
  Inform ClearQuery of which cleartrigger.exe is used for the **cleartrigger -check** command can is used to validate the clearbits file. Used when invoked from a **UNIX** OS.

- **#!clearquery_windows_cleartrigger_check=**
  Inform ClearQuery of which cleartrigger.exe is used for the **cleartrigger -check** command can is used to validate the clearbits file. Used when invoked from a **Windows** OS.

- **#!clearquery_unix_browser=**
  Inform ClearQuery of which UNIX **web browser** to use to display the html graphs that are created to show the results of ClearQuery "queries" reports, when requested from a **UNIX** OS.

- **#!clearquery_windows_browser=**
  Inform ClearQuery of which Windows **web browser** to use to display the html graphs that are created to show the results of ClearQuery "queries" reports, when requested from a **Windows** OS.

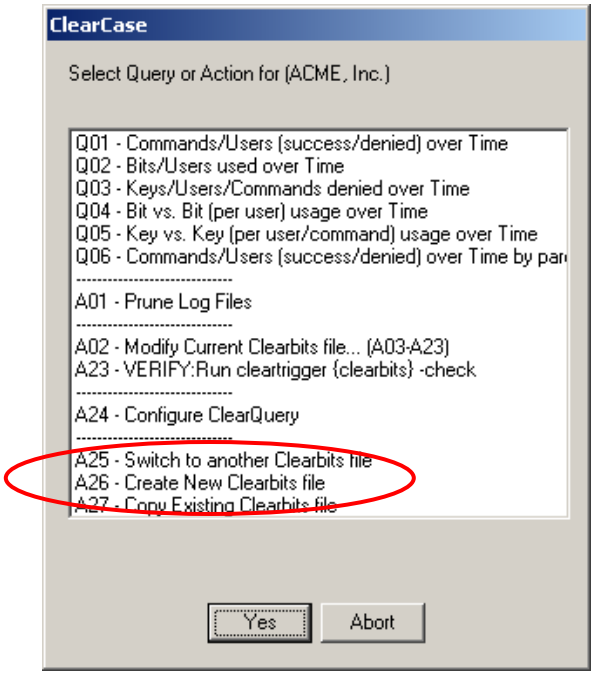
- **#!clearquery_relocated_logging_dir_unix=**
  Optionally, set to inform ClearQuery the Unix directory to use for the storage of logging files instead of the default depot's "log" directory.

- **#!clearquery_relocated_logging_dir_windows=**
  Optionally, set to inform ClearQuery the Windows directory to use for the storage of logging files instead of the default depot's "log" directory.

- **#!clearquery_relocated_results_dir_unix=**
  Optionally, set to inform ClearQuery the Unix directory to use for the storage of ClearQuery report html files instead of the default depot's "query_results" directory.

- **#!clearquery_relocated_results_dir_windows=**
  Optionally, set to inform ClearQuery the Windows directory to use for the storage of ClearQuery report html files instead of the default depot's "query_results" directory.

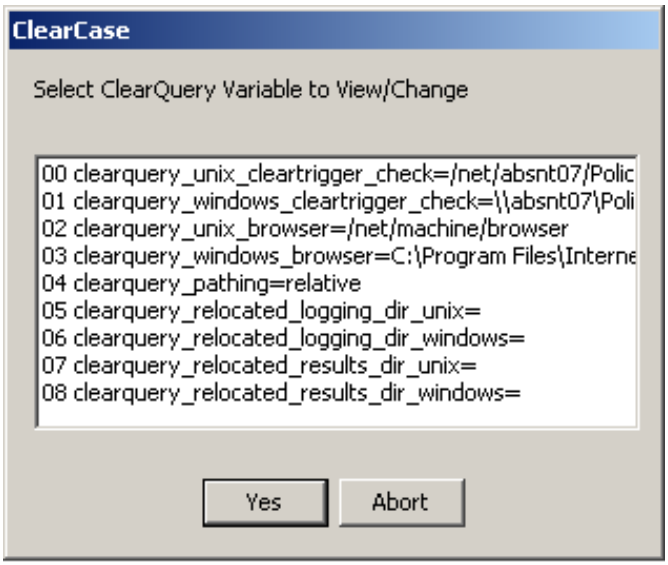
You can add/change these variables manually (some examples are in the distributed clearbits file) or use ClearQuery to configure the region.



To configure using ClearQuery itself, type **clearquery.exe F={path to clearbits_file}**.

Ex: **`clearquery.exe F=clearbits.txt`**

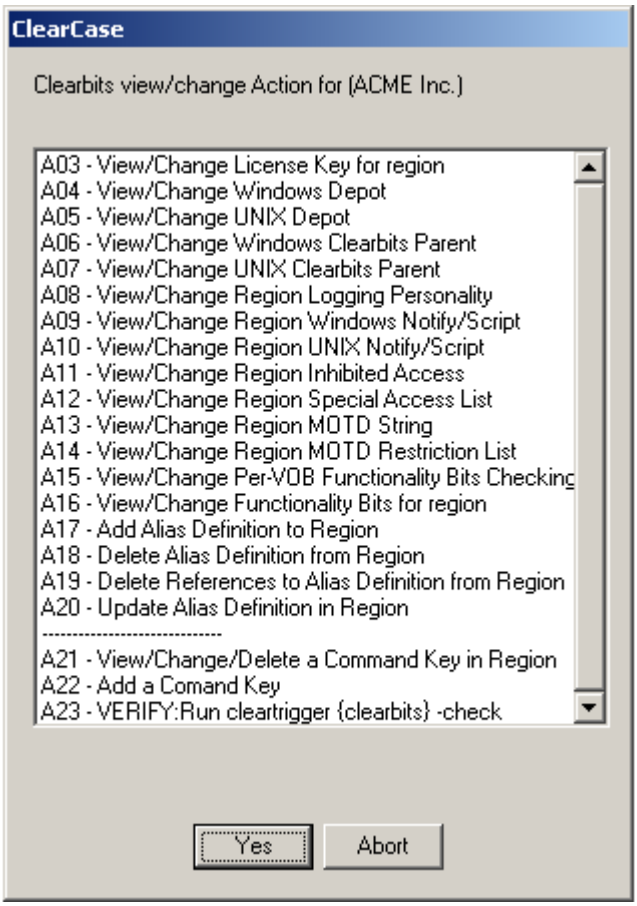
Then select A24 and "Yes" to configure.



This will display the current values and allowing one to change them. Any/all changes are written to the clearbits file.

Running ClearQuery – Modifying a clearbits file

Managing clearbits files is easier with ClearQuery. To modify a clearbits file, select A02 from the initial menu or pass A=2 to the command line to come directly to the menu entitled **"Clearbits view/change Action for (Region Name)"**. Select the field you want to change and let ClearQuery guide you through the add, change or delete. The **cleartrigger -check** can be invoked from here as well to ensure that that the clearbits file is correct



Use this menu to:

- Change region lists.
- Change region logging personality.
- Add/Change/Delete Notify Scripts.
- Add/Change/Delete ClearTrigger aliases.
- Add/Change/Delete MOTD and MOTD Restriction List.
- Toggle Functionality Bits.
- Add/Change/Delete command keys.
- Verify your clearbits file.

Select an "action" you are the current data (if any) and allowed to change it which performs the change and updates the clearbits file.

Running ClearQuery – Creating a report

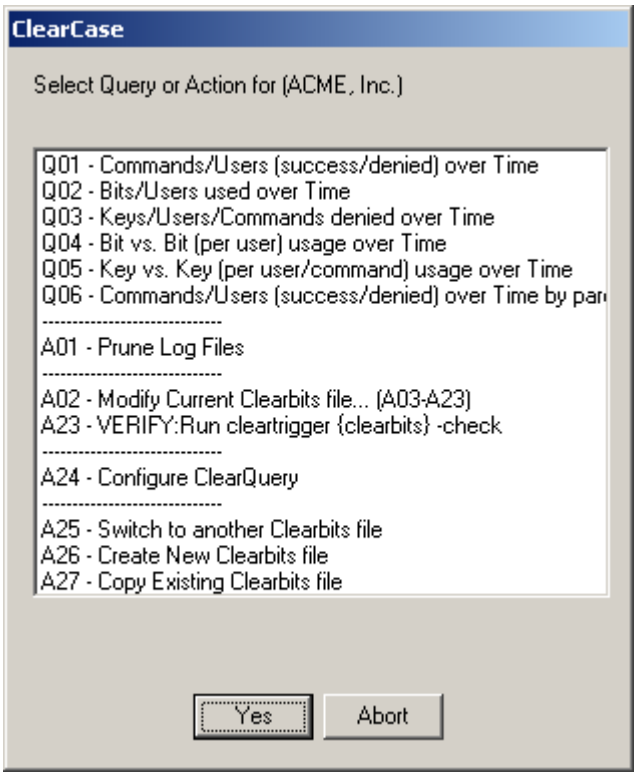
ClearQuery provides a **reporting facility** for ClearTrigger. One can track not only successful commands, but command denials as well (something ClearCase itself does not do). Use ClearQuery reports to determine:

- How often your policy prevented some unwanted action.
- How often command are being performed and by whom.
- If or when your developers actually "get" the policy.
- Which developers need "help".

When selecting a "query," the operator is asked for additional data unless that information was passed through the command line. The query that is built is run against the ClearTrigger log files (if ClearTrigger Logging is on) and results are displayed graphically via the selected web-browser.

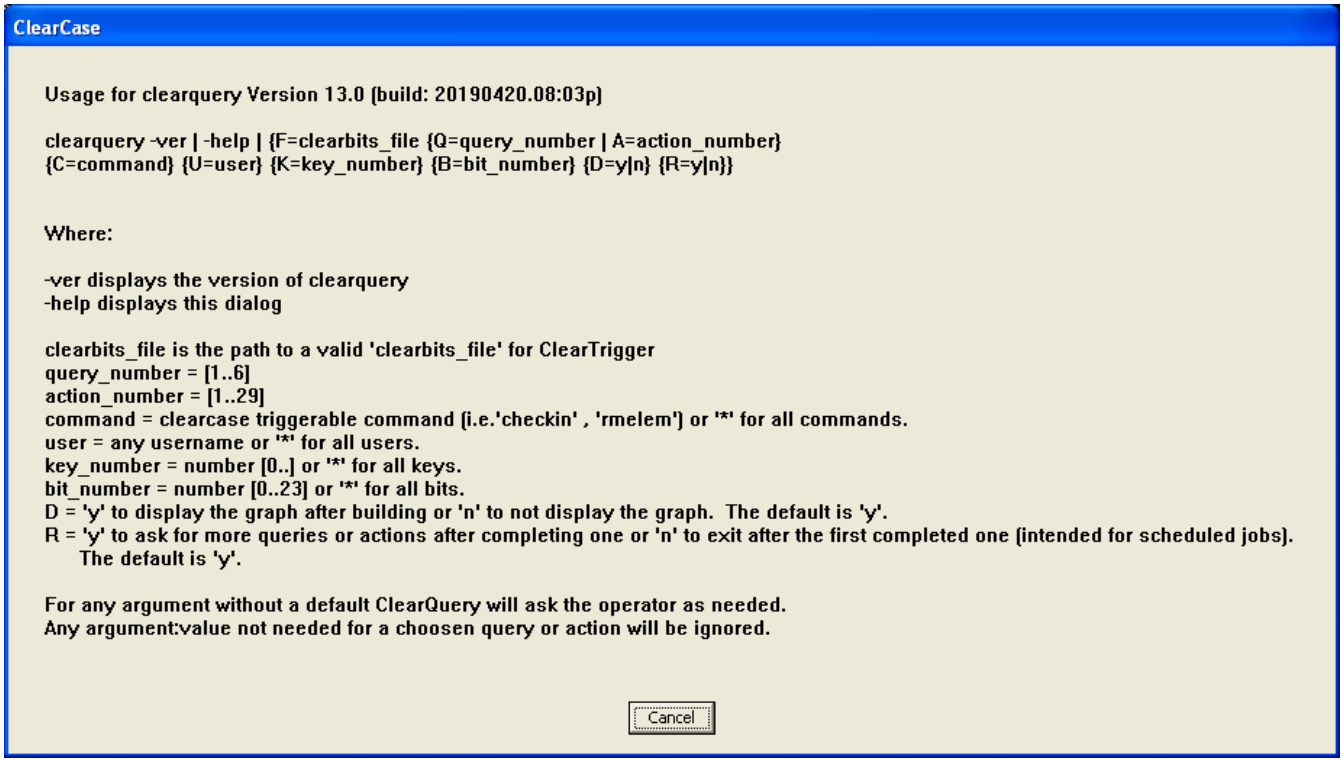
To view an example report, select a link from below:

- Q01
  - Q01 (all commands/all users)
  - Q01 (for single command/all users)
  - Q01 (for single command/single user)
- Q02
  - Q02 (all bits/all users)
  - Q02 (for single bit/all users)
  - Q02 (for single bit/single user)
- Q03
- Q04
- Q05
- Q06



Running ClearQuery- Command Line Interface (CLI)

The ClearQuery usage statement describing the CLI is below:



Environmental Variables and Defaults using ClearQuery

When executing ClearQuery, if any of these environmental variables are set they are used for prompt dialog defaults.

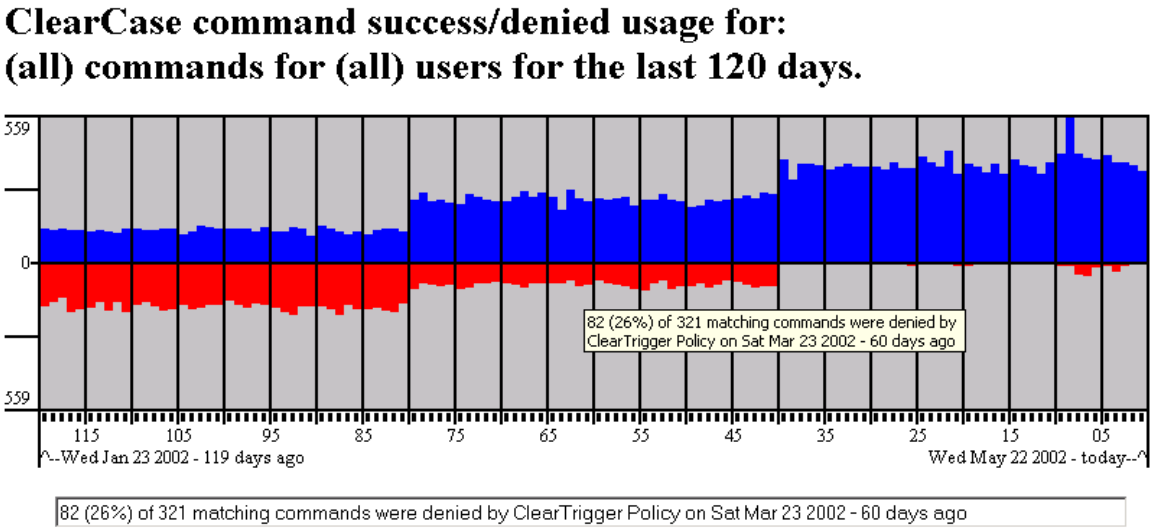
Variable Name	Description
ABS_CLEARQUERY_CLEARBITS_FILE	If set, this is used for the clearbits file – Allow a user to set this in their environment so that the path to the clearbits file does not have to be typed on each invocation to ClearQuery.
ABS_CLEARQUERY_VERBOSITY	If set to any value ClearQuery will confirm all actions that change the clearbits file before and after the change. When unset it just confirms before the change.
ABS_CLEARQUERY_BROWSER	Overrides the ClearQuery defined browser variable defined for the current OS. Overrides the **clearquery_unix_browser** or **clearquery_unix_browser** clearquery configuration values.

Sample Query Graphs

ClearQuery produces graphs that are full of information that can be used to make intelligent decisions. One can determine which policy is working and which is not, who is having trouble and with what commands, who needs training and who is trying to circumvent the company policy. ClearQuery is the only tool that provides reporting on not only command successes, but also command denials and trigger usage. We have selected "some" of the graphs you can get from ClearQuery so that you might get insight on how to interpret the graphs, though you will eventually come up with your own interpretations.

Q1 (all users – all commands)

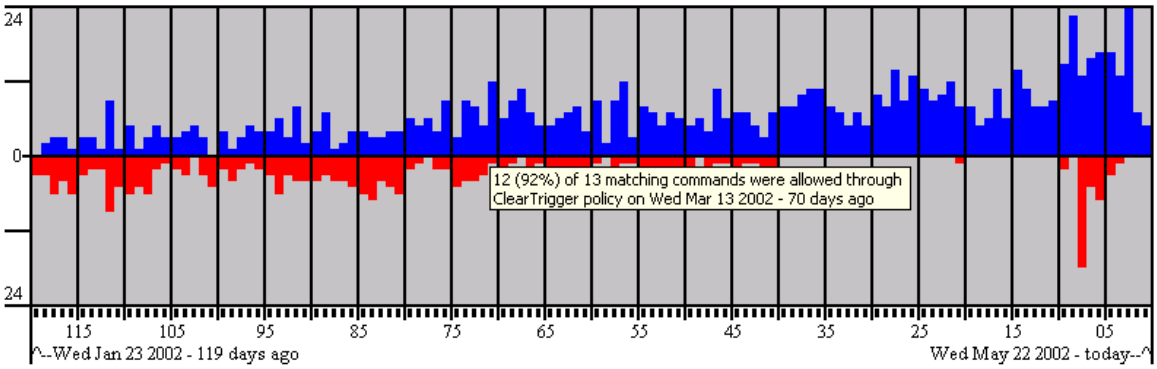
This type of graph is useful for generally finding out if your developers are "getting" your policy. The **Blue** portion depicts how many successful commands they are executing over time while the **Red** portion indicates how many denials they are getting from your policy. In theory as they "learn" what they are supposed to do, they are denied by your policy less as they just don't do what they are not supposed to do. At the same time they spend less time getting denied and have more time for successful commands, so productivity (amount of successes) increases. Every point on the graph displays information in the test block below the graph



Q1 (all users – a single command)

This type of graph is useful for generally finding out if your developers are having any difficulty with a particular command within your policy. The **Blue** portion depicts how many successful commands they are executing over time while the **Red** portion indicates how many denials they are getting from your policy. Every point on the graph displays information in the test block below the graph. In the Graph below the users seemed to fall in line with the company's rules around 40 days ago except for a lapse around 8 days ago (perhaps a new developer was added to the team).

ClearCase command success/denied usage for:
the (checkin) command for (all) users for the last 120 days.

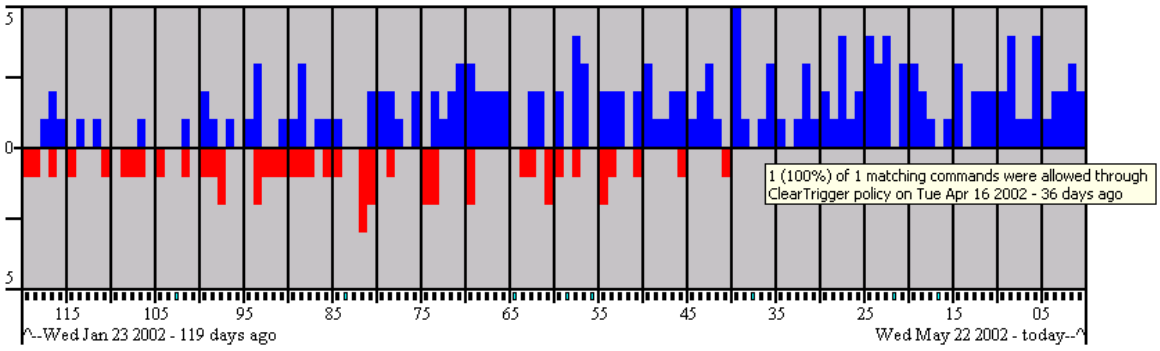


12 (92%) of 13 matching commands were allowed through ClearTrigger policy on Wed Mar 13 2002 - 70 days ago

Q1 (single user – a single command)

This type of graph is useful for finding out if a single developer is having difficulty with a particular command. In the Graph below the user (user1) seemed to fall in line with the company's rules around 40 days ago.

ClearCase command success/denied usage for:
the (checkin) command for the user (user1) for the last 120 days.

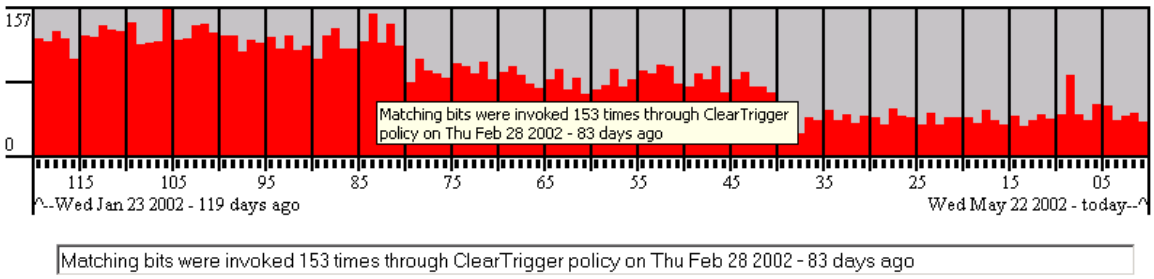


1 (100%) of 1 matching commands were allowed through ClearTrigger policy on Tue Apr 16 2002 - 36 days ago

Q2 (single bit – all users)

This type of graph is useful for finding out if any functionality bits turned on are doing their job. Again, the bit usage should generally be high in the beginning as the developers learn your policy and lower as time passes. This graph should not go to zero as some of the bits "provide" functionality (e.g. the remove empty branch).

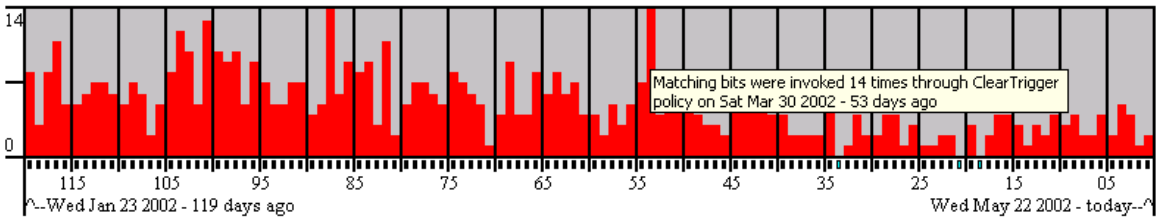
ClearCase Functionality Bit usage for:
(all) bits for (all) users for the last 120 days.



Matching bits were invoked 153 times through ClearTrigger policy on Thu Feb 28 2002 - 83 days ago

Q2 (single bit – all users)

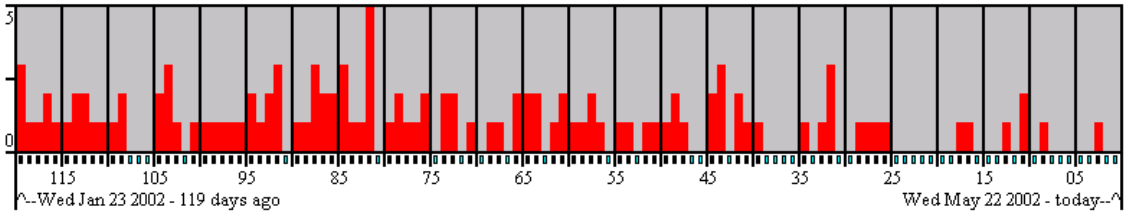
ClearCase Functionality Bit usage for:
bit #2 (BIT__EVIL_TWIN) for (all) users for the last 120 days.



Matching bits were invoked 14 times through ClearTrigger policy on Sat Mar 30 2002 - 53 days ago

Q2 (single bit – single user)

ClearCase Functionality Bit usage for:
bit #2 (BIT__EVIL_TWIN) for the user (user1) for the last 120 days.

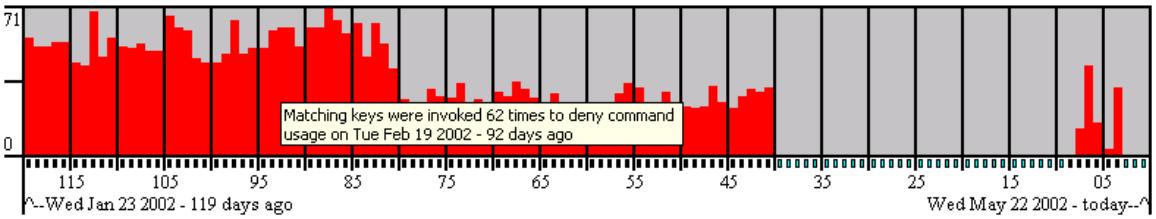


Matching bits were invoked 2 times through ClearTrigger policy on Sun Feb 03 2002 - 108 days ago

Q3 (all keys – all users – all commands)

This type of graph is useful for determining if the custom keys created are doing their job. Again the trend should yield to fewer denials over time.

ClearCase Command Key denials for:
(all) keys for (all) users for (all) commands for the last 120 days.



Matching keys were invoked 62 times to deny command usage on Tue Feb 19 2002 - 92 days ago

Note: When reporting on a ClearTrigger Lite region this graph displays CPD Denials instead of Command Keys.

Q4 (all bits – all users)

This is useful for determining bit-by-bit usage. From here one can see just how many empty branches or evil twins were avoided or how many times users were prevented from making a new element with spaces in the name.

ClearCase Functionality Bit usage for:
(all) bits for (all) users for the last 120 days.

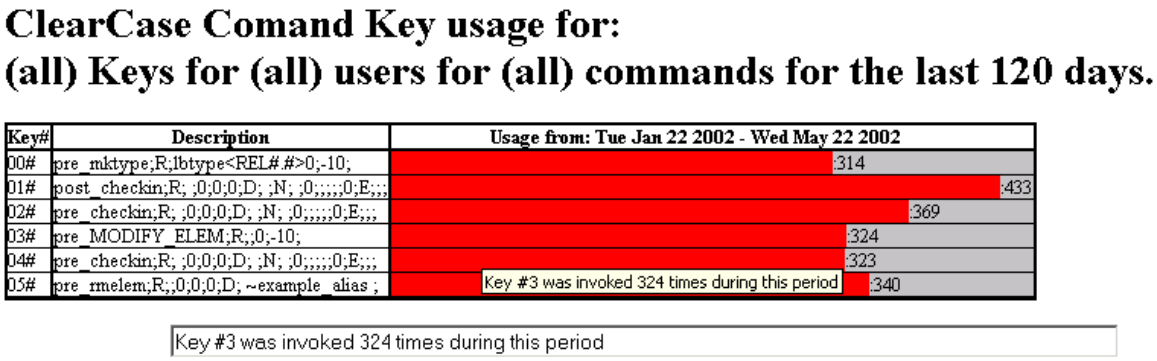
Bit#	Bit Description	Usage from: Thu Aug 15 2002 - Thu Dec 12 2002
00#	BIT__AUTO_BRANCH_REMOVAL	:498
01#	BIT__CHOWN_VOB_OWNER	:509
02#	BIT__EVIL_TWIN	:473
03#	BIT__ONE_CO_PER_BRANCH	:503
04#	BIT__CHECK_BRANCH_LABEL_NAMES	:467
05#	BIT__VOB_OWNER_CI_MAIN	:460
06#	BIT__RECURSIVE_CI_CO_UNCO	:555
07#	BIT__CHECK_MAGIC_PATH	:554
08#	BIT__NO_SPACES_IN_ELEMENT_NAMES	:510
09#	BIT__NO_MULTIPLE_MERGES	:542
10#	BIT__SMART_CHECKIN	:498
11#	BIT__CASE_CHECKING	:517
12#	BIT__ATOMIC_CHANGE	:499
13#	BIT__REQUIRE_EXTENSION	:513
14#	BIT__NOTIFY_PARENT_CHANGE_ON_CO	:485
15#	BIT__NOTIFY_PARENT_CHANGE_ON_CI	:473
16#	BIT__PREVENT_DATA_LOSS	:479
17#	BIT__AUTO_EXEC_BIT	:516
18#	BIT__PREVENT_CORE_ELEMENTS	
19#	BIT__AUTO_DIRECTORY_PROTECT	

The BIT__PREVENT_DATA_LOSS bit was invoked 479 times during this period

The BIT__PREVENT_DATA_LOSS bit was invoked 479 times during this period

Q5 (Key by Key – all users)

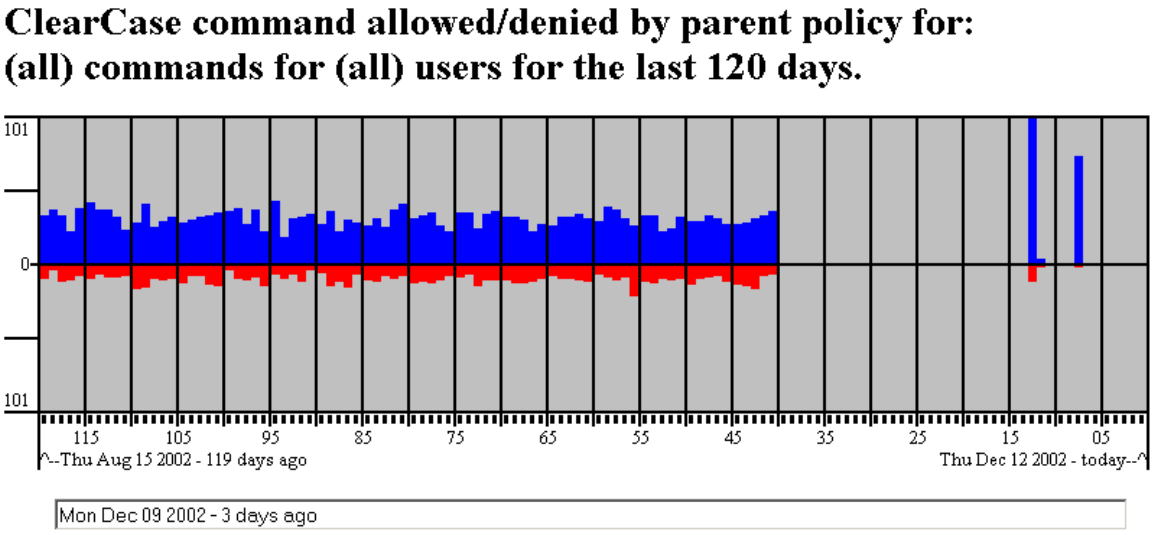
This graph is useful for determining how active keys are or how often any custom triggers that might have been created are called.



Note: When reporting on a ClearTrigger Lite region this graph displays CPDs instead of Command Keys.

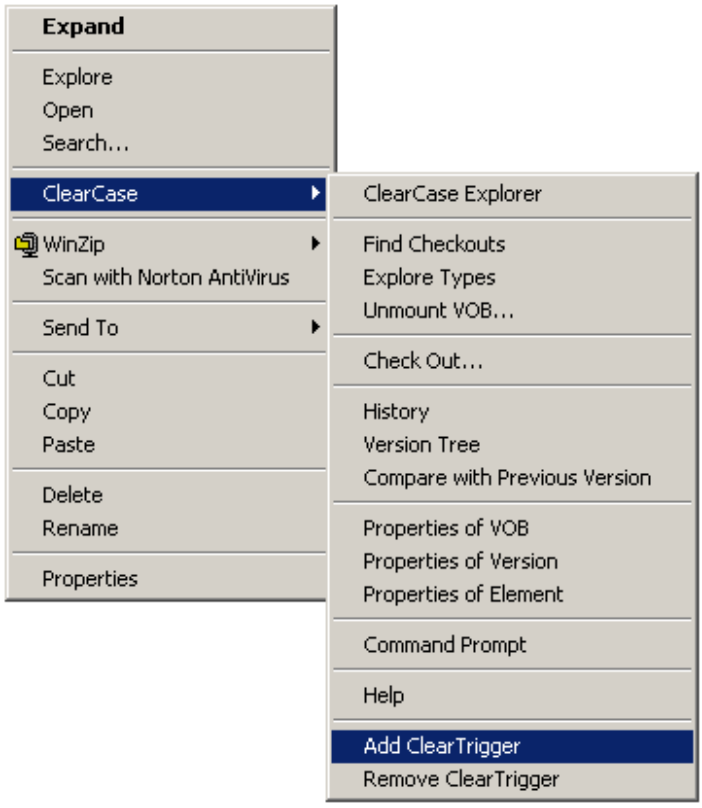
Q6 (parent policy – all users – all commands)

This graph is useful for determining how often activity is being allowed or disallowed by the parent policy defined in any [parent clearbits file].



Pull-down Menu Integrations

It is easy to integrate the application of **ClearTrigger** to the Pull-down menus of **Windows Explorer** and **ClearCase Explorer** such that one can apply or remove ClearTrigger when VOBs are selected. Once completed, the administrator can apply ClearTrigger without having to enter paths to cleartrigger or the clearbits file. This is a customization that the ClearCase/ClearTrigger administrator might want for ***their*** desktop. The **clearmenuadmin** application distributed with ClearCase is helpful for creating these integrations.

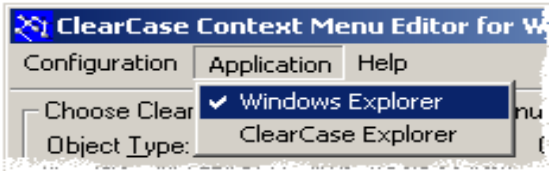


The steps to create the menu integrations are as follows.

- **Start the clearmenuadmin executable** from the command line in a DOS window. The executable is already in your path if ClearCase is installed on the machine.

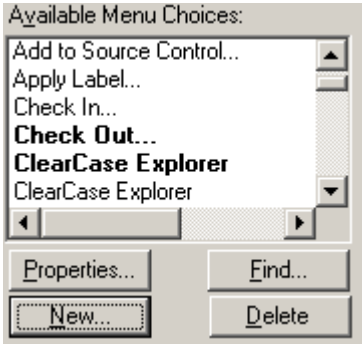
- Select the **Windows Explorer** or the **ClearCase Explorer** Application.

  Once selected, the instructions are the same for either application.



You should perform the remaining steps for each application.

- **Add new menu items in the "Available Menu Choices" section**.
  - Select the new button in the "Available Menu Choices" section. You will add two (2) new Menu Choices.



- Create an "**Add ClearTrigger**" Menu Choice by completing the resulting dialog in a fashion similar to the one in **Fig. A**.

- Create a "**Remove ClearTrigger**" Menu Choice by completing the resulting dialog in a fashion similar to the one in **Fig. B**.

Fig. A

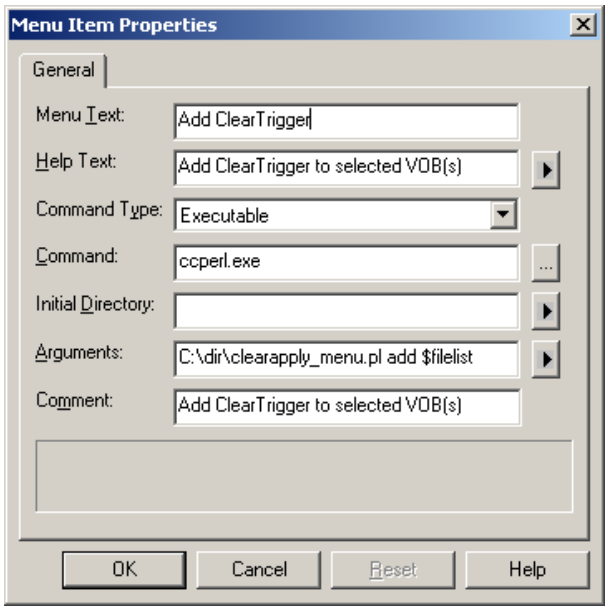
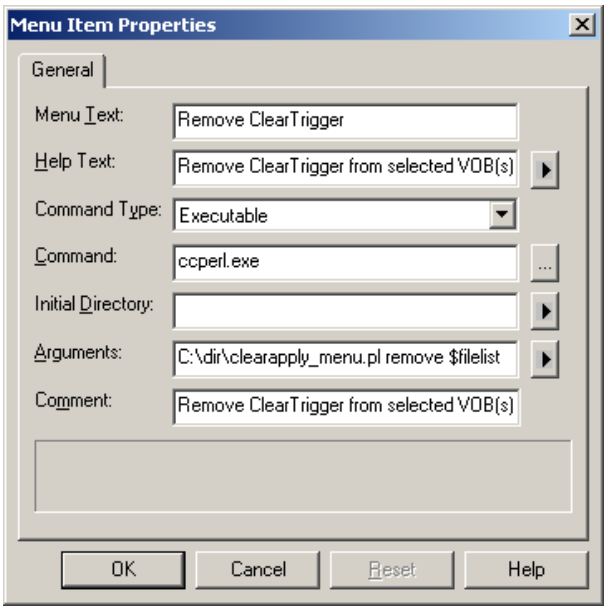


Fig B.

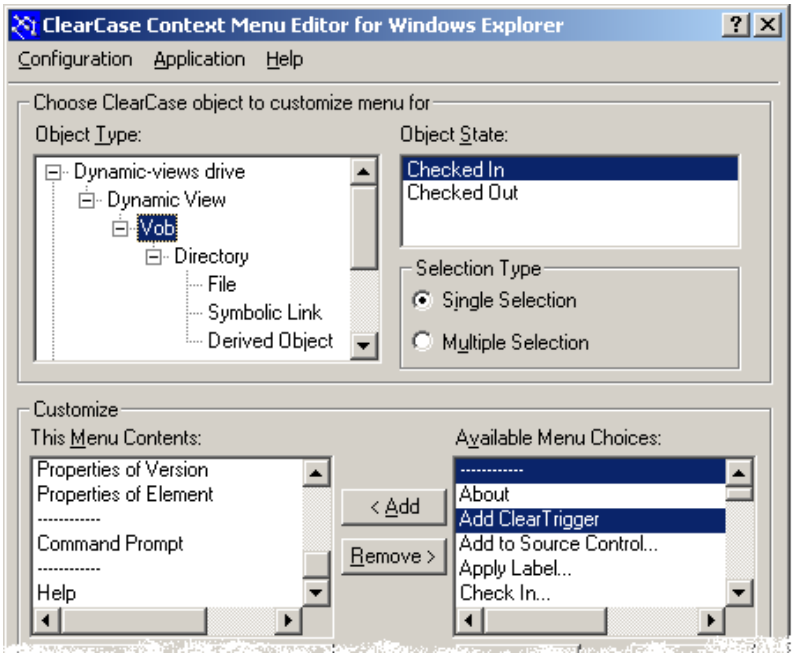


❖ **Add the new menu items.**

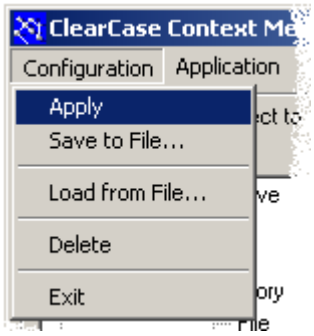
- Navigate to the "**Vob**" item in the "Object Type:" frame, then select the combination "**Checked In**" and "**Single Selection**"

- Select a **separator** item, your recently added "**Add ClearTrigger**" item and the "**Remove ClearTrigger**" item (the list is alphabetized so "Remove ClearTrigger" appears lower in the list).

- Select the "**<Add**" button to add the menu items.



- Repeat these steps for the "**Checked In/Multiple Selection**", "**Checked Out/Multiple Selection**", "**Checked Out/Single Selection**" combinations.



- **Apply the new menu configuration to your desktop**. You could also "Save to File" to save a registry file that can be used again later on your machine or another machine to create the menu integration.

- **Remember to repeat all of these steps for the "ClearCase Explorer" application**.

- **Place the menu script in the anticipated location**. In the examples we referred to the "**c:\dir\clearapply_menu.pl**" script, make sure that script exist. The code for this script exists as **Appendix A**. Modify the code so that the paths are appropriate for your organization.

- **Now you should be able to apply ClearTrigger to VOBs from the Pull-down Menus**. You will have to restart any Windows Explorer or ClearCase Explorer applications for the setting to take affect.

Appendix A (Clearapply_menu.pl)

```
# clearapply_menu.pl
#
# Call clearapply.exe with selected VOB(s).
#
# NOTE: A copy of this code is available from ABS at:
#       https://www.abs-consulting.com/_faq/scriptS/clearapply_menu.pl.txt
#
# Author: Charles W. Clarke III (ABS)
# email:  charles@abs-consulting.com
# URL:    https://www.abs-consulting.com
# Date:   Apr. 08, 2001
############################################################
# History: 04/08/01 : Created for A Better Solution, Inc.
############################################################

############################
# Site-specific variables #
############################

$clearapply = "C:\\ccadmin_scripts\\clearapply.exe";           # Local to machine                       #
$ct_unix = "/net/machine/share/depot/bin/cleartrigger";        # Network available cleartrigger:Unix    #
$cb_unix = "/net/machine/share/depot/clearbits.txt";           # Network available clearbits:Unix       #
$ct_win = "\\\\machine\\share\\depot\\bin\\cleartrigger.exe";  # Network available cleartrigger:Windows #
$cb_win = "\\\\machine\\share\\depot\\clearbits.txt";          # Network available clearbits:Windows    #

########
# Main #
########

if (($#ARGV < 1) || (($ARGV[0] ne "add") && ($ARGV[0] ne "remove")))
{
   $USAGE="\"USAGE ccperl.exe $0 add | remove VOB_directory(s)\"";

   `clearprompt yes_no -mask abort -default abort -pre -prompt $USAGE`;

   exit 1;
}

############################################################
# For every VOB in the list - add or remove ClearTrigger #
############################################################

if ($ARGV[0] eq "add")
{
   for ($i=1; $i <= $#ARGV; $i++)
   {
      @PARTS = split (/\\/, $ARGV[$i]);
      printf ("Applying ClearTrigger to VOB \\$PARTS[$#PARTS] ...\n");

      $COMMAND="$clearapply $ct_unix $cb_unix $ct_win $cb_win \\$PARTS[$#PARTS]";
      system ($COMMAND);
   }
}
else
{
   for ($i=1; $i <= $#ARGV; $i++)
   {
      @PARTS = split (/\\/, $ARGV[$i]);
      printf ("Removing ClearTrigger from VOB \\$PARTS[$#PARTS] ...\n");

      $COMMAND="$clearapply remove \\$PARTS[$#PARTS]";
      system ($COMMAND);
   }
}
```