Rational Software Corporation®

RATIONAL® CLEARCASE®

COMMAND REFERENCE (A–L)

UNIX/WINDOWS EDITION

VERSION: 2002.05.00 AND LATER

PART NUMBER: 800-025071-000

Rational®
the software development company

Command Reference
Document Number 800-025071-000 October 2001
Rational Software Corporation 20 Maguire Road Lexington, Massachusetts 02421

Technical Acknowledgments

This software and documentation is based in part on BSD Networking Software Release 2, licensed from the Regents of the University of California. We acknowledge the role of the Computer Systems Research Group and the Electrical Engineering and Computer Sciences Department of the University of California at Berkeley and the Other Contributors in its development.

This product includes software developed by Greg Stein <gstein@lyra.org> for use in the mod_dav module for Apache (http://www.webdav.org/mod_dav/).

Contents

Reference Pages

Tables

Preface

Rational ClearCase is a comprehensive software configuration management system. It manages multiple variants of evolving software systems, tracks the versions that were used in software builds, performs builds of individual programs or entire releases according to user-defined version specifications, and enforces site-specific development policies.

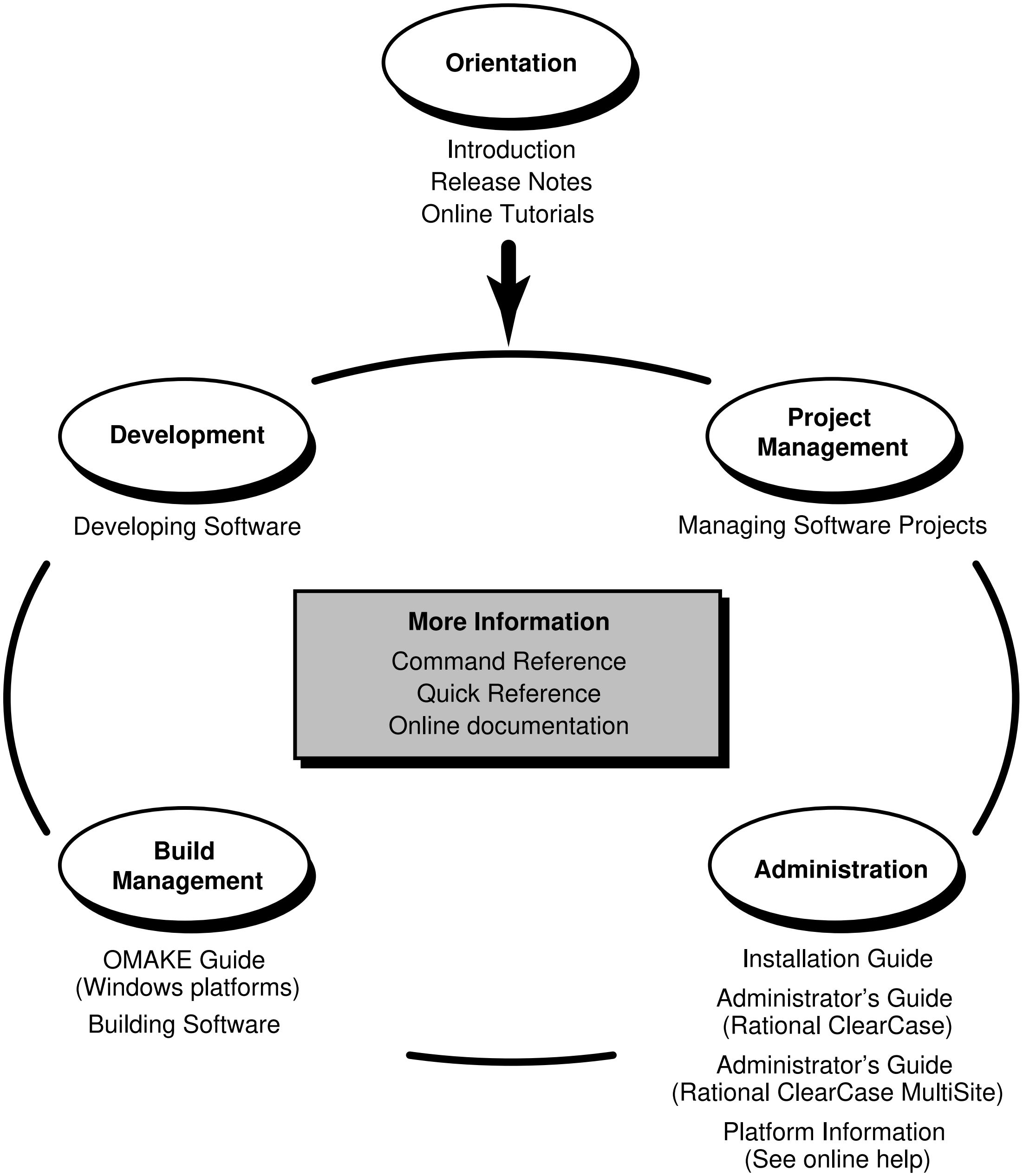
Rational ClearCase LT offers capabilities like those of ClearCase, but for the smaller software development group. Rational ClearCase Attache provides a ClearCase client solution for Microsoft Windows users (see the *ClearCase Attache Manual*). Rational ClearCase MultiSite is a layered product option for ClearCase. It supports parallel software development and software reuse across project teams that are distributed geographically.

About This Manual

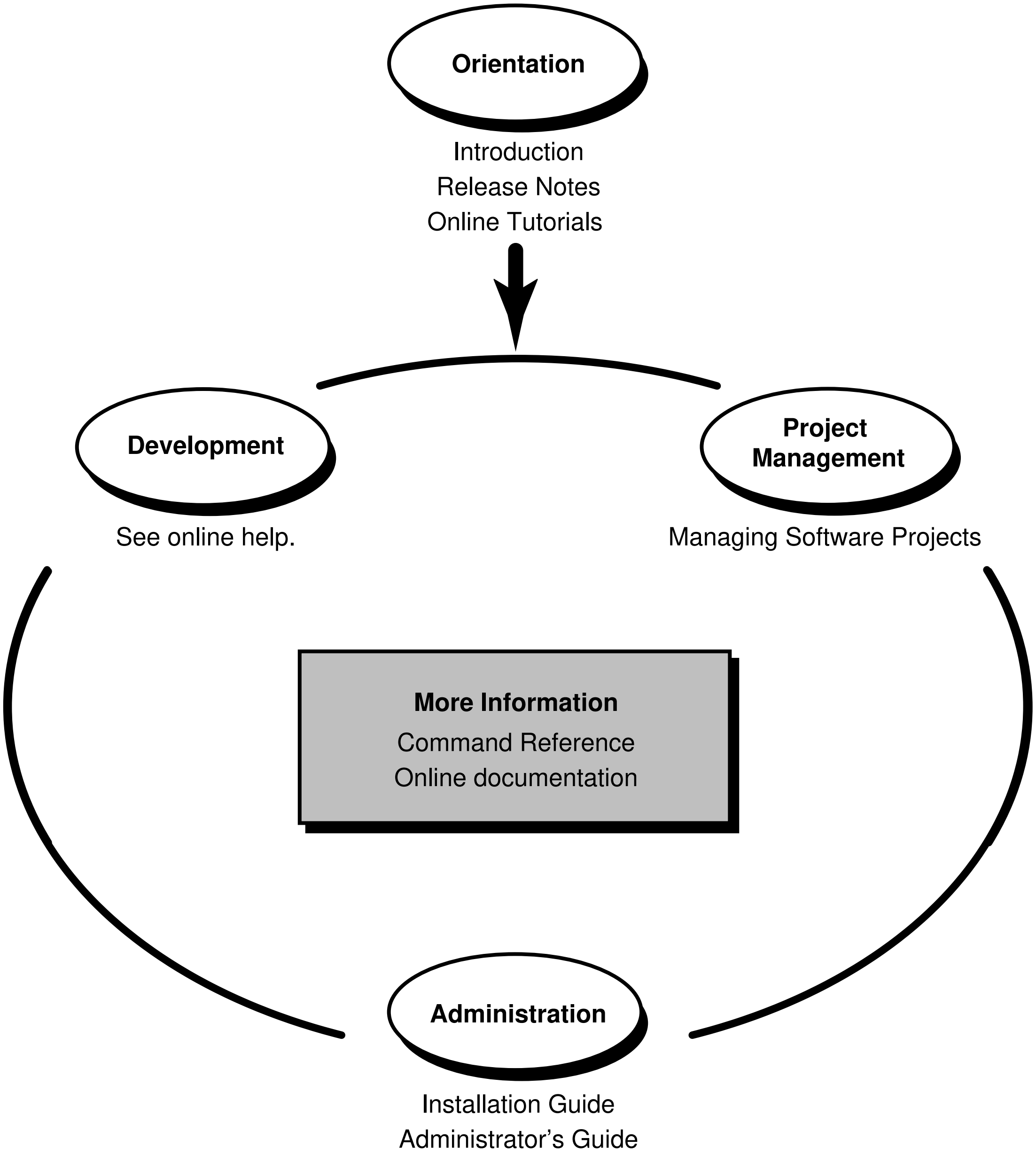
This manual includes detailed reference information about ClearCase, ClearCase LT, Attache, and MultiSite on UNIX and Windows. It is not intended to be a learning tool—it assumes you have already learned about these products through other means. For a general orientation to the contents of this manual, read the **intro** reference page.

The reference pages are in alphabetical order in two volumes. Each reference page has an Applicability section that lists the products and operating system platforms that support the command described therein. Within each reference page, product-specific information is annotated "ClearCase only," "ClearCase LT only," and so on. In this context, the term *ClearCase* always refers only to ClearCase, not to ClearCase LT, Attache, MultiSite, nor to the ClearCase Product Family (CPF) in general.

ClearCase Documentation Roadmap



ClearCase LT Documentation Roadmap



Typographical Conventions

This manual uses the following typographical conventions:

- *ccase-home-dir* represents the directory into which the ClearCase Product Family has been installed. By default, this directory is **/usr/atria** on UNIX and **C:\Program Files\Rational\ClearCase** on Windows.

- *attache-home-dir* represents the directory into which ClearCase Attache has been installed. By default, this directory is **C:\Program Files\Rational\Attache**, except on Windows 3.*x*, where it is **C:\RATIONAL\ATTACHE**.

- **Bold** is used for names the user can enter; for example, all command names, file names, and branch names.

- *Italic* is used for variables, document titles, glossary terms, and emphasis.

- `A monospaced font` is used for examples. Where user input needs to be distinguished from program output, **bold** is used for user input.

- Nonprinting characters are in small caps and appear as follows: **<EOF>**, **<NL>**.

- Key names and key combinations are capitalized and appear as follows: SHIFT, CTRL+G.

- [ ] Brackets enclose optional items in format and syntax descriptions.

- { } Braces enclose a list from which you must choose an item in format and syntax descriptions.

- | A vertical bar separates items in a list of choices.

- ... In a syntax description, an ellipsis indicates you can repeat the preceding item or line one or more times. Otherwise, it can indicate omitted information.

  NOTE: In certain contexts, ClearCase recognizes "**...**" within a pathname as a wildcard, similar to "*" or "?". See the **wildcards_ccase** reference page for more information.

- If a command or option name has a short form, a "medial dot" (·) character indicates the shortest legal abbreviation. For example:

  lsc·heckout

  This means that you can truncate the command name to **lsc** or any of its intermediate spellings (**lsch**, **lsche**, **lschec**, and so on).

Online Documentation

The ClearCase Product Family (CPF) graphical interfaces include an online help system.

There are three ways to access the online help system: the **Help** menu, the **Help** button, or the F1 key. **Help > Contents** provides access to the complete set of online documentation. For help on a particular context, press F1. Use the **Help** button on various dialog boxes to get information specific to that dialog box.

CPF products also provide access to full reference pages (detailed descriptions of commands, utilities, and data structures) using the **man** command. Without any argument **man** displays the overview reference page for the command line interface. For information about using a particular command, specify the command name as an argument.

Examples:

cleartool man	*(display the cleartool overview page)*
multitool man mkreplica	*(display the multitool mkreplica reference page)*
`attache-workspace>` **man checkout**	*(display the Attache checkout reference page)*

CPF products provide access to syntax for individual commands. The **–help** command option displays individual subcommand syntax. For example:

cleartool uncheckout –help
```
Usage: uncheckout | unco [-keep | -rm] [-cact | -cwork ] pname ...
```

Without any argument, **cleartool help** displays the syntax for all **cleartool** commands.

On UNIX, the **apropos** command displays command summary information and entries from the ClearCase glossary. See the **apropos** reference page for more information.

Additionally, the online tutorials provide important information on setting up a user's environment, along with a step-by-step tour through each product's most important features.

Technical Support

If you have any problems with the software or documentation, please contact Rational Technical Support via telephone, fax, or electronic mail as described below. For information regarding support hours, languages spoken, or other support information, click the **Technical Support** link on the Rational Web site at **www.rational.com**.

Your Location	Telephone	Facsimile	Electronic Mail
North America	800-433-5444 toll free or 408-863-4000 Cupertino, CA	408-863-4194 Cupertino, CA 781-676-2460 Lexington, MA	**support@rational.com**
Europe, Middle East, and Africa	+31-(0)20-4546-200 Netherlands	+31-(0)20-4546-201 Netherlands	**support@europe.rational.com**
Asia Pacific	61-2-9419-0111 Australia	61-2-9419-0123 Australia	**support@apac.rational.com**

intro

Introduction to ClearCase, ClearCase LT, ClearCase MultiSite, and Attache reference pages

APPLICABILITY

Product	Command Type
ClearCase	general information
ClearCase LT	general information
Attache	general information
MultiSite	general information

Platform
UNIX
Windows

DESCRIPTION

This reference page provides a general orientation to the contents of this manual.

cleartool Subcommands

The majority of ClearCase and ClearCase LT commands are **cleartool** subcommands; see the **cleartool** reference page.

MultiSite Reference Pages

This manual includes only those MultiSite commands that you can invoke through **cleartool** as well as through **multitool**:

- **chmaster**
- **lsmaster**
- **lsreplica**
- **reqmaster**

The *Administrator's Guide* for Rational ClearCase MultiSite contains reference pages for all MultiSite commands.

Information of General Interest

The following reference pages are of interest to all users, as the information they contain is useful in various contexts:

cleartool	Object selector syntax, escaping characters, quoting command arguments, object naming rules and restrictions
env_ccase	Environment variables
events_ccase	VOB event records
fmt_ccase	Command output formatting directives
pathnames_ccase	Pathname interpretation and resolution
permissions	Summary of permissions checking for **cleartool** commands
query_language	VOB query language
version_selector	Version selector syntax
wildcards_ccase	Wildcards

Object-Type Independent Commands

The majority of commands manipulate specific object types, and it is easy to think of the commands in those narrow terms. However, it is useful to remember that some commands can operate on various object types:

checkvob	Checks hyperlinks to objects
chevent	Changes object event records
chmaster	Transfers mastership of objects to other VOB replicas
cleardescribe	Describes objects
clearhistory	Lists the histories (event records) of objects
describe	Describes objects
lock	Locks objects
lshistory	Lists the histories (event records) of objects
lslock	Lists locks on objects
mkattr	Attaches attributes to objects
protect	Changes ownership or permissions of objects
rename	Renames objects
rmattr	Removes attributes from objects
unlock	Unlocks objects

UNIX Graphical User Interfaces

The reference pages listed below describe how to start the UNIX graphical user interfaces. Also see the **schemes** reference page, which describes X Window System resources for these GUIs.

cleardescribe	Displays objects
cleardiffbl	Compares UCM baselines
cleargetlog	Starts the graphical log browser
clearhistory	Displays histories of VOB objects
clearjoinproj	Enables you to join UCM projects
clearmrgman	Merges element versions
clearprojexp	UCM project browser
clearviewupdate	Updates snapshot views
clearvobadmin	Starts the graphical VOB admin browser
hyperhelp	GUI to the documentation
xclearcase	Primary GUI
xcleardiff	Compares/merges element versions

UCM Commands

Following is a summary of UCM commands:

chactivity	Modifies an activity
chbl	Changes a baseline
chfolder	Modifies a folder
chproject	Modifies a project
chstream	Modifies a stream
cleardiffbl	Starts the diffbl browser
clearjoinproj	Starts the Join Project Wizard
clearprojexp	Starts the Project Explorer
deliver	Delivers development stream changes to the integration stream
diffbl	Compares the contents of baselines or streams
lsactivity	Lists information about activities
lsbl	Lists information about baselines
lscomp	Lists information about components
lsfolder	Lists information about folders
lsproject	Lists information about projects
lsstream	Lists information about streams
mkactivity	Creates an activity
mkbl	Creates baselines
mkcomp	Creates a component
mkfolder	Creates folders
mkproject	Creates a project
mkstream	Creates a stream
rebase	Changes the configuration of a stream
rmactivity	Removes an activity
rmbl	Removes a baseline
rmcomp	Removes a component
rmfolder	Removes a folder
rmproject	Removes a project
rmstream	Removes a stream
setactivity	Specifies the current activity for your view
setplevel	Changes the list of promotion levels in a project VOB

Reference Pages by User Role

This section lists reference pages by user role: developer, software builder, project manager, and administrator. For more information, see *Developing Software*, *Building Software*, *Managing Software Projects*, and one or more of the following titles, as appropriate for the products you use:

- *Administrator's Guide* for Rational ClearCase

- *Administrator's Guide* for Rational ClearCase MultiSite
- *Administrator's Guide* for Rational ClearCase LT

Table 1 Developer Reference Pages

apropos	**comments**	**lshistory**	**reserve**
catcs	**config_spec**	**lslock**	**rmattr**
cc.icon	**cptype**	**lsprivate**	**rmfolder**
cc.magic	**deliver**	**lsproject**	**rmhlink**
cd	**describe**	**lsstream**	**rmlabel**
chactivity	**diff**	**lsview**	**rmmerge**
chbl	**edcs**	**lsvtree**	**rmname**
checkin	**endview**	**man**	**rmview**
checkout	**env_ccase**	**merge**	**schemes**
chfolder	**file**	**mkattr**	**setactivity**
chproject	**find**	**mkdir**	**setcs**
chstream	**fmt_ccase**	**mkelem**	**setview**
chview	**get**	**mkhlink**	**shell**
clearbug	**getcache**	**mklabel**	**startview**
cleardescribe	**help**	**mkview**	**uncheckout**
cleardiff	**hyperhelp**	**mv**	**unreserve**
clearhistory	**ln**	**pathnames_ccase**	**update**
clearjoinproj	**ls**	**profile_ccase**	**version_selector**
clearmrgman	**lsactivity**	**pwd**	**wildcards_ccase**
clearprojexp	**lsbl**	**pwv**	**xclearcase**
clearprompt	**lscheckout**	**query_language**	**xcleardiff**
cleartool	**lscomp**	**rebase**	**xmldiffmrg**
clearviewupdate	**lsfolder**	**recoverview**	

Table 2 Build-Related Reference Pages

catcr	**diffcr**	**makefile_ccase**	**makefile_sun**
clearaudit	**lsdo**	**makefile_gnu**	**omake**
clearmake	**make**	**makefile_pmake**	**rmdo**
clearmake.options	**makefile_aix**	**makefile_smake**	**winkin**

Table 3 Project Manager Reference Pages

annotate	**mkbl**	**mkproject**	**rmelem**
chevent	**mkbranch**	**mkstream**	**rmproject**
chtype	**mkbrtype**	**mktrigger**	**rmstream**
cleardiffbl	**mkcomp**	**mktrtype**	**rmtrigger**
diffbl	**mkeltype**	**rmactivity**	**rmtype**
findmerge	**mkfolder**	**rmbl**	**rmver**
mkactivity	**mkhltype**	**rmbranch**	**setplevel**
mkattype	**mklbtype**	**rmcomp**	**type_manager**

Table 4 Administrator Reference Pages

checkvob	**export_mvfs**	**msdostext_mode**	**rmregion**
chflevel	**exports_ccase**	**mvfscache**	**rmstgloc**
chpool	**getlog**	**mvfslog**	**rmtag**
clearexport_ccase	**hostinfo**	**mvfsstat**	**rmvob**
clearexport_cvs	**import**	**mvfsstorage**	**schedule**
clearexport_pvcs	**init_ccase**	**mvfstime**	**scrubber**
clearexport_rcs	**lock**	**mvfsversion**	**setcache**
clearexport_sccs	**lsclients**	**permissions**	**setsite**
clearexport_ssafe	**lspool**	**promote_server**	**snapshot.conf**
clearfsimport	**lsregion**	**protect**	**softbench_ccase**
cleargetlog	**lssite**	**protectvob**	**space**
clearimport	**lsstgloc**	**reformatview**	**umount**
clearlicense	**lstype**	**reformatvob**	**unlock**
clearvobadmin	**lsvob**	**register**	**unregister**
config_ccase	**mkpool**	**relocate**	**view_scrubber**
credmap	**mkregion**	**rename**	**vob_sidwalk**
creds	**mkstgloc**	**rgy_backup**	**vob_restore**
CtCmd	**mktag**	**rgy_check**	**vob_scrubber**
dospace	**mkvob**	**rgy_passwd**	**vob_snapshot**
events_ccase	**mount**	**rgy_switchover**	**vob_snapshot_setup**
	mount_ccase	**rmpool**	

Attache Reference Pages

The reference pages listed in Table 5 are applicable to Attache only.

Table 5 Attache Reference Pages

att_clnt	**attache_graphical_interface**	**mvws**	**setws**
attache	**lslocal**	**put**	**wildcards**
attcmd	**lsws**	**quit**	**ws_helper**
attache_command_line_interface	**mkws**	**rmws**	**wshell**

annotate

Annotates lines of text file / time stamps, user names, and so on

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

ann·otate [**–a·ll** | **–rm**] [**–nco**] [**–out** *pname*]
[**–s·hort** | **–l·ong** | **–fmt** *format*[**,***hdr-format*[**,***elide-format*]]]
[**–rmf·mt** *rm-format*] [**–nhe·ader**]
[**–nda·ta** | **–f·orce**] *pname* ...

DESCRIPTION

The **annotate** command lists the contents of a version, annotating each line to indicate when, and in which version, the line was added. You can customize the annotations using the **–fmt** option, which is described in the **fmt_ccase** reference page. By default, **annotate** writes its output to a file with the **.ann** extension.

Line of Descent

Each version has a line of descent, a sequence of ancestor versions going all the way back to the mainline's version 0. The default listing has a header section that includes the event records of all the versions in the line of descent of the annotated version.

Type Manager Interface

The **annotate** command extracts information from the element's versions. To do so, it invokes the **annotate** method of the element's type manager. Only the **text_file_delta** and **z_text_file_delta** type managers (which correspond to the predefined element types **text_file** and

compressed_text_file) include an **annotate** method. You must use the **–ndata** option when annotating versions of other element types.

REPORT FORMAT

The default report format includes the following components:

- **Element pathname —** Shows the path of the element being annotated.
- **Heading section —** Lists the event record for each version along the line of descent, in standard **cleartool lshistory** format.
- **Text line annotations —** Includes a bar graph indicating how long ago the line first appeared in an ancestor version, along with that version's time stamp, creator, and version-ID.
- **Elision strings —** Replace text line annotations that would duplicate the annotation on the preceding line. An elision string includes the bar graph and a single dot (`.`) character.
- **Source lines from the specified version —** Any TAB characters in source lines are expanded according to the value of environment variable **CLEARCASE_TAB_SIZE** (default: 8).

If you use the **–rm** or **–all** option, the report also includes deletion annotations. These appear on text lines that are not in the annotated version, but do exist in some other version of the element.

RESTRICTIONS

See the *Type Manager Interface* section.

OPTIONS AND ARGUMENTS

INCLUDING OTHER TEXT LINES. *Default:* The listing includes only text lines that are present in the specified version.

–a·ll

Expands the listing to include all text lines that occurred in *any* version of the element, including lines in versions that are not along the line of descent. (Lines from versions outside the line of descent are annotated as `UNRELATED`; this annotation appears in the same column used to annotate deletion lines.)

–rm

Also includes removed lines—text lines that were present in one or more versions along the line of descent, but do not appear in the specified version. See also the **–rmfmt** option.

HANDLING OF CHECKED-OUT VERSIONS. *Default:* An error occurs if you specify a checked-out version. (The type manager can annotate checked-in versions only.)

–nco

If you specify a checked-out version, **annotate** uses the version from which it was checked out.

DESTINATION OF LISTING. *Default:* Command output is sent to the file *input-file***.ann**.

–out *output-pname*

If *output-pname* is a file name, redirects command output to the specified file (overwriting the file if it already exists). If *output-pname* is a single hyphen character (**–out –**), sends command output to **stdout**. If *output-pname* is a directory, places command output for each annotated version in a file within that directory (which must already exist).

If you use this option when annotating more than one version, *output-pname* must be a directory.

REPORT FORMAT. *Default:* The source file is listed as described in *REPORT FORMAT on page 9*.

–s·hort

Uses predefined annotation format strings that yield an abbreviated report.

–l·ong

Uses predefined annotation format strings that yield a verbose report.

–fmt *format*[,*hdr-format*[,*elide-format*]]

Specifies a display format for primary annotations, and optionally, for the header section and/or elision strings. Format strings must be enclosed in quotes. The default *format* is **"%BAd %Sd %-8.8u %-16.16Vn | "** .

Use a hyphen (-) to designate a default format string. For example, to supply a *hdr-format*, but not a primary annotation format, use the construction **–fmt** –, *hdr-format*. It is usually desirable to terminate the *hdr-format* with a <NL> character, by using **\n**.

If you omit the *elide-format*, it is computed based on the primary line-by-line annotation: all characters except <TAB> and the vertical bar (|) in the primary annotation are replaced by <SPACE>, and the middle character, if it is a <SPACE>, is replaced by a period (.).

In general, it is simpler to use fixed-width fields, not <TAB>-character specifiers (**\t**), to create aligned columns of annotations. See the **fmt_ccase** reference page for more details on composing format strings.

–rmf·mt *rm-format*

Specifies a format for deletion annotations (see also **–rm** and **–all**). The default format is **"DEL %Sd %-8.8u | "** .

–f·orce

Displays each text-line's annotation, even if it duplicated the previous line's annotation. This option suppresses use of elision strings.

PARTIAL REPORTS. *Default:* The report includes both a header section and the annotated text lines.

–nhe·ader
Suppresses the header section; the report consists of the annotated text lines only.

–nda·ta
Suppresses the annotated text lines; the report consists of the header section only.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Annotate a source file, using the short format.

  cmd-context **annotate -short msg.c**

  ```
  Annotated result written to "msg.c.ann".
  ```

  % **cat msg.c.ann**

```
/vobs/src/msg.c
---------------
24-Apr-99 anne /main/rel2_bugfix/9
12-Mar-99 ravi /main/rel2_bugfix/8
.
.
.
23-Apr-99 rks /main/48 (REL2)
20-Apr-99 spc /main/47
.
.
.
-------------------------------------------------
-------------------------------------------------
20-May-98     | #include "hello.h"
 .            |
 .            | char *
21-Apr-99     | env_user() {
 .            | char * user_env;
 .            | user_env = getenv("USER");
.
.
.
 .            | time_t clock;
24-Mar-99     | char *s;
20-Sep-98     |
14-Jun-99     | s = ctime(&clock);
 .            | s[ strlen(s)-1 ] = ' ';
 .            | return s;
20-May-98     | }
```

- Annotate a source file, using the long format.

  cmd-context **annotate -long msg.c**

  ```
  Annotated result written to "msg.c.ann".
  ```

  type msg.c.ann

  ```
  02-Apr-99.10:51:54 ##### Steve (scd.user@reach)\main\rel2_bugfix\1
  a test
  .
  .
  .
  ------------------------------------------------
  ------------------------------------------------
  ```

```
##### 01-Apr-99.16:19:25 scd \main\1             | #include "hello.h"
##### 02-Apr-99.10:51:54 scd \main\rel2_bugfix\1 | /* a test */
##### 01-Apr-99.16:19:25 scd \main\1             |
.
.
.
##### .                                         |char *
##### .                                         | hello_msg() {
.
.
.
```

- Annotate a source file and write the output to standard output. Display deletion lines, customize the annotation format, and suppress the header output.

  cmd-context **annotate -out - -fmt "%Sd %-8.8u | " -rm -nheader util.c**

```
20-May-98 anne |                      | #include "hello.h"
.              |                      |
.              |                      | char *
.              |                      | env_user() {
.              | DEL 08-Feb-99 gcd    | return getenv("USER");
08-Feb-99 gcd  |                      | char *str = getenv("USER");
.              |                      | if ( strcmp(str,"root") == 0 )
.
.
.
```

- Customize the header format, but use the default format for text line annotations.

  cmd-context **annotate -out - -fmt "-,Version %Vn created by %u.\n" util.c**

```
version \main\3 created by anne.
version \main\2 created by anne.
version \main\1 created by rick.
version \main\0 created by rick.

------------------------------------------------
------------------------------------------------
```

```
# 20-May-98 rick          \main\1         | #include "hello.h"
#                         .               |
#                         .               | char *
#                         .               | env_user() {
### 08-Feb-99 anne        \main\3         | char *str = getenv("USER");
###                       .               | if ( strcmp(str,"root") == 0 )
.
.
.
```

SEE ALSO

fmt_ccase, **type_manager**

apropos

Displays command summary information

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
MultiSite	multitool subcommand

Platform
UNIX

SYNOPSIS

- Extract information from the standard **whatis** file:
  apr·opos *topic ...*
- Extract information from auxiliary **whatis** file:
  apr·opos –glo·ssary [*topic-args*]

DESCRIPTION

This command does not require a product license.

The **apropos** command extracts information from the product-specific **whatis** file in *ccase-home-dir***/doc/man/**. Use **apropos** as you use the standard UNIX **whatis(1)** or **apropos(1)** command. Alternatively, use the **–glossary** option to extract a glossary definition or other help information from the auxiliary **whatis** file.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

Default: A lookup is performed in the standard **whatis** file.

topic ...

apropos makes a separate search for each *topic* character string in the standard **whatis** file. The string can occur anywhere within the line.

–glo·ssary [*topic-args*]

Combines all topic-args into a single character string, then displays all sections in the auxiliary **whatis** file whose header lines include this character string. Omitting the *topic-args* causes the entire auxiliary file to be displayed.

EXAMPLES

- Search for lines with the string "reserve" in the **whatis** file.

  cleartool apropos reserve
  ```
  reserve                 Convert an unreserved checkout to reserved
  unreserve               Change a reserved checkout to unreserved
  ```

- Search for lines with the string "epoch" in the standard MultiSite **whatis** file.

  multitool apropos epoch
  ```
  chepoch    change epoch information
  lsepoch    display epoch information
  ```

- Search in the auxiliary whatis file for glossary terms that include the phrase "symbolic link."

  cleartool apropos –glossary symbolic link
  ```
  +++ VOB symbolic link
  An object, cataloged in a (version of a) directory element, whose
  contents is a pathname. ClearCase does not maintain a version history
  for a VOB symbolic link.
  ```

FILES

ccase-home-dir/**doc/man/cc_whatis**
ccase-home-dir/**doc/man/cc_whatis.aux**
ccase-home-dir/**doc/man/ms_whatis**
ccase-home-dir/**doc/man/ms_whatis.aux**

SEE ALSO

help, **man**

att_clnt

Attache user-level commands (command-line interface)

APPLICABILITY

Product	Command Type
Attache	command

Platform
UNIX
Windows

SYNOPSIS

att_clnt [**–exit**] { {**checkin** | **ci**} [**–c**·**omment** *comment* | **–nc**·**omment**] *pname*...
{**checkout** | **co**} [**–c**·**omment** *comment* | **–nc**·**omment**] *pname*...
{**uncheckout** | **unco**} *pname*...
import [**–ci**] [**–c**·**omment** *comment* | **–nc**·**omment**] *pname*...
[**exec**] *command* [*arguments* ...] }

DESCRIPTION

A command-line interface that makes it possible to execute any Attache command and any **cleartool** command supported by Attache (except **lsvtree –graphical**) from a command shell or from scripts. For a list of Attache commands, see the **attache_command_line_interface** reference page.

att_clnt.exe is a Windows program, and can be run from a DOS command line in Windows NT or Windows 95. It cannot be executed from a command line in Windows 3.x, but it can be invoked from another program.

NOTE: Do not use **att_clnt** scripts on Windows 95. The DOS shell in Windows 95 does not wait for a Windows program to complete. Once the program starts, another command can be executed. Therefore, a script file containing several **att_clnt** commands in sequence will run in parallel, rather than one at a time.

USAGE OVERVIEW

There are four commands specially recognized by **att_clnt**: **checkin**, **checkout**, **uncheckout**, and **import**. For each of these commands, **att_clnt** displays a special dialog box. In addition, for **import**, **att_clnt** excludes intermediate build files produced by Visual C++.

Using the **exec** argument passes the arguments following it to the Attache command interpreter with no special treatment. Therefore,

```
att_clnt exec checkin
```

is not the same as

```
att_clnt checkin
```

OPTIONS AND ARGUMENTS

EXITING THE ATTACHE INTEGRATION CLIENT. *Default:* **att_clnt** does not exit after the command execution completes, so the command results can be read from the output window. To terminate the program, click **Close**.

–exit
Causes **att_clnt** to terminate after command execution. It can be used in scripts for unattended operation.

RUNNING AN ATTACHE COMMAND FROM THE ATTACHE INTEGRATION CLIENT. *Default:* None.

[**exec**] *command* [*arguments* ...]
Passes the *arguments* to the Attache command interpreter. File-name arguments can have absolute or relative pathnames.

The Attache commands **checkin**, **checkout**, **uncheckout**, and **import** can be run with or without the **exec** argument. If used without **exec**, a special dialog box is displayed for each command. Used with **exec**, these commands work as specified in their respective reference pages. For all other commands, specifying **exec** has no effect.

SEE ALSO

attache, **attcmd**

Attache

Overview of the Attache client program

APPLICABILITY

Product	Command Type
Attache	command

Platform
UNIX
Windows

SYNOPSIS

attache [*ws-name* | **–n**]

DESCRIPTION

The Attache client program runs on your personal computer and enables you to access ClearCase versioned object bases (VOBs) on UNIX or Windows NT hosts running the workspace helper program **ws_helper**.

OPTIONS AND ARGUMENTS

SPECIFYING THE INITIAL WORKSPACE. *Default:* The workspace, if any, that was active when Attache was last exited.

ws-name

Specifies the workspace name or the view-tag name of an existing workspace to which Attache will be set on startup.

–n

Starts Attache with no initial workspace.

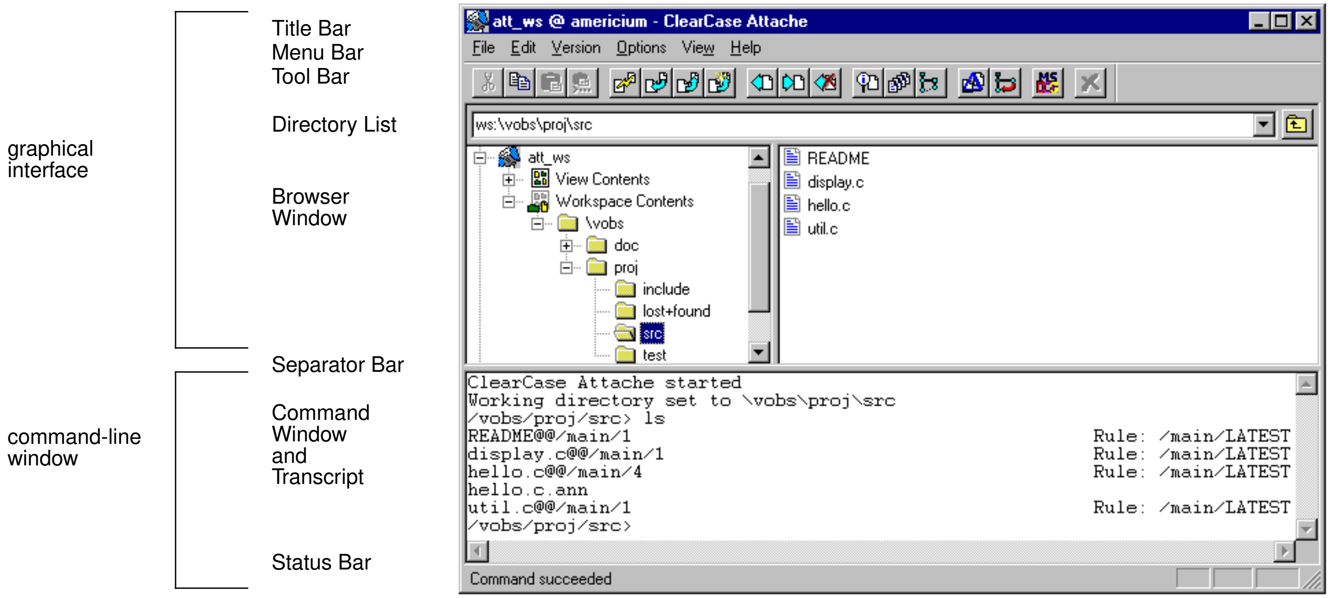
ATTACHE INSTALLATION DIRECTORY

Attache documentation refers to the installation directory with the symbol *attache-home-dir*, which by default is **C:\Program Files\Rational\Attache**, except on Windows 3.*x*, where it is **C:\RATIONAL\ATTACHE**.

STARTING ATTACHE

On both Windows 95 and Windows NT, Attache can be started from the command line. On Windows 95 or Windows NT 4.0 Attache can also be started by clicking **> Start > Programs >**

ClearCase > Attache. On other Windows platforms, Attache can be started via the Attache icon in the ClearCase Program Group or by clicking **File > Run** in File Manager or the Program Manager. Starting Attache opens an Attache window as shown here.



See the **attache_graphical_interface** reference page for a description of the menus, browsers and buttons, and the **attache_command_line_interface** reference page for an overall description of the commands.

ATTACHE COMMAND TOOL

The Attache command tool (**attcmd.exe**) is a command-line interface primarily intended for use in scripts. It can be used in a single-command mode or interactively. The command tool is available on Windows 95 and Windows NT, but not on Windows 3.x.

The Attache command tool can be started from the command line. The Attache command tool can also be started by clicking **> Start > Programs > ClearCase > Attache Command Tool**.

See the **attcmd** reference page for a description of the command tool.

ATTACHE INTEGRATION CLIENT

The Attache Integration Client is a command-line interface primarily intended to be invoked by other tools, for example, Visual C++ 1.5 or 2.x. It can be invoked from the command shell on Windows 95 or Windows NT. The integration client is available on Windows 3.x but it cannot be invoked from the command shell. See the **att_clnt** reference page for a description of the Attache integration tool.

ATTACHE'S STARTUP DIRECTORY

There is a separate startup directory associated with the Attache client process. This directory changes depending on how Attache is started:

- If started from the command line, it is the directory from which you start **attache**.
- In Windows 95 and Windows NT 4.0:
    - If you are using a shortcut, the startup directory defaults to the **bin** subdirectory of *attache-home-dir*, but it can be changed using Properties.
- In Windows 3.1 and Windows NT 3.51:
    - It is the "working directory" specified in Attache's program item properties, if Attache is started from the icon.

After the Attache client process is started, this directory never changes. This start-up directory serves as the default location for a newly created workspace storage directory if a full local pathname is not specified.

USER NAME AND PASSWORD

User name and password information for the helper host is required for Attache use. If this information has not been set up in the configuration database, it will be requested when you make or set to a workspace. User names on UNIX and Windows NT helper hosts take different forms. On Windows NT helper hosts, the username is a combination of domain name and user name, for example, **rational\jed**, and on UNIX hosts, the user name stands alone.

GETTING HELP

Attache provides an online help facility:

- **Hypertext Help System —** From the menu bar **Help > Contents** to enter Attache's hypertext online help system.

ERROR LOG

Some of the warning and error messages displayed by Attache commands are also written to log files located in directory **/var/adm/atria/log** on a UNIX ClearCase host, or to the Windows NT event log on a Windows NT ClearCase host.

SEE ALSO

attache_graphical_interface, **attache_command_line_interface**, **attcmd**, **att_clnt**, **ws_helper**

attache_command_line_interface

Using Attache commands

APPLICABILITY

Product	Command Type
Attache	general information

Platform
UNIX
Windows

SYNOPSIS

command [*options/args*]

DESCRIPTION

Attache commands create, modify, and manage the information in the workspace, that is, local files. Attache, the PC interface to ClearCase version-control and configuration management software, has a rich set of commands that create, modify, and manage the information in ClearCase VOBs and views. Commands are entered in the Attache client Command window or by using the menus and buttons provided through the graphical user interface. See the **attache_graphical_interface** reference page for a description of the graphical user interface.

ATTACHE COMMANDS

This reference page does not describe the individual commands:

annotate
catcr
catcs
cd
checkin
checkout
checkvob
chevent
chpool
chtype
clearlicense
cptype
describe
diff
diffcr
edcs
find
findmerge
get
getcache
getlog
help
hostinfo
import
ln
lock
ls
lscheckout
lsclients
lsdo
lshistory
lslock
lslocal
lspool
lsprivate
lsregion
lsreplica
lstype
lsview
lsvob
lsvtree
lsws
make
man
merge
mkattr
mkattype
mkbranch
mkbrtype
mkdir
mkelem
mkeltype
mkhlink
mkhltype
mklabel
mklbtype
mkpool
mkregion
mktag
mktrigger
mktrtype
mkview
mkvob
mkws
mount
mv
mvws
protect
put
pwd
pwv
quit
recoverview
reformatview
reformatvob
register
relocate
rename
reserve
rmattr
rmbranch
rmdo
rmelem
rmhlink
rmlabel
rmmerge
rmname
rmpool
rmregion
rmtag
rmtrigger
rmtype
rmver
rmvob
rmview
rmws
setcache
setcs
setws
shell
space
startview
umount
uncheckout
unlock
unregister
unreserve
update
winkin
wshell

ATTACHE COMMAND WINDOW

The Command window is an editable text window with keyboard input available at the command prompt line. By default, the Command window occupies the portion of the Attache Window directly above the Status Bar. The Command window also serves as a transcript pad, retaining approximately (limited by available memory in the system) the most recent 200 lines; you can scroll the transcript horizontally and vertically.

You enter all Attache commands at the command prompt, which is the name of your current working directory. You can type commands, or copy them from other places in the window and paste them at the prompt.

You can move the cursor with the keyboard's arrow keys or with the mouse. To reexecute a previously entered command, place the cursor anywhere in the line of that command and press ENTER. You can edit the command before pressing ENTER; this makes it easy to correct typing errors and to modify previously entered commands.

GETTING HELP FOR COMMANDS

Attache provides several online help facilities for its commands:

- **Syntax summary** — To display a syntax summary for an individual command, use the **–help** option:

  mklabel –help *(syntax of one command)*

- **Reference pages** — Use **man** *command_name* or **help** *command_name* to display the reference page for an Attache or ClearCase command.

- **Hypertext Help System** — On the command menu, click **Help > Contents** to enter Attache's hypertext online help system and choose **ClearCase Reference** from the **Main Contents**.

USAGE OVERVIEW

A single Attache command can be invoked from the prompt in the Command window using this syntax:
command [*options-and-args*]

Command Options

Command options may appear in any order, but all options must precede any nonoption arguments (typically, names of files, versions, branches, and so on). If an option is followed by an additional argument, such as **–branch /main/bugfix**, there must be white space between the option string and the argument. If the argument itself includes space characters, it must be enclosed in quotes.

Command Abbreviations and Aliases

Many command names and option words can be abbreviated. A command's syntax summary indicates all valid abbreviations. For example:
–pre·decessor

This means that you can abbreviate the option to the minimal **–pre**, or to any intermediate spelling: **–pred**, **–prede**, and so on.

For option words, the minimal abbreviation is always three characters or fewer.

A few commands have a built-in command alias. For example, **checkin**'s alias is **ci**; **checkout**'s alias is **co**. These commands are equivalent:

cmd-context **checkin test.c**

cmd-context **ci test.c**

PATHNAMES IN ATTACHE COMMANDS

Many Attache commands require a pathname as an argument, such as the name of a file element, directory element, or view-private file. In most cases you can use full remote, drive-relative, or relative pathnames, but full local pathnames have very limited use. A full local pathname begins with a drive letter and designates a file that physically exists on the PC; a full remote pathname begins with a slash or backslash and designates a file in your view. Full local pathnames are allowed in only four cases: the *ws-pname* argument to **mkws**, the *@pname* argument for **put**, **get**, **find**, **update**, or **findmerge**, the project configuration file for **Update**, and the *file* argument to the **–exclude** option for **import**.

Although, in general, remote pathnames are correct in Attache commands, there are restrictions:

- If your view host supports names longer than those supported on your local host, attempts to download a pathname with any component longer than that maximum result in an error.
- If your view host supports pathnames longer than those supported on your local host, attempts to download a pathname longer than that maximum result in an error.
- If your view host supports file names with mixed case, attempts to download a file whose name contains any uppercase characters result in a warning. Uploading a file using wildcard expansion, or by uploading its directory, creates the remote file with all lowercase characters.
- If your view host is a Windows NT machine, remote pathnames cannot specify drive names or UNC-prefixes, except in cases that access only remote files, for example, the **lshistory** or **lsvob** commands. Using a command such as **get** will attempt to download the pathname, resulting in an error.

  For help in addressing these issues in your environment, see *Resolving File-Naming Issues in Cross-Platform Development* in the *ClearCase Attache Manual*.

For a workspace using a UNIX helper, the helper's process is set to the view of your workspace.

For a workspace using a Windows NT helper, there is an implicit **subst** of a drive, for example, **Z:**, for the current view, and a change to that drive. Thus, **Z:** is equivalent to **M:\myview,** and a pathname reference such as **\my_vob** is really a reference to **Z:\my_vob**. Pathnames beginning with a backslash (\) or slash (/), such as **\my_vob** are called drive-relative, rather than full, because they do not begin with a drive letter. Drive-relative pathnames assume the current drive.

In many cases, you can also use a ClearCase-defined variant: a version-extended pathname (full, drive-relative, or relative) or in some cases a view-extended pathname (full or relative).

Slash (/) and backslash (\) are interchangeable in pathnames. For example:

`C:\users\smg\test\test.c`	*(full local pathname to workspace file)*
`/vobs/myvob/src/main.c`	*(full remote pathname to UNIX helper host; also called a 'VOB pathname', because it begins with a VOB-tag (/vobs/myvob))*
`/view/smgvw/vobs/myvob/src/main.c`	*(view-extended full pathn/test.c/main/bugfix/4ame (VOB object); the /view directory constitutes 'view-extended namespace')*
`/vobs/myvob/src/main.c@@/main/3`	*(version-extended full remote pathname)*

`C:\users\smg\test\test.c`	*(full local pathname to workspace file)*
`\vob_proj\test\test.c`	*(drive-relative remote pathname to Windows NT helper host; also called an 'VOB pathname', because it begins with a VOB-tag (\vob_proj))*
`M:\smg_view\myvob\src\main.c`	*(view-extended full pathname (VOB object); the M: drive constitutes 'view-extended namespace')*
`\myvob\src\main.c@@\main\3`	*(version-extended drive-relative remote pathname)*

A relative pathname does not begin with a slash or backslash and is relative to your working directory. For example:

`test.c`	*(relative pathname)*
`tcp/libw.a`	*(relative pathname)*
`test.c@@/main/4`	*(version-extended relative pathname)*

`test.c`	*(relative pathname)*
`..\lib`	*(relative pathname)*

For both full or relative pathnames:

- The standard pathname of an element implicitly references the version selected by the current view. (This feature is called transparency.)
- A view-extended pathname references the version of the element selected by the specified view. However, using view-extended names to access files in another workspace does not work because the version selected by that view may be different from the file in the associated workspace.
- A VOB-extended pathname references an object using a VOB database identifier. The most commonly used is a version-extended pathname, which references a particular version of

an element using its unique version-ID (**test.c@@\main\bugfix\4**) or using a version label (**test.c@@\RLS2.0**). Other kinds of VOB-extended pathnames:

`hello.c@@`	*(extended pathname to element object)*
`hello.c@@\main\bugfix`	*(extended pathname to branch object)*

For more information, see the **version_selector** and **pathnames_ccase** reference pages.

PROCESSING OF SYMBOLIC LINKS

Downloading a pathname that contains a symbolic link downloads a copy of the file or directory the link points to, rather than the link itself.

In addition, Attache commands do not traverse VOB symbolic links; rather, they operate on the link objects themselves. For example:

- You cannot check out a VOB symbolic link, even if it points to an element.
- A **describe** command lists information on a VOB symbolic link object, not on the object to which it points.
- A **mklabel –recurse** command walks the entire subtree of a directory element, but it does not traverse any VOB symbolic links it encounters.

COMMAND-LINE PROCESSING

Attache interprets the command line and recognizes various special characters and constructs:

character escape (**^**)	The two-character sequence **^***special-char* suppresses the special meaning of the character.
single-quoting (**′ ′**)	Allows white space characters and other special characters to be included as part of the command argument. Within a single-quoted string (**′** ... **′**), a double-quote character has no special meaning, and **^′** is replaced by **′**.
double-quoting (**" "**)	Allows white space characters and other special characters to be included as part of the command argument. Within a double-quoted string (**"** ... **"**), **^"** is replaced by **"**, and **^′** is replaced by **′**.
commenting (**#**)	Command lines that begin with a number sign (**#**) are ignored.
wildcards	File-name patterns (including *****, **?**, and so on) that are not enclosed in quotes are expanded as described in the **wildcards** reference page. These patterns are also supported in config specs. (The meaning of ellipsis is slightly different in config specs; see the **config_spec** reference page.)

Attache does not expand environment variables.

RESTRICTIONS ON COMMAND USE

Each Attache command description lists the restrictions for the command, including required identities.

OBJECT LOCKING

ClearCase provides for temporary access control through explicit locking of individual objects with the **lock** command. When an object is locked, it cannot be modified by identity (except those explicitly excluded from the lock).

Attache command descriptions list the locks that can prevent a command from being executed, even if you have the necessary permissions. For example, the **chtype** command lists three locks that would prevent you from changing an element type:

```
VOB, element type, pool (non-directory elements only)
```

This means that **chtype** would fail if the VOB containing the element were locked, if the element's type were locked (such as the **text_file** type), or the storage pool containing the (nondirectory) element were locked.

SEE ALSO

man, **help**, **attache**, **attache_graphical_interface**, **fmt_ccase**, **lock**, **permissions**, **profile_ccase**, **version_selector**, **attcmd**, **att_clnt**

attache_graphical_interface

Attache windows, toolbar, and menus

APPLICABILITY

Product	Command Type
Attache	general information

Platform
UNIX
Windows

SYNOPSIS

Attache's windows, browsers, toolbar, and menus

DESCRIPTION

The graphical user interface enables you to interact with the Attache client through its toolbar, menus, and browsers.

GETTING HELP

Attache provides online help facilities:

- **Hypertext Help System** — On the command menu, click **Help > Contents** to enter Attache's hypertext online help system to see the Table of Contents for the Help System.

ATTACHE'S WINDOWS

Command Window

The Command window is an editable text window with keyboard input available at the command prompt. By default, the Command window occupies the portion of the Attache Window directly above the Status Bar. To change the size of the Command window, or to completely obscure it, move the separator bar up or down. The size is remembered, and this configuration is used each time you start Attache. The Command window is the main component of Attache's command-line interface. See the **attache_command_line_interface** reference page for more information.

Browser Window

The Browser window is an adjustable pane above the Command window, containing the File Browser. You can change the size of this window, or obscure it completely by moving the separator bar up or down. The size is remembered, and this configuration is used each time you start Attache. The File Browser window displays your local files (**Workspace Contents**) and all the VOBs mounted on your helper host and visible through the current view (**View Contents**).

Selections you make in the Browser window enable the different toolbars and commands. You can select items in several ways:

In the left side of this window:

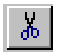
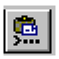
- Clicking a **+** expands the directory or icon; clicking a **–** collapses it
- Clicking a folder or icon displays its contents in the right side of the window

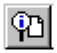
In the right side of this window:

- A single click selects a single item
- Selecting a single item followed by pressing SHIFT and clicking a second item selects the range of items from the first to the second, inclusive
- Pressing CTRL and clicking selects discontiguous items
- Double-clicking a folder opens it; double-clicking a file item invokes its associated application. If the file is not already local, Attache first opens the **get** dialog box for the file.

To refresh the Browser window manually, use **View > Refresh**.

ATTACHE'S TOOLBAR

Button	Name	Function	Menu Equivalent	Corresponding Command
	**Cut**	Cuts the selected text to the clipboard	**Edit > Cut**	CTRL+Q
	**Copy**	Copies the selected text to the clipboard	**Edit > Copy**	CTRL+C
	**Paste**	Pastes the text from the clipboard	**Edit > Paste**	CTRL+V
	**Paste submit**	Pastes the text from the clipboard after moving the insertion point to the end of the buffer	**Edit > Paste submit**	CTRL+SHIFT+INS

Button	Name	Function	Menu Equivalent	Corresponding Command
	**Update**	Downloads all nonwritable files created since you last updated your workspace	**File > Update Workspace**	**update**
	**Get**	Opens the **Get** dialog box for the Browser selection	**Version > Get**	**get**
	**Put**	Uploads specified writable files from your workspace to the associated view	**Version > Put**	**put**
	**Import**	Creates an element corresponding to each selected file or directory in a workspace subtree	**Version > Import**	**import**
	**Checkout**	Opens the **Checkout** dialog box for the Browser selection	**Version > Checkout**	**checkout**
	**Checkin**	Opens the **Checkin** dialog box for the Browser selection	**Version > Checkin**	**checkin**
	**Uncheckout**	Opens the **Uncheckout** dialog box for the Browser selection	**Version > Uncheckout**	**uncheckout**
	**Properties**	Displays descriptive information for the Browser selection	**Version > Properties**	**describe**
	**History**	Displays the version history for the Browser selection	**Version > History**	**lshistory**
	**Version Tree**	Displays the graphical version tree for the Browser selection	**Version > Version Tree**	**lsvtree –graphical**
	**Diff vs. Predecessor**	Displays the differences between the Browser selection and its predecessor version	**Version > Diff vs. Predecessor**	**diff –pred –graphical**
	**Merge**	Executes the **findmerge** command for the Browser selection	**Version > Merge**	**findmerge**

Button	Name	Function	Menu Equivalent	Corresponding Command
	**Command Shell**	Starts up the MS-DOS Prompt in the current working directory of the workspace	**File > Command Shell**	**wshell**
	**Stop Execution**	Cancels the currently executing command as soon as possible. The **Stop Execution** button is available only when a command is active.	**File > Stop Execution**	—

ATTACHE'S MENUS

Summary

Commands	Corresponding Command	When Is It Enabled?
File Menu		
New Workspace	**mkws**	Always
Open Workspace	**setws**	If workspaces have been created
Update Workspace	**update**	If there is a current workspace
Delete Workspace	**rmws**	If workspaces have been created
Workspace Properties	—	If a workspace is selected in the file browser
Edit Config Spec	**edcs**	If there is a current workspace
Change Directory	**cd**	If a directory is selected in the File Browser
Command Shell	**wshell**	If there is a current workspace
Stop Execution	—	Only when a command is active
Exit	**quit**	Always
Edit Menu		
Cut	CTRL+X	If text is selected in the Command Window

Commands	Corresponding Command	When Is It Enabled?
Copy	CTRL+C	If text is selected in the Command or Browser Window
Paste	CTRL+V	If text has been cut or copied to the clipboard
Paste/Submit	CTRL+SHIFT+INS	If text has been cut or copied to the clipboard
Find	These are not ClearCase **find**	Always
Find Next		If **Find** has been invoked with a search string
Version Menu		
Import	**import**	If one or more files or directories are selected in the File Browser
Remove	**rmname**	If one or more files or directories are selected in the File Browser
Get	**get**	If one or more files or directories are selected in the File Browser
Put	**put**	If one or more files or directories are selected in the File Browser
Checkout	**checkout**	If one or more files or directories are selected in the File Browser
Checkin	**checkin**	If there is a current workspace
Uncheckout	**uncheckout**	If there is a current workspace
Properties	**describe**	If one or more files or directories are selected in the File Browser
History	**lshistory**	If one or more files or directories are selected in the File Browser
Version Tree	**lsvtree –graphical**	If one or more files or directories are selected in the File Browser
Diff vs. Predecessor	**diff –graphical –predecessor**	If one or more files or directories are selected in the File Browser

Commands	Corresponding Command	When Is It Enabled?
Merge	**findmerge**	If there is a current workspace
Options Menu		
Preferences	—	Always
Fonts	—	If the command-line interface is active
View Menu		
Toolbar	—	always
Status Bar	—	Always
Refresh	—	Always
Help Menu		
Contents	—	Always
How to Use Help	—	Always
Tutorial	—	Always
About Attache	—	Always

File > New Workspace

New Workspace creates a workspace and, if **Use existing view** is not selected, its associated view for a view host running ClearCase. You are prompted for the workspace name (also the view-tag name), the workspace storage directory, the workspace helper host (defaults to the view host), and if you are creating a new view, the view host, the view storage directory in which to create the view, and the global pathname to that directory.

File > Open Workspace

Open Workspace sets you to a workspace. Choosing a workspace name from the list will change your environment to the selected workspace.

File > Update Workspace

Update Workspace downloads the files specified in the project configuration file to your workspace. You can specify a log file for the operation by choosing **Log update activity to this file**. You are prompted for the full pathname of the file. You can specify whether to **Get all**

versions or to **Find versions checked in since** the specified time and download only them. You can also choose whether to **Recurse into the subdirectories** of the directories specified in the project configuration file. After an **Update** has been performed using a given project configuration file, the time of the **Update** is remembered. This time is used automatically by the **Find versions checked in since** option the next time the **Update** dialog box is opened for that configuration file, unless you specify a different time. You can list the files that would be downloaded without performing the **Update** by selecting **Display changed versions but do not download** in conjunction with the **Find versions checked in since** option.

File > Delete Workspace

Delete Workspace removes the selected workspace. Removing a workspace causes all its local files and subdirectories to be deleted including its workspace storage directory, even if it is being shared with another workspace.The view associated with the workspace is not deleted unless it was created with the same **mkws** or **File > New Workspace** command as the workspace.

File > Workspace Properties > Workspace

Workspace specifies the following attributes for the selected workspace: **Workspace and view tag name**, **Workspace storage directory**, and the **Initial working directory**. The Workspace and view tag name and the Workspace storage directory are set when you create the workspace using **mkws** or **File > New Workspace**. The default **Initial working directory** is set to root (****).Change the default by entering the full pathname of the preferred **Initial working directory** in this field.

File > Workspace Properties > Helper

Helper is used to specify the **Helper host**, **Login username**, and **Login password** of the selected workspace. The default **Helper host** is set when the workspace is created using **mkws** or **File > New Workspace**. Change the default by entering a new **Helper host** name in this field. You can edit the **Login username** and **Login password** fields. The **Login password** is encrypted. Delete the current password by clicking **Delete Password**. You can also choose to **Delete view when workspace is deleted**.

File > Workspace Properties > Update Status

For a selected workspace, **Update Status** displays the **Time of last successful update** and the **Project config file used** in that update.

File > Edit Config Spec

Edit Config Spec brings up the **Notepad** in which to modify the current config spec. On return, you are prompted about whether to accept this as the current config spec or to abort.

File > Change Directory

Change Directory changes your working directory to that of the selected directory in both the workspace and the view.

File > Command Shell

Command Shell starts the MS-DOS Prompt in the current working directory of the workspace.

File > Stop Execution

Stop Execution cancels the currently executing command as soon as possible.

File > Exit

Exit quits the Attache program.

Edit > Cut

Cut moves the text selected in the Command Window to the clipboard.

Edit > Copy

Copy makes a copy of the text selected in the Command Window on the clipboard. Executing a **Copy** command while the browser has the focus puts the pathnames of selected items on the clipboard.

Edit > Paste

Paste copies the text from the clipboard to the Command Window.

Edit > Paste/Submit

Paste/Submit moves the insertion point to the end of the buffer before pasting.

Edit > Find

Find searches for an occurrence of a text string in the Command window. You are prompted for the string, whether to **Match case**, and the **Direction** in which to search.

Edit > Find Next

Find Next reexecutes the search with the same string.

Version > Import

Import creates an element corresponding to each selected file or directory in a workspace, if the element does not already exist in the VOB. You can enter a new comment or accept the last comment entered in this Attache session. Selecting **Exclude files matching patterns contained in this file** specifies the pathname of a file containing file-name patterns that are not to be imported. You can use **Browse** to specify an existing file or **Edit** to change a file. If **Checkin initial versions of new elements** is selected, **Import** creates the new element and version **\main\0**, checks out the element, then uploads and checks in a new version containing the data in the workspace file. Selecting **Lower-case new element names** causes new element names to be created all lower-case in Windows 95 and Windows NT. Windows 3.x creates element names in lowercase by default.

Version > Remove

Remove executes the **rmname** command for the selected element name or VOB symbolic link listed in the **Remove Names** dialog box. The directory is checked out, if needed.

Version > Get

Get downloads files or directories selected from the File Browser to your workspace. You can specify a log file for the operation by choosing **Log download activity to this file**. You are prompted for the full pathname of the file. You can choose to **Overwrite local files** for any existing writable file of the same name, to **Prompt for Overwrite confirmation** with existing writable files, or to select the **Do Not Overwrite local files** option. There is also a **Recurse through subdirectories** toggle switch to download the full directory tree beneath your selection and a **Preserve timestamps from remote files** toggle switch to allow you to download the files while maintaining their date and time.

Version > Put

Put uploads files or directories selected from your workspace to the associated **view**. You can specify a log file for the operation by choosing **Log download activity to this file**. You are prompted for the full pathname of the file. There is also a **Recurse through subdirectories** toggle switch to upload the full directory tree beneath your selection and a **Preserve timestamps from remote files** toggle switch to allow you to upload the files while maintaining their date and time.

Version > Checkout

Checkout checks out files or directories selected from the File Browser. You can enter a new checkout comment or accept the last checkout comment entered in this Attache session. If you click **Query for each file**, you are prompted for comments in the Command window; otherwise, the comment is used for each checkout. The **Comment** text box may be left empty. You can also select the check box for an Unreserved checkout, which is not selected by default.

Version > Checkin

Checkin checks in a list of versions selected from the File Browser. You can enter a new checkin comment or (by leaving the Comment text box empty) accept the default comment which is the checkout comment for the first selected version. Any comment specified is used for all checked-in files, as long as **Query for each file** is not selected. To use the checkout comment for all files being checked in, select **Use Checkout comment for each file.** If you select **Query for each file**, you are prompted for comments in the Command window. You can also select **Checkin even if identical**, which is not selected by default.

Version > Uncheckout

Uncheckout cancels the checkout of one or more files or directories as selected from the File Browser. You can specify which checkouts to cancel by selecting **All selected elements** or **Checked-out elements in all current work activities**. To save the current checked-out version as a view-private file, select the **Keep view-private copies** check box .

Version > Properties > General

For a selected element, **General** displays as a **Description** the comment the user supplied when creating the element, and the **Element Type**.

For a version, **General** displays as a **Description** the last comment issued on the checkin or checkout of the version.

For a checked-out version, **General** also displays the view from which the version is checked-out (**Checkout View**), whether it is a **Reserved** checkout, and the name the **Predecessor** version for the checked-out file.

Version > Properties > Labels

Labels displays any labels attached to the version selected in the File Browser.

Version > Properties > Attributes

Attributes displays the **Type** and **Value** of any attribute for the version or element selected in the File Browser.

Version > Properties > HyperLinks

HyperLinks displays the list of hyperlinks for the version or element selected in the File Browser.

Version > Properties > Triggers

For directory elements, **Triggers** displays the list of triggers that fire on operations involving the element (**Attached Triggers**) and the list of triggers inherited by elements created within the directory (**Inherited Triggers**).

For file elements, **Triggers** displays an Attached Triggers list.

Version > Properties > Protection

Protection displays the meaningful access permissions (read, write, execute) for the particular element type selected in the File Browser. Access permissions are displayed in three sections: Owner, Group, and World. Not all permissions are meaningful in the VOB. Available check boxes indicate that the permission is meaningful; dimmed check boxes indicate that the permission is not.

Version > Properties > Lock

Lock displays whether the element selected in the File Browser is **Locked** with a **lock** command or **Obsolete** with a **lock –obsolete** command. The **Description** field displays the comment the user specified when locking or obsoleting the element, and **Excluded Users** displays a list of user names to which the lock does not apply.

Version > History

History executes the **lshistory** command for the file or directory selected in the File Browser.

Version > Version Tree

Version Tree executes the **lsvtree –graphical** command for the file or directory selected in the File Browser.

Version > Diff vs. Predecessor

Diff vs. Predecessor executes the **diff –graphical –predecessor** command for the file or directory selected in the File Browser.

Version > Merge

Merge executes the **findmerge** command for one or more files selected in the File Browser. Merge allows you to consider merges either for **Selected elements only**, or **All elements in containing VOBs** (**–all**), or in **All VOBs** (**–avobs**), or for a specific **Project configuration file**. The merge action can be one of **Display needed merges only** (**–print**), **Character mode merge** (**–merge**), **Graphical mode merge** (**–graphical**), or **Graphical mode only when there are conflicts** (**–merge –graphical**). The source version can be one of the specified **Version specifier** (**–fversion**), the **Latest version on the checked-out branch** (**–flatest**), or the **Version selected by this view** (**–ftag**). Click **Browse** to see a list of views. As options, you can specify **Recurse into subdirectories** (**–nrecurse**), **Prompt for each needed merge** (**–okmergel** or **–okgraphical**), and **Serial output format** (**–serial**). The Comment box may be left empty. For more information, see the **findmerge** reference page.

Options > Preferences > Login Info

Login Info stores your user name and password information, which is used when you connect to a workspace using **File > New Workspace** or the **mkws** or **setws** commands. If you have different user name and password combinations for different ClearCase hosts, you can store a different combination for each workspace by selecting **Current workspace only**, or you may choose to be prompted for this information, rather than to store it.

Options > Preferences > Registry

Registry is used to specify the ClearCase registry host and the ClearCase network region.

Options > Preferences > View Attributes

View Attributes specifies the interop text mode for new views created by the **mkws** command or **File > New Workspace**. Each view has an associated text mode that determines whether line terminators are presented to the view exactly as stored. This is important if you are sharing files between the PC and UNIX machines, and if you run applications on the PC requiring that lines be terminated with line feed and carriage return characters. The values you can specify are those allowed by the **–tmode** option to the **mkview** command; see the **mkview** reference page.

Options > Preferences > Options

Compress during file transfer specifies whether to compress files during transfer between the workspace and the view and to uncompress them after the transfer to improve performance over slow communications lines. **Display properties in workspace contents browser** enables the display in the Workspace browser of properties that are always visible in the View browser. **Unreserved flag set for checkout operations** allows you to change the default checkout mode to unreserved.

Options > Font

Font is used to specify the **Font**, **Font style**, **Size**, and **Script** used in the Command Window.

View > Toolbar

If selected, the Toolbar is displayed. This is a toggle switch.

View > Status Bar

If selected, the Status Bar is displayed. This is a toggle switch.

View > Refresh

Refresh updates the display for the Browser window.

Help > Contents

Contents starts the online help system at the main Table of Contents.

Help > How to Use Help

Help on Help provides information about using the Windows Help System.

Help > Tutorial

Tutorial starts the interactive *Attache online Tutorial.*

Help > About

About provides information about the Attache executable, including the version number.

SEE ALSO

attache, **attache_command_line_interface**, **attcmd**, **att_clnt**

attcmd

Attache user-level commands (command-line interface)

APPLICABILITY

Product	Command Type
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- Single-command mode:
  attcmd [**–ws** *ws-name*] *subcommand* [*options/args*]
- Interactive mode:
  > attcmd [**–ws** *ws-name*]
  `attcmd>` *subcommand* [*options/args*]
  .
  .
  .
  `attcmd>` **quit**
- Display version information for Attache:
  attcmd –ver·sion

DESCRIPTION

attcmd is a console version of the Attache command-line interface. It is similar to the command window in the Attache graphical user interface (see the **attache_command_line_interface** reference page for more information). It is intended primarily for users who want to embed Attache operations in scripts or invoke them from other tools.

attcmd accepts all commands which are valid in the Attache command window, except for these:

- **describe –graphical**
- **lshistory –graphical**
- **lsvtree –graphical**

attcmd

NOTE: **attcmd** is a 32-bit Windows console application and is available on Windows 95 and Windows NT only.

attcmd SUBCOMMANDS

This reference page does not describe the individual **attcmd** subcommands. For a list of the **attcmd** subcommands, see the **attache_command_line_interface** reference page.

USAGE OVERVIEW

You can use **attcmd** in either single-command mode or interactive mode. A single Attache command can be invoked from the shell using this syntax:

attcmd [**–ws** *ws-name*] *subcommand* [*options/args*]

If you want to enter a series of subcommands, enter the **attcmd** command with no subcommand arguments. This places you at the interactive mode prompt:

```
attcmd>
```

You can then issue any number of subcommands (called “commands” from now on), ending with **quit** to return to the shell.

IDENTIFYING THE WORKSPACE

The **–ws** option can be used to specify which workspace in which to start. If not specified, then the current working directory is used to determine the workspace. If the current working directory is in or under a workspace, that workspace is set. If it is not, no workspace is set on startup and the **setws** command must be used to set a workspace.

After startup, if the current working directory is within the startup workspace, **attcmd** attempts to change to that directory after connecting to the workspace helper host. The attempt may fail if the directory does not exist in the view associated with the workspace. In this case, the initial working directory is the root directory.

INPUT REDIRECTION

Command input is read from the standard input, but input redirection generally does not work. Whenever a remote command is executed, any input which occurs during command execution is sent to the remote command process. Therefore, if input were redirected from a file, for example, the first command which caused remote command execution would cause the rest of the input file to be read and sent to the remote process.

EXIT STATUS

If you exit **attcmd** by entering a **quit** command in interactive mode, the exit status is `0` (zero). The exit status from single-command mode depends on whether the command succeeded (zero exit status) or generated an error message (nonzero exit status).

SEE ALSO

attache, attache_command_line_interface

catcr

Displays configuration record created by **clearmake**, **omake** (Windows only), or **clearaudit**

APPLICABILITY

Product	**Command Type**
ClearCase	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- Display a configuration record:

  catcr [**–r·ecurse** | **–fla·t** | **–uni·on** | **–che·ck** [**–uni·on**] | **–mak·efile**]
  [**–sel·ect** *do-leaf-pattern*] [**–ci**] [**–typ·e** { **f** | **d** | **l** } ...]
  [**–ele·ment_only**] [**–vie·w_only**] [**–cri·tical_only**] [**–nam·e** *tail-pattern*]
  [**–zer·o**] [**–wd**] [**–nxn·ame**] [**–fol·low**] [**–l·ong** | **–s·hort**] *do-pname* ...

- Display only the configuration record header and build scripts:

  catcr –scr·ipts_only [**–rec·urse**] [**–l·ong** | **–s·hort**] *do-pname* ...

DESCRIPTION

The **catcr** command displays the configuration records (CRs) for the specified derived objects (DOs) and, optionally, for their build dependencies. The ClearCase make tool (**clearmake**, or **omake** for Windows) creates a CR each time it executes a build script that creates one or more DOs. You can also list the configuration record header and build scripts only.

NOTE: ClearCase creates configuration records for dynamic views only.

For more information about configuration records and derived objects, see *Building Software*.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

REPORTING ON DERIVED-OBJECT SUBTARGETS. *Default:* **catcr** lists the derived-object subtargets used to build *do-pname*, but it does not examine or display subtarget CRs. The **–recurse**, **–flat**, **–union**, **–check**, and **–makefile** options direct **catcr** to recurse into subtarget CRs. Use **–select** to isolate the CRs of one or more subtargets; use **–ci** to examine the CRs of pre-built, checked-in DO versions.

–r·ecurse

Displays the CRs of any derived objects that are subtargets of *do-pname*. Each CR is displayed separately.

–fla·t

Similar to **–recurse**, but consolidates the CRs into a single list of versions and derived objects, with no duplicate entries. **–flat** produces one report for each *do-pname* on the command line. The report includes file-system objects only; no headers, variables and options, or build scripts. A number preceding each filename indicates the total number of times it was referenced during the build.

–uni·on

Produces one report for all derived objects on the command line. Like **–flat**, it consolidates the CRs of each *do-pname* and its subtargets into a single list of objects, with no duplicate entries. It then combines the separate lists into a single report with no duplicates. The report includes file-system objects only—no headers, variables and options, or build scripts are included.

–che·ck [**–uni·on**]

Flags entries in the CR that have unusual characteristics, including references to view-private files (but not references to view-private directories).

You can use **–check** with **–union**. This combination determines whether a CR contains any of the following:

- Versions that are not currently checked in. This includes versions that no longer exist (an intermediate version that only existed as a view-private file, for example), versions that are currently checked out, and versions that were explicitly removed with the **rmver** command.
- Multiple versions of the same element. This can occur, for example, if a build used multiple libraries, which were built from different source versions.
- Multiple references to the same element with different names, such as a renamed element in different directory versions.

–mak·efile

Similar to **–recurse**, but displays the CR in simple **makefile** format. The listing includes the dependencies and build script for each of the derived object's subtargets. Always include the **–wd** option with **–makefile**; this causes **catcr** to list pathnames with respect to the initial working directory of the build. (Note that this differs from the standard behavior of **–wd**). If you fail to include **–wd**, **cleartool** displays a warning message, and then displays the makefile without modifying dependency pathnames.

–sel·ect *do-leaf-pattern*

Starts the listing at the subtarget of *do-pname* that match the specified pattern. *do-leaf-pattern* can be a pattern (see the **wildcards_ccase** (ClearCase) or **wildcards** (Attache) reference page) that matches a simple file-name; it must not include a slash character (/\) or the ellipsis wildcard (...). Alternatively, it can be a standard pathname of a derived object.

This option is useful for isolating a derived object that was built as a dependency of another one. For example, this command displays the CR of the derived object named **hello.obj** that was used to build **hello.exe** in the current view:

cmd-context **catcr -select hello.obj hello.exe**

–ci (for use in recursive listings only)

By default, recursive listings do not display CRs for DO versions. This option displays the CRs for DO versions. **–ci** only has effect with **–recurse**, **–flat**, **–union**, and **–makefile**.

SPECIFYING KINDS OF OBJECTS TO DISPLAY. *Default:* **catcr** reports on all objects in the CR, which may include source files, directories, and symbolic links; derived objects; makefiles; view-private files; and non-MVFS objects that were explicitly declared as dependencies.

–typ·e { **f** | **d** | **l** } ...

Restricts the listing to files only (**f**), or to directories only (**d**), or to links only (**l**). If you omit **–type**, a **–short** listing includes files only and a **–long** listing includes all three kinds. To specify multiple kinds of objects, group them into a single argument: **–type fd**.

–ele·ment_only

Lists versions of elements only, including checked-out versions. This option excludes from the listing derived objects (except DO versions), view-private files and directories, symbolic links, and non-MVFS objects.

NOTE: If a view-private file listed in the CR is converted to an element after the creation of the CR, and has at least one checked-in version, it is considered to be an element and is listed by **–element_only**.

–vie·w_only

Lists view-private objects only, including checked-out versions of elements. If you specify this option along with **–element_only**, the listing includes only checked-out versions of elements.

–cri·tical_only

Excludes from the listing any objects marked as “noncritical” in the CR. Objects with that property typically have it because the user specified the objects as dependents of the **.NO_DO_FOR_SIBLING** special target in a **clearmake** makefile or an **omake** makefile.

–nam·e *tail-pattern*
: Restricts the MVFS objects listing to those whose final pathname component match the specified pattern. *tail-pattern* can include any of the wildcard characters described in the **wildcards_ccase** (ClearCase) or **wildcards** (Attache) reference page.

–scr·ipts_only
: Restricts the listing to the configuration record header and build script.

CONTROLLING REPORT APPEARANCE. *Default:* **catcr** reports, in three sections, on MVFS objects, variables and options, and the build script. The report uses full pathnames, and it omits comments and directory versions.

–zer·o
: Includes the CR header and options section, even if the specified filters remove all objects. The listing includes the target name, current view, and so on, but no information on particular file-system objects.

–wd
: Lists pathnames relative to the current working directory, rather than full pathnames. With **–makefile**, displays pathnames relative to the initial working directory of the build.

–nxn·ame
: Lists simple pathnames for MVFS objects, rather than version-extended pathnames or DO-IDs.

–fol·low
: Lists the link targets only (that is, suppresses printing of the link text).

–l·ong
: Expands the listing to include the kinds of objects in the CR and comments. With **–makefile**, adds comments only. For example, an object may be listed as a version, a directory version, or derived object. (See **ls –long** for a complete list.) Comments indicate whether an object is in a makefile, a referenced derived object, or a new derived object.

–s·hort
: Restricts the listing to file-system objects only (omits header information, variables and options, and build scripts). With **–makefile**, the listing also includes build scripts.

SPECIFYING THE DERIVED OBJECT. *Default:* None.

do-pname ...
: One or more pathnames, specifying the derived objects whose CRs are to be included in the listing. A standard or view-extended pathname specifies the DO in the view. An

extended pathname with a DO-ID specifies a particular DO, irrespective of view (for example, **hello.obj@@24–Mar.11:32.412)**.

Use the **lsdo** command to list derived objects with their DO-IDs.

do-pname can be a DO version, specified with any version-specification method (standard pathname, version-extended pathname, and so on).

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: Most examples show the same CR processed with different options. Some output lines have been split for clarity.

- List the CR for a derived object in the current view named **bgrs**.

  cmd-context **catcr bgrs**

```
Target bgrs built by jones.dvt
Host "oxygen" running SunOS 4.1.1 (sun4c)
Reference Time 11-Dec-94.12:02:39, this audit started 11-Dec-94.12:04:52
View was oxygen: /home/jones/views/920615.vws
Initial working directory was vob1/docaux/bgr/sun4
----------------------------
MVFS objects:
----------------------------
/vobs/docaux/bgr/libbgr/sun4/libbgr.a@@10-Dec.16:45.1893
/vobs/docaux/bgr/sun4/bgrs@@11-Dec.12:05.1956
/vobs/docaux/bgr/sun4/buga@@11-Dec.12:04.1926
.
.
.
```

```
/vobs/docaux/bgr/sun4/bugs.o@@11-Dec.12:03.1902
/vobs/docaux/bgr/sun4/bugsched.o@@11-Dec.12:04.1953
.
.
.
----------------------------
Variables and Options:
----------------------------
CC=/usr/bin/cc
CFLAGS=-I../I../libbgr -DBSD -DSCCS -g
ENV_LDFLAGS=../libbgr/sun4/.a
OBJECTS=main.o pick.o bugs.o bugr.o bugi.o bugf.o bugc.o bugl.o buge.o
bugd.o buga.o bugh.o bugw.o bugfld.o bugdt.o bugu1.o bugu2.o bugsched.o
----------------------------
Build Script:
----------------------------
/usr/bin/cc -I ../libbgr -DBSD -DSCCS -g main.o pick.o bugs.o
bugr.o bugi.o bugf.o bugc.o bugl.o buge.o bugd.o buga.o bugh.o bugw.o
bugfld.o bugdt.o bugu1.o bugu2.o bugsched.o
-o bgrs ../libbgr/sun4/libbgr.a
----------------------------
```

- Combine all CRs associated with **bgrs.exe** and its subtargets into a single listing.

  cmd-context **catcr -flat bgrs.exe**

```
----------------------------
MVFS objects:
----------------------------
1   \vob1\docaux\bgr\buga.c@@\main\1 <19-Dec-94.11:49:03>
1   \vob1\docaux\bgr\bugc.c@@\main\1 <19-Dec-94.11:49:09>
1   \vob1\docaux\bgr\bugd.c@@\main\1 <19-Dec-94.11:49:14>
20  \vob1\docaux\bgr\bugs.h@@\main\3 <17-Jun-94.23:55:22>
1   \vob1\docaux\bgr\bugsched.c@@\main\1 <19-Dec-94.11:50:07>
.
.
.
 2  \vob1\docaux\bgr\bugw.obj@@11-Dec.12:04.1932
 2  \vob1\docaux\bgr\main.obj@@11-Dec.12:03.1896
```

The integer at the beginning of an entry indicates the number of times the object was referenced during the build. For example, **\vob1\docaux\bgr\bugs.h** was referenced 20 times.

- Excerpt from the CR for the **bugsched.o** subtarget of **bgrs** the versions of elements involved in the build.

  cmd-context **catcr -select bugsched.o -element_only bgrs**

```
Target bugsched.o built by akp.user
Host "oxygen" running SunOS 4.1.1 (sun4c)
Reference Time 11-Dec-94.15:23:21, this audit started
11-Dec-.94.15:23:39
View was neptune:/usr/people/akp/views/940615.vws
Initial working directory was /vobs/docaux/bgr/sun4
----------------------------
MVFS objects:
----------------------------
/vobs/docaux/bgr/bugs.h@@/main/3 <17-Jun-94.23:55:22>
/vobs/docaux/bgr/bugsched.c@@/main/2 <11-Dec-94.15:23:04>
/vobs/docaux/bgr/libbgr/stint.h@@/main/2 <08-Sep-94.10:06:04>
----------------------------
Variables and Options:
----------------------------
CC=/usr/bin/cc
CFLAGS=-I../libbgr -DBSD -DSCCS -g
RM=rm -f
SRC=..

----------------------------
Build Script:
----------------------------
rm -f bugsched.o ; /usr/bin/cc -c -I../libbgr -DBSD -DSCCS -g
../bugsched.c
----------------------------
```

- List only header files (**.h** extension) involved in the build of a particular derived object.

  cmd-context **catcr -name '*.h' bgrs.exe**

```
----------------------------
MVFS objects:
----------------------------
20  \vob1\docaux\bgr\bugs.h@@\main\3 <17-Jun-94.23:55:22>
19  \vob1\docaux\bgr\libbgr\intstint.h@@\main\1 <19-Dec-94.11:54:50>
36  \vob1\docaux\bgr\libbgr\stint.h@@\main\2 <08-Sep-94.10:06:04>
1   \vob1\docaux\bgr\spar.h@@\main\1 <19-Dec-94.11:50:42>
```

- List the configuration record header and build script only.

  cmd-context **catcr –scripts_only sun5/clearmake**

```
Target clearmake built by bert.user
Host "bosco" running SunOS 5.7 (sun4u)
Reference Time 27-Mar-01.13:30:31, this audit started 27-Mar-01.13:30:52
View was bosco:/export/home/bert/views/bert_V5.0.BL1.vws
Initial working directory was /vobs/atria/bin/mmake/sun5
----------------------------
Build Script:
----------------------------
        @echo "linking clearmake"
```

SEE ALSO

clearaudit, **clearmake**, **config_spec**, **diffcr**, **ls**, **lsdo**, **rmdo**, **wildcards**, **wildcards_ccase**, *Building Software*

catcs

Displays the config spec of a view

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

catcs [**–tag** *view-tag*]

DESCRIPTION

The **catcs** command displays a view's config spec. This command does not require a product license.

In a dynamic view, if the working directory view differs from the set view (sometimes established with the **setview** command), **catcs** displays a warning and uses the working directory view. You cannot set a snapshot view; therefore, there is no distinction between the working directory view and the set view.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE VIEW. *Default:* Displays the config spec of the current view.

–tag *view-tag*
The view-tag of any view; the view need not be active.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Display the current view's config spec.

  cmd-context **catcs**

  ```
  element * CHECKEDOUT
  element * /main/LATEST
  ```

- Display the config spec of the view with view-tag **jackson_fix**.

  cmd-context **catcs -tag jackson_fix**

  ```
  element * CHECKEDOUT
  element * ...\rel2_bugfix\LATEST
  element * \main\LATEST -mkbranch rel2_bugfix
  ```

SEE ALSO

edcs, **lsview**, **mktag**, **pwv**, **setcs**, **setview**, **config_spec**

cc.icon, default icon

File type to icon mapping rules (graphical interface)

APPLICABILITY

Product	Command Type
ClearCase	data structure
ClearCase LT	data structure

Platform
UNIX

SYNOPSIS

file-type [*file-type* ...] : *icon-name* ;
.
.
.

DESCRIPTION

An icon file contains an ordered set of rules that maps file types to names of bitmap files, which contain icon bitmaps.

In **xclearcase**, a file browser uses a series of lookups to determine how to represent a file system object:

1. It searches one or more magic files to determine the list of file types for the file system object. (See the **cc.magic** reference page for details.)
2. It searches one or more icon files for a match with the first file type. Finding a match yields the name of a bitmap file. For example, this entry maps the file type `text_file` to the icon bitmap file name `text: text_file : -icon text ;`
3. The semicolon (`;`) that terminates an icon rule must be preceded by white space.
4. If no match can be found for the first file type, **xclearcase** searches the same set of icon files for a match with the second file type, and so on through the entire list of file types, if necessary. (If none of the file types produces a match in any icon file, an error occurs.)
5. Having determined the name of a bitmap file, **xclearcase** searches for an actual file in one or more directories containing bitmap files. (If it cannot locate a bitmap file with this name, an error occurs.)

6. Bitmap file names must have a numeric extension, indicating the size of the bitmap. For example, **text.40**. **xclearcase** selects that bitmap file whose name begins with the string specified by **–icon**, and whose size is 40x40 pixels.

If the file-system object is selected, this process includes an extra step: **xclearcase** tries to match a **–selected** icon rule for each relevant file type before accepting a bitmap specified by **–icon**. For example, the following rule specifies both generic and "when selected" icons for use with elements of type text_file:

```
text_file : -icon text -selected text_selected;
```

Selecting and deselecting a text_file object from a file browser switches between the two icons.

Search Paths

ClearCase and ClearCase LT support search paths both for icon files and for bitmap files:

- **Icon file search path** — If ICON_PATH is set in your environment (to a colon-separated list of directories), **xclearcase** searches files with a **.icon** extension in these directories. In each directory, files are processed in alphabetical order. As soon as a matching rule is found, the search ends; thus, if multiple rules match a file type, the first rule encountered is used.

  If ICON_PATH is not set, this default search path is used:

  `home-directory/.icon:${`*ccase-home-dir*`: /usr/atria}/config/ui/icon`

- **Bitmap file search path** — If BITMAP_PATH is set in your environment (to a colon-separated list of directories), **xclearcase** searches for bitmap files with a **.40** suffix in these directories.

  If BITMAP_PATH is not set, this default search path is used:

  `home-directory/.bitmaps:${`*ccase-home-dir*`:-/usr/atria}/config/ui/bitmaps`

EXAMPLES

- For file type **c_source**, use the icon file named **c**. When a **c_source** element is selected, use the icon file **c_select**.

  c_source : -icon c -selected c_select ;

- For file type **postscript**, use the icon file named **ps**.

  postscript : -icon ps ;

SEE ALSO

cc.magic

cc.magic, default.magic

File typing rules

APPLICABILITY

Product	Command Type
ClearCase	data structure
ClearCase LT	data structure

Platform
UNIX
Windows

SYNOPSIS

- File-typing rule:

  file-type-list **:** *selection-expression* **;**

- File type list:

  file-type [*file-type* ...]

- Selection expression:

  selection-op [*arg(s)*] [*logical-op selection-op* [*arg(s)*]] ...

DESCRIPTION

A magic file contains an ordered set of file-typing rules, which ClearCase and ClearCase LT use to determine a list of file types for an existing file system object, or for one that is about to be created (see the **mkeltype** reference page for information on predefined element types). A rule can use the object's name, its **file(1)** or **stat(2)** data, or its contents. File-typing involves searching one or more magic files for the first rule that matches a file-system object; finding a match yields a single file type or an ordered list of file types; failing to find a match produces an error.

On UNIX systems, file-typing is performed in the following situations:

- When you create a new element with **mkelem**, but do not specify an element type (with **–eltype**), the element's name is file-typed. (If you are converting a view-private file to an element with **mkelem –ci** or **mkelem –nco**, the file's contents are also used in the file-typing.) The resulting file type list is compared with the VOB's set of element types (which includes both element types in the VOB and element types in the Admin VOB hierarchy associated with the VOB). The first file type that matches an element type is chosen as the element type; if no file type matches any existing element type, an error occurs:

- The file browsers have a graphical mode, in which each file-system object is displayed as an icon. The icon is selected first by file-typing the object, and then using one of its file types to select a bitmap from the ones listed in an **icon** file. (See the **cc.icon** reference page.)

NOTE: In an MVFS directory (any directory accessed through a VOB-tag), file-typing by a browser uses only a file's name and its **stat** data; for performance reasons, the file's contents are not used.

On Windows, ClearCase and ClearCase LT perform file-typing when you create a new element with **mkelem**, but do not specify an element type (with **–eltype**). (If you are converting a view-private file to an element with **mkelem –ci** or **mkelem –nco**, the file's contents are also used in the file-typing.) The resulting file type list is compared with the VOB's set of element types (which includes both element types in the VOB and element types in the Admin VOB hierarchy associated with the VOB). The first file type that matches an element type is chosen as the element type; if no file type matches any existing element type, an error occurs:

```
cleartool: Error: Can't pick element type from rules ...
```

Following are examples of file-typing rules:

```
directory : -stat d ;
c_source source text_file : -printable & -name "*.c" ;
sh_script script text_file : -printable & (-name ".profile" | -name "*.sh") ;
archive library file: !-printable & -name "*.a" ;
```

Search Path

ClearCase and ClearCase LT support a search path for magic files. If **MAGIC_PATH** is set in your environment (to a colon-separated list of directories (UNIX) or to a semicolon-separated list of directories (Windows)), **xclearcase** or ClearCase searches files with a **.magic** extension in these directories. In each directory, files are processed in alphabetical order. As soon as a matching rule is found, the search ends. If multiple rules match a file type, the first rule encountered is used.

If **MAGIC_PATH** is not set, this default search path is used:

(UNIX) `home-directory/.magic:${`*ccase-home-dir*`:-/usr/atria}/config/magic`

(Windows) `%HOME%\.magic;`*ccase-home-dir*`\config\magic`

FILE-TYPING RULES

Each file-typing rule has the following format:

file-type-list **:** *selection-expression* **;**

A single text line can contain multiple rules. Conversely, a single rule can span several lines; each intermediate line must end with a backslash (\). A line that begins with a number sign (**#**) is a comment.

NOTE: The semicolon (**;**) that terminates a rule must be separated from the preceding characters by white space.

FILE TYPE LIST

A file-type-list is an ordered list of one or more names, separated by white space. Only letters, digits, and underscores (_) are permitted in these names. Depending on the file-typing situation, each name must match either an element type defined in some VOB, or an icon name specified in an icon file. To avoid errors, always make the final name one of the predefined element types (see the **mkeltype** reference page). These names are also included in the system-default icon file.

Following are some file-type-list examples:

```
text_file
backup_dir directory
manual_page text_file
cplusplus_src src_file text_file
```

Here is a UNIX system scenario that calls for a lengthy file type list:

Your host mounts several VOBs, in which different sets of element types are defined. Perhaps one VOB defines element type **bshell** for Bourne shell scripts, a second VOB defines element type **shell_script** for all shell scripts, and a third VOB does not define any special element type for scripts. Your file-typing rules must be appropriate for all the VOBs. For example:

```
bshell shell_script text_file : -name "*.sh" ;
shell_script text_file : -name "*.csh" ;
```

With the above file-typing rules, **xclearcase** uses the file type **text_file** to select the same icon for all shell script files. A user who wanted to distinguish Bourne shell scripts from C shell scripts may add a **cshell** file type, and create different bitmaps to correspond to the unique file types **bshell** and **cshell**.

Magic File:

```
bshell shell_script text_file : -name "*.sh" ;
cshell shell_script text_file : -name "*.csh" ;
```

Icon File:

```
bshell : -icon bourne_shell_icon ;
cshell : -icon C_shell_icon ;
```

SELECTION EXPRESSION

A selection-expression consists of one or more selection operators and their arguments, connected by
logical operators. Examples:

```
-name "*.c"
-name "*.[ch]"
-name "*.c" | -name "*.h"
-printable
!-printable
-stat d
```

Selection Operators and Arguments—General Information

Any abbreviation of a selection operator name is accepted. For example, you can abbreviate **–name** to **–n**, **–na**, or **–nam**.

All **string** arguments must be enclosed in double quotes. Use \" to include a double-quote character in a string argument.

If the file-system object already exists, any of the selection operators listed below can produce a match. If you are determining the file type for a nonexistent object (for example, an element that is about to be created with **mkelem**), only the **–name** operator can produce a match.

Selection Operators and Arguments—UNIX Dynamic Views Only

When an **xclearcase** browser performs file-typing in an MVFS directory to determine which icons to display, only the **–name** and **–stat** operators can produce a match. All other operators are invalid; expressions derived from the invalid operations with logical NOT and logical AND are also invalid, and can never produce a match. Examples:

- Both of the following rules always fail when applied by a ClearCase or ClearCase LT browser in a VOB directory:

```
text_file : -printable ;
file : ! -printable ;
```

- This rule always fails when applied by a ClearCase or ClearCase LT browser in a VOB directory, because **–magic** is invalid:

```
xyz_file : -magic 0, "<xyz>" & -name "*.xyz" ;
```

- When applied by a browser in a VOB directory, the subexpression before the logical OR never produces a match, but the subexpression after the logical OR can produce a match, making the entire expression `TRUE`:

```
bourne_shell : (-magic 0, "#!/bin/sh" & -stat f) | -name "*.sh"
```

–name *pattern*

Matches an object's simple filename (leaf name) against *pattern*. *pattern* is a double-quoted string, and can include any ClearCase/ClearCase LT wildcard, except for the ellipsis (**...**). See the **wildcards_ccase** reference page for a complete list.

–stat *stat_char*

Matches an object against the specified **stat** file type. **stat_char** is a single character:

r	Regular file
d	Directory
c	Character device; not supported for Windows files
b	Block device; not supported for Windows files
f	FIFO (named pipe); not supported for Windows files
s	Socket; not supported for Windows files
l	Symbolic link

NOTE: The types **c**, **b**, **f**, and **s** can be used on Windows if they are pointed at a UNIX file.

On UNIX systems, the selection expression **–stat l & –stat r** is `TRUE` for a symbolic link that points to a regular file. In general, however, testing for symbolic links is not particularly useful. **xclearcase** displays an icon for the object it finds at the end of a chain of symbolic links.

–magic *byte_offset*, *data_type*, *value*
–magic *byte_offset*, *string*

Matches an object against a magic value: a number or string at a specified offset within the object's first physical block (512 bytes).

byte_offset	The byte offset from the beginning of the file.	
data_type	The architecture-specific data format of the numeric *value* argument that follows:	
	byte	*value* is an 8-bit byte.
	l_short	*value* is a little-endian 16-bit shortword.
	l_long	*value* is a little-endian 32-bit longword.
	b_short	*value* is a big-endian 16-bit shortword.
	b_long	*value* is a big-endian 32-bit longword.
value	A numeric magic value, expressed as an integer in hexadecimal, octal, or decimal:	
	0x ...	A hexadecimal value

	0 ...	An octal value
	...	(Any other form) A decimal value
string	A nonnumeric magic value, expressed as a double-quoted string.	

–printable

Matches an object if it is a printable file:

- Its first block must contain only characters evaluating to `TRUE` by the X/Open **isprint** and **isspace** routines.
- Its first block must have an average line length <= 256.

Remember that **mkelem** can create an element object that corresponds to an empty (and therefore unprintable) file.

–token *string*

Matches an object if the specified double-quoted string occurs in its first physical block (512 bytes).

–file *string* (UNIX only)

Matches an object if the leading characters in its **file(1)** command output match the specified double-quoted string.

Logical Operators

File-typing rules can use the following logical operators, listed in decreasing order of precedence:

(0)	Parentheses for grouping
!	Unary NOT
&	Logical AND
&&	Logical AND
|	Logical OR
||	Logical OR

NOTE: The effect of the unary NOT operator may depend on whether or not an object exists. It cannot produce a match if the selection operator is inappropriate. For example, on UNIX systems, attempting to get the file status of a nonexistent object:

`! -stat f` *(produces a match when file-typing the name of an existing directory)*
`! -stat f` *(fails to produce a match when file-typing a name for which no object currently exists)*

EXAMPLES

- Assign the file types **source_file** and **text_file** to files whose file-name extension is **.c** or **.h**.

```
source_file text_file : -name "*.c" | -name "*.h" ;
```

- Assign the file types **cplspls_source** and **text_file** to printable files whose file-name extension is **.cxx** or **.c++**.

```
cplspls_source text_file : -printable & (-name "*.cxx" | -name "*.c++") ;
```

- Assign the file types **csh_script** and **text_file** to printable files that begin with the character string **#!**, and whose first block contains the string **csh**.

```
csh_script text_file : -printable & -magic 0,"#!" & -token "csh" ;
```

- Assign the file type **directory** to all directory objects.

```
directory : -stat d ;
```

- Assign the file types **cpio** and **file** to objects that the standard UNIX **file(1)** programs reports as "cpio archive".

```
cpio file : -file "cpio archive" ;
```

- Assign the file types **doc_file** and **text_file** to printable files with the file-name extension **.txt** or **.doc**.

```
doc_file text_file : -printable & (-name "*.doc" | -name "*.txt");
```

FILES

*ccase-home-dir***/config/magic/default.magic**

SEE ALSO

cc.icon, **mkelem**, **mkeltype**, **wildcards_ccase**, **file(1), stat(2)**

cd

Changes the current working directory

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command
MultiSite	multitool subcommand

Platform
UNIX
Windows

SYNOPSIS

cd [*dir-pname*]

DESCRIPTION

The **cd** command works differently depending on whether you are using a dynamic view or a snapshot view.

Changing Directories in a Dynamic View

The **cd** command changes the current working directory, as does the UNIX **cd(1)** command. In Attache, the current directory is changed both in the workspace and the view, and is set as the command prompt in the Command Window. In ClearCase and MultiSite, this command is intended for use in interactive **cleartool** and **multitool** sessions and in shell scripts or batch files that simulate interactive sessions.

With a view-extended pathname, **cd** also changes your working directory view. The specified view's config spec determines which versions of elements are visible in your new working directory.

With a version-extended pathname that specifies an element or branch, **cd** changes your current working directory to a location in version-extended namespace, wherein element and branch names are treated like directories in a read-only file system. The best way to leave version-extended namespace is to change directories to a full pathname. Typing **cd ..** does not exit version-extended namespace until you ascend past the VOB root directory. (See the **pathnames_ccase** reference page.)

Changing Directories in a Snapshot View

The **cd** command changes the current working directory. If *dir-pname* specifies a snapshot view, **cd** changes the view context to that of the snapshot view.

View Selection Precedence

Regardless of the view type, view-selection precedence is as follows:

1. If you specify a a view-extended name, that view is used.
2. Otherwise, the view implied by the current directory is used.
3. Otherwise, the view that has been set (if any) is used.

Attache's Client Process Startup Directory

A separate startup directory is associated with the Attache client process. This directory changes depending on how Attache is started. For example, it is the working directory specified in Attache's program item properties if Attache is started from the icon. Once the Attache client process is started, this directory never changes.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE NEW WORKING DIRECTORY. *Default:* Changes to your home directory in ClearCase, ClearCase LT and MultiSite; or the home directory on the helper host in Attache, determined in UNIX by examining the password database and in Windows by the value of the **HOME** environment variable, the value of the **HOMEDRIVE** and **HOMEPATH** environment variables, or the user profile.

dir-pname

The pathname of the directory to become your current working directory. You can specify a view-extended or version-extended pathname, as described above.

In Attache, the pathname may or may not exist locally, and may even be invalid on the local file system. No error occurs unless you try to download a file at that pathname.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Change to your home directory.

  cmd-context **cd**

- Change to the **release** subdirectory of the current working directory's parent.

  cmd-context **cd ..\release** *(in Attache, type this command at a workspace prompt)*

- (ClearCase/ClearCase LT/MultiSite) Use a view-extended pathname to change to the **src** directory in the context of the **jackson_old** view.

  cmd-context **cd /view/jackson_old/usr/hw/src**

- Change to the directory in extended namespace that represents the **main** branch of element **hello.c**.

  cmd-context **cd hello.c@@\main** *(in Attache, type this command at a workspace prompt)*

- (ClearCase/ClearCase LT/MultiSite) Change to a directory in extended namespace, and then return to the original directory.

  cmd-context **cd src@@**

  cmd-context **pwd**
  `/view/jackson_vu@@/usr/hw/main/2/src`

  cmd-context **cd /usr/hw/src**

  cmd-context **pwd**
  `/usr/hw/src`

- (Attache) Change to a directory in extended namespace, and then return to the original directory.

  `\vobs1\>` **cd src@@**

  `\vobs1\src@@>` **pwd**
  `\vob1\src@@`

  `\vobs1\src@@>` **cd \vob1\hw\src**

  `\vobs1\hw\src>` **pwd**
  `\vob1\hw\src`

SEE ALSO

attache_command_line_interface, **attache_graphical_interface**, **cd (1)**, **config_spec**, **pathnames_ccase**, **pwd**, **pwv**, **setview**, *Administrator's Guide*

chactivity

Changes an activity

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

chact·ivity [**–c·omment** *comment* | **–cfi·le** *pname* | **–cq·uery** | **–cqe·ach** | **–nc·omment**]
{ [**–hea·dline** *headline activity-selector …*] |
[**–fcs·et** *src-activity-selector* **–tcs·et** *dest-activity-selector version-pname*[**,**...] }

DESCRIPTION

The **chactivity** command modifies one or more activities. Use this command for these tasks:

- Changing an activity's headline
- Moving versions from the change set of one activity to the change set of another activity

Note that changing the headline for an activity does not affect its name (its unique identifier). For information about changing the activity's name, see **rename**.

The destination activity must exist before you can move a change set and both the source and destination activities must be in the same stream. Use **lsactivity –long** to list the pathnames of change-set versions associated with an activity.

RESTRICTIONS

Identities: No special identity required.

Locks: An error occurs if there are locks on any of the following objects: the UCM project VOB, the activity.

Mastership: (Replicated VOBs only) Your current replica must master the activity.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default*: Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–nc**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
: Overrides the default with the option you specify. See the **comments** reference page.

MODIFY AN ACTIVITY'S HEADLINE. *Default*: None.

–hea·dline *headline*
: Specifies a new headline for the activity. The *headline* argument can be a character string of any length. Use double quotes to enclose a headline with spaces or special characters.

SPECIFYING THE ACTIVITY. *Default*: None.

activity-selector ...
: Specifies one or more activities to modify.

You can specify an activity as a simple name or as an object selector of the form [**activity**]**:***name@vob-selector*, where *vob-selector* specifies a project VOB (see the **cleartool** reference page). If you specify a simple name and the current directory is not a project VOB, then this command assumes the activity resides in the project VOB associated with the stream attached to the current view. If the current directory is a project VOB, then that project VOB is the context for identifying the activity.

SPECIFYING THE SOURCE AND DESTINATION ACTIVITIES. *Default*: None.

–fcs·et *src-activity-selector*
: Specifies the activity from which to move versions.

You can specify an activity as a simple name or as an object selector of the form [**activity**]**:***name@vob-selector*, where *vob-selector* specifies a project VOB (see the **cleartool** reference page). If you specify a simple name and the current directory is not a project VOB, then this command assumes the activity resides in the project VOB associated with the stream attached to the current view. If the current directory is a project VOB, then that project VOB is the context for identifying the activity.

–tcs·et *dest-activity-selector*
: Specifies the activity to move versions to. These versions are recorded in the activity's change set.

You can specify an activity as a simple name or as an object selector of the form [**activity**]**:***name@vob-selector*, where *vob-selector* specifies a project VOB (see the **cleartool** reference page). If you specify a simple name and the current directory is not a project VOB, then this command assumes the activity resides in the project VOB associated with the stream attached to the current view. If the current directory is a project VOB, then that project VOB is the context for identifying the activity.

version-pname[**,**...]
: One or more version-extended pathnames that specify the versions to be moved to another change set.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Change an activity's headline.

  cmd-context **chactivity -headline "Fix front matter" fix_copyright**

  ```
  Changed activity "fix_copyright".
  ```

- Move a version from one activity's change set to another activity's change set.

  cmd-context **chactivity -fcset update_date **
  -tcsets fix_copyright add_proc@@/main/chris_webo_dev/1

  ```
  Moved version "add_proc@@/main/chris_webo_dev/1" from activity
  "update_date" to activity "fix_copyright".
  ```

SEE ALSO

lsactivity, **mkactivity**, **rename**, **rmactivity**

chbl

Changes a baseline

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

chbl [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**]
{ [[**–inc·remental** | **–fu·ll**] [**–nre·curse**]] [**–level** *promotion-level*] }
baseline-selector ...

DESCRIPTION

The **chbl** command modifies one or more baselines. You can modify a baseline's labeling status or assign a new promotion level to a baseline.

For information about changing the baseline's name, see **rename**.

Baseline Labels

Baselines can be unlabeled, incrementally labeled, or fully labeled. Only labeled baselines can be used to configure streams (see **rebase** and **mkstream**). After they are applied, baseline labels cannot be moved.

Promotion Levels

Promotion levels must be defined in the baseline's project VOB before they can be applied to baselines. For information on promotion levels, see the **setplevel** reference page.

The promotion levels available in a VOB can be listed by running the **describe** command on the UCM project VOB object.

RESTRICTIONS

Identities: You have one of the following identities:

- Project VOB owner
- Baseline owner
- Stream owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows only)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows only)

Locks: An error occurs if there are locks on any of the following objects: the UCM project VOB, the baseline.

Mastership: (Replicated VOBs only) Your current replica must master the baseline.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default*: Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–nc**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
: Overrides the default with the option you specify. See the **comments** reference page.

CHANGING A BASELINE'S LABELING STATUS. *Default*: None.

–inc·remental
: Changes the labeling status for an unlabeled baseline to incremental. This option has no effect if the baseline is already incrementally or fully labeled.

–fu·ll
: Changes the labeling status for a baseline from unlabeled or incremental to full. This option has no effect if the baseline is already fully labeled. A **chbl –full** operation may take a long time for components with many elements.

–nre·curse
: By default, when **–incremental** and **–full** are used on a composite baseline, the specified change is applied recursively to its member baselines. The **–nrecurse** option is used to prevent the default behavior. It causes the changes to apply only to the composite baseline and not its members.

ASSIGNING PROMOTION LEVELS. *Default*: No change in promotion level.

–level *promotion-level*
: Sets the promotion level for the specified baselines. The specified promotion level must be defined in the baseline's project VOB. When used on a composite baseline, the change is not applied recursively to its members.

SPECIFYING THE BASELINE. *Default*: None.

baseline-selector ...

Specifies one or more baselines to modify.

baseline-selector is of the form [**baseline:**]*baseline-name*[@*vob-selector*], where *vob-selector* specifies the baseline's project VOB.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Change an unlabeled baseline to be incrementally labeled. The baseline specifier includes a VOB component, which must be the baseline's project VOB.

  cmd-context **chbl -incremental testbl.121@/vobs/core_projects**

  ```
  Begin incrementally labeling baseline "testbl.121".
  Done incrementally labeling baseline "testbl.121".
  ```

- Change a baseline's promotion level and check the labeling status. The baseline specifier includes a VOB component, which must be the baseline's project VOB.

  cmd-context **chbl -full -level TESTED testbl.121@\vobs\core_projects**

  ```
  Change baseline "testbl.121".
  Baseline "testbl.121" is already fully labeled.
  ```

SEE ALSO

describe, **diffbl**, **lsbl**, **lscomp**, **mkbl**, **rmbl**, **setplevel**

checkin

Creates a permanent new version of an element

APPLICABILITY

Product	**Command Type**
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

checkin | **ci** [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery**
| **–cqe·ach** | **–nc·omment**] [**–nwa·rn**]
[**–cr**] [**–pti·me**] [**–kee·p** | **–rm**] [**–fro·m** *source-pname*]
[**–ide·ntical**] { **–cact** | *activity-selector* ... | *pname* ... }

DESCRIPTION

To create a new version of an element, **checkin** makes changes in the VOB and in the view.

Actions Taken in the VOB

For one or more elements, **checkin** creates a successor to a version that was previously checked out in the current view: the predecessor version. The version number of the successor is the next unused number on the branch. (If one or more versions have been deleted from the end of the branch with **rmver**, it may seem that some version numbers have been skipped.) An appropriate message is displayed:

```
Checked in "msg.c" version "/main/bugfix/26".
```

In Attache, any existing local files corresponding to pathname arguments are uploaded before performing the **checkin** remotely; directories are not uploaded.

A checkin event record is created, which can be listed with the **lshistory** command:

cmd-context **lshistory msg.c**

```
06-Aug.12:09 akp create version "msg.c@@\main\bugfix\26"
```

.
.
.

Only elements can be checked in. You cannot check in a view-private or local file; you must first make an element of the same name. Use the **mkelem –ci** command to simultaneously create an element and check in a view-private or local file as its first version.

Actions Taken in the View

checkin works differently in different contexts.

- **Dynamic view.** By default, the new version of a file element is created by copying the contents of the view-private file named *pname* (the checked-out version) to the VOB, and then deleting that file. The **–keep** and **–from** options alter this behavior.
- **Snapshot view.** By default, the new version of a file element is created by copying the contents of the file named *pname* (the checked-out version) to the VOB. The checked-in version remains in the view. (This version may not be the one specified by the config spec.) The **–keep** and **–from** options alter this behavior. If multiple instances of this file element are loaded into the view (because the load rules specify a hard-linked UNIX file or a linked Windows file in more than one location), **checkin** updates each instance of the file with the checked-in version.
- **Attache**. By default, the new version of a file element is created by uploading the local file named *pname* to the view, copying the contents of the uploaded view-private file named *pname* (the checked-out version) to the VOB, deleting that file in the view, and then setting the local file to read-only. The **–keep** and **–from** options alter this behavior.

After the element is checked in, your view typically selects the version you just created. However, in a dynamic view and Attache it is possible that your view selects another version (perhaps on another branch), because that version is specified by your config spec rules. In this case, **checkin** displays a warning message. In Attache, the workspace copy is not updated.

In Attache, a warning is issued for each argument that has no corresponding local file, but the command will still execute remotely. For each successfully checked-in version, the local file is changed to be read-only.

METADATA AND THE CHECKED-IN VERSION

From the viewpoint of the VOB database, the new, checked-in version is the same object as the checked-out version. Thus, any metadata items (version labels, attributes, hyperlinks) that were attached to the checked-out version remain attached to the new version. And, for example, **checkin** followed by **mklabel** is equivalent to **mklabel** followed by **checkin**.

CHECKIN OF RESERVED AND UNRESERVED CHECKOUTS

At the time you enter a **checkin** command, there may be several checkouts of the same version. At most one of the checkouts (perhaps yours) is reserved; all the others are unreserved. Your **checkin** command succeeds in either of these cases:

- Yours was a reserved checkout.
- All checkouts were unreserved, and no one has checked in a successor version.

If the command fails because someone else has a reserved checkout, you must wait until that checkout is resolved, with **checkin**, **uncheckout**, or **unreserve**. If the command fails because someone has checked in a successor version ahead of you, you can check in your work by performing the following steps:

1. Merge from the current **LATEST** version on the branch to your checked-out version.
2. Enter the **checkin** command again.

CHECKIN OF DERIVED OBJECTS

(Dynamic views and Attache) You can check in a derived object to make it a version of an element (a DO version). By default, both the data and configuration record of a derived object are checked in. To save disk storage, you can use the **–cr** option to check in only the configuration record, not the data. Checking in a nonshareable DO converts the DO, its sibling DOs, and its sub-DOs to shareable DOs.

clearmake can reuse or winkin a derived object only if it is stored under its original pathname. Thus, a DO version created under an alternate name with **checkin –from** cannot be used by **clearmake** for build avoidance. (**clearmake** can still use the derived object named in the **–from** option, which is unaffected by this command.)

See the **mkelem** reference page for information on creating a file element for a DO, and see *Building Software* for information regarding subsequent operations on DO versions.

RESTRICTIONS

Identities: You must have one of the following identities:

- User who checked out the element
- Element owner
- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

Additional restrictions on UNIX:

- If the element's **set-UID** bit is set, only the element's owner, the VOB owner, or **root** can check in the version.

- If the element's **set-GID** bit is set, only a member of the element's group, the VOB owner, or **root** can check in the version.

Locks: An error occurs if one or more of these objects are locked: VOB, element type, branch type, element, branch, pool (file elements).

Mastership: (Replicated VOBs) Your current replica must master the branch on which you are checking in the version.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, with commenting controlled by your home directory's **.clearcase_profile** file in ClearCase and ClearCase LT or your remote home directory's **.clearcase_profile** file in Attache (default: **–cqe**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**

Overrides the default with the option you specify. See the **comments** reference page.

NOTE: If a checkout comment exists (specified with the **checkout** command and/or generated to record changes to a checked-out directory), you can make it the checkin comment by using either of the following commands:

- **checkin –nc**
- **checkin –cqe**; at the prompt, press CTRL+D (UNIX) or CTRL+Z ENTER (Windows), or .RETURN
- **checkin**; at the prompt, press CTRL+D (UNIX) or CTRL+Z ENTER (Windows), or **.**RETURN

Any other entry at the **–cqe** prompt specifies a new checkin comment, discarding the checkout comment (if any) for that element. The **–c** and **–cq** options always discard the checkout comment (if any) for each element processed.

SUPPRESSING WARNING MESSAGES *Default:* Warning messages are displayed.

–nwa·rn

Suppresses warning messages.

CHECKING IN DERIVED OBJECTS. *Default:* **checkin** checks in both the data and configuration record for a derived object.

–cr (For derived-object checkin)

Checks in only the configuration record for the specified derived objects. Each new DO version will have a configuration record, but no data. You can use many **cleartool** commands with such DO versions, such as **catcr**, **diffcr**, and **mklabel** (but not **lsdo**). DO versions are also visible when you use the **ls** command. However, a version created with this option cannot be opened or executed, because there is no data.

MANAGING SOURCE FILES. *Default:*

- In a dynamic view, **checkin** deletes each view-private, checked-out *pname* file after using it to create a new version.
- In a snapshot view, **checkin** uses the checked-out *pname* file to create a new version, then loads the checked-in version into the view.

You can use the following options (which have no meaning for directory elements) to save view-private copies, or to check in source files from other locations.

–kee·p

Saves the current contents of each checked-out version in a view-private file, in addition to creating a new version. The view-private file gets a name of the form *pname***.keep** (or possibly, *pname***.keep.***n*). In Attache, this file is not downloaded to the workspace. **–keep** is the default when you use the **–from** option, because the current contents of the checked-out version would otherwise be lost.

–rm

Removes each *pname* file after creating a new version. In a dynamic view, this is the default if you do not use the **–from** option. This option does not affect the Attache workspace.

–fro·m *source-pname*

Uses the contents of *source-pname* as the new version, instead of the view-private file *pname*. By default, **–keep** is invoked to preserve the contents of the view-private *pname*. In Attache, if *source-pname* exists in the workspace, it is uploaded first. The *source-pname* file itself is not affected. This option makes it easy to copy data from another location (outside the VOB, perhaps) into an element's version tree.

When using this option, specify only one *pname* argument.

NOTES:

- In a snapshot view, you cannot use a view-extended pathname as *source-pname*.
- This option will not work on directories.

MISCELLANEOUS OPTIONS. *Default:* **checkin** resets the new version's modification time to the check-in time. Also, **checkin** cancels the checkin operation for files managed by certain type managers, if the contents of the files match their predecessor versions.

–pti·me

Preserves the modification time of the file being checked in. If you omit this option, **cleartool** or Attache sets the modification time of the new version to the checkin time.

NOTE: On some UNIX platforms, it is important that the modification time be preserved for archive files (libraries) created by **ar(1)** (and perhaps updated with **ranlib(1)**). The link editor, **ld(1)**, will complain if the modification time does not match a time recorded in the archive itself. Be sure to use this option, or (more reliably) store archive files as elements of a user-defined type, created with the **mkeltype –ptime** command. This causes **–ptime** to be invoked when the element is checked in.

–ide·ntical

Checks in the element even if the predecessor version is identical to the checked-out version. By default, the checkin operation is canceled in such cases.

NOTE: This situation applies only to elements whose **type manager** computes version-to-version deltas (for example, elements of type **text_file**, **binary_delta_file**, and **compressed_text_file**). If an element's type manager does not compute deltas, **checkin** always creates a new version, whether or not it is identical to its predecessor. For example, a new version is always created for an element of type **file**, which uses the **whole_copy** type manager.

SPECIFYING OBJECTS TO CHECK IN. *Default:* None.

–cact

Checks in each checked-out version in the change set of the current UCM activity in your view.

activity-selector ...

Checks in each checked-out version in the change set of each specified activity. Specify *activity-selector* in the form **activity:***activity-name*[@*vob-selector*]

activity-name	name of the activity

pname ...

The pathnames of one or more elements to be checked in.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- After verifying its checkout comment, check in element **util.c**, using that comment.

  cmd-context **lscheckout –long util.c**
  ```
  10-May-99.16:11:07 Chuck Jackson (jackson.dvt@oxygen)
  checkout version "util.c" from /main/4 (reserved)
  by view: "oxygen/home/jackson/cj.vws"
  "revise syntax"
  ```
  cmd-context **checkin –nc util.c**
  ```
  Checked in "util.c" version "\main\5".
  ```
- Check in an element from an alternate file, discarding the checked-out version. Provide a comment on the command line.

  cmd-context **checkin –rm –from c:\users\cep\util.c –c "Release 1.1 update" util.c**
  ```
  Checked in "util.c" version "\main\6".
  ```
- (ClearCase) Check in only the configuration record of a derived object, discarding its data.

  cmd-context **checkin –nc –cr hello**
  ```
  Checked in "hello" version "/main/1".
  ```

SEE ALSO

attache_command_line_interface, **attache_graphical_interface**, **checkout**, **clearmake**, **config_spec**, **get**, **lshistory**, **merge**, **mkelem**, **mkeltype**, **mklabel**, **profile_ccase**, **put**, **rmver**, **uncheckout**

checkout

Creates a modifiable copy of a version

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

checkout | **co** [**–res·erved** | **–unr·eserved** [**–nma·ster**]]
[**–out** *dest-pname* | **–nda·ta**] [**–pti·me**]
[**–bra·nch** *branch-pname* | **–ver·sion**] [**–nwa·rn**]
[**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**]
[**–q·uery** | **–nq·uery**]
pname ...

DESCRIPTION

For one or more elements, the checkout **command** checks out a branch (typically, the most recent version on a branch). In most cases, this creates a writable copy of that version in the current view (the checked-out version), but see the *CHECKING OUT A DO VERSION* section. An appropriate message is displayed. For example:

```
Checked out "msg.c" from version "/main/bugfix/25"
```

If you are checking out in a UCM view, the view must be set to a UCM activity (see **setactivity**). Checked-out elements are added to the change set of the UCM activity you set.

In Attache, files checked-out successfully are downloaded to the workspace after the **checkout** command is executed remotely. If a local file exists, and is both writable and different from the checked-out version, the user is queried before the local file is overwritten, but the file is always checked out in the view. Checked-out directories are created locally if they do not already exist in the workspace. All downloaded files are made writable.

A checkout record is created; it can be listed with the **lscheckout** command:

cmd-context **lsco msg.c**

```
05-Aug.20:50 akp checkout version "msg.c" from \main\bugfix\25 (reserved)
```

If a view-private object already exists with the same name as an element being checked out, **checkout** responds as follows:

- In a dynamic view, it displays this error message:

  `Not a vob object:` *pname*

  To check out the element, rename or remove the view-private object with the standard operating system command and enter the **checkout** command again.

- In a snapshot view, the behavior is different for view-private directories and view-private files:

  - A view-private directory that corresponds to a directory in the VOB namespace is checked out. That is, **checkout** creates a checkout record in the VOB for the directory element. Any changes to the checked-out directory in the view are added to the VOB at checkin.

  - A view-private file with the same name as an element being checked out is treated as a hijacked file. **checkout** asks whether you want to use the file as the checked-out version; if you do not, the view-private file is renamed.

- In Attache, **checkout** saves the private object in the view, not in the workspace.

Before using a command that changes the contents of a directory (**mkelem**, **mkdir**, **rmname**, **ln**, or **mv**), you must first check out the directory. Each of these commands appends an appropriate line to the directory's checkout comment. For example, using **mkelem** to create a new element within a directory adds a line like this one:

```
Added file element "wel.c".
```

RESERVED AND UNRESERVED CHECKOUTS

A version can have at most one reserved checkout and any number of unreserved checkouts. Performing a reserved checkout (without using the **–version** option) guarantees you the right to create a successor to the version you checked out. If several users perform unreserved checkouts, any one of them (and only one) can create a successor version.

The predecessor version of your checked-out file may not be the latest on the branch from which you checked out your version; this situation can occur if the **–version** option or unreserved checkouts are used. In this case, you must merge from the latest version on the branch to your checked-out version before you can check in your version.

You can change the reserved state of a checked-out version with the **reserve** and **unreserve** commands.

MultiSite: Checking Out a Branch Mastered at Another Site

If the VOB containing the element is replicated, the **checkout** command fails if you try to check out a branch mastered by a different replica:

cleartool checkout –nc file1.txt

```
cleartool: Error: Unable to perform operation "checkout" in replica
"lexington" of VOB "/vobs/dev".
cleartool: Error: Master replica of branch "/main" is "london".
cleartool: Error: Unable to check out "file1.txt".
```

If you need to do work on a branch mastered by another replica, you have two choices:

- Request mastership of the branch and wait until the mastership is transferred to your current replica before checking out the branch.
- Check out the branch and do your work while waiting for mastership to be transferred. You can request mastership before or after checking out the branch. To check out the branch, use **checkout –unreserved –nmaster**, which performs a nonmastered checkout. When the mastership of the branch is transferred to your current replica, you may have to perform a merge before checking in your work. Therefore, do not use this option if you cannot merge versions of the element (for example, if the versions are in binary format).

To request mastership, ask the administrator at the master replica to transfer mastership, or use the **reqmaster** command. Consult your ClearCase administrator to make sure mastership requests with **reqmaster** are enabled and that the replicas are at the correct feature level.

NONSTANDARD CHECKOUTS

By default, the **checkout** command checks out these versions:

- The most recent version on a branch, if you are using a dynamic view
- The version currently loaded in the view, if you are using a snapshot view

To modify a different version, you can either use the **–version** option or create a subbranch at that version. (See the **mkbranch** reference page). Furthermore, from a single view, you can have only one checkout per element at a time.

NOTE: When working in a snapshot view, the only version of a directory element that can be checked out is the version currently loaded in the view. Therefore, the **–version** and **–branch** options will not work in this case.

When you use the **–version** option, you can specify the version either by setting your config spec to use that version, or by specifying a version-extended pathname as the *pname* argument. After you make your changes, you must merge from the latest version of the element before you can perform a checkin.

You can check out a version that your config spec does not currently specify, either by using the **–branch** option or by specifying a *pname* argument that is a branch pathname (for example, **msg.c@@/main/rel4_bugfix**). In such cases, a warning message appears:

```
cleartool: Warning: Version checked out is different from version previously
selected by view.
```

CHECKING OUT A DO VERSION

(Dynamic view) If the version being checked out is a derived object (DO version), **checkout** attempts to winkin the DO to your view. If it cannot perform the winkin, it copies the DO's data instead. A winkin cannot be performed if you use the **–out** option to specify a destination in another VOB, or in a non-VOB location, such as **/tmp**.

See *Building Software* for additional information on the behavior of checked-out DO versions.

AUTO-MAKE-BRANCH

If the config spec specifies a version using a rule with a **–mkbranch** *branch-type* clause (see also **config_spec**), **checkout** works as follows:

1. Creates a branch of type *branch-type*.
2. Checks out (version 0 on) the new branch.

Except for some extra messages, the behavior is no different from an ordinary checkout. The checked-out version has the expected contents, because version 0 on the new branch has the same contents as the version at the branch point.

NOTE: (MultiSite sites) If the VOB is replicated, the current replica must master all the branch types specified in your config spec. Otherwise, auto-make-branch fails.

Multiple-Level Auto-Make-Branch

A config spec can include a cascade of auto-make-branch rules, causing **checkout** to create multiple branching levels at once. **checkout** keeps performing auto-make-branch until version 0 on the newly created branch is not selected by a rule with a **–mkbranch** clause. For example:

```
1    element * CHECKEDOUT
2    element * .../br2/LATEST
3    element * .../br1/LATEST –mkbranch br2
4    element * MYLABEL –mkbranch br1
5    element * /main/LATEST
```

If you check out an element in a view that currently selects the version labeled **MYLABEL**:

1. A branch of type **br1** is created at the **MYLABEL** version (Rule 4).
2. Rule 3 now selects the newly-created version **.../br1/0**, so a branch of type **br2** is created at that version.

3. Version **.../br1/br2/0** is checked out. The checked-out version has the same contents as the **MYLABEL** version, and is selected by Rule 1. When you edit and check in a new version, **.../br1/br2/1**, the view selects it with Rule 2.

CHECKED-OUT FILES

Any checked-out file can be read, edited, and deleted like any ordinary file.

Dynamic Views

You have write permission on a checked-out file only if you have write permission on the set view's view storage directory. If you have write permission on the view storage directory for the view you are using, you have write permission on a checked-out file in that view.

Snapshot Views

The initial permissions on the checked-out file are determined by this algorithm:

- Start with the permissions of the element itself. (See the **mkelem** and **protect** reference pages.)
- Add a write permission wherever the element itself has a read permission (user, group, and/or other).
- (UNIX) Subtract read, write, and/or execute permissions according to your current **umask(1)** value.

You can change the permissions of the checked-out file with the standard UNIX **chmod(1)** command or by changing the Windows file properties, but you must use the **protect** command to change the permissions of the element itself.

Attache

A checked-out file is a workspace-local object.

CHECKEDOUT BUT REMOVED FILES

There may be no object in the view located at the pathname of the checked-out version. This can happen if any of these conditions are true:

- You have deleted the file.
- You renamed the file.
- You used **checkout –out** to copy the checked-out version to another location.
- You used **checkout –ndata** to create only a checkout record for the version.
- A permission problem occurred and **checkout** was unable to write the file. In this case, cancel the checkout (use **uncheckout**), fix the permission problem, and check out the file again.

An OS-level listing command does not show the missing file, but the **cleartool ls** and Attache **ls** commands display the pathname of the checked-out version with the notation " checked out but removed."

RESTRICTIONS

Identities: You must have one of the following identities:

- Element owner
- Element group member
- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

Additional restrictions on UNIX:

- If the element's **set-UID** bit is set, only the element's owner, the VOB owner, or **root** can check out the version.
- If the element's **set-GID** bit is set, only a member of the element's group, the VOB owner, or **root** can check out the version.

Locks: An error occurs if one or more of these objects are locked: VOB, element type, branch type, element, branch.

Mastership: (Replicated VOBs) Your current replica must master the branch you are checking out unless you use **–unreserved –nmaster**.

OPTIONS AND ARGUMENTS

RESERVED AND UNRESERVED CHECKOUTS. *Default:* **checkout** reserves the branch unless a different default has been specified in **profile_ccase**.

–res·erved

Reserves the branch: no user in another view can perform a reserved checkout of the same branch (but any number of unreserved checkouts can be performed); no new versions can be created on the branch until your checkout is changed to unreserved with **unreserve** or resolved with **checkin** or **uncheckout**.

–unr·eserved [–nma·ster]

Leaves the branch unreserved: other users, in other views, can check out the same version (but at most one of the checkouts can be reserved).

With **–nmaster**, checks out the branch even if the current replica does not master the branch. Do not use this option if you cannot merge versions of the element.

See the **checkin** reference page for a discussion of how new versions are created from reserved and unreserved checkouts.

CREATION OF CHECKED-OUT VERSION IN VIEW. *Default:* (file elements) Creates in the view:

- A view-private file for the version being checked out with the same pathname as the element (dynamic view).
- A modifiable copy of the version being checked out with the same pathname as the element (snapshot view).

Attache downloads a copy of that file to the workspace.

EXCEPTION: (Dynamic views) If the version being checked out is a derived object, it is winked in to the view.

–out *dest-pname*

(Does not apply to directories or DO versions) Creates a writable file under an alternate filename (perhaps in a different directory); in Attache, the file is downloaded to the workspace. No view-private file named *pname* is created. The **cleartool ls** and Attache **ls** commands list the element as `checkedout but removed`.

–nda·ta

(Does not apply to directories) Creates a checkout record for the version, but does not create an editable file containing its data; in Attache, no file is downloaded to the workspace. The **ls** command lists the file as `checked out but removed`. This option is useful for checking out files that will be completely overwritten (for example, staged binaries or other files that are copied into place).

PRESERVING MODIFICATION TIME. *Default:* In a dynamic view, **checkout** resets the file's modification time to the checkout time. In a snapshot view, **checkout** preserves the file's modification time.

–pti·me

Preserves the modification time of the file being checked out. This option is silently ignored when you use it in a snapshot view.

NON-STANDARD CHECKOUTS. *Default:* If *pname* specifies a particular branch, check out that branch, that is, the latest version on the branch. Otherwise, do the following:

- In a dynamic view, check out the latest version on the branch.
- In a snapshot view, check out the version that is currently in the view.

checkout creates a copy of each checked-out version and names it *pname*.

–bra·nch *branch-pname*

Specifies the branch whose most recent version is to be checked out. For example, to check out the latest version on branch **ports**, specify **–branch \main\ports**.

–ver·sion

Allows the checkout of a version that is not the latest on its branch.

SUPPRESSING WARNING MESSAGES *Default:* Warning messages are displayed.

–nwa·rn
Suppresses warning messages.

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–cqe**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
Overrides the default with the option you specify. See the **comments** reference page.

QUERYING FOR THE RESOLUTION OF CHECKOUT PROBLEMS. *Default:* No querying.

–q·uery
Query for the resolution of a checkout problem.

–nq·uery
Do not query for the resolution of a checkout problem.

ELEMENT ARGUMENT. *Default:* None.

pname ...
Pathnames of one or more elements to be checked out.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Check out the currently selected version of element **hello.c**, with no comment.

  cmd-context **checkout –nc hello.c**
  ```
  Checked out "hello.c" from version "/main/3".
  ```

- Check out the latest version on the **rel2_bugfix** branch of file **msg.c**, to another file name.

  cmd-context **checkout –nc –branch \main\rel2_bugfix –out msg_test.c msg.c**
  ```
  Checked out "msg.c" from version "\main\rel2_bugfix\1".
  ```

<u>*cmd-context*</u> **ls msg_test.c msg.c**

```
msg_test.c
msg.c@@\main\rel2_bugfix\CHECKEDOUT from \main\rel2_bugfix\1 [checked out
but removed]
```

- (ClearCase and ClearCase LT) Check out the latest version on the **rel2_bugfix** branch of file **msg.c**, using an extended pathname to indicate the branch. This command checks out the same version as the preceding example.

  <u>*cmd-context*</u> **checkout –nc msg.c@@/main/rel2_bugfix**

  ```
  Checked out "msg.c" from version "/main/rel2_bugfix/1".
  ```

- Check out an old version of the file **hello.h**, using an extended pathname to indicate the version. (Before you check in your revised version, you must perform a merge.)

  <u>*cmd-context*</u> **checkout –c "attempt fix of old bug" -version hello.h@@\main\1**

  ```
  Checked out "hello.h" from version "\main\1".
  ```

- Perform an unreserved checkout of element **hello.h**. Provide a comment on the command line.

  <u>*cmd-context*</u> **checkout –c "modify local defines"–unreserved hello.h**

  ```
  Checked out "hello.h" from version "/main/2"
  ```

- Check out **hello.c**. Then, change your mind and cancel the checkout, removing the view-private copy.

  <u>*cmd-context*</u> **checkout –nc hello.c**

  ```
  Checked out "hello.c" from version "\main\1".
  ```

  <u>*cmd-context*</u> **uncheckout –rm hello.c**

  ```
  Checkout cancelled for "hello.c".
  ```

SEE ALSO

checkin, **config_spec**, **lscheckout**, **merge**, **profile_ccase**, **reserve**, **uncheckout**, **unreserve**

checkvob

Finds and fixes inconsistencies between VOB database and storage pools, problems with hyperlinks, and problems with global types

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- Check/fix storage pools

  checkvob [**–vie·w** *view-tag*] [**–log** *log-dir-pname*] [**–fix** [**–f·orce**] [**–ign·ore**]]
  [**–dat·a**] [**–pro·tections**] [**–deb·ris**] [**–set·up**]
  { **–poo·l** [**–sou·rce**] [**–der·ived**] [**–cle·artext**] { *vob-stg-pname* | *pname-in-vob* }
  | [**–loc·k**] *file-pname* ...
  }

- Check/fix hyperlinks

  checkvob –hli·nks [**–to** | **–fro·m**] [**–hlt·ype** *hltype-selector*]
  [**–f·orce**] [**–pna·me**] *object-selector* ...

- Check/fix global types

  checkvob –glo·bal [**–log** *log-pname*] [**–fix** [**–f·orce**]]
  [**–acq·uire**] [**–pro·tections**] [**–loc·k** | **–unl·ock**]
  { *vob-selector* | *global-type-selector* }

DESCRIPTION

checkvob can find and fix problems with storage pools, with hyperlinks, and with global types in an admin VOB hierarchy. For more information, see the *Administrator's Guide*.

RESTRICTIONS

Identities:

For **–fix**, you must have one of the following identities:

- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

For **–hlink**, you must have one of the following identities:

- Object owner
- Object group member
- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

Locks: Without **–fix** (check-only mode), no locks apply. With **–fix**, the following restrictions apply:

- The VOB must be unlocked (or locked with **lock –nusers** *user-running-checkvob*).
- Problems cannot be fixed if the affected element or pool is locked. Use **–ignore** to modify this behavior.

It may be difficult to fix ownership and naming problems with global types if local copies or eclipsing ordinary types are locked.

Mastership: (Replicated VOBs only) No mastership restrictions.

OPTIONS AND ARGUMENTS

The following sections describe the options and arguments for storage pool mode, hyperlink mode, and global types mode. For more details on fix-mode processing, see the *Administrator's Guide*.

Storage Pool Mode

SPECIFYING A VIEW. *Default:* Uses the current view context. If you attempt to run **checkvob** without a view context, you are prompted to continue. Without a view context, **checkvob** cannot generate VOB pathnames in problem object reports, so it reports OIDs instead of pathnames. In general, run **checkvob** from a view.

–vie·w *view-tag*

Uses view *view-tag* to resolve any *file-pname* arguments, and to construct VOB object pathnames in output. This option exists primarily to permit **checkvob** to run on VOB servers where the MVFS is not installed (hosts where you cannot establish a working view context).

LOG FILE DIRECTORY. *Default:* **checkvob** creates in the current directory a log file directory named **checkvob.***date-time*. With **–pool**, several log files are created, including a **summary** file and one file per pool analyzed. Otherwise, a single **transcript** file stores a report on each individual file examined.

–log *log-dir-pname*

Specifies an alternative directory for the log file directory. If *log-dir-pname* already exists, **checkvob** returns an error.

FIX MODE. *Default:* Reports any problems, but does not try to fix them.

WARNING: Fixing problems detected with **–data** can update the VOB irreversibly. If source or DO data containers are missing from the storage pools when **checkvob** runs, it updates the VOB database, dereferencing these containers with the equivalents of **rmver -data** (for missing source containers) and **rmdo** (for missing DO containers).

–fix

Directs **checkvob** to try to correct any problems it finds. Without **–force**, **–fix** prompts you before fixing any problem object. You must run **checkvob** from the VOB server host to use **–fix**.

For details on how **checkvob** tries to fix the various problems it detects, see the *Administrator's Guide*.

–f·orce

Minimizes interactive prompts when **checkvob** runs with **–fix**.

–ign·ore

Ignores element and pool locks during fix processing. Use of **–ignore** requires that the VOB be locked for all users except the user running **checkvob** (**lock –nusers**). This option is not recommended for general use. It exists primarily to support automatic **checkvob** invocations when **vob_restore** is run.

DATABASE/POOL INCONSISTENCIES. *Default:* Scans pools or individual file containers looking for all detectable problems.

–dat·a

Identifies missing data containers. **checkvob** scans the VOB database and source pools to confirm the existence of each data container known to the database.

NOTE: During check processing, a "healthy" element is one whose containers have the right names, in the right locations, with the right permissions. **checkvob** does not detect container data corruption.

–pro·tections

Identifies access control problems on data containers.

–deb·ris

Scans storage pools for data containers not referenced by the VOB database. **–debris** is meaningful only when used with **–pool**. In general, **checkvob –fix –debris** moves debris to the applicable pool's **lost_found** directory. See the *Administrator's Guide* for details.

SETUP MODE. *Default:* None. Specify **–setup** to run **checkvob** in setup mode.

–set·up

Prepares newly reformatted VOB for **checkvob** processing.
Prepares a VOB for **vob_snapshot/vob_restore/checkvob** processing. See the *Administrator's Guide*.

POOL MODE. *Default:* None. Specify **–pool** and a *vob-stg-pname* argument in order to process one or more storage pools.

–poo·l

Runs **checkvob** in pool mode. See the *Administrator's Guide*.

–sou·rce
–der·ived
–cle·artext

Processes the VOB's source, derived object (DO), and/or cleartext pools. If you omit all of these options, **checkvob** processes all pool kinds.

vob-stg-pname
pname-in-vob

Identifies the VOB; required with **–pool**.

INDIVIDUAL FILE MODE. *Default:* None. If you do not use the **–pool** option, you must specify one or more *file-pname* arguments.

–loc·k

Locks each element during check processing. **checkvob** always locks an element during fix processing.

file-pname ...

Specifies one or more VOB objects having associated data containers—file elements, versions, or DOs. **checkvob** compares each data container's location and permissions against what is expected by the VOB database.

Hyperlink Mode

HYPERLINK MODE. *Default:* None. Use **–hlinks** to run **checkvob** in hyperlink mode. **checkvob** prompts for confirmation before deleting each partially unavailable hyperlink it detects.

–hli·nks

Runs **checkvob** in hyperlink mode.

–to
–fro·m
Checks only hyperlinks to or from the specified objects.

–hlt·ype *hyperlink-type-selector*
Checks only hyperlinks of type *hyperlink-type-selector*. Specify *hyperlink-type-selector* in the form **hltype:***type-name*[@*vob-selector*]

type-name	Name of the hyperlink type
vob-selector	Object-selector for a VOB, in the form [**vob:**]*pname-in-vob*. The *pname-in-vob* can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted).

–f·orce
Deletes broken hyperlinks without prompting for confirmation.

–pna·me
Interprets each *object-selector* argument as a pathname.

object-selector ...
Specifies the objects whose broken hyperlinks are to be found and deleted. Note that if you specify a VOB, **checkvob** does not check all hyperlinks in that VOB; it checks only the hyperlinks attached to the VOB object itself. Specify *object-selector* in one of the following forms:

pname

- A standard or view-extended pathname to an element specifies the version in the view.
- A version-extended pathname specifies an element, branch, or version, independent of view.
- The pathname of a VOB symbolic link.

NOTE: If *pname* has the form of an object selector, you must include the **–pname** option to indicate that *pname* is a pathname.

vob-selector	**vob:***pname-in-vob* *pname-in-vob* can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted). It cannot be the pathname of the VOB storage directory.
attribute-type-selector	**attype:***type-name*[@*vob-selector*]
branch-type-selector	**brtype:***type-name*[@*vob-selector*]

element-type-selector	**eltype:***type-name*[@*vob-selector*]
hyperlink-type-selector	**hltype:***type-name*[@*vob-selector*]
label-type-selector	**lbtype:***type-name*[@*vob-selector*]
trigger-type-selector	**trtype:***type-name*[@*vob-selector*]
pool-selector	**pool:***pool-name*[@*vob-selector*]
oid-obj-selector	**oid:***object-oid*[@*vob-selector*]

The following object selector is valid only if you use MultiSite:

replica-selector	**replica:***replica-name*[@*vob-selector*]

Global Types Mode

GLOBAL TYPES MODE. *Default:* None. You must specify **–global**.

–global
: Runs **checkvob** in global types mode.

LOG FILE. *Default:* **checkvob** creates in the current directory a file named **checkvob.***date***.***time*.

–log *log-pname*
: Specifies an alternative pathname for the log file. If *log-pname* already exists, **checkvob** returns an error.

FIX MODE. *Default:* Reports any problems, but does not try to fix them.

–fix
: Directs **checkvob** to try to correct any problems it finds. Without **–force**, **–fix** prompts you before fixing any problem object.

  For details on how **checkvob** tries to fix the various problems it detects, see the *Administrator's Guide*.

–f·orce
: Minimizes interactive prompts when **checkvob** runs with **–fix**.

–acquire
: Lists/fixes eclipsing local copies and eclipsing ordinary types

–protections
: Lists/fixes mismatched protections between global types and their local copies.

–lock
–unlock
: Lists/fixes eclipsing local locks. In fix mode, **–lock** locks the global type and **–unlock** removes the lock entirely.

vob-selector

Specifies a VOB in an admin VOB hierarchy. **checkvob** checks/fixes all global types found in the hierarchy. Specify *vob-selector* in the form **vob:***pname-in-vob*

pname-in-vob can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any filesystem object within the VOB (if the VOB is mounted). It cannot be the pathname of the VOB storage directory.

global-type-selector

Specifies a global type to be checked for problems. Specify *global-type-selector* in one of the following forms:

attribute-type-selector	**attype:***type-name*[@*vob-selector*]
branch-type-selector	**brtype:***type-name*[@*vob-selector*]
element-type-selector	**eltype:***type-name*[@*vob-selector*]
hyperlink-type-selector	**hltype:***type-name*[@*vob-selector*]
label-type-selector	**lbtype:***type-name*[@*vob-selector*]

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, cmd-context represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, cmd-context represents the interactive **cleartool** prompt. In Attache, cmd-context represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Check a single element's data container.

  cmd-context **checkvob /vobs/lib/gui.c**

```
The session's log directory is 'checkvob.11-Apr-98.05:21:17'.

=================================================================
Processing element "/vobs/lib/gui.c@@".
Checking status of 1 referenced containers in pool "s/sdft"...
Initial container status: 0 missing, 0 misprotected.
=================================================================
```

- Perform a routine check on a small, healthy VOB's source pools.

<u>*cmd-context*</u> **lsvob \vob_lib2**

```
* \vob_lib2    \\saturn\vobstore\vob_lib2.vbs
```

<u>*cmd-context*</u> **checkvob –pool –source \\saturn\vobstore\vob_lib2.vbs**

```
=================================================================
Starting "source pool" processing at 10-Apr-98.06:35:11

Running from host: saturn
VOB hostname: saturn
VOB host storage pathname: C:\vobstore\lib2.vbs
VOB global storage pathname: \\saturn\vobstore\lib2.vbs
VOB replica oid: 0cdc7b37.f48611cc.b3d5.00:01:80:02:bc:53
VOB host reference time: 10-Apr-98.06:29:59
Processing pools: sdft
Processing of misprotected containers is: ENABLED
Processing of ndata containers is: ENABLED
Processing of unreferenced containers is: ENABLED
Fix processing mode: DISABLED

Poolkind transcript log:
checkvob.20-Apr-98.12.10.40\poolkind_source\transcript
=================================================================
... progress messages ...
=================================================================
Completed "source pool" processing at 10-Apr-98.06:35:37

"source pool" Processing Summary:
Referenced Container Check Processing Time: 00:00:20
*** Referenced Container Fix Processing was not performed.
Unreferenced Container Check Processing Time: 00:00:05
*** Unreferenced Container Fix Processing was not performed.

Installed type managers are OK.

Pool root storage areas are OK.
```

```
Pool: s\sdft
Referenced container check processing:
    229 containers checked
             0 ndata          0 misprotected
    22 objects checked
             0 ndata          0 misprotected
Unreferenced container check processing:
    229 containers checked     (47778 kbytes)
             0 unreferenced but under age      (0 kbytes)
             0 unreferenced but maybe needed   (0 kbytes)
    0 unreferenced containers (0 kbytes, 0 empty)


The VOBs source pools are healthy.

Poolkind transcript log:
checkvob.20-Apr-98.12.10.40\poolkind_source\transcript
=================================================================
```

- Check all global types in the admin VOB hierarchy containing the VOB **/vobs/dev**.

  cmd-context **checkvob –global vob:/vobs/dev**

```
The session's log file is "checkvob.30-Jul-99.17:28:55".
Starting analysis of Admin VOB hierarchy.


Analysis of Admin VOB hierarchy complete.
5 VOBs analyzed, no hierarchy errors found.

Starting "global type" processing.

Detection of eclipsing local copies is: ENABLED
Detection of protection mis-matches is: ENABLED
Detection of eclipsing local locks is: ENABLED
Correction of detected errors is: DISABLED

Completed "global type" processing.
Processed 8 global types in 5 VOBs.
```

UNIX FILES

current-dir/**checkvob.***date-time* (default *log-pname* for **–global**)
current-dir/**checkvob.***date-time***/** (default *log-dir*)
log-dir/**.summary**
log-dir/**poolkind_cleartext/transcript**
log-dir/**poolkind_derived/transcript**

log-dir/**poolkind_source/transcript**
log-dir/**summary**
vob-storage-dir/**s/sdft/pool_id**
vob-storage-dir/**c/cdft/pool_id**
vob-storage-dir/**d/ddft/pool_id**

WINDOWS FILES

*current-dir***checkvob.***date-time* (default *log-pname* for **–global**)
*current-dir***checkvob.***date-time*\\ (default *log-dir*)
*log-dir***.summary**
*log-dir***poolkind_cleartext\\transcript**
*log-dir***poolkind_derived\\transcript**
*log-dir***poolkind_source\\transcript**
*log-dir***summary**
*vob-storage-dir***vob_server.conf**
*vob-storage-dir***s\\sdft\\pool_id**
*vob-storage-dir***c\\cdft\\pool_id**
*vob-storage-dir***d\\ddft\\pool_id**

SEE ALSO

reformatvob, **rmdo**, **rmver**, **type_manager**, **vob_restore**, **vob_snapshot**, *Administrator's Guide*

chevent

Changes the comment string in an event record

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

chevent [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**]
[**–app·end** | **–ins·ert** | **–rep·lace**]
{ **–eve·nt** [**–inv·ob** *vob-selector*] *event-ID* ...
| [**–pna·me**] *pname* ...
| *object-selector* ...
}

DESCRIPTION

The **chevent** command modifies or replaces the comment string in one or more existing event records. It is useful for correcting typing errors, and for including information that was omitted in the original comment.

There are several ways to specify an event record whose comment you want to change:

- If you specify a checked-out version, **chevent** changes the comment in the checkout event record.
- If you specify any other object, **chevent** changes that object's creation event record. For example, if you specify a label type object, **chevent** changes the comment supplied when that label type was created with **mklbtype**.

- You can change the comment in an arbitrary event record by passing its event-ID to the **–event** option. Use the command **lshistory –eventid** to capture event-IDs. (Event-IDs remain valid until the VOB is reformatted with **reformatvob**.)

See the **events_ccase** reference page for details on the operations that cause event records to be created, and how event records are attached to objects. See also the **comments** reference page.

RESTRICTIONS

Identities: You must have one of the following identities:

- User associated with the event
- Object owner
- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

Locks: The following restrictions apply:

Object	**Locks that Prevent Changing the Object's Events**
VOB	VOB
Pool	VOB, pool
Element	VOB, element type
Type	VOB, type
Branch, Version	VOB, element type, element, branch type, branch
Hyperlink	VOB, hyperlink type

Mastership: (Replicated VOBs only) For a version, your current replica must master the branch containing the version. For any other object, your current replica must master the object.

OPTIONS AND ARGUMENTS

SPECIFYING THE COMMENT CHANGE. *Default:* For each object or event, **chevent** prompts for a comment string to apply to the corresponding event record.

–c·omment *comment*

Specifies a character string to replace the existing comment or be added to it.

–cfi·le *comment-file-pname*

Specifies a text file whose contents are to be placed in the event record.

NOTE: A final newline character in this file is included in the comment.

–cq·uery

Prompts for one comment, which will be used to update all of the event records.

–cqe·ach
Same as default—prompts for a separate comment string for each object or event ID.

–nc·omment
No comment. When combined with **–replace**, this option removes the existing comment. Otherwise, it nullifies the effect of **chevent**.

SPECIFYING HOW TO CHANGE THE COMMENT. *Default:* The new comment is appended to the existing one. On UNIX systems, it is preceded by a newline character.

–app·end
Same as default.

–ins·ert
The new comment is inserted before the existing one. On UNIX systems, the new comment is followed by a newline character.

–rep·lace
The existing comment is discarded; the new comment replaces it.

SPECIFYING EVENT RECORDS TO BE CHANGED. You can indicate which event record is to be changed by specifying a file-system object, a non-file-system object, or a numerical event-ID. *Default:* None.

–eve·nt [**–inv·ob** *vob-selector*] *event-ID* ...
Specifies one or more events by their numeric event-IDs. The **–event** keyword can appear anywhere an option is valid; the event-ID arguments must appear at the end of the command (that is, after all options). By default, event-IDs specify events in the VOB containing the current working directory; use **–invob** *vob-selector* to specify another VOB.

To determine the event-ID of an event, use **lshistory –eventid**.

[**–pna·me**] *pname* ...

A standard pathname or VOB-extended pathname, indicating the creation event record for an element, branch, or version object. The standard pathname of an element specifies the version in your view. The **–pname** option is required only if the pathname looks like an object-selector (for example, an element named **pool:one**).

Specifying a checked-out version changes its `checkout version` comment. You can use any of the following to specify the checked-out version:

`hello.h`	(*standard pathname*)
`hello.h@@\main\rel2_bugfix\CHECKEDOUT`	(*extended pathname to checked-out "placeholder" version*)
`hello.h@@/main/rel2_bugfix/CHECKEDOUT.465`	(*placeholder version has unique numeric suffix)*

object-selector ...

One or more *object-selectors*, in any of these forms:

vob-selector	**vob:***pname-in-vob*
	pname-in-vob can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted). It cannot be the pathname of the VOB storage directory.
attribute-type-selector	**attype:***type-name*[@*vob-selector*]
branch-type-selector	**brtype:***type-name*[@*vob-selector*]
element-type-selector	**eltype:***type-name*[@*vob-selector*]
hyperlink-type-selector	**hltype:***type-name*[@*vob-selector*]
label-type-selector	**lbtype:***type-name*[@*vob-selector*]
trigger-type-selector	**trtype:***type-name*[@*vob-selector*]
pool-selector	**pool:***pool-name*[@*vob-selector*]
hlink-selector	**hlink:***hlink-id*[@*vob-selector*]
oid-obj-selector	**oid:***object-oid*[@*vob-selector*]

The following object selector is valid only if you use MultiSite:

replica-selector	**replica:***replica-name*[@*vob-selector*]

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Add a creation comment for an element, verifying the change with **describe**.

  cmd-context **chevent hello.c@@**

  ```
  Comments for "hello.c":
  Main module of greeting program.
  .
  Modified event of file element "hello.c".
  ```

<u>*cmd-context*</u> **describe hello.c@@**

```
file element "hello.c@@"
created 04-Dec-98.14:38:26 by anne.user
  "Main module of greeting program"
  element type: text_file
source pool: p1 cleartext pool: pc1
```

- Add a header to a checked-out version's checkout comment.

<u>*cmd-context*</u> **lscheckout bye.c**

```
13-May.13:58 anne checkout version "bye.c" from /main/1 (reserved)
"Improve error handling."
```

<u>*cmd-context*</u> **chevent –insert –c "Fix bug #2493:" bye.c**

```
Modified event of version "bye.c".
```

<u>*cmd-context*</u> **lscheckout bye.c**

```
13-May.13:58 anne checkout version "bye.c" from /main/11 (reserved)
"Fix bug #2493:
 Improve error handling."
```

- Update a branch type creation comment.

<u>*cmd-context*</u> **chevent –append brtype:v1_bugfix**

```
Comments for "v1_bugfix":
```

Branches should sprout from the version labeled 'V1'

.

```
Modified event of branch type "v1_bugfix".
```

<u>*cmd-context*</u> **lstype brtype:v1_bugfix**

```
28-Mar.16:26 ali branch type "v1_bugfix"
 "Branch for fixes to version 1.
 Branches should sprout from the version labeled 'V1'"
```

- Delete the comment on a branch object.

<u>*cmd-context*</u> **chevent –replace –nc welcome.c@@/main/v1_bugfix**

```
Modified event of branch "welcome.c".
```

- Find the event-ID for an operation and append a comment string to the one already assigned to that event. Then verify that the new comment was added.

<u>*cmd-context*</u> **lshistory –long –eventid util.c**

```
event 45678:
21-Mar-99.14:45:20 Anne Duvo (anne@neptune)
destroy sub-branch "bugfix" of branch "util.c@@\main"
"Destroyed branch "\main\bugfix"."
.
.
.
```

cmd-context **chevent –c "bugfix merge completed." –append –event 45678**

```
Modified event "45678".
```

cmd-context **lshistory –long –eventid util.c**

```
event 45678:
21-Mar-99.14:45:20 Anne Duvo (anne@neptune)
destroy sub-branch "bugfix" of branch "util.c@@\main"
 "Destroyed branch "\main\bugfix".
"bugfix merge completed."
.
.
.
```

SEE ALSO

events_ccase, **lock**, **lshistory**, **mktrtype**, **vob_scrubber**

chflevel

Raises the feature level of a VOB

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- Analyze and possibly raise the feature level of a VOB on the local host:

  chflevel [**–f·orce**] **–auto**

- Raise the feature level of a MultiSite replica:

  chflevel –rep·lica *feature-level replica-selector*

- Raise the feature level of a MultiSite VOB family:

  chflevel [**–f·orce**] [**–ove·rride**] **–fam·ily** *feature-level vob-selector*

DESCRIPTION

The **chflevel** command raises the feature level of a VOB. A feature level is an integer that is incremented at each ClearCase and ClearCase LT release that introduces features that affect VOBs created in an earlier ClearCase or ClearCase LT release. The purpose of raising feature levels is to make all features in a release available to users of the VOB that was created in the earlier release.

Every ClearCase and ClearCase LT release is associated with a feature level. Read the *Release Notes* for your product for information on which releases correspond to which feature levels.

Raising the Feature Levels of VOBs

To raise the feature level of a VOB, use the **chflevel** command with the **–auto** option at the host running the VOB server.

In some circumstances—for example, when you **describe** a VOB—you will notice MultiSite terms such as *VOB family*. This kind of information is of interest only to MultiSite users.

Raising the Feature Levels of MultiSite VOBs

Raising the feature level of a MultiSite VOB prevents features from being used at one replica that are not meaningful to other replicas that are at a lower feature level.Thus, feature level control makes it unnecessary to upgrade all replicas in a VOB family to a new ClearCase release simultaneously.

Every VOB replica has a feature level, and every VOB family has a feature level:

- The replica feature level is the feature level that is equal to or less than the feature level of the ClearCase release installed on the host where the replica's server runs.
- The family feature level is the feature level that is equal to or less than the lowest replica feature level found among members of the VOB family.

You must raise the replica feature levels before raising the VOB family feature level. After raising the feature level of replicas in the VOB family, raise the VOB family feature level to the lowest feature level of any replica in the family.

For more information, see the *Administrator's Guide* for Rational ClearCase MultiSite.

RESTRICTIONS

Identities: You must have one of the following identities:

- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

Locks: An error occurs if one or more of these objects are locked: VOB, VOB replica.

Mastership: (Replicated VOBs only) A replica whose feature level is to be raised must master its own replica object. The family feature level can be raised only through the replica that masters the VOB object.

Other: (Replicated VOBs only) If the current family feature level is less than or equal to 1, the first replica whose feature level is raised must be the replica that masters the VOB object.

OPTIONS AND ARGUMENTS

–aut·o

Lists each VOB on the local host, annotated with its replication state, family feature level, and replica feature level. For unreplicated VOBs only, this option offers to raise the feature levels. You must raise the feature levels of replicated VOBs using the command synopses for MultiSite.

–f·orce

When specified with the **–auto** option, this option raises the feature levels of unreplicated VOBs without prompting for confirmation.

When specified with the **–family** option, this option forces MultiSite replicas on the local host to the feature level specified by **–family** without prompting. This option may fail to force the family feature level unless you also specify **–override**.

–rep·lica *feature-level replica-selector*

Raises the feature level of the specified MultiSite replica.

–ove·rride

Overrides the check that ensures that the feature level specified by **–family** is less than or equal to the lowest feature level found among replicas in the family. When specified with the **–force** option, forcibly raises the VOB family feature level without prompting. When specified without **–force**, **–override** lists replicas that are below the specified family feature level.

NOTE: Do not use the **–force** and **–override** options together unless you are certain that all replicas are at the feature level specified by **–family**.

–fam·ily *feature-level vob-selector*

Raises the feature level of the specified MultiSite VOB family.

EXAMPLES

- Raise the feature levels of any unreplicated VOBS running on the local host without prompting for confirmation and list the feature levels of any replicated VOBs on this host.

  cmd-context **chflevel –force –auto**

- Raise the feature level of the replica **rome** to **2**.

  cmd-context **chflevel –replica 2 replica:rome**

- Raise the feature level of the VOB family **\tmp\testvob** to **2**.

  cmd-context **chflevel –family 2 vob:\tmp\testvob**

- Raise the family feature level of the current VOB to **2**. Override the check to ensure that family feature level **2** is no higher than the lowest replica feature level found among replicas in this VOB family.

  cmd-context **chflevel –force –override –family 2 vob:.**

SEE ALSO

chmaster, **describe**, *Release Notes* for your product

chfolder

Modifies a UCM folder

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

chfolder [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |
–cq·uery | **–cqe·ach** |**–nc·omment**]
–to *to-folder-selector*
folder-selector ...

DESCRIPTION

The **chfolder** command moves a folder to another location in the folder hierarchy of a project VOB. The **RootFolder** cannot be moved.

For information about changing the folder's name, see **rename**.

RESTRICTIONS

Identities: No special identity required.

Locks: An error occurs if one or more of these objects are locked: the folder, the UCM project VOB.

Mastership: (Replicated VOBs only) Your current replica must master the folder.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default*: Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–nc**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
Overrides the default with the option you specify. See the **comments** reference page.

SPECIFYING THE DESTINATION OF A FOLDER. *Default*: None.

–to *to-folder-selector*

Specifies the new parent folder. The to-folder and the folder you are moving must belong to the same UCM project VOB.

folder-selector is of the form [**folder:**]*folder-name*[@*vob-selector*], where *vob-selector* specifies the folder's project VOB.

SPECIFYING THE FOLDER TO CHANGE. *Default*: None.

folder-selector ...

Specifies one or more folders to modify. **RootFolder** cannot be moved.

folder-selector is of the form [**folder:**]*folder-name*[@*vob-selector*], where *vob-selector* specifies the folder's project VOB.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, <u>*cmd-context*</u> represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, <u>*cmd-context*</u> represents the interactive **cleartool** prompt. In Attache, <u>*cmd-context*</u> represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Make the folder **Core_Parsers** a subfolder of **RootFolder**. Note that the folder's project VOB is given as the VOB component of the *folder-specifier*.

  <u>*cmd-context*</u> **chfolder -to RootFolder Core_Parsers@/vobs/core_projects**
  ```
  Changed folder "Core_Parsers@\vobs\core_projects".
  ```

SEE ALSO

lsfolder, **mkfolder**, **rmfolder**, **rename**

chmaster

Transfers mastership of VOB-database object

APPLICABILITY

Product	Command type
ClearCase	cleartool subcommand
MultiSite	multitool subcommand

Platform
UNIX
Windows

SYNOPSIS

chmaster [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* | **–cq·uery**
| **–cqe·ach** | **–nc·omment**]
{ *master-replica-selector object-selector* ...
| [**–pname**] *master-replica-selector branch-or-element-pname* ...
| **–str·eam** [**–ove·rride**] *master-replica-selector stream-selector* ...
| **–def·ault** [**–pname**] *branch-pname* ...
| **–def·ault** *brtype-selector* ...
| **–all** [**–obsolete_replica** *old-replica-selector*]
[**–l·ong**] [**–vie·w** *view-tag*] *master-replica-selector*
}

DESCRIPTION

This command transfers the mastership of one or more objects from one VOB replica to another. Only the current replica is affected immediately; other replicas are notified of the mastership transfers through the normal exchange of update packets.

To limit use of this command to a certain set of users, you can create triggers. For more information, see *Managing Software Projects*.

SPECIFYING A VIEW CONTEXT

The **chmaster** command requires a view context. If you are not in a set view or working directory view on UNIX or a view drive on Windows, you can specify a view on the command line, as shown in the following table. If you specify a dynamic view, it must be active on your host.

NOTE: A view you specify in the **chmaster** command takes precedence over your current set view, working directory view, or view drive.

Argument	How to specify a view
object-selector *brtype-selector*	Use a view-extended pathname as the *vob-selector* portion of the argument. For example: **lbtype:LABEL1@/view/jtg/vobs/dev** **brtype:v1.0_bugfix@/view/jtg/vobs/dev** **lbtype:LABEL1@s:\dev** **brtype:v1.0_bugfix@s:\dev**
branch-pname *element-pname*	Specify *branch-pname* or *element-pname* as a view-extended pathname. For example: **/view/jtg/vobs/dev/cmd.c@@** **/view/jtg/vobs/dev/cmd.c@@/main** **s:\dev\cmd.c@@** **s:\dev\cmd.c@@\main**
master-replica-selector (for the **chmaster –all** variant)	Use the **–view** option or use a view-extended pathname as the *vob-selector* portion of the argument. For example: **–view jtg replica:boston_hub@\dev** **replica:boston_hub@/view/jtg/vobs/dev** **replica:boston_hub@s:\dev**

RESTRICTIONS

Identities: For all UCM objects except baselines, no special identity is required. For baselines and all other non-UCM objects, you must have one of the following identities:

- Object creator (except for replicas)
- Object owner (except for replicas)
- VOB owner
- **root** (UNIX)
- Member of the ClearCase administrators group (Windows)

Locks: Restrictions depend on the kind of object:

Object whose mastership is changing	Locks on these objects cause the chmaster command to fail
Element	Element, element type, VOB

Object whose mastership is changing	Locks on these objects cause the chmaster command to fail
Branch	Branch, branch type, VOB
Type object	Type object, VOB
Hyperlink	Hyperlink type, VOB
Baseline	Baseline, VOB, replica, components associated with the baseline
Stream	Stream, activity
Component	Component, VOB, replica

Mastership: Your current replica must master the object. Using both **–all** and **–obsolete_replica** overrides this restriction, but you must not use the **–obsolete_replica** option except in special circumstances. (See the description of the **–all** option.)

Other: You cannot transfer mastership of a branch if the branch is checked out reserved or if it is checked out unreserved without the **–nmaster** option.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, with commenting controlled by the standard ClearCase user profile (default: **–nc**). See *EVENT RECORDS AND COMMENTS* in the **multitool** reference page. To edit a comment, use **cleartool chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* | **–cq·uery** | **–cqe·ach** | **–nc·omment**
Overrides the default with the specified comment option.

SPECIFYING THE OBJECTS. *Default:* None.

master-replica-selector object-selector ...
Transfers mastership of objects specified with *object-selector* to the VOB replica specified with *master-replica-selector*. Specify *master-replica-selector* in the form [**replica:**]*replica-name*[@*vob-selector*]

replica-name	Name of the replica (displayed with **lsreplica**)
vob-selector	VOB family of the replica; can be omitted if the current working directory is within the VOB.
	Specify *vob-selector* in the form [**vob:**]*pname-in-vob*

pname-in-vob	Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

Specify *object-selector* in one of the following forms:

vob-selector	**vob:***pname-in-vob* where *pname-in-vob* — Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)
attribute-type-selector	[**attype:**]*type-name*[@*vob-selector*]
branch-type-selector	[**brtype:**]*type-name*[@*vob-selector*]
element-type-selector	[**eltype:**]*type-name*[@*vob-selector*]
hyperlink-type-selector	[**hltype:**]*type-name*[@*vob-selector*]
label-type-selector	[**lbtype:**]*type-name*[@*vob-selector*]
hlink-selector	[**hlink:**]*hlink-id*[@*vob-selector*]
oid-obj-selector	**oid:***object-oid*[@*vob-selector*]
replica-selector	[**replica:**]*replica-name*[@*vob-selector*]
baseline-selector	[**baseline:**]*baseline-name*[@*vob-selector*]
component-selector	[**component:**]*component-name*[@*vob-selector*]

[**–pname**] *master-replica-selector branch-or-element-pname* ...

Transfers mastership of the specified branches or elements to the VOB replica specified with *master-replica-selector*. A branch pathname takes the form *element-name***@@/***branch*..., for example, **cmdsyn.c@@/main/bugfix**, and an element pathname takes the form *element-name***@@**, for example, **cmdsyn.c@@**. If *branch-or-element-pname* has the form of an object selector, you must include the **–pname** option to indicate that *pname* is a pathname.

–str·eam [**–ove·rride**] *master-replica-selector stream-selector* ...

Transfers mastership of the specified streams and their associated objects to the VOB replica specified with *master-replica-selector*. Specify *stream-selector* in the following form:

stream-selector	[**stream:**]*stream-name*[@*vob-selector*]

Use the **–override** option only if the **chmaster –stream** command fails. With **–override**, **chmaster** attempts to transfer mastership of objects whose mastership was not transferred during the original invocation of the command. For more information, see *Managing Mastership* in *Administrator's Guide* for Rational ClearCase MultiSite.

–a·ll [**–obsolete_replica** *old-replica-selector*] [**–l·ong**] [**–vie·w** *view-tag*] *master-replica-selector*

CAUTION: Incorrect use of the **–obsolete_replica** form of the command can lead to divergence among the replicas in a VOB family.

Transfers to *master-replica-selector* mastership of all objects that are located in and mastered by the current replica. (The **chmaster** command determines the current replica by using the *vob-selector* you specify as part of *master-replica-selector*. If you do not include a *vob-selector*, **chmaster** uses the replica containing the current working directory.)

If errors occur, the command continues. After finishing, it reports that not all mastership changes succeeded.

With **–long**, **chmaster** lists the objects whose mastership is changing.

With **–view**, **chmaster** uses the specified view as the view context.

With **–obsolete_replica**, **chmaster** transfers mastership of all objects in the replica specified with *old-replica-selector*. Also, **chmaster** associates nonmastered checkouts with the new replica. Use this form of **chmaster** only when replica *old-replica-selector* is no longer available (for example, was deleted accidentally). Before entering this command, you must make sure that *old-replica-selector* masters itself or is mastered by the replica that it last updated. Then, enter the **chmaster** command at the last-updated replica. You must also send update packets from the last-updated replica to all other remaining replicas in the VOB family. For more information, see the **rmreplica** reference page.

RETURNING MASTERSHIP OF BRANCHES TO DEFAULT STATE. *Default:* None.

–def·ault [**–pname**] *branch-pname* ...

Transfers mastership of *branch-pname* to the replica that masters the branch type. If *branch-pname* has the form of an object selector, you must include the **–pname** option to indicate that *branch-pname* is a pathname.

–def·ault *brtype-selector* ...

Removes explicit mastership of branches that are mastered explicitly by the current replica and are instances of the type *brtype*.

NOTE: You can use this command only at the replica that masters the branch type.

EXAMPLES

- At replica **boston_hub**, transfer mastership of label type **V1.0_BUGFIX** to the **sanfran_hub** replica.

  multitool chmaster sanfran_hub lbtype:V1.0_BUGFIX
  ```
  Changed mastership of "V1.0_BUGFIX" to "sanfran_hub"
  ```

- At replica **sanfran_hub**, transfer mastership of element **list.c** to the **sydney** replica.

  multitool chmaster sydney list.c@@
  ```
  Changed mastership of "list.c" to "sydney"
  ```

- At replica **sanfran_hub**, transfer mastership of the stream **v2.1.bl5** and its associated objects to the **boston_hub** replica.

  multitool chmaster –stream boston_hub@/vobs/dev stream:v2.1.bl5@/vobs/dev

- At the replica that is the master of replica **sanfran_hub**, make **sanfran_hub** self-mastering.

  multitool chmaster sanfran_hub replica:sanfran_hub
  ```
  Changed mastership of "sanfran_hub" to "sanfran_hub"
  ```

- At replica **buenosaires**, transfer mastership of branch **cache.c@@/main/v3_dev** to **boston_hub**.

  multitool chmaster boston_hub cache.c@@/main/v3_dev
  ```
  Changed mastership of branch "/vobs/dev/cache.c@@/main/v3_dev" to
  "boston_hub"
  ```

- For all objects mastered by the current replica, transfer mastership to **sanfran_hub**.

  multitool chmaster –all sanfran_hub
  ```
  Changed mastership of all objects
  ```

- Same as the preceding example, but have **chmaster** list each object whose mastership is changing, and specify a view context.

  multitool chmaster –all –long sanfran_hub@/view/jtg/vobs/dev
  ```
  Changed mastership of branch type sydney_main
  Changed mastership of label type SYDNEY_V2.0
  Changed mastership of replica sydney
  Changed mastership of all objects
  ```

- Return mastership of a branch to the replica that masters the branch type, and then remove its explicit mastership.

  At the replica that masters the branch:

  multitool describe –fmt "%[master]p\n" brtype:v3_bugfix
  ```
  boston_hub@\dev
  ```

  multitool chmaster boston_hub@\dev \dev\acc.c@@\main\v3_bugfix
  ```
  Changed mastership of branch "\dev\acc.c@@\main\v3_bugfix" to
  "boston_hub@\dev"
  ```

  multitool syncreplica –export –fship boston_hub@\dev
  ```
  Generating synchronization packet c:\Program Files\Rational\ClearCase\var
  \shipping\ms_ship\outgoing\sync_bangalore_19-Aug-00.09.33.02_3447_1
  ...
  ```

At the replica that masters the branch type:

multitool syncreplica –import –receive

```
Applied sync. packet
/var/adm/atria/shipping/ms_ship/incoming/sync_bangalore_19-Aug-00.09.33.02
_3447_1
to VOB /net/minuteman/vobstg/dev.vbs
```

multitool chmaster –default brtype:v3_bugfix

```
Changed mastership of branch type "v3_bugfix" to "default"
```

SEE ALSO

reqmaster, **syncreplica**
Chapter 8, *Managing Mastership* in the *Administrator's Guide* for Rational ClearCase MultiSite.

chpool

Changes the storage pool to which an element is assigned

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

chpool [**–f·orce**] [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery**
| **–cqe·ach** | **–nc·omment**] *pool-selector pname* ...

DESCRIPTION

The **chpool** command changes the source storage pool, derived object storage pool, or cleartext storage pool to which one or more elements are assigned.

For a file element:

- Changing the source pool moves the data containers that store all existing versions from the current pool to the specified pool.
- Changing the cleartext pool designates a different location for new *cleartext* versions. Existing cleartext versions remain where they are, and are eventually scrubbed. (See the **scrubber** reference page.)
- An error occurs if you attempt to assign a file element to a derived object pool; file elements have source and cleartext pools only.

For a directory element:

- Changing the source pool or the cleartext pool affects pool inheritance by new elements. Elements created within the directory are assigned to the new pool; the pool assignments of existing elements do not change.

- Changing the derived object pool designates a new location for shared derived objects with pathnames in that directory. The **promote_server** program copies data containers to the new pool; he existing contents of the old pool do not change, and are eventually deleted by **scrubber**.

Commands for Listing Pools

The **lspool** command lists a VOB's storage pools. The **describe** command includes storage pool assignments in its listing for an element. To reference an element (rather than one of its versions), append the extended naming symbol to the element's standard pathname:

cmd-context **describe msg.c@@**

RESTRICTIONS

Identities: You must have one of the following identities:

- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)

Locks: An error occurs if one or more of these objects are locked: VOB, element type, element, pool.

Mastership: (Replicated VOBs only) No mastership restrictions.

OPTIONS AND ARGUMENTS

USER INTERACTION. *Default:* Prompts for confirmation before moving data containers.

–f·orce
Suppresses the confirmation step.

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–nc**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
Overrides the default with the option you specify. See the **comments** reference page.

SPECIFYING THE POOL. *Default:* None.

pool-selector
An existing storage pool. Specify *pool-selector* in the form [**pool:**]*pool-name*[@*vob-selector*]

pool-name	Name of the storage pool See the the **cleartool** reference page for rules about composing names.
vob-selector	VOB specifier Specify *vob-selector* in the form [**vob:**]*pname-in-vob*

pname-in-vob	Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted).

SPECIFYING THE ELEMENTS. *Default:* None.

pname ...

One or more pathnames, each of which specifies a file or directory element. A standard pathname is valid; you do not need to append the extended naming symbol. (Specifying a version or a branch is generally equivalent to specifying its element.)

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Reassign all elements in the current directory that have a **.c** extension to cleartext pool **cltxt2**.

  cmd-context **chpool –force cltxt2 *.c**

  ```
  Changed pool for "cm_add.c" to "cltxt2".
  Changed pool for "cm_fill.c" to "cltxt2".
  Changed pool for "convolution.c" to "cltxt2".
  Changed pool for "msg.c" to "cltxt2".
  Changed pool for "test_cmd.c" to "cltxt2".
  Changed pool for "util.c" to "cltxt2".
  ```

- Change the default source pool for the **src** directory, so that new elements created in this directory are assigned to the **c_pool** pool.

  cmd-context **chpool c_pool src**

  ```
  Changed pool for "src" to "c_pool".
  ```

- Change the source pool for **hello.c** to **sdft**, the VOB's default source pool. (Assumes the element had been assigned to a different pool.)

  cmd-context **chpool sdft hello.c**

```
Move all versions of element "hello.c"? [no] yes
Changed pool for "hello.c" to "sdft".
```

SEE ALSO

lspool, **mkdir**, **mkelem**, **mkpool**, **profile_ccase**, **promote_server**, **scrubber**

chproject

Modifies a UCM project

APPLICABILITY

Product	**Command Type**
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

chproj·ect [**–c·omment** *comment* | **–cfi·le** *pname* | **–cq·uery** | **–cqe·ach** | **–nc·omment**]
{
[**–amo·dcomp** *component-selector*[**,**...]] [**–dmo·dcomp** *component-selector*[**,**...]]
[**–to** *to-folder-selector*] [**–reb·ase_level** *promotion-level*]
[**–pol·icy** *policy-keyword*[**,**...]] [**–npo·licy** *policy-keyword*[**,**...]]
[**–spo·licy** *policy-keyword*[**,**...]]
[**–crm·enable** *ClearQuest-user-database-name* | **–ncr·menable**]
}
project-selector ...

DESCRIPTION

The **chproject** command modifies one or more UCM projects in the following ways:

- Adds one or more modifiable components to a project.
- Removes one or more components from the project's modifiable component list.
- Moves a project to another folder.
- Changes the promotion level required of a baseline before it can be recommended by a stream in the project.
- Sets policy for a project.
- Enables or disables a project for use with Rational ClearQuest.

For information about changing the project's name, see **rename**.

Adding Modifiable Components

Over time, a project's scope can expand, and you may need to add modifiable components to the project. The **–amodcomp** option allows you to add one or more modifiable components. For the streams in the project to capture the changes in component modifiability, do one of two things:

- If the components have a baseline in the stream, regenerate the view attached to the stream by running **setcs –stream**.
- If the components don't yet have a baseline in the stream, add baselines for these components with **rebase –baseline**.

Converting Components from Modifiable to Read-Only

If you made some components modifiable when you set up a project and later change your mind, the **–dmodcomp** option allows you to remove one or more components from the project's modifiable component list. A component cannot be removed from the list if any changes have been made to it in the project. Before performing this operation, verify that no development stream has changes to the component. A warning is issued if any of the following is true:

- The component or VOB is replicated. Any modification in development streams of replicated VOBs cannot be delivered.
- One or more of the project's streams are unavailable.
- The component is already unmodifiable.

After you convert the components to read-only, reconfigure the views or streams as needed. See instructions in *Adding Modifiable Components*.

Setting Required Promotion Levels for Recommended Baselines

A project's rebase level is the minimum promotion level a baseline must have in order to be recommended by a stream in the project. For example, if Project A has three promotion levels—**REJECTED**, **TESTED**, and **RELEASED** (in ascending order)—and **TESTED** is the rebase level, the latest baselines labeled **TESTED** or **RELEASED** can be recommended by a stream with **chstream –recommended –default**. See **rebase** and **setplevel** for more information.

Using Rational ClearQuest with UCM Projects

You can link or unlink a UCM project to a ClearQuest database with the **–crmenable** or **–ncrmenable** options. When you enable ClearQuest for a UCM project that contains UCM activities, then for each UCM activity, a ClearQuest record of type UCMUtilityActivity is created and linked to the activity. This process is called activity migration. If you disable a link to ClearQuest from a UCM project that contains activities, all its activities are unlinked from their ClearQuest records.

The **–crmenable** and **–ncrmenable** options display a summary of the number of activities that have been migrated or unlinked.

In MultiSite, you are informed if activities cannot be migrated or linked because they are not mastered in the current UCM project VOB replica. If there are any, they are listed along with a list of replicas from which to run the command again to correct the problem.

You can also use the **–crmenable** and **–ncrmenable** options to check for possible linking errors. If you believe that your project enabled for ClearQuest may contain activities that are not linked to a ClearQuest record, run **chproject –crmenable**. This operation scans all activities in the project, migrating all activities that are not linked. To check for linked activities in projects that have been disabled for use with ClearQuest, run **chproject –ncrmenable**. This operation removes links between activities as needed. See *Managing Software Projects* for more information.

RESTRICTIONS

Identities: You must have one of the following identities:

- Project owner
- Project VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows only)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows only)

Locks: An error occurs if there are locks on any of the following objects: the project, the UCM project VOB.

Mastership: (Replicated VOBs only) Your current replica must master the project.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default*: Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–cq**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**

Overrides the default with the option you specify. See the **comments** reference page.

ADDING TO THE LIST OF MODIFIABLE COMPONENTS FOR A PROJECT. *Default*: None.

–amo·dcomp *component-selector*[**,**...]

Adds one or more components to the project’s set of modifiable components.

component-selector is of the form [**component:**]*component-name*[@*vob-selector*], where *vob-selector* specifies the component’s project VOB.

REMOVING FROM THE LIST OF MODIFIABLE COMPONENTS FOR A PROJECT. *Default*: None.

–dmo·dcomp *component-selector*[**,**...]
Removes one or more components from the project's set of modifiable components.

component-selector is of the form [**component:**]*component-name*[@*vob-selector*], where *vob-selector* specifies the component's project VOB.

MOVING THE PROJECT TO ANOTHER FOLDER. *Default*: None.

–to *to-folder-selector*
Moves one or more projects to the specified folder. The to-folder and project must have the same UCM project VOB.

folder-selector is of the form [**folder:**]*folder-name*[@*vob-selector*], where *vob-selector* specifies the folder's project VOB.

CHANGING THE RECOMMENDED PROMOTION LEVEL FOR A REBASE OPERATION. *Default*: None.

–reb·ase_level *promotion-level*
Changes the promotion level required for baselines to be recommended by a stream in the project. For each component, the latest baseline in the integration stream at or above this promotion level is recommended.

SETTING PROJECT POLICY. *Default*: None.

–pol·icy *policy-keyword*
Enables the specified policy. For information about project policies, see **mkproject**.

–npo·licy *policy-keyword*
Disables the specified policy. For information about project policies, see **mkproject**.

–spo·licy *policy-keyword*
Allows the specified policy to be enabled or disabled by individual streams. For information about project policies, see **mkproject**.

LINKING A PROJECT TO RATIONAL CLEARQUEST. *Default*: No linking.

–crm·enable *ClearQuest***-***user-database-name*
Enables a link from the project to the specified Rational ClearQuest database. The schema of the ClearQuest database must be enabled for UCM, and your system must be configured for the correct schema repository.

–ncr·menable
Disables use of Rational ClearQuest for the specified project.

SELECTING A PROJECT. *Default*: None.

project-selector ...
Specifies one or more projects to modify.

project-selector is of the form [**project:**]*project-name*[@*vob-selector*], where *vob-selector* specifies the project's project VOB.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Add the modifiable component, **webo_modeler**, to the project.

  cmd-context **chproject -amod webo_modeler webo_proj1@/vobs/webo_pvob**

  ```
  Changed modifiable component list for project
  "webo_proj1@/vobs/webo_pvob".
  ```

SEE ALSO

chbl, **deliver**, **lscomp**, **lsproject**, **mkproject**, **mkcomp**, **rebase**, **rmproject**

chstream

Modifies a UCM stream

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

chstream [**–c·omment** *comment* | **–cfi·le** *pname* | **–cq·uery** | **–cqe·ach** | **–nc·omment**]
[**–tar·get** *stream-selector* | **–nta·rget**] [**–gen·erate**]
[**–pol·icy** *policy-keyword*[**,**...]] [**–npo·licy** *policy-keyword*[**,**...]]
[**–rec·ommended** { *baseline-selector*[,...] | **–def·ault** } | **–nre·commended**]
stream-selector ...

DESCRIPTION

The **chstream** command allows you to set stream policies, specify the default deliver target for integration streams, and set recommended baselines.

For information about changing the stream's name, see **rename**.

RESTRICTIONS

Identities: You must have one of the following identities:

- Project VOB owner
- Stream owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows only)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows only)

Locks: An error occurs if there are locks on the following objects: the UCM project VOB, the stream.

Mastership: (Replicated VOBs only) Your current replica must master the stream.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default*: Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–nc**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
: Overrides the default with the option you specify. See the **comments** reference page.

SETTING THE DEFAULT DELIVER TARGET FOR INTEGRATION STREAMS ONLY. *Default*: None.

–tar·get *stream-selector* | **–ntarget**
: Specifies the default deliver stream of an integration stream. The target must be a stream in a different project.

  Using **–ntarget** clears the default deliver stream setting.

  stream-selector is of the form: [**stream:**]*stream-name*[@*vob-selector*] where *vob-selector* is the project VOB of a different project.

UPDATING THE STREAM'S COPY OF THE CONFIG SPEC. *Default:* None.

–gen·erate
: Forces an update of the stream's copy of the config spec. Use this option after renaming a baseline (or its label type) and after renaming a stream (or its branch type).

SETTING PROJECT POLICY. *Default*: None.

–pol·icy *policy-keyword*[**,**...]
: Enables the specified policy. For information about stream policies, see **mkstream**.

–npo·licy *policy-keyword*[**,**...]
: Disables the specified policy. For information about stream policies, see **mkstream**.

SETTING RECOMMENDED BASELINES. *Default*: None.

–rec·ommended { *baseline-selector*[**,**...] | **–def·ault** } | **–nre·commended**
: Names a list of baselines that a stream recommends. You must always specify the full list of recommended baselines to change the list. With composite baselines, you may only need to name one baseline.

  baseline-selector is of the form [**baseline:**]*baseline-name*[@*vob-selector*], where *vob-selector* specifies the baseline's project VOB.

  Using the **–default** option sets the default recommended baselines. The default recommended baselines are dynamic; they are the latest baselines created in the stream at or above the specified promotion level for each component, or its foundation baseline if none has been created in the stream.

Using the –**nrecommended** option clears the list of recommended baselines previously set in a stream.

SPECIFYING THE STREAM. *Default:* None.

stream-selector ...

Specifies one or more streams to be modified.

stream-selector is of the form [**stream:**]*stream-name*[*@vob-selector*], where *vob-selector* specifies the stream's project VOB.

You can specify the stream as a simple name or as an object selector of the form [**stream**]**:***name@vob-selector*, where *vob-selector* specifies a project VOB (see the **cleartool** reference page). If you specify a simple name and the current directory is not a project VOB, then this command assumes the stream resides in the project VOB associated with the current view. If the current directory is a project VOB, then that project VOB is the context for identifying the stream.

SEE ALSO

deliver, **lsstream**, **mkstream**, **rebase**, **rename**, **rmstream**

chtype

Changes the type of an element or renames a branch

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

chtype [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* | **–cq·uery** | **–cqe·ach** | **–nc·omment**]
[**–f·orce**] [**–pna·me**] *type-selector* { *pname* ... | *object-selector* ... }

DESCRIPTION

The **chtype** command changes the element type of one or more existing elements or renames one or more existing branches. These operations involve changing the type object associated with the element or branch.

RESTRICTIONS

Identities:

For an element, you must have one of the following identities:

- Element owner
- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

For a branch, you must have one of the following identities:

- Branch creator
- Element owner

- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

Locks: An error occurs if one or more of these objects are locked:

Element:	VOB, element type, element, pool
Branch:	VOB, element type, element, branch type, branch.

Mastership: (Replicated VOBs only) For an element, your current replica must master the element. For a branch, your current replica must master the new branch type and the branch you are changing.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–nc**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
: Overrides the default with the option you specify. See the **comments** reference page.

CONFIRMATION STEP. *Default:* **chtype** prompts for confirmation if changing an element's type will change the way its versions are stored in the VOB storage pool.

–f·orce
: Suppresses the confirmation step.

SPECIFYING THE NEW TYPE. *Default:* None.

type-selector
: An element type or branch type. The type must already exist. (Exception: If you specify a global element type or global branch type, a local copy of the type is created if one does not already exist.) Specify *type-selector* in the form [*type-kind***:**]*type-name*[@*vob-selector*]

type-kind	One of **brtype** branch type **eltype** element type
type-name	Name of the type object
vob-selector	Object-selector for a VOB, in the form [**vob:**]*pname-in-vob*. The *pname-in-vob* can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

SPECIFYING THE ELEMENTS, BRANCHES, OR ACTIVITIES. *Default:* None.

[**–pna·me**] *pname* ...

One or more pathnames, each of which specifies a file or directory element. A standard pathname is valid; you need not append the extended naming symbol. That is, specifying a version is equivalent to specifying its element. Specifying a branch (for example, **foo.c@@/main/bugfix** causes an error.

If *pname* has the form of an object selector (for example, **eltype:fl2**), you must use the **–pname** option to indicate that it is a pathname. The **–pname** option must precede non-option arguments; for example:

cmd-context **chtype –nc –force –pname eltype:c_source eltype:fl2**

object-selector ...

One or more extended pathnames, each of which specifies a particular branch of an element. For example:

```
foo.c@@/main/bugfix
bar.@@/main/maint/bug405
```

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Convert an element to type **file**.

  cmd-context **chtype file hello.c**

  ```
  Change version manager and reconstruct all versions for "hello.c"? [no]
  yes
  Changed type of element "hello.c" to "file".
  ```

- Change branch **rel2_bugfix** to branch **maintenance**, providing a comment.

  cmd-context **chtype -c "rel2_bugfix no longer in use" maintenance util.c@@\main\rel2_bugfix**

  ```
  Changed type of branch "util.c@@\main\rel2_bugfix" to "maintenance".
  ```

- Convert an archive library to **compressed_file** format, suppressing confirmation prompts.

  cmd-context **chtype -force compressed_file libutil.a**

  ```
  Changed type of element "libutil.a" to "compressed_file".
  ```

SEE ALSO

cc.magic, **mkbrtype**, **mkelem**, **mkeltype**, **profile_ccase**, **rename**

chview

Changes properties of a view

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- ClearCase:

  chview { [**–cac·hesize** *size*] [**–sha·reable_dos** | **–nsh·areable_dos**] [**–reado·nly** | **–readw·rite**] } { **–cvi·ew** | *view-tag* }

- ClearCase LT:

  chview { [**–cac·hesize** *size*] [**–reado·nly** | **–readw·rite**] } { **–cvi·ew** | *view-tag* }

DESCRIPTION

The **chview** command changes various properties of a view, including the cache size, the type of DOs the view creates, and the access mode. The view server can be running when you enter this command.

Cache Size

The **–cachesize** option changes the cache size for a view and is equivalent to the **setcache –view –cachesize** command. For information on view caches, see the **setcache** reference page.

ClearCase—Type of Derived Objects Built in a Dynamic View

The **–shareable_dos** and **–nshareable_dos** options change the properties of the derived objects (DOs) created by future builds in the dynamic view. Shareable DOs are available for winkin by other views; nonshareable DOs are not available for winkin by other views. Both kinds of DOs have configuration records, but **clearmake** does not write shopping information for nonshareable DOs into the VOB. For more information about shareable and nonshareable DOs, see *Building Software*.

Using the **–shareable_dos** or **–nshareable_dos** option does not change the properties of the existing DOs in the view. To make a nonshareable DO shareable, you must use the **winkin** command. You cannot make a shareable DO nonshareable.

NOTE: You can change the DO property of the view while a **clearmake** or **omake** build is running in the view. The build will use the new property after completing the current target build.

ClearCase—Access Mode in a Dynamic View

The **–readonly** and **–readwrite** options change the access mode for the view's private data-storage area. By default, views have read-write access. If you change a view's property to read-only, you cannot use the view to perform any operation that creates new files in view-private storage (for example, checkouts or builds).

Changing a view's property to read-only does not prevent users from changing the view's config spec. For information on restricting changes to the config spec, see the **mkview** reference page.

RESTRICTIONS

Identities: You must have one of the following identities:

- View owner
- On UNIX: **root** on the **view_server** host (For more information about **view_server**, see the *Administrator's Guide*.)

Locks: No locks apply.

Mastership: (Replicated VOBs) No mastership restrictions.

OPTIONS AND ARGUMENTS

–cac·hesize *size*
Specifies a size for the **view_server** cache. *size* must be an integer value of bytes, optionally followed by the letter **k** to specify kilobytes or **m** to specify megabytes; for example, **800k** or **3m**.

–sha·reable_dos
Specifies that DOs created by future builds in the view are shareable.

–nsh·areable_dos
Specifies that DOs created by future builds in the view are not shareable.

–reado·nly | **–readw·rite**
Changes the access mode of the view.

–cvi·ew

Changes the properties of the current view. On UNIX systems, if there is a working directory view, **chview** changes it; otherwise, **chview** changes the set view.

view-tag

Changes the properties of the view specified by *view-tag*.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Change the cache size for view **smg_test**.

  cmd-context **chview –cachesize 500k smg_test**

  ```
  The new view server cache limits are:
  Lookup cache:              49152 bytes
  Readdir cache:            204800 bytes
  File stats cache:          86016 bytes
  Object cache:             172032 bytes
  Total cache size:            512000 bytes
  ```

- (ClearCase) Change the current view to create nonshareable DOs.

  cmd-context **chview –nshareable_dos –cview**

  ```
  Properties: nshareable_dos
  ```

SEE ALSO

mkview, **lssite**, **setsite**, **winkin**

clearaudit

Non-**clearmake** build and shell command auditing facility for dynamic views

APPLICABILITY

Product	Command Type
ClearCase	command

Platform
UNIX
Windows

SYNOPSIS

clearaudit [[**–c**] *shell_cmd*]

DESCRIPTION

NOTE: clearaudit is applicable to dynamic views only.

The **clearaudit** command runs an audited shell with the same view and working directory as the current process. MVFS files created within an audited shell (or any of its children) are derived objects (DOs). When it exits, an audited shell creates a configuration record (CR) and associates it with each of the newly created DOs.

The CR and DOs produced by **clearaudit** are similar to those created by **clearmake**. They can be listed, compared, and deleted with the same **cleartool** commands used for other DOs (see below). They can be shared with other views through explicit **winkin** commands, but they cannot be winked in by **clearmake**. They can be checked in as DO versions. For more information about configuration records, see *Building Software*.

clearaudit itself is not a shell. It starts an audit and then executes an underlying shell. **clearaudit** determines which shell to run as follows:

- First choice: the value of environment variable **CLEARAUDIT_SHELL**, which must be the full pathname of a program
- Second choice: the value of the UNIX environment variable **SHELL**,or the Windows environment variable **COMSPEC**. These environment variables must be set to the full pathname of a program
- If no EV is set: the Bourne shell, **/bin/sh** (UNIX) or **cmd.exe** (Windows)

View Context

On UNIX systems, the process from which you invoke **clearaudit** must have a view context: set view or working directory view. In either case, the audited process is set to that view. An error occurs if the invoking process has no view context, or if its working directory view differs from its set view. (See the **pwv** reference page.)

On Windows systems as well, the process from which you invoke **clearaudit** must have a view context for the audited process. An error occurs if the invoking process has no view context.

Location of Temporary Build Files

clearaudit creates temporary build files in the directory specified by the CCASE_AUDIT_TMPDIR environment variable. If this EV is not set or is set to an empty value, **clearaudit** creates temporary files in the directory specified as follows:

- On UNIX systems, by the TMPDIR environment variable. If neither EV is set, **clearaudit** creates temporary files in the **/tmp** directory.
- On Windows systems, by the TMP environment variable.

All temporary files are deleted when **clearaudit** exits. If the value of CCASE_AUDIT_TMPDIR is a directory under a VOB-tag, **clearaudit** prints an error message and exits.

Auditing Any Process

clearaudit can be used to document the work performed by any process. For example, you can use **clearaudit** to audit the creation of a UNIX **tar(1)** file or a Windows **backup** operation, producing a configuration record that describes exactly which files and/or versions were archived.

Auditing a Non-ClearCase make

You can also use **clearaudit** to produce derived objects and configuration records for software builds performed with another **make** program, such as UNIX **make(1)** or Windows **nmake**. Follow these guidelines:

- On UNIX systems:
  - Set the value of SHELL to **/usr/atria/bin/clearaudit** in the makefile.
  - Set your process's CLEARAUDIT_SHELL environment variable to your normal shell, for example, **/bin/sh**. This prevents recursive invocation of **clearaudit**: if CLEARAUDIT_SHELL is not set, **clearaudit** attempts to start the shell specified in SHELL, which was set to **clearaudit**.
  - If you want to produce a single CR for each target's build script, structure your makefiles so that each build script is a single shell command. Use continuation lines (\) as necessary.

- On Windows systems:
  - Set the value of **COMSPEC** to *ccase-home-dir***bin****clearaudit** in the makefile.
  - Set your process's **CLEARAUDIT_SHELL** environment variable to your normal shell, for example,**%SYSTEMROOT%\system32\cmd.exe**. This prevents recursive invocation of **clearaudit**: if **CLEARAUDIT_SHELL** is not set, **clearaudit** attempts to start the shell specified in **COMSPEC**, which was set to **clearaudit**.
  - If you want to produce a single CR for each target's build script, structure your makefiles so that each build script is a single shell command. Use continuation lines (^) as necessary.

UNIX Systems Only—Auditing a Shell Script

A shell script that begins with the following line is executed in an audited shell:

```
#! /usr/atria/bin/clearaudit
```

Be sure that the process from which the script is invoked has **CLEARAUDIT_SHELL** set, as described above.

OPTIONS AND ARGUMENTS

–c

Most UNIX shells (including **sh**, **csh**, **tcsh**, and **ksh**) and some Windows shells (including **cmd.exe**), require the use of the **–c** option, which tells the shell what command to execute. This option must precede any *shell_cmd* arguments.

shell_cmd

One or more words, which are passed as arguments to **$CLEARAUDIT_SHELL**, **$SHELL**, or **/bin/sh** (UNIX); or to **%CLEARAUDIT_SHELL%**, **%COMSPEC%**, or **cmd.exe** (Windows).

EXAMPLES

- Run program **myscr** in an audited C shell.

  `%` **env SHELL=/bin/csh clearaudit -c myscr**

- Run program **validation_suite** in an audited third-party shell tool.

  `C:\>` **set CLEARAUDIT_SHELL= R:\MKSNT\mksnt\bin\sh.exe**

  `C:\>` **clearaudit /c validation_suite**

- This example shows a typical CR produced by **clearaudit**. It describes all files produced by a software build with UNIX **make**. View-private files are marked with time stamps.

```
Target ClearAudit_Shell built by block.user
Host "starfield" running IRIX 4.0.1 (IP6)
Reference Time 16-May-99.10:24:08, this audit started 16-May-99.10:24:08
View was starfield:/usr/people/block/cc_views/view.bl62
Initial working directory was /vobs/doc/reference_man/test
----------------------------
MVFS objects:
----------------------------
/vobs/doc/reference_man/test/hello@@16-May.10:25.16742
/vobs/doc/reference_man/test/hello.c <16-May-99.10:11:34>
/vobs/doc/reference_man/test/hello.o@@16-May.10:25.16740
/vobs/doc/reference_man/test/makefile <16-May-99.10:23:57>
```

- Run a batch file that performs a backup in an audited shell; create an empty derived object (**bkup_do**) whose CR lists all of the backed-up objects.

  `C:\>` **clearaudit /c audit_bkup C:\users e:**

  `Batch file` **audit_bkup**`:`

```
rem
echo Audited backup of %1
echo Backup destination is %2
backup %1 %2 /s
rem
echo Creating derived object bkup_do
echo "" > .\bkup_dofc
```

SEE ALSO

catcr, **clearmake**, **diffcr**, **lsdo**, **omake**, **pwv**, **rmdo**, **scrubber**, **setview**, **make(1)**, **sh(1)**, **tar(1)**, *Building Software*

clearbug

Creates problem report for Rational Technical Support

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX

SYNOPSIS

clearbug [**–s·hort**] [**–p** *bug-priority*] [**–r yes/no**] [**–l** *alternate-logfile-name*]

DESCRIPTION

clearbug gathers information from your current processing context: date/time, version of operating system, versions of ClearCase or ClearCase LT tools, your UNIX and ClearCase or ClearCase LT contexts, system error logs, and so on. It sends this information to `stdout`, from which you can cut-and-paste it into a problem report for Technical Support. Run **clearbug** from a directory somewhere below the root of the VOB in question, so that the view and VOB information are recorded.

clearbug is self-documenting, displaying detailed instructions before it prompts you for information. **clearbug** first prompts you for the priority of the bug and whether it is reproducible, then gathers the following information:

- System information
- Version numbers of ClearCase or ClearCase LT programs
- User and group information
- Working directory pathname
- VOB information (from **describe**.)
- View information
- Information about mounts
- Active VOBs on the current machine

- The last 20 lines of the system log file

Send the problem report to Technical Support.

OPTIONS AND ARGUMENTS

–s·hort

Displays only the prompts for priority and reproducibility; suppresses the initial text that explains how to submit the problem, lists log files to examine, and describes the priorities.

–p *bug-priority*

Sets priority level of the bug:

1	Urgent problem; no useful work can be done.
2	Serious problem; a major function is experiencing a reproducible problem which causes major inconvenience; no easy workaround.
3	Problem; an important function is experiencing an intermittent problem, or a common nonessential operation is failing consistently.
4	Minor problem; all other errors. Inconvenience can be tolerated.
5	Request for enhancement.

–r yes/no

Specifies whether the bug is reproducible.

–l *alternate-logfile-name*

Specifies an alternate name for the log file. By default, the log is displayed on the screen and written to `./clearbug.log.`*date*`.`*time*. If you do not want a log file, specify **–l /dev/null**.

cleardescribe

Lists or changes the properties of an object graphically

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

cleardescribe { *object-selector* | *pname* } ...

DESCRIPTION

The **cleardescribe** command invokes the **describe** command with the **–graphical** option.

OPTIONS AND ARGUMENTS

The syntax of the **cleardescribe** command is the same as that for the graphical version of **describe**. See the **describe** reference pages for a description of the command-line options.

SEE ALSO

describe, **chevent**, **lstype**, **lslock**, **mklabel**, **rmlabel**, **mktrigger**, **rmtrigger**, **lock**, **unlock**, **reserve**, **unreserve**, **protect**, **mkattype**, **mkbrtype**, **mkeltype**, **mkhltype**, **mklbtype**, **mktrtype**

cleardiff

Compares or merges text files

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

- UNIX only—Compare files:

  cleardiff [**–tin·y** | **–win·dow**] [**–dif·f_format** | **–ser·ial_format** | **–col·umns** *n*]
  [**–hea·ders_only** | **–qui·et** | **–sta·tus_only**] [**–b·lank_ignore**] *pname1 pname2* ...

- UNIX only—Merge files:

  cleardiff –out *output-pname* [**–bas·e** *pname*] [**–q·uery** | **–qal·l** | **–abo·rt**]
  [**–tin·y** | **–win·dow**] [**–dif·f_format** | **–ser·ial_format** | **–col·umns** *n*]
  [**–hea·ders_only** | **–qui·et** | **–sta·tus_only**] *pname1 pname2* ...

- Windows only—Compare files:

  cleardiff [**–dif·f_format** | **–ser·ial_format** | **–col·umns** *n*]
  [**–hea·ders_only** | **–qui·et** | **–sta·tus_only**] [**–b·lank_ignore**] *pname1 pname2* ...

- Windows only—Merge files:

  cleardiff –out *output-pname* [**–bas·e** *pname*] [**–q·uery** | **–qal·l** | **–abo·rt**]
  [**–dif·f_format** | **–ser·ial_format** | **–col·umns** *n*]
  [**–hea·ders_only** | **–qui·et** | **–sta·tus_only**] *pname1 pname2* ...

DESCRIPTION

cleardiff is a line-oriented file comparison and merge utility with a character-based user interface. It can process up to 32 files. Alternative interfaces to **cleardiff** are:

- All platforms: **cleardiff** can be invoked with the **cleartool diff** command to perform a file comparison, or with the **cleartool merge** subcommand to perform a merge.
- UNIX only: ClearCase and ClearCase LT include a corresponding GUI tool, **xcleardiff**. This tool can be invoked with the **diff –graphical** and **merge –graphical** subcommands and through **xclearcase**.

NOTE: You cannot compare directory versions with **cleardiff**; you must use **diff**. (The **diff** command first analyzes the directory versions, then calls **cleardiff**, using the type manager mechanism.)

See the **diff** and **merge** reference pages for discussions of how files are compared and merged.

OPTIONS AND ARGUMENTS

–tin·y | **–win·dow** (UNIX only)

–window creates a child process, which displays a side-by-side report in a separate 120-character difference window. The **diff** command returns immediately. To exit the difference window, type a UNIX **INTR** character (typically, CTRL+C).

–tiny is the same as **–window**, but uses a smaller font in a 165-character difference window.

–dif·f_format | **–ser·ial_format** | **–col·umns** *n*

–diff_format reports both headers and differences in the same style as UNIX **diff**, and suppresses the file summary from the beginning of the report.

–serial_format reports differences with each line containing output from a single file, instead of using a side-by-side format.

–columns establishes the overall width of a side-by-side report. The default width is 80 (that is, only the first 40 or so characters of corresponding difference lines appear). If *n* does not exceed the default width, this option is ignored.

NOTE: Any of the following options can be invoked with the **diff –options** or **merge –options** commands.

–hea·ders_only | **–qui·et** | **–sta·tus_only**

–headers_only lists only the header line of each difference. The difference lines themselves are omitted.

–quiet suppresses the file summary from the beginning of the report.

–status_only suppresses all output, returning only an exit status: a `0` status indicates that no differences were found; a `1` status indicates that one or more differences were found. This option is useful in shell scripts.

–b·lank_ignore

Ignores extra white space characters in text lines: leading and trailing white space is ignored altogether; internal runs of white space characters are treated like a single <SPACE> character.

–out *output-pname*
Stores the output of a merge in file *output-pname*. This file is not used for input, and must not already exist.

–bas·e *pname*
Makes file *pname* the base contributor for the comparison or merge. If you omit this option, the *pname1* argument becomes the base contributor, and the comparison or merge automatically runs with the **–qall** option invoked.

–q·uery | **–qal·l** | **–abo·rt**
–query turns off automatic merging for nontrivial merges (where two or more contributors differ from the base contributor) and prompts you to proceed with every change in the from-versions. Changes in the to-version are accepted unless a conflict exists.

–qall turns off automatic acceptance of changes in which only one contributor differs from the base contributor. **cleardiff** prompts for confirmation of such changes, as it does when two or more contributors differ from the base contributor.

–abort is intended for use with scripts or batch jobs that involve merges. It allows completely automatic merges to proceed, but aborts any merge that requires user interaction.

pname1 pname2 ...
The pathnames of contributors to compare or merge. These can be view-extended or version-extended pathnames. Only one such argument is required if you also specify a file with the **–base** option.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Compare the current version of an element with a scratch copy in your home directory.

  `z:\myvob>` **cleardiff msg.c C:\users\susan\msg.c.tmp**

```
********************************
<<< file 1: msg.c
>>> file 2: C:\users\susan\msg.c.tmp
********************************
------------[changed 5]--------|-----------[changed to 5]------------
         static char msg[256]; | static char msg[BUFSIZ];
                              -|-
-----------[changed 9-11]------|-----------[changed to 9]------------
                 env_user(),   | env_user(), env_home(), e+
                 env_home(),   |-
                 env_time() ); |
                              -|
```

- Compare the same files, this time in a separate window and using a small font.

  `%` **cleardiff -tiny msg.c ~/msg.c.tmp**

- Compare the most recent versions on two branches of an element.

  `y:\lib_vob>` **cleardiff util.c@@\main\LATEST util.c@@\main\rel2_bugfix\LATEST**

SEE ALSO

diff, **diff(1)**, **merge**, **type_manager**, **xcleardiff**

cleardiffbl

Starts the **diffbl** browser

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

cleardiffbl [*baseline-selector1 baseline-selector2*]

DESCRIPTION

The **cleardiffbl** command starts the **diffbl** browser, which compares two baselines and displays differences in terms of activities or versions.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

baseline-selector1 baseline-selector2
Specifies the two baselines to compare. baseline-selector is of the form [**baseline:**]*baseline-name*[*@vob-selector*], where *vob-selector* specifies the baseline's project VOB.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Display the differences between the baselines **rev17bl1** and **rev17bl2.**

  cmd-context **cleardiffbl rev17bl1 rev17bl2**

SEE ALSO

diffbl

clearexport_ccase

Copies ClearCase data to a different VOB

APPLICABILITY

Product	Command Type
ClearCase	command

Platform
UNIX
Windows

SYNOPSIS

clearexport_ccase [**–r**] [[**–s** *date-time*] [**–p** *date-time*] | **–I** { **now** | *date-time* }]
[**–t** *temp-dir-pname*] [**–T** *translation-pname*]
[**–o** *datafile-pname*] [*source-name* ...]

DESCRIPTION

The **clearexport_ccase** utility plays a central role in cross-VOB maintenance by copying VOB objects from one VOB to another, specifically:

- Elements
- All the elements and links cataloged within a directory
- A hierarchy of directories, file elements, and VOB symbolic links

For information on moving elements from one VOB to another or splitting a VOB into two or more VOBs, see **relocate**.

The copy procedure involves two stages: export and import. During the export stage, you invoke **clearexport_ccase** in the VOB where the data to be moved resides. **clearexport_ccase** creates a *datafile* (by default, named **cvt_data**), and places in it descriptions of the objects in the VOB (for details, see Table 6).

In the import stage, you invoke **clearimport** on the *datafile*. **clearimport** reads the descriptions in the datafile and imports the information into the new VOB. Use the same config spec (the same view) for both the export and import phases.

NOTE: You can run this command only in a dynamic view. Additionally, you cannot run this command on UNIX and then run **clearimport** on Windows to import the data, or vice versa.

However, you can transfer data in either direction between a UNIX VOB and a Windows VOB by mounting the UNIX VOB on your Windows machine and running both **clearexport_ccase** and **clearimport** on the Windows machine.

Contents of the Datafile

Table 6 describes which aspects of objects **clearexport_ccase** includes in the data file. Not all objects are included in all circumstances; for example, derived objects are not available in snapshot views.

Table 6   ClearCase Items Included in Data File

Item	Description included in data file?	Notes
Directory element	Yes	The datafile includes descriptions only of the elements and cataloged links in the directory version selected by the current view; thus, for example, metadata associated with the directory version is not exported. Even exporting all directories in a VOB may miss the elements not included in the VOB as it is currently seen by the view.
File element	Yes	If the element has a user-defined element type, an error occurs if you invoke **clearimport** in a VOB in which that element type is not defined. (**clearimport** makes no effort to verify that the element type is defined the same way in both VOBs.) **clearexport_ccase** includes in the datafile descriptions of any attributes attached to an element object itself. By default, **clearexport_ccase** includes descriptions of all versions in the datafile, but you can specify command options to limit the versions that are included.
Checked-out versions	No	When **clearexport_ccase** processes a checked-out version, it issues a warning message and does not include a description of the checked-out version in the datafile.
Symbolic links	Yes	UNIX
Checked-in DOs	Yes	Config records of checked-in DOs are copied to the new VOB when you invoke **clearimport** (ClearCase).
Event records	Yes	
Type objects	Yes	

Table 6 ClearCase Items Included in Data File

Item	Description included in data file?	Notes
Attributes	Yes	
Labels	Yes	
Hyperlinks	Some	Only hyperlinks that represent merges (hyperlinks of type **Merge**) are described in the datafile.
Triggers	No	
Contents of **lost+found** directory	See Notes	To include the contents of **lost+found** in the datafile, make **lost+found** the current directory, and then run **clearexport_ccase**.

TRANSLATION OF BRANCHES AND VERSION LABELS

A label type cannot have the same name as a branch type within the same VOB. If **clearexport_ccase** encounters a label-branch naming conflict, it renames one of them. For example, the label **rel2** may become **rel2_1**. Such renaming can introduce inconsistencies over multiple runs of **clearexport_ccase**. The same label may be renamed during one run, but not during others. You can enforce consistency by using the same translation file in multiple invocations of **clearexport_ccase**. If you name such a file, using the **–T** option, **clearexport_ccase** uses it to:

- Look up each label or branch to determine whether it has been translated previously. If a match is found, the current name is translated the same way.
- Record each translation of a new label or branch for use in future lookups.

The first time you use **clearexport_ccase**, use **–T** to create a new translation file. On subsequent invocations of **clearexport_ccase**, use **–T** again and specify the same translation file, for consistent name translation.

Syntax of Translation File

The translation file consists of one or more lines in the following form:

{ **label** | **branch** } *old-name new-name*

For example, to rename the branch type **pre_import_work** to **post_import_work** and the label **BL1.7** to **IMPORT_BASE**, the translation file contains the lines:

```
branch pre_import_work post_import_work
label BL1.7 IMPORT_BASE
```

No blank lines are allowed in the file.

HANDLING OF ELEMENTS THAT CANNOT BE EXPORTED

When **clearexport_ccase** encounters an element that cannot be exported (for example, a file with format problems or a broken symbolic link), it prints an error and continues. After creating the data file, the command prints a summary of the elements that could not be exported.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. *Default:* The datafile includes the version of a directory or file element currently selected by your view. If you specify a directory as a *source-name* argument: **clearexport_ccase** processes the files in that directory but ignores the contents of the subdirectories; and **clearimport** creates a directory element for *source-name* and for each of its subdirectories.

–r

clearexport_ccase descends recursively into all *source-name* arguments that are directories. The recursive descent involves only the currently selected version of each directory element.

SELECTIVE CONVERSION OF FILES. *Default:* **clearexport_ccase** processes all elements it encounters.

–s *date-time*

clearexport_ccase processes only versions modified with new metadata (labels, branches, attributes, and so on.) or created since the specified time. Exception: **clearexport_ccase** processes a branch created at an old version if one or more new versions exist on the branch. Use this option for regular, incremental updating of an element from another one that is still under development. Be sure to specify a *date-time* that covers the entire period since the preceding update. In other situations, it is probably better to use **–I** instead of **–s**.

NOTE: In an incremental updating situation, if you remove a label or branch from an imported version, **clearimport** does not remove the label or branch from the target element.

Specify the time in one of the following formats:

*date***.***time* | *date* | *time* | **now**
where:

date := *day-of-week* | *long-date*

time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify the *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit the *date*, the default is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want to resolve the time to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, Greenwich Mean Time (GMT) is used. (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

–p *date-time*

Like **-s**, but processes only versions modified with new metadata (labels, branches, attributes, and so on) or created prior to the specified time.

–I { **now** | *date-time* }

Processes only the important versions of an element, but includes all versions created since the specified time. A version is important if any of these conditions are true:

- It is the most recent version on its branch.
- It has a version label.
- It has an attribute.
- A subbranch is sprouted from it.
- Either end of a merge arrow hyperlink is connected to it.

DIRECTORY FOR TEMPORARY FILES. *Default on UNIX systems:* the value of **P_tmpdir** (set in the **stdio.h** system include file; you can override this value by setting the **TMPDIR** environment variable). *Default on Windows systems:* the value of the **TMP** environment variable.

–t *temp-dir-pname*

Specifies an alternate directory for temporary files. This directory must already exist.

TRANSLATION OF BRANCHES AND LABELS. *Default:* As described in the section *TRANSLATION OF BRANCHES AND VERSION LABELS* on page 150, **clearexport_ccase** may rename a branch or label type to avoid naming conflicts.

–T *translation-file*

Uses the specified translation file to control conversion of label and branch names.

STORAGE LOCATION OF DATAFILE. *Default:* **clearexport_ccase** creates *datafile* **cvt_data** in the current working directory.

–o *datafile-pname*

Stores the *datafile* in the specified location. An error occurs if *datafile* already exists.

SPECIFYING FILES TO BE PROCESSED. *Default:* **clearexport_ccase** processes the current working directory (equivalent to specifying "." as the *source-name* argument). **clearimport** creates an element in the target VOB for each element in the current working directory. **clearimport** creates a directory element in the target VOB for each subdirectory of the current working directory.

source-name ...

One or more pathnames, specifying elements and/or directory versions:

- For each specified element, **clearimport** re-creates some or all of its versions.
- For each specified directory version, **clearexport_ccase** places descriptions in the *datafile* for all the elements it catalogs. **clearimport** either reuses an existing directory (creating a new version if new elements are added) or creates a directory element with a single version for the specified directory itself, and for its subdirectories.

Each *source-name* must be a simple file or directory name. This enables **clearimport** to reliably access the source data. Specifying a parent directory (**..**) causes an error, as does any UNIX pathname that includes a slash or any Windows path that includes a slash or backslash. Thus, before entering this command, change to the directory in or under which the elements to be exported reside.

EXAMPLES

- Create entries in the *datafile* for the entire tree under directory element **src**, exporting important versions created before 1999 and all versions created since the beginning of 1999.

  clearexport_ccase -r -I 1-Jan-1999 src

- Create entries in the *datafile* for the elements in the current working directory, but not in any subdirectories; store the *datafile* in a file named **newcvt**.

  clearexport_ccase -o newcvt .

SEE ALSO

clearexport_*, **clearimport**, **events_ccase**, **relocate**, **rcs(1)**, **rsh(1)** or **remsh(1)**, **sccs(1)**

clearexport_cvs

Converts CVS files to elements

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

clearexport_cvs [**–r** [**–A**]] [[**–s** *date-time*] [**–p** *date-time*] | **–I** { **now** | *date-time* }]
[**–V**] [**–S**] [**–t** *temp-dir-pname*] [**–T** *translation-file*]
[**–o** *datafile-pname*] [*source-name* ...]

DESCRIPTION

The **clearexport_cvs** command processes Concurrent Versions Systems (CVS) files so they can be imported into ClearCase or ClearCase LT elements and versions. The source data can range from a single file to an entire directory tree.

During the export stage, you invoke **clearexport_cvs** in the area where the CVS files reside. **clearexport_cvs** creates a *datafile* (by default, named **cvt_data**), and places in it descriptions of elements, branches, and versions. **clearexport_cvs** follows symbolic links it encounters during the export stage.

In the import stage, you invoke **clearimport** on the *datafile* to import information into the new VOB.

clearexport_cvs ignores most information in CVS files that is not related to version-tree structure. **clearexport_cvs** converts each CVS symbol, which names a revision or branch, into the appropriate construct: version label or branch. You can specify a translation file to control naming, enforcing consistency over multiple invocations of **clearexport_cvs**. You can use the **–S** and **–V** options to preserve CVS state attributes and CVS revision numbers as attributes of the corresponding ClearCase or ClearCase LT versions. The **–A** option enables you to export CVS

Attic subdirectories if you choose to recursively descend *source_name* arguments that specify directories.

clearexport_cvs and **clearimport** use magic files to determine which element type to use for each element **clearimport** creates. For more information on magic files and file typing, see the **cc.magic** reference page.

NOTE: You cannot run **clearexport_cvs** on UNIX and then run **clearimport** on Windows to import the data, or vice-versa. However, you can transfer data in either direction between UNIX and Windows by mounting the UNIX VOB or file-system on the Windows machine and running both **clearexport_cvs** and **clearimport** on the Windows machine.

CVS Files, Working Files, and Locks

clearexport_cvs works directly with the structured CVS files. It does not process the working files created with **co** and **co –l** commands. Be sure to check in working files with the **ci** command before running the exporter. **clearexport_cvs** issues warning messages when it encounters checked-out files, but it still processes them.

clearexport_cvs ignores all CVS locks.

CVSROOT Environment Variable

You must set the environment variable **CVSROOT** for the **cvs** command to work. If, for example, **CVSROOT** is set to **/usr/src/cvs** and an element archive is found in the CVS repository as **/usr/src/cvs/gui/windows/main.cxxx,v**, then an extraction command for a version of the element would look like

cvs get –Q –p –r1.1 gui/windows/main.cxx

SPECIAL CHARACTERS IN FILE NAMES

During import, **clearimport** invokes a shell to extract data from the datafile. **clearimport** can handle some, but not all, characters that are special to shells. Import fails for any file name that includes any of these characters:

```
`  '  "  <Tab>  [  ]  ?  *  %
```

For example:

Succeeds	Fails
`foo&bar`	`foo[bar`
`$MY_LIB`	``yellow`sunset``
`file name`	`file*name`

Before running **clearexport_cvs**, rename any file whose name contains these characters.

NOTE: If you specify *datafile-pname* or *source-name* and any of the names include spaces, you must escape the space characters (UNIX) or enclose the name in double quotes (Windows). For example, on UNIX systems:

% **clearexport_cvs src\ files**

and on Windows systems:

> **clearexport_cvs "src files"**

HANDLING OF CVS SYMBOLS

A *CVS symbol* is a mnemonic name for a particular revision or branch of a CVS file. **clearexport_cvs** translates the symbols to version labels and branch names (more precisely, to names of label types and branch types).

- Translation to version labels — Suppose a CVS symbol, **RLS_1.3**, names a revision, **3.5**. **clearexport_cvs** places a description of label type **RLS_1.3** in the *datafile*, and **clearimport** imports that label type and assigns a label of that type to the version created from the CVS revision.
- Handling of magic branches — When **clearexport_cvs** encounters a magic branch with a symbolic name in a CVS archive, it determines whether any versions have been checked in on that branch. If there are any, the magic branch's symbolic name is used as the name of the ClearCase/ClearCase LT branch; otherwise, the branch is ignored.
- Translation to branch names — Suppose a CVS symbol, **rls_1.3_fixes**, names a branch **3.5.1**. **clearexport_cvs** outputs a description of branch type **rls_1.3_fixes**, and **clearimport** creates a branch of that type at the ClearCase or ClearCase LT version created from CVS revision **3.5**.

Because there is no concept of a subbranch of the **main** branch, **clearexport_cvs** does not process single-digit symbols that name CVS branches. If a CVS symbol includes characters that are not valid in names of label types or branch types, **clearexport_cvs** replaces the offending name. For example, the CVS symbol **C++** can be renamed to "**C..**".

A label type cannot have the same name as a branch type within the same VOB. If the same CVS symbol names both a revision and a branch—not necessarily in the same CVS file—**clearexport_cvs** renames one of them. For example, after exporting a symbol **FX354**, which names a branch, it may encounter the same symbol as the name of a revision in another CVS file. In this case, it creates label type **FX354_1**.

Translation File

This renaming of CVS symbols can introduce inconsistencies over multiple runs of **clearexport_cvs**. The same symbol may be renamed during processing of some CVS files, but not change during processing of other files. You can enforce consistency by using the same translation file in multiple invocations of **clearexport_cvs**. If you name such a file, using the **–T** option, **clearexport_cvs** uses it as follows:

- To look up each CVS symbol to see how to translate it to a label type or branch type. If a match is found, the symbol is translated the same way.
- To record each translation of a new CVS symbol for use in future lookups.

The first time you use **clearexport_cvs**, use **–T** to create a new translation file. On subsequent invocations of **clearexport_cvs**, use **–T** again, specifying the same translation file for consistent name translation.

The translation file consists of one or more lines in the following form:

{ **label** | **branch** } *old-name new-name*

For example, to rename the branch type **pre_import_work** to **post_import_work** and the label **BL1.7** to **IMPORT_BASE**, the translation file contains the lines:

```
branch pre_import_work post_import_work
label BL1.7 IMPORT_BASE
```

No blank lines are allowed in the file.

HANDLING OF OBJECTS THAT CANNOT BE EXPORTED

When **clearexport_cvs** encounters a file or directory that cannot be exported (for example, a file with format problems, or a broken symbolic link), it prints an error and continues. After creating the data file, the command prints a summary of the files and directories that could not be exported.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. *Default:* If you specify a directory as a *source-name* argument: (1) **clearexport_cvs** Processes the files in that directory but ignores the contents of the subdirectories; (2) **clearimport** creates a directory element for *source-name* and for each of its subdirectories.

–r

clearexport_cvs descends recursively into all *source-name* arguments that are directories.

SELECTIVE CONVERSION OF FILES. *Default:* **clearexport_cvs** processes all CVS revisions it finds.

–s *date-time*

clearexport_cvs processes only CVS revisions that have been modified since the time specified. Use this option for regular, incremental updating of an element from a CVS file that is still under development. Be sure to specify a *date-time* that covers the entire period since the preceding update. In other situations, it is probably better to use **–I** instead of **–s**.

clearexport_cvs determines whether to process a CVS archive by using the last-modified date/time of the archive. If this date/time is before the *date-time* you specify with **–s**, **clearexport_cvs** does not process any of the revisions in the archive. If the archive's date/time is after the *date-time* you specify, **clearexport_cvs** processes the following revisions in the archive:

- All revisions created since the specified *date-time*
- All revisions that have labels
- All revisions from which branches sprout

NOTE: In an incremental updating situation, if you remove a label or branch from a CVS revision, **clearimport** does not remove the label or branch from the element.

–p *date-time*

Like **-s**, but processes only versions modified with new metadata (labels, branches, attributes, and so on) or created prior to the specified time.

–I { **now** | *date-time* }

Processes important revisions only, but includes all revisions created since the specified time. A revision is important if any of these conditions is true:

- It is the most recent version on its branch.
- It has a label.
- A subbranch is sprouted from it.

Specify the time in one of the following formats:

*date***.***time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit *date*, the default is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want to resolve the time to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, Greenwich Mean Time (GMT) is used. (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

PRESERVATION OF CVS INFORMATION AS ATTRIBUTES. *Default:* **clearexport_cvs** does not attach attributes to versions exported from CVS revisions.

–V

Attaches an attribute of type **CVS_REVISION** to each newly created version. The string value of the attribute is the CVS revision number of the exported revision. (**clearimport** creates attribute type **CVS_REVISION**, if necessary.)

If you use the **–s** option with this option, **clearimport** attaches **CVS_REVISION** attributes only to revisions created after the *date-time* you specified.

Each attribute requires about 1 KB of storage in the VOB database.

–S

If a CVS revision's state is not the default (**Exp**), attaches an attribute of type **CVS_STATE** to the newly created version. The string value of the attribute is the CVS state attribute of the exported revision.

–A

Specifies that files found in CVS **Attic** subdirectories are to be exported as if they were part of the main repository directory. For example, the CVS file, **./proj/Attic/main.c,v** is exported as the element **./proj/main.c**.

DIRECTORY FOR TEMPORARY FILES. *Default on UNIX systems:* the value of **P_tmpdir** (set in the **stdio.h** system include file; you can override this value by setting the **TMPDIR** environment variable). *Default on Windows systems:* the value of the **TMP** environment variable.

–t *temp-dir-pname*

Specifies an alternate directory for temporary files. This directory must already exist.

HANDLING OF BRANCHES AND LABELS. *Default:* As described in the section *HANDLING OF CVS SYMBOLS* on page 156, **clearexport_cvs** may rename a branch or label type to avoid naming conflicts.

–T *translation-file*

Uses the specified translation file to control and record the conversion of CVS symbols to version labels and branch names.

STORAGE LOCATION OF DATAFILE. *Default:* **clearexport_cvs** creates *datafile* **cvt_data** in the current working directory.

–o *datafile-pname*

Stores the *datafile* at the specified location. An error occurs if *datafile* already exists.

SPECIFYING FILES TO BE EXPORTED. *Default:* **clearexport_cvs** processes the current working directory (equivalent to specifying "." as the *source-name* argument). If you specify a directory as a *source-name* argument: (1) **clearexport_cvs** processes the files in that directory but ignores the contents of the subdirectories; (2) **clearimport** creates a directory element for *source-name* and for each of its subdirectories (except one named **CVS** or **cvs**).

source-name ...

One or more pathnames, specifying CVS files and/or directories:

- For each specified CVS file, **clearexport_cvs** places a description in the *datafile*.
- For each specified directory, **clearexport_cvs** places descriptions in the *datafile* for each of the CVS files it contains. **clearimport** creates a directory element for the specified directory itself, and for its subdirectories (except one named **CVS**).

Each *source-name* must be a simple file or directory name. This enables **clearimport** to reliably access the source data when it is executed. Specifying the parent directory (**..**) causes an error, as does as does any UNIX pathname that includes a slash or any Windows path that includes a backslash. Thus, before entering this command, change to the directory in or under which the elements to be exported reside. If the CVS files reside in **CVS** subdirectories, use the **–r** option to enable **clearexport_cvs** to find them.

EXAMPLES

- Create a datafile for a single CVS file.

  clearexport_cvs myprogram.c,v

- Process three CVS files in the current working directory and store the datafile in file **cvt_include**.

  clearexport_cvs -o cvt_include bgr1.h,v bgr2.h,v bgr3.h,v

SEE ALSO

clearexport_*, **clearimport**, **events_ccase**, **relocate**, **cvs(1)**, **rsh(1)** or **remsh(1)**, **sccs(1)**

clearexport_pvcs

Converts PVCS files to elements

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

clearexport_pvcs [**–r**] [[**–s** *date-time*] [**–p** *date-time*] | **–I** { **now** | *date-time* }]
[**–V**] [**–G**] [**–t** *temp-dir-pname*] [**–T** *translation-file*]
[**–o** *datafile-pname*] [*source-name* ...]

DESCRIPTION

The **clearexport_pvcs** command processes PVCS files so they can be imported into elements and versions. The source data for export can range from a single file to an entire directory tree.

During the export stage, you invoke **clearexport_pvcs** in the directory where the PVCS files reside. **clearexport_pvcs** creates a *datafile* (by default, named **cvt_data**) and places in it descriptions of elements, branches, and versions.

In the import stage, you invoke **clearimport** on the *datafile* to import information into the new VOB.

clearexport_pvcs ignores most information in PVCS files that is not related to version-tree structure. **clearexport_pvcs** converts each PVCS label, which names a revision or branch, into the appropriate construct: version label or branch. (You can specify a translation file to control naming, enforcing consistency over multiple invocations of **clearexport_pvcs**.) You can use the **–V** option to preserve PVCS revision numbers as attributes of the corresponding ClearCase or ClearCase LT versions.

clearexport_pvcs and **clearimport** use magic files to determine which element type to use for each element **clearimport** creates. For more information on magic files and file typing, see the **cc.magic** reference page.

NOTE: You cannot run **clearexport_pvcs** on UNIX and then run **clearimport** on Windows to import the data, or vice-versa. However, you can transfer data in either direction between UNIX and Windows by mounting the UNIX VOB or file-system on your Windows machine and running both **clearexport_pvcs** and **clearimport** on the Windows machine.

PVCS Files, Workfiles, and Locks

clearexport_pvcs works directly with *PVCS files*. It does not process the *workfiles* created with the **get** and **get –l** commands. Be sure to check in workfiles with the **put** command before running the exporter.

clearexport_pvcs issues warning messages when it encounters checked-out files, but it still processes them. **clearexport_pvcs** ignores all PVCS locks.

If PVCS files are stored in **VCS** (or **vcs**; case is not important) subdirectories, **clearexport_pvcs** collapses the subdirectory level. For example, PVCS file **./proj/VCS/main.c,v** becomes element **./proj/main.c**.

SPECIAL CHARACTERS IN FILE NAMES

During import, **clearimport** invokes a shell to extract data from the datafile. **clearimport** can handle some, but not all, characters that are special to shells. Import fails for any file name that includes any of these characters:

```
`  '  "  <Tab>  [  ]  ?  *  %
```

For example:

Succeeds	**Fails**
`foo&bar`	`foo[bar`
`$MY_LIB`	``yellow`sunset``
`file name`	`file*name`

Before running **clearexport_pvcs**, rename any file whose name contains these characters.

NOTE: If you specify *datafile-pname* or *source-name* and any of the names include spaces, you must escape the space characters (UNIX):

`%` **clearexport_pvcs src\ files**

or enclose the name in double quotes (Windows):

`>` **clearexport_pvcs "src files"**

HANDLING OF PVCS SYMBOLS

A *PVCS symbol* is a mnemonic name for a particular revision or branch of a PVCS file. **clearexport_pvcs** translates the symbols to version labels and branch names (more precisely, to names of label types and branch types).

- **Translation to version labels** — Suppose a PVCS symbol, **RLS_1.3**, names a revision, **3.5**. **clearexport_pvcs** places a description of label type **RLS_1.3** in the *datafile*, and **clearimport** imports that label type and assigns a label of that type to the appropriate version.
- **Translation to branch names** — Suppose a PVCS symbol, **rls_1.3_fixes**, names a branch, **3.5.1**. **clearexport_pvcs** outputs information about branch type **rls_1.3_fixes**, and **clearimport** creates a branch of that type at the appropriate version.

Because there is no concept of a subbranch of the **main** branch, **clearexport_pvcs** does not process single-digit symbols that name PVCS branches. If a PVCS symbol includes characters that are not valid in names of label types or branch types, **clearexport_pvcs** replaces the offending name. For example, the PVCS symbol **C++** may be renamed to "**C..**".

A label type cannot have the same name as a branch type within the same VOB. If the same PVCS symbol names both a revision and a branch—not necessarily in the same PVCS file—**clearexport_pvcs** renames one of them. For example, after exporting a symbol **FX354**, which names a branch, it may encounter the same symbol as the name of a revision in another PVCS file. In this case, it creates label type **FX354_1**.

Translation File

Renaming PVCS symbols can introduce inconsistencies over multiple runs of **clearexport_pvcs**. The same symbol may be renamed during processing of some PVCS files, but not chang during processing of other files. You can enforce consistency by using the same translation file in multiple invocations of **clearexport_pvcs**. If you name such a file, using the **–T** option, **clearexport_pvcs** uses it as follows:

- To look up each PVCS symbol to see how to translate it to a label type or branch type. If a match is found, the symbol is translated the same way.
- To record each translation of a new PVCS symbol, for use in future lookups.

The first time you use **clearexport_pvcs**, use **–T** to create a new translation file. On subsequent invocations of **clearexport_pvcs**, use **–T** again, specifying the same translation file, for consistent name translation.

The translation file consists of one or more lines in the following form:

{ **label** | **branch** } *old-name new-name*

For example, to rename the branch type **pre_import_work** to **post_import_work** and the label **BL1.7** to **IMPORT_BASE**, the translation file contains the lines:

```
branch pre_import_work post_import_work
label BL1.7 IMPORT_BASE
```

No blank lines are allowed in the file.

HANDLING OF OBJECTS THAT CANNOT BE EXPORTED

When **clearexport_pvcs** encounters a file or directory that cannot be exported (for example, a file with format problems or a broken symbolic link), it prints an error and continues. After creating the data file, it prints a summary of the files and directories that could not be exported.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. *Default:* If you specify a directory as a *source-name* argument: (1) **clearexport_pvcs** processes the files in that directory but ignores the contents of the subdirectories; (2) **clearimport** creates a directory element for *source-name* and for each of its subdirectories.

–r

clearexport_pvcs descends recursively into all *source-name* arguments that are directories.

SELECTIVE CONVERSION OF FILES. *Default:* **clearexport_pvcs** processes all files it encounters.

–s *date-time*

clearexport_pvcs processes only versions modified since the time specified. Use this option for regular, incremental updating of an element from a PVCS file that is still under development. Be sure to specify a *date-time* that covers the entire period since the preceding update. In other situations, it is better to use **–I** instead of **–s**.

clearexport_pvcs determines whether to process a PVCS archive by using the last-modified date/time of the archive. If this date/time is before the *date-time* you specify with **–s**, **clearexport_pvcs** does not process any of the revisions in the archive. If the date/time is after the *date-time* you specify, **clearexport_pvcs** processes the following revisions:

- All revisions created since the specified *date-time*
- All revisions that have labels

NOTE: In an incremental updating situation, if you remove a label or branch from a PVCS version, **clearimport** does not remove the label or branch from the ClearCase/ClearCase LT element.

–p *date-time*

Like **-s**, but processes only versions modified with new metadata (labels, branches, attributes, and so on) or created prior to the specified time.

–I { **now** | *date-time* }

Processes important versions only, but includes *all* versions created since the specified time. A version is important if any of these conditions is true:

- It is the most recent version on its branch
- It has a label
- A subbranch sprouts from it

Specify the time in one of the following formats:

date.*time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit *date*, the default is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want to resolve the time to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, Greenwich Mean Time (GMT) is used. (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

PRESERVATION OF PVCS INFORMATION AS ATTRIBUTES. *Default:* **clearexport_pvcs** does not attach attributes to versions exported from PVCS revisions.

–V

Attaches an attribute of type **PVCS_REVISION** to each newly created version. The string value of the attribute is the PVCS revision number of the exported revision. (**clearimport** creates attribute type **PVCS_REVISION**, if necessary.)

If you use the **–s** option with this option, **clearimport** attaches **PVCS_REVISION** attributes only to revisions created after the *date-time* you specified.

Each attribute requires about 1 KB of storage in the VOB database.

–G

If a PVCS revision has a promotion group, attaches an attribute of type **PVCS_GROUP** to the newly created version. The string value of the attribute is the promotion group of the exported revision. (**clearimport** creates attribute type **PVCS_GROUP**, if necessary.)

If you use the **–s** option with this option, **clearimport** attaches **PVCS_GROUP** attributes only to revisions created after the *date-time* you specified.

Each attribute requires about 1 KB of storage in the VOB database.

DIRECTORY FOR TEMPORARY FILES. *Default on UNIX systems:* the value of **P_tmpdir** (set in the **stdio.h** system include file; you can override this value by setting the **TMPDIR** environment variable). *Default on Windows systems:* the value of the **TMP** environment variable.

–t *temp-dir-pname*
Specifies an alternate directory for temporary files. This directory must already exist.

TRANSLATION OF BRANCHES AND LABELS. *Default:* As described in the section *HANDLING OF PVCS SYMBOLS* on page 162, **clearexport_pvcs** may rename a branch or label type to avoid naming conflicts.

–T *translation-file*
Uses the specified translation file to control and record the conversion of PVCS symbols to version labels and branch names.

STORAGE LOCATION OF DATAFILE. *Default:* **clearexport_pvcs** creates *datafile* **cvt_data** in the current working directory.

–o *datafile-pname*
Stores the *datafile* at the specified location. An error occurs if *datafile* already exists.

SPECIFYING FILES TO BE EXPORTED. *Default:* **clearexport_pvcs** processes the current working directory (equivalent to specifying "."as the *source-name* argument). **clearimport** creates an element in the new VOB for each element in the current working directory. **clearimport** creates a directory element in the new VOB for each subdirectory of the current working directory (except one named **PVCS** or **pvcs**).

source-name ...
One or more pathnames, specifying PVCS files and/or directories:

- For each specified PVCS file, **clearexport_pvcs** places a description in the *datafile*.
- For each specified directory version, **clearexport_pvcs** places descriptions in the *datafile* for all the elements it catalogs. **clearimport** creates a directory element for the specified directory itself, and for its subdirectories.

Each *source-name* must be a simple file or directory name. This enables **clearimport** to reliably access the source data. Specifying a parent directory (**..**) causes an error, as does any UNIX pathname that includes a slash or any Windows path that includes a backslash. Thus, before entering this command, change to the directory in or under which the elements to be exported reside.

EXAMPLES

- Create entries in the *datafile* for the entire tree under directory element **src**, exporting important versions created before 1999 and all versions created since the beginning of 1999.

  clearexport_pvcs -r -I 1-Jan-1999 src

- Create entries in the *datafile* for the elements in the current working directory, but not in any subdirectories; store the *datafile* in a file named **newcvt**.

  clearexport_pvcs -o newcvt .

SEE ALSO

clearexport_*, **clearimport**, **events_ccase**, **relocate**, **rcs(1)**, **rsh(1)** or **remsh(1)**, **sccs(1)**

clearexport_rcs

Converts RCS files to elements

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

clearexport_rcs [**–r**] [[**–s** *date-time*] [**–p** *date-time*] | **–I** { **now** | *date-time* }]
[**–V**] [**–S**] [**–t** *temp-dir-pname*] [**–T** *translation-file*]
[**–o** *datafile-pname*] [*source-name* ...]

DESCRIPTION

The **clearexport_rcs** command processes Revision Control System (RCS) files so they can be imported into ClearCase or ClearCase LT elements and versions. The source data can range from a single file to an entire directory tree.

During the export stage, you invoke **clearexport_rcs** in the area where the RCS files reside. **clearexport_rcs** creates a *datafile* (by default, named **cvt_data**), and places in it descriptions of elements, branches, and versions. **clearexport_rcs** follows symbolic links it encounters during the export stage.

In the import stage, you invoke **clearimport** on the *datafile* to import information into the new VOB.

clearexport_rcs ignores most information in RCS files that is not related to version-tree structure. **clearexport_rcs** converts each RCS symbol, which names a revision or branch, into the appropriate construct: version label or branch. You can specify a translation file to control naming, enforcing consistency over multiple invocations of **clearexport_rcs**. You can use the **–S** and **–V** options to preserve RCS state attributes and RCS revision numbers as attributes of the corresponding ClearCase or ClearCase LT versions.

clearexport_rcs and **clearimport** use magic files to determine which element type to use for each element **clearimport** creates. For more information on magic files and file typing, see the **cc.magic** reference page.

NOTE: You cannot run **clearexport_rcs** on UNIX and then run **clearimport** on Windows to import the data, or vice-versa. However, you can transfer data in either direction between UNIX and Windows by mounting the UNIX VOB or file-system on the Windows machine and running both **clearexport_rcs** and **clearimport** on the Windows machine.

RCS Files, Working Files, and Locks

clearexport_rcs works directly with the structured *RCS files*. It does not process the *working files* created with **co** and **co –l** commands. Be sure to check in working files with the **ci** command before running the exporter. **clearexport_rcs** issues warning messages when it encounters checked-out files, but it still processes them.

clearexport_rcs ignores all RCS locks.

If RCS files are stored in **RCS** (or **rcs**; case is not important) subdirectories, **clearexport_rcs** collapses the subdirectory level in the export process. For example, RCS file **./proj/RCS/main.c,v** becomes element **./proj/main.c**.

SPECIAL CHARACTERS IN FILE NAMES

During import, **clearimport** invokes a shell to extract data from the datafile. **clearimport** can handle some, but not all, characters that are special to shells. Import fails for any file name that includes any of these characters:

```
`  '  "  <Tab>  [  ]  ?  *  %
```

For example:

Succeeds	**Fails**
`foo&bar`	`foo[bar`
`$MY_LIB`	`` yellow`sunset ``
`file name`	`file*name`

Before running **clearexport_rcs**, rename any file whose name contains these characters.

NOTE: If you specify *datafile-pname* or *source-name* and any of the names include spaces, you must escape the space characters (UNIX):

`%` **clearexport_rcs src\ files**

or enclose the name in double quotes (Windows):

`>` **clearexport_rcs "src files"**

HANDLING OF RCS SYMBOLS

An *RCS symbol* is a mnemonic name for a particular revision or branch of an RCS file. **clearexport_rcs** translates the symbols to version labels and branch names (more precisely, to names of label types and branch types).

- **Translation to version labels** — Suppose an RCS symbol, **RLS_1.3**, names a revision, **3.5**. **clearexport_rcs** places a description of label type **RLS_1.3** in the *datafile*, and **clearimport** imports that label type and assigns a label of that type to the version created from the RCS revision.
- **Translation to branch names** — Suppose an RCS symbol, **rls_1.3_fixes**, names a branch **3.5.1**. **clearexport_rcs** outputs a description of branch type **rls_1.3_fixes**, and **clearimport** creates a branch of that type at the ClearCase or ClearCase LT version created from RCS revision **3.5**.

Because there is no concept of a subbranch of the **main** branch, **clearexport_rcs** does not process single-digit symbols that name RCS branches. If an RCS symbol includes characters that are not valid in names of label types or branch types, **clearexport_rcs** replaces the offending name. For example, the RCS symbol **C++** can be renamed to "**C..**".

A label type cannot have the same name as a branch type within the same VOB. If the same RCS symbol names both a revision and a branch—not necessarily in the same RCS file—**clearexport_rcs** renames one of them. For example, after exporting a symbol **FX354**, which names a branch, it may encounter the same symbol as the name of a revision in another RCS file. In this case, it creates label type **FX354_1**.

Translation File

This renaming of RCS symbols can introduce inconsistencies over multiple runs of **clearexport_rcs**. The same symbol may be renamed during processing of some RCS files, but not chang during processing of other files. You can enforce consistency by using the same translation file in multiple invocations of **clearexport_rcs**. If you name such a file, using the **–T** option, **clearexport_rcs** uses it as follows:

- To look up each RCS symbol to see how to translate it to a label type or branch type. If a match is found, the symbol is translated the same way.
- To record each translation of a new RCS symbol for use in future lookups.

The first time you use **clearexport_rcs**, use **–T** to create a new translation file. On subsequent invocations of **clearexport_rcs**, use **–T** again, specifying the same translation file for consistent name translation.

The translation file consists of one or more lines in the following form:

{ **label** | **branch** } *old-name new-name*

For example, to rename the branch type **pre_import_work** to **post_import_work** and the label **BL1.7** to **IMPORT_BASE**, the translation file contains the lines:

```
branch pre_import_work post_import_work
label BL1.7 IMPORT_BASE
```

No blank lines are allowed in the file.

HANDLING OF OBJECTS THAT CANNOT BE EXPORTED

When **clearexport_rcs** encounters a file or directory that cannot be exported (for example, a file with format problems, or a broken symbolic link), it prints an error and continues. After creating the data file, the command prints a summary of the files and directories that could not be exported.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. *Default:* If you specify a directory as a *source-name* argument: (1) **clearexport_rcs** Processes the files in that directory but ignores the contents of the subdirectories; (2) **clearimport** creates a directory element for *source-name* and for each of its subdirectories.

–r

clearexport_rcs descends recursively into all *source-name* arguments that are directories.

SELECTIVE CONVERSION OF FILES. *Default:* **clearexport_rcs** processes all RCS revisions it finds.

–s *date-time*

clearexport_rcs processes only RCS revisions that have been modified since the time specified. Use this option for regular, incremental updating of an element from an RCS file that is still under development. Be sure to specify a *date-time* that covers the entire period since the preceding update. In other situations, it is probably better to use **–I** instead of **–s**.

clearexport_rcs determines whether to process an RCS archive by using the last-modified date/time of the archive. If this date/time is before the *date-time* you specify with **–s**, **clearexport_rcs** does not process any of the revisions in the archive. If the archive's date/time is after the *date-time* you specify, **clearexport_rcs** processes the following revisions in the archive:

- All revisions created since the specified *date-time*
- All revisions that have labels
- All revisions from which branches sprout

NOTE: In an incremental updating situation, if you remove a label or branch from an RCS revision, **clearimport** does not remove the label or branch from the element.

–p *date-time*

Like **-s**, but processes only versions modified with new metadata (labels, branches, attributes, and so on) or created prior to the specified time.

–I { **now** | *date-time* }

Processes important revisions only, but includes all revisions created since the specified time. A revision is important if any of these conditions is true:

- It is the most recent version on its branch.
- It has a label.
- A subbranch is sprouted from it.

Specify the time in one of the following formats:

*date***.***time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit *date*, the default is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want to resolve the time to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, Greenwich Mean Time (GMT) is used. (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

PRESERVATION OF RCS INFORMATION AS ATTRIBUTES. *Default:* **clearexport_rcs** does not attach attributes to versions exported from RCS revisions.

–V

Attaches an attribute of type **RCS_REVISION** to each newly created version. The string value of the attribute is the RCS revision number of the exported revision. (**clearimport** creates attribute type **RCS_REVISION**, if necessary.)

If you use the **–s** option with this option, **clearimport** attaches **RCS_REVISION** attributes only to revisions created after the *date-time* you specified.

Each attribute requires about 1 KB of storage in the VOB database.

–S

If an RCS revision's state is not the default (**Exp**), attaches an attribute of type **RCS_STATE** to the newly created version. The string value of the attribute is the RCS state attribute of the exported revision.

DIRECTORY FOR TEMPORARY FILES. *Default on UNIX systems:* the value of **P_tmpdir** (set in the **stdio.h** system include file; you can override this value by setting the **TMPDIR** environment variable). *Default on Windows systems:* the value of the **TMP** environment variable.

–t *temp-dir-pname*

Specifies an alternate directory for temporary files. This directory must already exist.

HANDLING OF BRANCHES AND LABELS. *Default:* As described in the section *HANDLING OF RCS SYMBOLS* on page 170, **clearexport_rcs** may rename a branch or label type to avoid naming conflicts.

–T *translation-file*

Uses the specified translation file to control and record the conversion of RCS symbols to version labels and branch names.

STORAGE LOCATION OF DATAFILE. *Default:* **clearexport_rcs** creates *datafile* **cvt_data** in the current working directory.

–o *datafile-pname*

Stores the *datafile* at the specified location. An error occurs if *datafile* already exists.

SPECIFYING FILES TO BE EXPORTED. *Default:* **clearexport_rcs** processes the current working directory (equivalent to specifying "." as the *source-name* argument). If you specify a directory as a *source-name* argument: (1) **clearexport_rcs** processes the files in that directory but ignores the contents of the subdirectories; (2) **clearimport** creates a directory element for *source-name* and for each of its subdirectories (except one named **RCS** or **rcs**).

source-name ...

One or more pathnames, specifying RCS files and/or directories:

- For each specified RCS file, **clearexport_rcs** places a description in the *datafile*.
- For each specified directory, **clearexport_rcs** places descriptions in the *datafile* for each of the RCS files it contains. **clearimport** creates a directory element for the specified directory itself, and for its subdirectories (except one named **RCS**).

Each *source-name* must be a simple file or directory name. This enables **clearimport** to reliably access the source data. Specifying a parent directory (**..**) causes an error, as does any UNIX pathname that includes a slash or any Windows path that includes a backslash.

Thus, before entering this command, you should change to the directory in or under which the RCS files to be exported reside. If the RCS files reside in **RCS** subdirectories, use the **–r** option to enable **clearexport_rcs** to find them.

EXAMPLES

- Create a datafile for a single RCS file.

  clearexport_rcs myprogram.c,v

- Process three RCS files in the current working directory and store the datafile in file **cvt_include**.

  clearexport_rcs -o cvt_include bgr1.h,v bgr2.h,v bgr3.h,v

SEE ALSO

clearexport_*, **clearimport**, **events_ccase**, **relocate**, **rcs(1)**, **rsh(1)** or **remsh(1)**, **sccs(1)**

clearexport_sccs

Converts SCCS files to ClearCase or ClearCase LT elements

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

clearexport_sccs [**–r**] [[**–s** *date-time*] [**–p** *date-time*] | **–I** { **now** | *date-time* }]
[**–V**] [**–t** *temp-dir-pname*] [**–T** *translation-file*]
[**–o** *datafile-pname*] [*source-name* ...]

DESCRIPTION

The **clearexport_sccs** command exports Source Code Control System (SCCS) files so they can be imported into ClearCase or ClearCase LT elements and versions. The source data can range from a single file to an entire directory tree.

During the export stage, you invoke **clearexport_sccs** in the area where the SCCS files reside. **clearexport_sccs** creates a *datafile* (by default, named **cvt_data**) containing descriptions of elements, branches, and versions. If any of the files to be processed reside below (rather than in) the current working directory, **clearexport_sccs** includes descriptions of the corresponding directory element(s) in *datafile*. **clearexport_sccs** follows symbolic links it encounters during the export stage.

In the import stage, you invoke **clearimport** on *datafile* to import information into the new VOB.

clearexport_sccs ignores information in SCCS files that is not related to version-tree structure; this includes flags, ID keywords, user lists, and Modification Request numbers. You can specify a translation file to control naming, enforcing consistency over multiple invocations of **clearexport_sccs**. You can use the **–V** option to preserve SCCS-IDs as attributes of the corresponding ClearCase or ClearCase LT versions.

clearexport_sccs and **clearimport** use magic files to determine which element type should be used for each element **clearimport** creates. For more information on magic files and file typing, see the **cc.magic** reference page.

NOTE: You cannot run **clearexport_sccs** on UNIX and then run **clearimport** on Windows to import the data, or vice-versa. However, you can transfer data in either direction between UNIX and Windows by mounting the UNIX VOB or file-system on your Windows machine and running both **clearexport_sccs** and **clearimport** on the Windows machine.

S-Files, G-Files, and P-Files

clearexport_sccs works directly with the structured SCCS s-files, which have the **s.** filename prefix. It does not process the g-files created with **get** and **get –e** commands. Be sure to check in such files with the **delta** command before running **clearexport_sccs**. **clearexport_sccs** issues warning messages when it encounters checked-out files, but it still processes them.

Other than issuing warning messages for checked-out files, **clearexport_sccs** ignores the p-files created by **get –e**.

If s-files are stored in **SCCS** (or **sccs**; case is not important) subdirectories, **clearexport_sccs** collapses the subdirectory level. For example, SCCS file **./proj/SCCS/s.main.c** becomes element **./proj/main.c**.

Multiple-Pass Export

You can process an SCCS file in several passes. For example, you can use **clearexport_sccs** to process major revision level 1, and use it again to process major revision level 2. On the subsequent passes, **clearimport** updates an existing element correctly if that VOB element has not been modified in the interim.

SPECIAL CHARACTERS IN FILE NAMES

During import, **clearimport** invokes a shell to extract data from the datafile. **clearimport** can handle some, but not all, characters that are special to shells. Import fails for any file name that includes any of these characters:

```
`  '  "  <Tab>  [  ]  ?  *  %
```

For example:

Succeeds	**Fails**
`foo&bar`	`foo[bar`
`$MY_LIB`	``yellow`sunset``
`file name`	`file*name`

Before running **clearexport_sccs**, rename any file whose name contains these characters.

NOTE: If you specify *datafile-pname* or *source-name* and any of the names include spaces, you must escape the space characters (UNIX):

% **clearexport_sccs src\ files**

or enclose the name in double quotes (Windows):

> **clearexport_sccs "src files"**

TRANSLATION FILE

An SCCS branch ID is a name for a particular branch of an SCCS file. **clearexport_sccs** translates the symbols to names of branch types. Suppose an SCCS symbol, **rls_1.3_fixes**, names a branch **3.5.1**. **clearexport_sccs** exports a description of branch type **rls_1.3_fixes**, and **clearimport** creates a branch of that type at the ClearCase or ClearCase LT version created from SCCS revision **3.5**.

You can enforce consistency of translation by using a translation file to control the names of branches created from SCCS branches. If you name such a file using the **–T** option, **clearexport_sccs** uses it as follows:

- To look up each SCCS branch ID to see how to translate it to the name of a branch type. If a match is found, the branch ID is translated the same way.
- To record each translation of a new SCCS branch ID for use in future lookups.

The first time you use **clearexport_sccs**, use **–T** to create a new translation file. On subsequent invocations of **clearexport_sccs**, use **–T** again, specifying the same translation file for consistent name translation.

Syntax of Translation File

The translation file consists of one or more lines in the following form:

branch *old-name new-name*

For example, to rename the branch type **pre_import_work** to **post_import_work** and the branch type **old_tests** to **obsolete_tests**, the translation file contains the lines:

```
branch pre_import_work post_import_work
branch old_tests obsolete_tests
```

No blank lines are allowed in the file.

HANDLING OF OBJECTS THAT CANNOT BE EXPORTED

When **clearexport_sccs** encounters a file or directory that cannot be exported (for example, a file with format problems, or a broken symbolic link), it prints an error and continues. After creating the data file, the command prints a summary of the files and directories that could not be exported.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. *Default:* If you specify a directory as a *source-name* argument: (1) **clearexport_sccs** processes the files in that directory but ignores the contents of the subdirectories; (2) **clearimport** creates a directory element for *source-name* and for each of its subdirectories.

–r

clearexport_sccs descends recursively into all *source-name* arguments that are directories.

SELECTIVE CONVERSION OF FILES. *Default:* **clearexport_sccs** processes all SCCS revisions it finds.

–s *date-time*

clearexport_sccs processes only SCCS revisions that have been modified since the time specified. Use this option for regular, incremental updating of an element from an SCCS file that is still under development. Be sure to specify a *date-time* that covers the entire period since the preceding update. In other situations, it is probably better to use **–I** instead of **–s**.

clearexport_sccs determines whether to process an SCCS archive by using the last-modified date/time of the archive. If this date/time is before the *date-time* you specify with **–s**, **clearexport_sccs** does not process any of the revisions in the archive. If the archive's date/time is after the *date-time* you specify, **clearexport_sccs** processes the following revisions:

- All revisions created since the specified *date-time*
- All revisions from which branches sprout

NOTE: In an incremental updating situation, if you remove a branch from an SCCS revision, **clearimport** does not remove the branch from the ClearCase or ClearCase LT element.

–p *date-time*

Like **-s**, but processes only versions modified with new metadata (labels, branches, attributes, and so on) or created prior to the specified time.

–I { **now** | *date-time* }

Processes important revisions only, but includes all revisions created since the specified time. A version is important if any of these conditions is true:

- It is the most recent version on its branch.
- It has a label.
- A subbranch is sprouted from it.

Specify the time as follows:

*date***.***time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit *date*, the default is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want to resolve the time to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, Greenwich Mean Time (GMT) is used. (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

PRESERVATION OF SCCS-IDS AS ATTRIBUTES. *Default:* **clearexport_sccs** does not attach attributes to versions exported from SCCS revisions.

–V

Attaches an attribute of type **SCCS_ID** to each newly created version. The string value of the attribute is the SCCS-ID of the exported SCCS revision. (**clearimport** creates attribute type **SCCS_ID**, if necessary.)

If you use the **–s** option with this option, **clearimport** attaches **SCCS_ID** attributes only to revisions created after the *date-time* you specified.

Each attribute requires about 1 KB of storage in the VOB database.

DIRECTORY FOR TEMPORARY FILES. *Default on UNIX systems:* the value of **P_tmpdir** (set in the **stdio.h** system include file; you can override this value by setting the **TMPDIR** environment variable). *Default on Windows systems:* the value of the **TMP** environment variable.

–t *temp-dir-pname*

Specifies an alternate directory for temporary files. This directory must already exist.

BRANCH NAME TRANSLATION. *Default:* Creates ClearCase or ClearCase LT branch names based on the SCCS revision IDs.

–T *translation-file*

Uses the specified translation file to control the mapping between SCCS branches and ClearCase or ClearCase LT branches. See also the *TRANSLATION FILE* on page 177.

STORAGE LOCATION OF DATAFILE. *Default:* **clearexport_sccs** creates *datafile* **cvt_data** in the current working directory.

–o *datafile-pname*

Stores the *datafile* in the specified location. An error occurs if *datafile* already exists.

SPECIFYING FILES TO BE EXPORTED. *Default:* **clearexport_sccs** processes the current working directory (equivalent to specifying "." as the *source-name* argument). If you specify a directory as a *source-name* argument: (1) **clearexport_sccs** processes the s-files in that directory but ignores the contents of the subdirectories; (2) **clearimport** creates a directory element for *source-name* and for each of its subdirectories (except one named **SCCS** or **sccs**).

source-name ...

One or more pathnames, specifying s-files and/or directories:

- For each specified s-file, **clearexport_sccs** converts some or all of its SCCS revisions to ClearCase or ClearCase LT versions.
- For each specified directory, **clearexport_sccs** places descriptions in the *datafile* for all the s-files it contains. **clearimport** creates a directory element for the specified directory itself, and for its subdirectories (except one named **SCCS** or **sccs**).

Each *source-name* must be a simple file or directory name. This enables **clearimport** to reliably access the source data. Specifying the parent directory (**..**) causes an error, as does any UNIX pathname that includes a slash character, or any Windows pathname that includes a backslash character.

Thus, before entering this command, you should change to the directory where (or under which) the s-files to be exported reside. If the s-files reside in **SCCS** subdirectories, use the **–r** option to enable **clearexport_sccs** to find them.

EXAMPLES

- Create a datafile for a single SCCS file.

  clearexport_sccs s.myprogram.c

- Process three SCCS files in the current working directory and store the datafile in file **cvt_include**.

  clearexport_sccs -o cvt_include s.bgr1.h s.bgr2.h s.bgr3.h

SEE ALSO

clearexport_*, **clearimport**, **events_ccase**, **relocate**, **rcs(1)**, **rsh(1)** or **remsh(1)**, **sccs(1)**

clearexport_ssafe

Convert SourceSafe files to ClearCase/ClearCase LT elements

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
Windows

SYNOPSIS

clearexport_ssafe [**–r**] [[**–s** *date-time*] [**–p** *date-time*] | **–I** { **now** | *date-time* }]
[**–V**] [**–b** *target-branch* [**–v** *version-id*]]
[**–T** *translation-file*] [**–o** *datafile-pname*] [*source-name*]

DESCRIPTION

clearexport_ssafe and **clearimport** allow you to copy SourceSafe files to a VOB. You can copy a single file, multiple files, or an entire project tree.

NOTE: Before using **clearexport_ssafe**, run **analyze.exe**. Resolve any problems detected by **analyze.exe** before proceeding with the export operation.

During the export stage, you invoke **clearexport_ssafe** while connected to any SourceSafe project. **clearexport_ssafe** creates a datafile (by default, **cvt_data**), and places in it the descriptions of elements and versions.

In the import stage, you invoke **clearimport** on the datafile. **clearimport** reads the descriptions in the datafile and imports the information into the VOB.

clearexport_ssafe converts each SourceSafe label into a ClearCase or ClearCase LT version label. **clearexport_ssafe** creates separate elements for each SourceSafe file. You can specify a translation file to control naming of labels, enforcing consistency over multiple invocations of the command. You can use the **–V** option to preserve SourceSafe version numbers as attributes of the corresponding ClearCase/ClearCase LT versions.

NOTE: **clearexport_ssafe** and **clearimport** use magic files to determine which element type should be used for each element **clearimport** creates. For more information on magic files and file typing, see the **cc.magic** reference page.

NOTE: **clearexport_ssafe** uses the value of the **TMP** environment variable as the directory for temporary files.

SourceSafe Checked-out Files

clearexport_ssafe issues warning messages when it encounters checked-out SourceSafe files, but still processes the files.

SourceSafe Executable in PATH

For **clearexport_ssafe** to work, the SourceSafe command-line executable, **ss.exe**, must be in your **PATH** environment variable. **ss.exe** must work without prompting you for a user name and password. To prevent prompting, you might need to use a SourceSafe user name that matches your domain login user name, or you might need to set the **SSUSER** environment variable and set the corresponding password to blank, or no password.

SPECIAL CHARACTERS IN FILENAMES

During import, **clearimport** invokes a shell to extract data from the datafile. **clearimport** can handle some, but not all, characters that are special to shells. The import fails for any filename that includes any of these characters:

```
`  '  "  <Tab>  [  ]  ?  *  %
```

For example:

Succeeds	**Fails**
`foo&bar`	`foo?bar`
`$MY_LIB`	``yellow`sunset``

Before running **clearexport_ssafe**, rename any filename that contains these characters.

TRANSLATION FILE

Renaming of SourceSafe labels can introduce inconsistencies over multiple runs of **clearexport_ssafe**. The same label might be renamed during processing of some SourceSafe files, but remain unchanged during processing of other files. You can enforce consistency by using the same translation file in multiple invocations of **clearexport_ssafe**. If you name such a file (using the **–T** option), **clearexport_ssafe** uses it to:

- Look up each SourceSafe label to determine how to translate it to a ClearCase/ClearCase LT label type. If a match is found, **clearexport_ssafe** translates the label the same way.
- Record each translation of a new SourceSafe label for use in future lookups.

The first time you use **clearexport_ssafe**, use **–T** to create a new translation file. On subsequent invocations of **clearexport_ssafe**, use **–T** again and specify the same translation file, for consistent name translation.

Syntax of Translation File

The translation file consists of one or more lines in the following form:

label *old-name new-name*

For example, to rename the label **BL1.7** to **IMPORT_BASE** and the label **BL2** to **IMPORT_BASE2**, the translation file would be:

```
label BL1.7 IMPORT_BASE
label BL2 IMPORT_BASE2
```

No blank lines are allowed in the file.

SHARES

clearexport_ssafe does not preserve shares as hard links during conversion. Instead, shares become separate elements.

LABELS

clearexport_ssafe converts a SourceSafe label into a ClearCase/ClearCase LT label with the same name.

NOTE: Labels on SourceSafe directories are not exported.

BRANCHES

clearexport_ssafe does not convert SourceSafe branches to ClearCase/ClearCase LT branches. Instead, **clearexport_ssafe** creates separate elements.

PINS

clearexport_ssafe translates SourceSafe pins to ClearCase/ClearCase LT version labels named **PINNED**.

NO DELTAS

SourceSafe lets you assign the **NO DELTAS** attribute to elements, which directs SourceSafe to keep only the current version of the element. SourceSafe keeps a history of all versions, but the contents of only the current version. **clearexport_ssafe** creates an empty version for each of the previous versions of the element, and a version with contents for the current version. Because ClearCase and ClearCase LT do not include a feature comparable to **NODELTAS**, when you create new versions of the element, the previous versions and their contents are stored. You can, however, use the **rmver –data** command to remove a version's contents.

HANDLING OF OBJECTS THAT CANNOT BE EXPORTED

When **clearexport_ssafe** encounters a file or directory that cannot be exported (for example, a file with format problems, or a broken symbolic link), it prints an error and continues. After creating the data file, the command prints a summary of the files and directories that could not be exported.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. **clearexport_ssafe** processes the files in your SourceSafe current project. You cannot specify a pathname with the *source-name* argument.

–r

clearexport_ssafe descends recursively into all subprojects of your current project.

NOTE: Do not specify the **RECURSIVE** variable in your SourceSafe initialization file (**.ini**).

SELECTIVE CONVERSION OF FILES. *Default:* **clearexport_ssafe** processes all files it encounters.

–s *date-time*

clearexport_ssafe processes only files that have been modified since the specified time. Use this option for regular, incremental updating of an element from a SourceSafe file that is still under development. Be sure to specify a *date-time* that covers the entire period since the preceding update. In other situations, it is better to use **–I** instead of **–s**.

clearexport_ssafe determines whether to process a SourceSafe file by using the last-modified date/time of the file. If this date/time is before the *date-time* you specified with **–s**, **clearexport_ssafe** does not process any of the versions of the file. If the date/time is after the *date-time* you specified, **clearexport_ssafe** processes the following versions:

- All versions created since the specified *date-time*
- All versions that have labels

NOTE: In an incremental updating situation, if you remove a label from a SourceSafe file, **clearimport** does not remove the label from the ClearCase/ClearCase LT element.

–p *date-time*

Like **-s**, but processes only versions modified with new metadata (labels, branches, attributes, and so on) or created prior to the specified time.

–I { **now** | *date-time* }

Processes important versions only, but includes *all* versions created since the specified time. A version is important if:

- it is the most recent version
- it has a label

To export only important versions, use **–I now**. The time is specified in one of the following formats:

date.*time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** |**yesterday** |**Sunday** | ... |**Saturday** |**Sun** | ... |**Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** |... |**December** |**Jan** |... |**Dec**

Specify the *time* in 24-hour format, relative to the local time zone. If you omit the time, it defaults to `00:00:00`. If you omit the *date*, it defaults to **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want to resolve the time to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, it defaults to Greenwich Mean Time (GMT). (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

PRESERVATION OF SOURCESAFE INFORMATION AS ATTRIBUTES. *Default:* **clearexport_ssafe** does not attach attributes to versions exported from SourceSafe versions.

–V

Attaches an attribute of type **SSAFE_VERSION** to each newly-created version. The string value of the attribute is the SourceSafe version number of the processed version. (**clearimport** creates attribute type **SSAFE_VERSION**, if necessary.)

If you use the **–s** option with this option, **clearimport** attaches **SSAFE_VERSION** attributes only to versions created after the *date-time* you specified.

Each attribute requires about 1KB of storage in the VOB database.

CREATING NEW VERSIONS ON A BRANCH. *Default:* **clearimport** creates new versions of a file or directory element on the element's **main** branch.

–b *target-branch* [**–v** *version-id*]

Converts each file to a version on branch *target-branch* of the new or existing element. Whenever **clearimport** creates a new element in the target VOB, it also revises the parent directory element on branch *target-branch*. To prevent directory branching, you can check out all directories on any branch before importing. **clearimport** then uses the checked-out directories.

If branch type *target-branch* does not already exist in the target VOB, **clearimport** creates it. If an existing element already has a branch of this type, the new version extends this branch; otherwise, **clearimport** sprouts *target-branch* from version **\main\LATEST** (**\main\0** for new elements). To specify another version from which the branch should sprout, use the **–v** option.

TRANSLATION OF LABELS. *Default:* **clearexport_ssafe** may automatically rename a label type to avoid naming conflicts.

–T *translation-file*
Uses the specified translation file to control and record the conversion of SourceSafe label names to ClearCase/ClearCase LT version labels.

STORAGE LOCATION OF DATAFILE. *Default:* **clearexport_ssafe** creates the *datafile* **cvt_data** in the current working directory.

–o *datafile-pname*
Stores the *datafile* at the specified location. An error occurs if *datafile* already exists.

NOTE: Do not specify the **OUTPUT** variable in your SourceSafe initialization file (**.ini**).

SPECIFYING FILES TO BE EXPORTED. *Default:* The SourceSafe current project (equivalent to specifying "."as the *source-name* argument). **clearexport_ssafe** processes each SourceSafe file in the current project, and creates a directory element for each subproject of the current project.

source-name
The name of the SourceSafe current project, or a file or subproject in that project:

- For each SourceSafe file, **clearexport_ssafe** places a description in the datafile.
- For the current project, **clearexport_ssafe** places a description in the *datafile* for all the elements it catalogs. **clearimport** creates a directory element for the specified project itself, and for its subprojects.

The *source-name* must be a SourceSafe file or project in the SourceSafe current project. This enables **clearimport** to reliably access the source data.

clearexport_ssafe processes files and subprojects from only your SourceSafe current project. You cannot use *source-name* to specify a pathname that is not in your SourceSafe current project.

EXAMPLES

- Create entries in the *datafile* for the elements in the current project, processing important versions created before 1999 and all versions created since the beginning of 1999.

  `c:\>` **clearexport_ssafe –r –I 1-Jan-1999**

- Create entries in the *datafile* for the elements in the current project, but not in any subprojects; store the *datafile* in a file named **newcvt**.

  `c:\>` **clearexport_ssafe –o newcvt .**

- In SourceSafe, display your SourceSafe current project.

  `c:\>` **ss cp**

- In SourceSafe, list the contents of your SourceSafe current project.

  `c:\>` **ss dir**

SEE ALSO

clearexport_*, **clearimport**, **events_ccase**, **relocate**

clearfsimport

Converts file system objects to element versions

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

- UNIX only:

  clearfsimport [**–preview**] [**–follow**] [**–recurse**] [**–rmname**] [**–comment** *comment*] [**–mklabel** *label*] [**–nsetevent**] [**–identical**] [**–master**] [**–unco**] *source-name* [. . .] *target-VOB-directory*

- Windows only:

  clearfsimport [**–preview**] [**–recurse**] [**–rmname**] [**–comment** *comment*] [**–mklabel** *label*] [**–nsetevent**] [**–identical**] [**–master**] [**–unco**] [**–downcase**] *source-name* [. . .] *target-VOB-directory*

DESCRIPTION

The **clearfsimport** command reads the specified file system source objects and places them in the target VOB. This command uses magic files to determine which element type to use for each element created (see the **cc.magic** reference page).

RESTRICTIONS

Identities: You must be **root** (UNIX) or the VOB owner to run **clearfsimport** unless you invoke it with the **-nsetevent** option.

Locks: If it encounters a VOB lock while trying to write data during an import operation, **clearfsimport** pauses and retries the operation every 60 seconds until it succeeds.

Mastership: (Replicated VOBs only) Your current replica must master the branch types, branches, elements, and type objects involved in the import. With the **–master** option, the importer can create a new element and its **main** branch even if the **main** branch is not mastered by your current replica. The **–master** option also allows auto-make-branch during element creation; mastership of newly created branches is assigned to the current replica. When you specify the **–master** option, you must be able to check out the parent directory of the new element.

OPTIONS AND ARGUMENTS

PREVIEWING THE RESULTS. *Default:* No preview.

–preview

Previews the import, listing all elements that the import would add or change, as well as any checkouts that would conflict with imports, but does not import anything.

HANDLING OF UNIX SYMBOLIC LINKS. *Default:* Processes each UNIX symbolic link as a VOB symbolic link with the same link text.

–follow

Processes the object to which a UNIX symbolic link points, instead of importing the link itself into the VOB.

HANDLING OF DIRECTORY ARGUMENTS. *Default:* If a *source-name* argument names a directory to be imported, an element version is created for the directory and for each file, directory, or UNIX symbolic link residing at the top level of the directory.

–recurse

Descends recursively into all *source-name* arguments that are directories.

HANDLING OF EXISTING VOB DIRECTORIES. Default: Existing VOB directories that are not present in the sources to be imported are left as is.

–rmname

For all *source-name*s that are directories, performs an **rmname** operation on elements that already existed in the VOB but are not present in the source directory. If used in combination with **-recurse**, performs this **rmname** operation in all directories traversed.

COMMENTS. *Default:* **created by clearfsimport**

–comment *comment*

Attaches the specified comment instead of the default comment to each element version checked in to the VOB.

LABELING. *Default:* No labeling.

–mklabel *label*

Attaches the specified label instance to each element version checked in. If the corresponding label type does not exist, it is created. If the label is already attached to an existing element version, it is moved.

EVENT RECORDS. *Default:* Historical information associated with the sources is preserved.

–nsetevent
Specifies that event records and historical information for new elements and element versions show the user who executed **clearfsimport** and the date of execution, not the original data associated with the sources. This option creates element versions that are newer than the original sources; thus, the **clearfsimport** operation is not restartable after you have invoked it with this option.

CREATION OF IDENTICAL SUCCESSOR VERSIONS. *Default:* Element versions that are identical to their predecessors in the source are not created.

–identical
Creates a new version of an element—even if it is identical to its predecessor—if the source has a more recent date than that of the version in the VOB.

BRANCH MASTERSHIP. *Default:* The main branch of the element is mastered by the replica that masters the branch's type.

–master
Assigns mastership of the main branch of the element to the VOB replica at which you execute this command.

HANDLING OF EXISTING CHECKED OUT ELEMENTS. *Default:* When **clearfsimport** encounters a checked out element that already exists in the target VOB and that corresponds to a source to be imported, it prints an error and continues.

–unco
If a checked-out file element corresponding to a source file to be imported already exists in the target VOB, an **uncheckout** operation is executed on the element and the corresponding view-private file is retained with a suffix of **.keep**. The import operation then proceeds to check the element out again, then checks in the imported version.

HANDLING OF CASE OF IMPORTED ELEMENTS. *Default:* Case preserving.

–downcase
Forces downcasing of imported elements.

NOTE: When importing files from a UNIX host into a VOB on a Windows NT host, **clearfsimport** may be unable to operate correctly on files and directories that have mixed-case names. For example, if a mixed-case name is specified on the command line, **clearfsimport** will create the element name as specified, even if the case mix of the source is different. In addition, the **-rmname** option will not work where source and target elements differ only in character case. Consistent use of **-downcase** can help avoid these problems. If **-downcase** is specified for an initial import from UNIX to Windows NT, it should be specified on any subsequent imports into the same VOB directories.

SPECIFYING THE SOURCE. *Default:* None.

source-name [. . .]

Flat files, directories, and UNIX symbolic links to be imported to the VOB.

For each pathname, the leaf of the pathname is imported into the target VOB. For example, **/usr/src/lib/foo.c** is imported into the VOB **/vobs/mylib** as **/vobs/mylib/foo.c** or into the VOB **/bigvob** as **/bigvob/foo.c**.

For each file, an element version is imported. If a corresponding version of the file already exists in the target VOB and is checked out, **clearfsimport** prints and error and continues unless the **-unco** option was specified. A summary of import failures due to checked out elements is printed at the completion of the import operation.

For each directory, versions are created for all files and UNIX symbolic links it contains. Also created is a directory element with one version for the directory and each of its subdirectories. Checked out VOB directories corresponding to source directories to be imported are re-used.

SPECIFYING THE TARGET VOB. *Default:* None.

target-VOB-directory

The VOB directory to which the sources are to be imported.

EXAMPLES

- Preview a VOB—**/vobs/projectx/src**—that is to be populated with the contents of **/usr/src/projectx**. Recursively descend all directories encountered and follow UNIX symbolic links to the target objects.

  clearfsimport –preview –follow –recurse /usr/src/projectx /vobs/projectx/src

SEE ALSO

cc.magic, **events_ccase**, **relocate**, **rmname**, **uncheckout**

cleargetlog

Displays the graphical log browser

APPLICABILITY

Product	Command Type
ClearCase	command

Platform
UNIX

SYNOPSIS

cleargetlog [**–hos·t** *hostname* | **–cvi·ew** | **–tag** *view-tag* | **–vob** *pname-in-vob*]

DESCRIPTION

The **cleargetlog** command invokes **getlog** with the **–graphical** option.

OPTIONS AND ARGUMENTS

See **getlog** for a description of the optional command line arguments.

SEE ALSO

getlog

clearhistory

Shows event records for VOB-database objects graphically

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

clearhistory [**–nop·references** [[**–min·or**] [**–nco**]
[**–use·r** *login-name*] [**–bra·nch** *branch-type-selector*]]]
[[**–r·ecurse** | **–d·irectory** | **–a·ll** | **–avo·bs**]
[**–pna·me**] *pname* ...
| *object-selector* ...
]

DESCRIPTION

The **clearhistory** command invokes **lshistory** with the **–graphical** option.

OPTIONS AND ARGUMENTS

The syntax of the **clearhistory** command is the same as the graphical version of **lshistory**. See the **lshistory** reference pages for a description of the command-line options.

SEE ALSO

chevent, **cleardescribe**, **describe**, **events_ccase**, **fmt_ccase**, **lscheckout**, **lshistory**, **lspool**, **lstype**, **lsvtree**

clearimport

Reads data files created by **clearexport** tools and import elements into a VOB

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

- UNIX only:

  clearimport [**–v·erbose**] [**–i·dentical**] [**–n·setevent**] [**–master**]
  [**–d·irectory** *destination-dir*] [**–c·omment** *comment*] [**–no·load**] [**–nolab·eldir**] *datafile*

- Windows only:

  clearimport [**–v·erbose**] [**–i·dentical**] [**–n·setevent**] [**–master**] [**–pcase**]
  [**–d·irectory** *destination-dir*] [**–c·omment** *comment*] [**–no·load**] [**–nolab·eldir**] *datafile*

DESCRIPTION

During the import stage, you invoke **clearimport** within an existing VOB on the *datafile* created by **clearexport_***. For each object processed by **clearexport_*** and entered in *datafile*, **clearimport** does one of the following things:

- Creates a new element with the same versions as the original.

  The user who invokes **clearimport** becomes the owner of the elements that **clearimport** creates. **clearexport_*** and **clearimport** use magic files to determine which element type to use for each element **clearimport** creates. For more information on magic files and file-typing, see the **cc.magic** reference page.

- Checks out an existing element (optionally, on a branch) and checks in a new version for each original version that has not already been imported

If any of the original files were located in subdirectories, **clearimport** creates corresponding directory elements.

To select the directory version into which it imports elements, **clearimport** uses your view (UNIX) or your view context (Windows). However, when **clearimport** creates directory versions, it creates them on the **main** branch. The exceptions are as follows:

- If a directory already exists in the target VOB, you can check it out on a branch, and **clearimport** uses that version
- If you are exporting with **clearfsimport** and you use the **–b** option, **clearimport** imports to the specified branch

When importing into a snapshot view, you can improve performance significantly by specifying **–noload**.

Creation of Event Records During the Import Phase

clearimport documents changes to the VOB by creating event records:

- Each time **clearimport** creates a new file element, it stores an `import file element` event record, along with the standard `create element` event record, in the VOB database. The `import file element` event record is associated with the parent directory element, not with the new file element itself. **clearimport** creates the `import` event record only if the object is more than 24 hours old.
- Each time **clearimport** creates a new version, it annotates the standard `create version` event record with the comment from the original version.
- Each time **clearimport** creates a new VOB symbolic link, it creates a standard `create symbolic link` event record.
- **clearimport** always stamps the `import file element` event record with the current time. It stamps the `create version` and `create element` event records according to the original data unless you use the **–nsetevent** option.
- **clearimport** stamps the event record for the creation of a branch with the same time stamp as the version at which it was created.

  NOTE: When **clearimport** creates a branch, the branch and version 0 of the element inherit the history information (user, group, and time stamp) of the version from which the branch sprouts.
- **clearimport** stamps the event record for attachment of an attribute, label, or merge arrow with the same time stamp as the associated version.
- **clearimport** stamps event records for the creation of directory elements and type objects with the current time, and attaches the comment `created by importer` or the comment given with the **–comment** option.

- **clearimport** labels all directories and their imported parents with the union of all labels applied during import to versions contained in those directories. To defeat this behavior, use the **-nolabeldir** option.

Incremental Import and Restartability

clearimport can skip certain versions or entire elements, which gives you some flexibility:

- You can import an element in several passes. You may use incremental import for time-budgeting reasons (when there are too many versions to import at once), or because the original element is still being developed.
- You can restart **clearimport** if it terminates prematurely for any reason. **clearimport** quickly updates versions it has already imported, effectively resuming where it left off.

CAUTION: If you invoke **clearimport** with the **–nsetevent** option, it creates ClearCase or ClearCase LT versions that are newer than all the original files to be imported; thus, it is not restartable.

For each source version, **clearimport** does not create a corresponding version if it already exists on the target branch—that is, if it has the same time stamp (or a more recent one). However, even when **clearimport** bypasses version creation, it still updates the new version's metadata, such as version labels, using information from the source version.

Preserving the Case of Files Imported to Windows Systems

By default, **clearimport** converts all file names to lower case. This is generally the simplest and most efficient way to import files when you use the MVFS option of *case insensitive* mode (the default). Converting file names on import mirrors the behavior of new files created in case-insensitive mode. For more information, see the *Administrator's Guide*.

Use the **–pcase** option when you need to preserve the case of files being imported; for example, when you import files whose names differ only in case (for example, **Makefile** and **makefile**) and you have disabled case-insensitive mode.

Handling of Unreadable or Troublesome Elements

clearimport prints an error when it cannot read an element version specified in the export data file. It creates version 0 of the unreadable element and continues to process the export datafile. Additionally, if **clearimport** has any difficulty importing any elements, it prints a list of such elements after it finishes.

RESTRICTIONS

Identities: You must be **root** (UNIX) or the VOB owner to run **clearfsimport** unless you invoke it with the **-nsetevent** option.

Locks: If it encounters a VOB lock while trying to write data during an import operation, **clearimport** pauses and retries the operation every 60 seconds until it succeeds.

Mastership: (Replicated VOBs only) Your current replica must master the branch types, branches, elements, and type objects involved in the import. With the **–master** option, the importer can create a new element and its **main** branch even if the **main** branch is not mastered by your current replica. The **–master** option also allows auto-make-branch during element creation; mastership of newly created branches is assigned to the current replica. When you specify the **–master** option, you must be able to check out the parent directory of the new element.

Other: The following restrictions apply:

- If you are importing ClearCase or ClearCase LT files, use the same config spec (the same view) for the export phase (invocation of **clearexport_ccase**) and the import phase (invocation of **clearimport**).
- Do not run **clearimport** in a view that has file elements checked out from the target VOB. If **clearimport** is importing to a checked-out element in the target VOB, it cancels the checkout (**uncheckout**) of that element and deletes the view-private file from the view storage directory. Any changes that you made to the file are lost.
- Do not run **clearimport** in a UCM view—use **clearfsimport** instead.
- When you import PVCS, RCS, SourceSafe, or SCCS files, **clearimport** uses an extraction command specific to the version-control product. You must have this extraction command in your path during import:

Product	Extraction Command
PVCS	**get**
RCS	**co**
SourceSafe	**ss.exe**
SCCS	**get**

  PVCS and SCCS use the same command; make sure the correct one is in your path.

- Do not run **clearexport_*** on UNIX and then run **clearimport** on Windows to import the data, or vice-versa. However, you can transfer data in either direction between UNIX and Windows by mounting the UNIX VOB or file-system on your Windows machine and running both **clearexport_*** and **clearimport** on the Windows machine.

OPTIONS AND ARGUMENTS

VERBOSITY OF OUTPUT. *Default:* **clearimport** prints a header for each kind of type creation (label, branch, attribute, and so on). When it creates directory elements, file elements, and VOB symbolic links, it prints a header as well as element names and version-IDs. When the import is completed, **clearimport** prints a message indicating that it has closed the directories.

–v·erbose

clearimport prints messages when it performs these operations: creates types, branches, directories, VOB symbolic links, attributes, or version labels; draws merge arrows; makes branches or elements obsolete; checks in or cancels checkouts of directories; and checks out onto a branch (when using a datafile created by **clearfsimport –b**).

CREATION OF IDENTICAL SUCCESSOR VERSIONS. *Default:* When you invoke **clearimport** on a datafile created by **clearfsimport**, it does not create a new version that is identical to its predecessor.

–i·dentical

Creates a new version even if it is identical to its predecessor, but only if the file has a more recent date than the date on the version in the VOB.

TRANSCRIPTION OF HISTORY INFORMATION. *Default:* The exporters extract historical information from each object and place it in the object's description in the *datafile*. The `create version` and `create element` event records created for the object by **clearimport** have the same information—user, group, and time stamp—as the original object.

NOTE: When **clearimport** creates a branch, the branch and version 0 of the element inherit the history information of the version from which the branch sprouts.

–n·setevent

Event records and historical information for new elements and versions reflect who ran the execution of **clearimport** and when, not the original data. You cannot use this option when you import a datafile created with **clearexport_ccase**.

CAUTION: If you invoke **clearimport** with the **–nsetevent** option, it is not restartable.

MASTERSHIP OF THE MAIN BRANCH. *Default:* Assigns mastership of the element's **main** branch to the VOB replica that masters the **main** branch.

–master

Assigns mastership of the **main** branch of the element to the VOB replica in which you execute the **clearimport** command.

PRESERVE CASE OF FILES. *Default:* **clearimport** converts all file names to lower case.

–pcase

Preserves the case of files being imported.

SPECIFYING A DESTINATION DIRECTORY. *Default:* **clearimport** imports elements into the current directory.

–d·irectory *destination-dir*

clearimport imports elements into the specified VOB directory.

EVENT RECORDS AND COMMENTS. *Default:* **clearimport** attaches the comment "created by importer" to any directories created during the import process.

–c·omment *comment*

clearimport attaches the specified comment instead of the default comment.

SUPPRESSING SNAPSHOT VIEW LOADS. *Default:* Imported elements are loaded into the snapshot view.

–no·load

Suppresses the loading of imported elements into snapshot views (this option is inapplicable to dynamic views). The view's **config_spec** must include a load rule that specifies the destination of the imported elements and a version-selection rule that specifies **/main/LATEST**. To see the new elements, you must update the view after the import operation (see **update**).

Specifying this option can improve **clearimport** performance substantially. If you also specify the **–identical** option, **clearimport** does not compare element versions to determine if they are identical. Used with **–noload**, **–identical** can result in a further improvement in **clearimport** performance.

SPECIFYING THE DATA FILE. *Default:* None. You must specify the *datafile* on which you want to invoke **clearimport**.

datafile

File created by **clearexport_*** command (by default, named **cvt_data**).

SUPPRESSING DIRECTORY LABELLING. *Default:* Directories and their imported parents are labelled with the union of all labels applied during import to versions contained in those directories.

–nolab·eldir

Suppresses the labelling of imported directory elements.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Invoke **clearimport** on **cvt_data**, forcing creation of identical versions and attaching a comment to new directories.

  `%` **cleartool setview newview** *(set a view)*

  `%` **cd /vobs/newvob/import** *(go to VOB directory where data is to be imported)*

  `%` **clearimport -identical -c "rick's import" ../../../src/cvt_data** *(invoke clearimport)*

- Invoke **clearimport** on **cvt_data**, enabling verbose output and importing elements into the **\newvob** directory.

  `c:\>` **net use y: \\view\view1**

  ```
  Drive Y: is now connected to \\view\view1.
  The command completed successfully.
  ```

  `c:\>` **y:**

  `y:\>` **clearimport -verbose -directory \newvob cvt_data**

SEE ALSO

chtype, **clearexport_ccase**, **clearexport_cvs**, **clearfsimport**, **clearexport_pvcs**, **clearexport_rcs**, **clearexport_sccs**, **clearexport_ssafe**, **events_ccase**, **protect**, **rename**, **setview**, **update**

clearjoinproj

Starts the UCM Join Project Wizard

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX

SYNOPSIS

clearjoinproj

DESCRIPTION

The **clearjoinproj** command starts the UCM Join Project Wizard, which takes you through the steps required to start work on an existing UCM project.

You can also start the Join Project Wizard by running **clearprojexp**.

RESTRICTIONS

Identities: No special identity required.

Locks: A stream cannot be created if there are locks on any of the following objects: the project VOB and, for integration streams, the project.

Mastership: (Replicated VOBs only) No mastership restrictions.

OPTIONS AND ARGUMENTS

None.

EXAMPLE

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Start the Join Project Wizard.

  cmd-context **clearjoinproj**

SEE ALSO

clearprojexp, **mkstream**, **mkview**

clearlicense

Monitors and controls the product license database

APPLICABILITY

Product	Command Type
ClearCase	command
MultiSite	command

Platform
UNIX
Windows

SYNOPSIS

clearlicense [**–pro·duct** *product-name*] [**–hos·tid** | **–rel·ease** [*username* | *user-ID*] ...]

DESCRIPTION

The **clearlicense** command reports the status of the ClearCase and Attache user licensing facility. You can also use this command to release users' licenses, making them available to other users.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

Default: A report on licenses and user activity for all products with valid licenses is displayed, in the format described above.

–pro·duct *product-name*

Specifies the product whose licensing information is to be displayed or changed. *product-name* must match (including capitalization) the word that follows **–license** in the product's license entry—for example: ClearCase, Attache, MultiSite, and Attache-MultiSite.

–hos·tid

Displays the machine identifier of the host on which you invoke the command (for ClearCase) or the helper host's machine identifier (for Attache).

To obtain the license-server-host-ID for the License Registration Form (located in the *Installation Guide*; to be used when you want to add licenses to an existing license database or add a license server host), log on to the current or future license server host and run **clearlicense –hostid**.

–rel·ease [*username* | *user-ID*] ...
Specifies users (by user name or by numeric user-ID) whose licenses are to be revoked. Using **–release** without an argument causes your own license to be revoked. To discourage license battles among users, **albd_server** prevents this option from being used an excessive number of times during any single period (for Attache) or day (for all other products).

LICENSING ERRORS

This section describes errors typically encountered in licensing.

Problems with License Host File

If the UNIX file **/var/adm/atria/config/license_host** or the Windows registry key value **LicenseHost** does not exist or is empty, this message appears:

`mvfs: ERROR: view` *view-tag* `not licensed!`	*(in Attache, appears on the console of helper host)*
command-name`: .: Input/Output error`	*(in Attache, appears in the Command window)*

Additional error messages may be displayed or written to **/var/adm/atria/log/view_log** or the the Windows Event Viewer:

```
Error: You do not have a license to run ClearCase.
```

```
Error: Your license server is not specified.
```

UNIX platform errors may include:

```
Error: Unable to open file "/var/adm/atria/config/license_host": No such
file or directory.
```

```
Error: Your license server is not specified.
Create "/var/adm/atria/config/license_host" and put the license server
hostname in it.
```

Windows platform errors may include:

`Error: Unable to query value of LicenseKeys in the NT Registry; NT error`
Windows-NT-status-code`.`

`Error: Unable to read LicenseKeys from the NT Registry; NT error`
Windows-NT-status-code`.`

Problems with License Server Host

If the license server host cannot be contacted, this message appears:

`mvfs: ERROR: view` *view-tag* `not licensed!`	*(in Attache, appears on the console of helper host)*
command-name`: .: I/O error`	*(in Attache, appears in the Command window)*

In addition, error messages are displayed or are logged to

- the file **/var/adm/atria/log/view_log** (UNIX):

  ```
  Error: Cannot contact license server host "hostname"
  defined in file /var/adm/atria/config/license_host
  ```

  ```
  Error: You do not have a license to run ClearCase.
  ```

- or to the Event Viewer (Windows):

  ```
  Error: Cannot contact license server host "hostname"
  defined in
  HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\LicenseHost.
  ```

  ```
  Error: You do not have a license to run ClearCase.
  ```

  ```
  Error: Error Windows-NT-status-code reading license server hostname from the
  NT Registry.
  ```

Losing a License

If you lose your license while a view is active, this message appears when you try to use the product:

`mvfs: ERROR: view shtest - all licenses in use!` *(in Attache, appears on the console of helper host)*

SEE ALSO

Administrator's Guide

clearmake

ClearCase build utility

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

- UNIX only—Build a target:

  clearmake [**–f** *makefile*] ... [**–cukinservwdpqUNR**]
  [**–J** *num*] [**–B** *bldhost-file*] [**–C** *compat-mode*] [**–V** | **–M**] [**–O** | **–T** | **–F**]
  [**–A** *BOS-file*] ... [*macro*=*value* ...] [*target-name* ...]

- Windows only—Build a target:

  clearmake [**–f** *makefile*] ... [**–cukinservwdpqUNR**]
  [**–C** *compat-mode*] [**–V** | **–M**] [**–O** | **–T** | **–F**]
  [**–A** *BOS-file*] ... [*macro*=*value* ...] [*target-name* ...]

- Display version information for **clearmake**:

  clearmake { **–ver·sion** | **–VerAll** }

DESCRIPTION

NOTE: The distinctive features of **clearmake**, such as build auditing, derived object sharing, and build avoidance, are supported in dynamic views only. In addition, while parallel building is supported in ClearCase snapshot views, it is not supported in ClearCase LT.

clearmake is ClearCase's variant of the UNIX **make(1)** utility. It includes most of the features of UNIX System V **make(1)**. It also features compatibility modes, which enable you to use **clearmake** with makefiles that were constructed for use with other popular **make** variants, including **Gnu make**.

clearmake features a number of ClearCase extensions:

- **Configuration Lookup—**A build-avoidance scheme that is more sophisticated than the standard scheme, which uses time stamps of built objects. Configuration lookup also includes automatic dependency detection. For example, this guarantees correct build behavior as C-language header files change, even if the header files are not listed as dependencies in the makefile.
- **Derived Object Sharing—**Developers working in different views can share the files created by **clearmake** builds.
- **Creation of Configuration Records—**Software bill-of-materials records that fully document a build and support the ability to rebuild.

RESTRICTIONS

Identities: No special identity required.

Locks: If it encounters a VOB lock while trying to write data, **clearmake** pauses and retries the operation every 60 seconds until it succeeds.

OPTIONS AND ARGUMENTS

clearmake supports the options below. In general, standard **make** options are lowercase characters and **clearmake** extensions are uppercase. Options that do not take arguments can be combined on the command line (for example, **–rOi**).

–f *makefile*

Use *makefile* as the input file. If you omit this option, **clearmake** looks for input files named **makefile** and **Makefile** (in that order) in the current working directory. You can use more than one **–f** *makefile* argument pair. Multiple input files are effectively concatenated.

–u

(Unconditional) Rebuild all goal targets specified on the command line, along with the recursive closure of their dependencies, regardless of whether they need to be rebuilt. (See also **–U**.)

–k

Abandon work on the current entry if it fails, but continue on other targets that do not depend on that entry.

–i

Ignore error codes returned by commands.

–n

(No-execute) List command lines from the makefile for targets which need to be rebuilt, but do not execute them. Even lines beginning with an at-sign (`@`) are listed. See *Building Software*.

Exception: A command containing the string **$(MAKE)** is always executed on Windows systems. On UNIX systems, it is executed unless you are using **sgismake** or **sgipmake** compatibility mode. These modes do not necessarily execute **$(MAKE)**.).

–s

(Silent) Do not list command lines before executing them.

–e

Environment variables override macro assignments within the makefile. (But *macro=value* assignments on the command line or in a build options spec override environment variables.)

–r

(No-rules) Do not use the built-in rules in file *ccase-home-dir***/etc/builtin.mk** (UNIX) or *ccase-home-dir***\etc\builtin.mk** (Windows). When used with **–C**, this option also disables reading platform-specific startup files. See the **–C** option for more information.

–v

(Verbose) Slightly more verbose than the default output mode. These features are particularly useful:

- Listing of why **clearmake** does not reuse a DO that already appears in your view (for example, because its CR does not match your build configuration, or because your view does not have a DO at that pathname)
- Listing of the names of DOs being created

–w

(Working directory) Prints a message containing the working directory both before and after executing the makefile.

–d

(Debug) Quite verbose and includes a list of the environment variables that **clearmake** reads during the build. Use this option only when debugging makefiles.

–p

(Print) Lists all target descriptions and all macro definitions, including target-specific macro definitions and implicit rules, and stops before executing anything.

–q

(Query) Evaluates makefile targets, but does not run the build scripts. **clearmake** returns 0 if the targets are up to date, and 1 if any targets need to be rebuilt. Note that **clearmake** treats a winkin of a derived object as a rebuild, so **clearmake –q** returns 1 if a DO can be winked in for a target.

–c

(Check out DOs) Before building or winking in a target, **clearmake** determines whether the target is a checked-in DO visible in the view at the path named in the makefile. If such a DO is found, **clearmake -c** checks it out before rebuilding it or winking it in. If a target creates sibling DOs, target group syntax must be used in the makefile or siblings will not be subject to this behavior.

–U

Unconditionally builds goal targets only. Subtargets undergo build avoidance. If you don't specify any target on the command line, the default target is the goal. (The **–u** option unconditionally builds both goal targets and build dependencies.)

–N

Disables the default procedure for reading one or more BOS files. For a description of the default procedure, see *Building Software*.

–R

(Reuse) Examines sibling derived objects (objects created by the same build rule that created the target) when determining whether a target object in a VOB can be reused (is up to date). By default, when determining whether a target can be reused, **clearmake** ignores modifications to sibling derived objects. **–R** directs **clearmake** to consider a target out of date if its siblings have been modified or deleted.

–J *num*

Enables **clearmake**'s parallel building capability. The maximum number of concurrent target rebuilds is set to the integer *num*. If *num*=0, parallel building is disabled. (This is equivalent to not specifying a **–J** option at all.)

Alternatively, you can specify *num* as the value of environment variable **CCASE_CONC**.

–B *bldhost-file*

Uses *bldhost-file* as the build hosts file for a parallel build. If you do not specify **–B**, **clearmake** uses the file **.bldhost.$CCASE_HOST_TYPE** in your home directory. When you use **–B**, you must also use **–J** or have the **CCASE_CONC** environment variable set. For more information, see *Building Software*.

–C *compat-mode*

(Compatibility) Invokes one of **clearmake**'s compatibility modes. (Alternatively, you can use environment variable **CCASE_MAKE_COMPAT** in a BOS file or in the environment to specify a compatibility mode.) *compat-mode* can be one of the following:

gnu

Emulates the Free Software Foundation's **Gnu make** program. To define built-in make rules, **clearmake** reads **gnubuiltin.mk** instead of **builtin.mk**.

See the **makefile_gnu** reference page.

std — Invokes **clearmake** with no compatibility mode enabled. Use this option to nullify a setting of the environment variable CCASE_MAKE_COMPAT.

On UNIX systems only, *compat-mode* can also have one of the values listed below. The **–C** option is UNIX-platform independent. However, some modes try to read system-specific files, and if the files do not exist, the command fails. For more information on the platform-specific methods for dealing with this failure, see *Release Notes*.

sgismake — Emulates the **smake(1)** native to IRIX systems. To define built-in make rules, **clearmake** reads file **/usr/include/make/system.mk** instead of *ccase-home-dir***/etc/builtin.mk**. See the **makefile_smake** reference page.

sgipmake — Emulates the **pmake(1)** native to IRIX systems. To define built-in make rules, **clearmake** reads file **/usr/include/make/system.mk** instead of *ccase-home-dir***/etc/builtin.mk**. See the **makefile_pmake** reference page.

sun — Emulates the standard **make(1)** native to SunOS systems. **clearmake** defines built-in make rules in the following ways:

- If you specify **–r**, **clearmake** reads *ccase-home-dir***/etc/sunvars.mk**.
- If you do not specify **–r**, **clearmake** reads *ccase-home-dir***/etc/sunvars.mk** and *ccase-home-dir***/etc/sunbuiltin.mk**. If the current directory contains a **default.mk** file, **clearmake** reads it; otherwise, **clearmake** reads **/usr/share/lib/make/make.rules** (Solaris) or **/usr/include/make/default.mk** (SunOS).

See the **makefile_sun** reference page.

aix — Emulates the standard **make(1)** native to IBM AIX systems. See the **makefile_aix** reference page.

–V

(View) Restricts configuration lookup to the current view only. Winkin is disabled. This option is mutually exclusive with **–M**.

–M

(Makefile) Restricts dependency checking to makefile dependencies only—those dependencies declared explicitly in the makefile or inferred from a suffix rule. All detected dependencies are ignored. For safety, this disables winkin.

For example, a derived object in your view may be reused even if it was built with a different version of a header file than is currently selected by your view. This option is mutually exclusive with **–V**.

–O (Objects)
–T (Time stamps)
–F (Files)

(**–O**, **–T**, and **–F** are mutually exclusive.)

–O compares only the names and versions of objects listed in the targets' CRs; it does not compare build scripts or build options. This is useful when this extra level of checking would force a rebuild that you do not want. Examples:

- The only change from the previous build is the setting or canceling of a "compile-for-debugging" option.
- A target was built using a makefile in the current working directory. Now, you want to reuse it in a build to be performed in the parent directory, where a different makefile builds the target (with a different script, which typically references the target using a different pathname).

–T makes rebuild decisions using the standard algorithm, based on time stamps; configuration lookup is disabled. (A CR is still created for each build script execution.)

NOTE: This option causes both view-private files and derived objects to be used for build avoidance. Because the view-private file does not have a CR to be included in the CR hierarchy, the hierarchy created for a hierarchical build has a gap wherever **clearmake** reuses a view-private file for a subtarget.

–F works like **–T**, but also suppresses creation of configuration records. All MVFS files created during the build are view-private files, not derived objects.

–A *BOS-file* ...

You can use this option one or more times to specify BOS files to be read instead of, or immediately after, the ones that are read by default. Using **–N** along with this option specifies "instead of"; omitting **–N** causes **clearmake** to read the **–A** file after reading the standard BOS files.

Alternatively, you can specify a colon-separated list of BOS file pathnames (UNIX) or a semicolon-separated list such pathnames as the value of environment variable **CCASE_OPTS_SPECS**.

–ver·sion

Prints version information about the **clearmake** executable.

–VerAll

Prints version information about the **clearmake** executable and the libraries (UNIX) or ClearCase DLLs (Windows) that **clearmake** uses.

EXAMPLES

- Unconditionally build the default target in a particular makefile, along with all its dependent targets.

  `%` **clearmake -u -f project.mk**

- Build target **hello** without checking build scripts or build options during configuration lookup. Be moderately verbose in generating status messages.

  `z:\src>` **clearmake -v -O hello**

- Build the default target in the default makefile, with a particular value of make macro INCL_DIR. Base rebuild decisions on time-stamped comparisons instead of performing configuration lookup, but still produce CRs.

  `y:\>` **clearmake -T INCL_DIR=\src\include_test**

- Perform a parallel build of target **bgrs**, using up to five of the hosts listed in file **.bldhost.solaris** in your home directory.

  `%` **setenv CCASE_HOST_TYPE solaris**

  `%` **clearmake -J 5 bgrs**

- Build target **bgrs.exe**, restricting configuration lookup to the current view only. Have environment variables override makefile macro assignments.

  `z:\src>` **clearmake -e -V bgrs.exe**

- Build the default target in Sun compatibility mode.

  `%` **clearmake -C sun**

UNIX FILES

*ccase-home-dir***/etc/builtin.mk**

WINDOWS FILES

*ccase-home-dir***\etc\builtin.mk**

SEE ALSO

clearaudit, **clearmake.options**, **env_ccase**, **make(1)**, **makefile_aix**, **makefile_ccase**, **makefile_gnu**, **makefile_pmake**, **makefile_smake**, **makefile_sun**, **omake**, **promote_server**, **scrubber**, **umask(1)**, *Building Software*

clearmake.options

clearmake build options specification (BOS) file

APPLICABILITY

Product	Command Type
ClearCase	data structure
ClearCase LT	data structure

Platform
UNIX
Windows

SYNOPSIS

One or more files read by **clearmake**, specifying make macros and special targets

DESCRIPTION

NOTE: The distinctive features of **clearmake**, such as build auditing, derived object sharing, and build avoidance, are supported in dynamic views only. In addition, while parallel building is supported in ClearCase snapshot views, it is not supported in ClearCase LT.

A build options specification (BOS) file is a text file containing macro definitions and/or ClearCase special targets. We recommend that you place temporary macros (such as **CFLAGS=–g** (UNIX) or **CFLAGS=/Zi** (Windows) and others not to be included in a makefile permanently) in a BOS file, rather than specifying them on the **clearmake** command line.

By default, **clearmake** reads BOS files in this order:

1. The default BOS files:
   a. The file **.clearmake.options** in your home directory (as indicated in the password database (UNIX) or by the HOME environment variable or in the user profile (Windows). This is the place for macros to be used every time you execute **clearmake**.
   b. One or more local BOS files, each of which corresponds to one of the makefiles specified with a **–f** option, or read automatically by **clearmake**. Each BOS file has a name in the form *makefile-name***.options**. For example:

      ```
      makefile.options
      Makefile.options
      ```

```
project.mk.options
```

2. BOS files in the **CCASE_OPTS_SPECS** environment variable.
3. BOS files specified on the command line with **–A**.

If you specify **–N**, **clearmake** does not read default BOS files.

clearmake displays the names of the BOS files it reads if you specify the **–v** or **–d** option, or if **CCASE_VERBOSITY** is set to 1.

The following sections describe the various kinds of BOS file entries.

Standard Macro Definitions

A standard macro definition has the same form as a make macro defined in a makefile:

macro_name **=** *string*

For example:

`CDEBUGFLAGS = -g` (UNIX)

`CDEBUGFLAGS = /Zi` (Windows)

Target-Dependent Macro Definitions

A target-dependent macro definition takes this form:

target-list **:=** *macro_name* **=** *string*

Any standard macro definition can follow the **:=** operator; the definition takes effect only when targets in *target-list* and their dependencies are processed. Targets in the *target-list* must be separated by white space. For example:

`foo.o bar.o := CDEBUGFLAGS=-g` (UNIX)

`foo.o bar.o := CDEBUGFLAGS=/Zi` (Windows)

Two or more higher-level targets can have a common dependency. If the targets have different target-dependent macro definitions, the dependency is built using the macros for the first higher-level target **clearmake** considered building (whether or not it actually built it).

Shell Command Macro Definitions

A shell command macro definition replaces a macro name with the output of a shell command:

macro_name **:sh =** *string*

This defines the value of *macro_name* to be the output of *string*, an arbitrary shell command. In command output, **<NL>** characters are replaced by **<SPACE>** characters. For example:

`BUILD_DATE :sh = date` (UNIX)

`NT_VER :sh = VER` (Windows)

NOTE: This syntax does not work in makefiles when you are using default compatibility mode.

Special Targets

You can use the following ClearCase special targets in a build options spec:

```
.DEPENDENCY_IGNORED_FOR_REUSE
.INCREMENTAL_REPOSITORY_SIBLING
.INCREMENTAL_TARGET
.NO_CMP_NON_MF_DEPS
.NO_CMP_SCRIPT
.NO_CONFIG_REC
.NO_DO_FOR_SIBLING
.NO_WINK_IN
.SIBLING_IGNORED_FOR_REUSE
.SIBLINGS_AFFECT_REUSE
```

On UNIX only, you can also use:

```
.NOTPARALLEL
```

See the **makefile_ccase** reference page for descriptions of these targets.

Include Directives

To include one BOS file in another, use the **include** or **sinclude** (silent include) directive. For example, on UNIX:

```
include /usr/local/lib/ux.options
```

```
sinclude $(OPTS_DIR)/clearmake.options
```

On Windows:

```
include \lib\aux.options
```

```
sinclude $(OPTS_DIR)\clearmake.options
```

Comments

A BOS file can contain comment lines, which begin with a pound sign (**#**) character.

Make Macros and Environment Variables

By default, the order of precedence of macros and environment variables is as follows:

1. Target-dependent macro definitions
2. Macros specified on the **clearmake** command line
3. Make macros set in a BOS file
4. Make macro definitions in a makefile

5. Environment variables

For example, target-dependent macro definitions override all other macro definitions, and macros specified on the **clearmake** command line override those set in a BOS file .

If you use the **–e** option to **clearmake**, environment variables override macro definitions in the makefile.

All BOS file macros (except those overridden on the command line) are placed in the build script's environment. If a build script recursively invokes **clearmake**:

- The higher-level BOS file setting (now transformed into an EV) is overridden by a make macro set in the lower-level makefile. However, if the recursive invocation uses **clearmake**'s **–e** option, the BOS file setting prevails.
- If another BOS file (associated with another makefile) is read at the lower level, its make macros override those from the higher-level BOS file.

SEE ALSO

clearmake, **env_ccase**, **makefile_ccase**

clearmrgman

Manages the merging of versions graphically

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX

SYNOPSIS

- Start the graphical user interface for a UCM deliver operation:

  clearmrgman –del·iver [**–str·eam** *stream-selector*] [**–to** *tartget-view-tag*]
  [**–tar·get** *stream-selector*] [**–q·uery** | **–qal·l**]

- Start the graphical user interface for a UCM rebase operation:

  clearmrgman –reb·ase [**–vie·w** *rebase-view-tag*] [**–str·eam** *stream-selector*] [**–q·uery** | **–qal·l**]

- Start the Merge Manager:

  clearmrgman [**–tta·g** *to-view-tag*]
  [{ [**–fta·g** *from-view-tag* | **–fbr·anch** *branch*-type | **–fla·bel** *label*-type
  | **–fve·rsion** *version-selector*]
  [**–d·irectory** | **–nr·ecurse**] [**–fol·low**] [**–noa·utomerge**] [**–q·uery** | **–qal·l**]
  [**–a·ll** *vob-names* | **–avo·bs** | *pname*...] }
  { **–fil·e** *mrgman-file* }]

DESCRIPTION

The **clearmrgman** command starts the Merge Manager, a graphical tool that manages the process of merging one or more versions. It automates the processes of gathering information for a merge, starting a merge, and tracking a merge. It can also save and retrieve the state of a merge for a set of versions.

You can also use UCM-specific options to start a graphical interface for a deliver or rebase operation.

RESTRICTIONS

Identities: No special identity is required to invoke **clearmrgman**. If an operation invokes **checkout** and/or **merge**, then the identity checking of those commands is in effect.

Locks: No locks apply.

Mastership: (Replicated VOBs only) No mastership restrictions.

OPTIONS AND ARGUMENTS

STARTING THE GRAPHICAL USER INTERFACE FOR A DELIVER OR REBASE OPERATION. *Default:* Starts the Merge Manager.

–deliver
Starts the graphical user interface for a UCM deliver operation.

–stream *stream-selector*
Specifies a stream to be used as the source for the UCM deliver or rebase operation.

–to *tartget-view-tag*
Specifies a view attached to the deliver target stream in the same project or in a different project. This option is not accepted for a remote post deliver operation.

–target *stream-selector*
Specifies a non-default deliver target stream in the same or a different project. For more information about delivering to an alternative target, see **deliver**.

–rebase
Starts the graphical user interface for a UCM rebase operation.

–view *rebase-view-tag*
Specifies a view to be used for the UCM rebase operation.

SPECIFYING THE DESTINATION VIEW. *Default:* None.

–ttag *to-view-tag*
Specifies a view that is used as the target of the merge operation. Merge results are created in this view.

SPECIFYING THE FROM–VERSION. *Default:* None.

–ftag *from-view-tag*
Specifies a view that is used as the from view in the merge operation. Compares this version with the version in your view. A version of the same version is always used, even if the version has a different name in the other view.

–fbranch *branch-type*
Compares the version in your view with the latest version on the specified branch.

–flabel *label-type*
Compares the version in your view with the version selected by the specified label.

–fversion *version-selector*
Compares the version in your view with the version specified by *version-selector*.

NARROWING THE LIST OF VERSIONS TO BE CONSIDERED. Use the following options to select a subset of the versions specified by *pname* arguments and the **–all** or **–avobs** option.

–d·irectory
For each directory, considers only the directory itself, not the directory or file versions, or VOB symbolic links it catalogs.

–nr·ecurse
For each directory version, considers the file and directory versions within it, but does not descend into its subdirectories.

–fol·low
Causes VOB symbolic links to be traversed.

TURNING OFF AUTOMATIC MERGING.

–noa·utomerge
Turns off automatic merging of directories.

–q·uery
Turns off automatic merging for nontrivial merges and prompts you to proceed with every change in the from-versions. Changes in the to-version are accepted unless a conflict exists.

–qal·l
Turns off automated merging. Prompts you to determine whether you want to proceed with each change.

SPECIFYING THE VERSIONS TO BE CONSIDERED. *Default:* None.

–all *vob-names*
Considers all the versions in the specified VOB or VOBs, whether or not they are visible in your view.

–avobs
Considers all the versions in all the VOBs active (mounted) on the local host. (If environment variable CLEARCASE_AVOBS is set to a colon-separated list of VOB-tags, this set of VOBs is used instead.

pname...
Considers only the specified file versions and the subtrees under the specified directory versions.

LOADING A MERGE MANAGER FILE. *Default:* Starts a new session.

–fil·e *mrgman-file*
Loads the specified merge manager file.

EXAMPLES

- Start the Merge Manager.

  % **clearmrgman**

- Start the Merge Manager and compare the version of **foo.c** selected by the view **joe_view** with the version of **foo.c** selected by the active view.

  % **clearmrgman -ftag joe_view foo.c**

- Start the graphical user interface for a UCM deliver operation.

  % **clearmrgman -deliver rt_int/@vobs/mypvob**

SEE ALSO

deliver, **findmerge**, **merge**, **rebase**, **rmmerge**, **xmldiffmrg**

clearprojexp

Starts the UCM Project Explorer

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

clearprojexp

DESCRIPTION

The **clearprojexp** command starts the Project Explorer, a graphical utility that lets you create, manage, work in, or view UCM projects.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

None.

EXAMPLE

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Start the Project Explorer.

  clearprojexp

SEE ALSO

chbl, **chfolder**, **chproject**, **chstream**, **clearjoinproj**, **clearmrgman**, **deliver**, **mkbl**, **mkcomp**, **mkfolder**, **mkproject**, **mkstream**, **mkview**, **rebase**

clearprompt

Prompt for user input

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

- UNIX only—Prompt for text:

  clearprompt text –out·file *pname* [**–mul·ti_line**]
  [**–def·ault** *string* | **–dfi·le** *pname*]
  –pro·mpt *prompt_string* [**–pre·fer_gui**]

- UNIX only—Prompt for pathname:

  clearprompt file –out·file *pname* [**–pat·tern** *match_pattern*]
  [**–def·ault** *filename* | **–dfi·le** *pname*] [**–dir·ectory** *dir_path*]
  –pro·mpt *prompt_string* [**–pre·fer_gui**]

- UNIX only—Prompt for list:

  clearprompt list –out·file *pname* [**–items** *choice*[,*choice*] [**–choices**] | **–dfi·le** *pname*]
  –pro·mpt *prompt_string* [**–pre·fer_gui**]

- UNIX only—Prompt for continue-processing choice:

  clearprompt proceed [**–typ·e** *type*] [**–def·ault** *choice*]
  [**–mas·k** *choice*[,*choice*]] **–pro·mpt** *prompt_string* [**–pre·fer_gui**]

- UNIX only—Prompt for yes-no choice:

  clearprompt yes_no [**–typ·e** *type*] [**–def·ault** *choice*]
  [**–mas·k** *choice*[,*choice*]] **–pro·mpt** *prompt_string* [**–pre·fer_gui**]

- Windows only—Prompt for text:

  clearprompt text –out·file *pname* [**–mul·ti_line**]
  [**–def·ault** *string* | **–dfi·le** *pname*]
  –pro·mpt *prompt_string*

- Windows only—Prompt for pathname:

  clearprompt file –out·file *pname* [**–pat·tern** *match_pattern*]
  [**–def·ault** *filename* | **–dfi·le** *pname*] [**–dir·ectory** *dir_path*]
  –pro·mpt *prompt_string*

- Windows only—Prompt for list:

  clearprompt list –out·file *pname* [**–items** *choice*[**,***choice*] [**–choices**] | **–dfi·le** *pname*]
  –pro·mpt *prompt_string*

- Windows only—Prompt for continue-processing choice:

  clearprompt proceed [**–typ·e** *type*] [**–def·ault** *choice*]
  [**–mas·k** *choice*[**,***choice*]] **–pro·mpt** *prompt_string* [**–newline**]

- Windows only—Prompt for yes-no choice:

  clearprompt yes_no [**–typ·e** *type*] [**–def·ault** *choice*]
  [**–mas·k** *choice*[**,***choice*]] **–pro·mpt** *prompt_string* [**–newline**]

proceed *choice* is one of: **proceed**, **abort**

yes_no *choice* is one of: **yes**, **no**, **abort**

type is one of: **ok**, **warning**, **error**

DESCRIPTION

The **clearprompt** command prompts the user for input, then either stores the input in a file or returns an appropriate exit status. **clearprompt** is designed for use in trigger action and GUI scripts. (See the **mktrtype** reference page.)

NOTE: On Windows 98 and Windows Me systems, you must invoke **clearprompt** from the command prompt's **start /wait** command.

On UNIX systems, **clearprompt** can interact with the user either through **stdin** and **stderr** (CLI mode), or through a pop-up window (GUI mode). It uses the latter style when a trigger fires on an operation invoked through the GUI program **xclearcase**.

A trigger action script (or any other script) can use the exit status of `clearprompt proceed` or `clearprompt yes_no` to perform conditional processing:

User Selection	Exit Status
yes	0
proceed	0
no	256 (hex 100)
abort	512 (hex 200)

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

text [–mul·ti_line]
file
list
proceed
yes_no

(Mutually exclusive) Specifies the kind of user input to be prompted for:

text prompts for a single text line (with no trailing <NL> character). **text –multi_line** works just like **cleartool** comment input: in command-line mode, the user can enter any number of lines (on UNIX systems, terminated with RETURN or CTRL+D). If the **-multi_line** option is not used, a text string that exceeds 138 characters (all uppercase) or 193 characters (all lowercase) will be truncated.

file prompts for a file name or, if **–prefer_gui** is specified, opens a file browser window.

list prompts for a choice from a list of items.

proceed prompts for a choice between the alternatives **proceed** and **abort**. The default for this option is **proceed** unless you override it by specifying **–default abort**.

yes_no prompts for a choice among the alternatives **yes**, **no**, and **abort**. The default for this option is **yes** unless you override it by specifying **–default no** or **–default abort**.

–out·file *pname*

Specifies the file to which the user's input is written.

–def·ault *string*

Specifies the default text to be written to the **–outfile** file if the UNIX user presses RETURN (in CLI mode) or clicks **OK** (in GUI mode), or if the Windows user clicks **OK**.

–def·ault *filename*

Specifies the default file name string to be written to the **–outfile** file if the UNIX user presses RETURN (in CLI mode) or clicks **OK** (in GUI mode), or if the Windows user clicks **OK**.

–dfi·le *pname*

A variant of **–default**; reads the default text from a file instead of the command line. With the **list** argument, **–dfile** reads a list of comma-separated items from a file instead of from the command line.

–def·ault *choice*

Specifies the choice made if the UNIX user presses RETURN (in CLI mode) or clicks **OK** (in GUI mode), or if the Windows user clicks **OK**. The specified default is silently included in the **–mask** list.

–typ·e *type*

Specifies the *severity level*: **ok**, **warning**, or **error**. The only effect is in the way the user is prompted for input.

–ite·ms *choice*[,*choice*]

Restricts the universe of choices for a **list** interaction.

–choices

Allows the user to select more than one choice from the list.

–mas·k *choice*[,*choice*]

Restricts the choices for a **proceed** or **yes_no** interaction. Defaults for **proceed** and **yes_no**, whether or not they are explicitly specified, are included among the **–mask** arguments.

–newline

With **proceed** or **yes_no** on Windows, forces all "\n" sequences in *prompt_string* to be displayed as newline characters. Ignored in other modes (which already interpret "\n" this way) and on UNIX.

–pat·tern *match_pattern*
–dir·ectory *dir_path*

On UNIX systems, when **clearprompt file** executes in GUI mode, the file browser window contains a pathname filter. On Windows systems, the file prompt window contains the pathname filter.

By default, this window displays the names of all files in the current working directory. You can use the **–directory** and/or **–pattern** option to specify a different directory and/or file name pattern (for example, ***.c**) to restrict which file names are displayed. The user can change the filter after the file browser appears.

–pro·mpt *prompt_string*

Specifies the prompt message to be displayed.

–pre·fer_gui

Causes **clearprompt** to try to work in GUI mode; but if the attempt to open an interaction window fails, falls back to CLI mode.

Exceptions: GUI mode is forced if any of these conditions are true:

- **clearprompt** is invoked by a trigger firing on an **xclearcase** (not **cleartool**) operation. If an interaction window cannot be created, an error occurs.
- The environment variable **ATRIA_FORCE_GUI** is set to **1**.

EXAMPLES

NOTE: See the **mktrtype** reference page for additional examples.

- Prompt the user to enter a name, writing the user's input to file **uname**. Use the value of the **USER** environment variable if the user presses RETURN.

  `%` **clearprompt text -outfile uname -default $USER -prompt "Enter User Name:"**

- Ask a question and prompt for a **yes**/**no** response. Make the default response **no**.

  `y:\>` **clearprompt yes_no -prompt "Do You Want to Continue?" ^**
  -default no -mask yes,no

  On Windows 98 or Windows Me, prepend the command shell's **start /wait** command:

  `y:\>` **start /wait clearprompt yes_no -prompt "Do You Want to Continue?" ^**
  -default no -mask yes,no

- Ask a question and prompt for a **yes**/**abort** response, excluding **no** as a choice, and using a separate window if possible. The default is **yes** because no default is explicitly specified.

  `%` **clearprompt yes_no -prompt "OK to continue?" -mask abort -prefer_gui**

- Prompt for a file name. Restrict the choices to files with a **.c** extension, and write the user's selection to a file named **myfile**.

  `c:\>` **clearprompt file -prompt "Select File From List" -outfile myfile ^**
  -pattern '*.c'

SEE ALSO

mktrigger, **mktrtype**, **xclearcase**

cleartool

ClearCase and ClearCase LT user-level commands (command-line interface)

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

- Single-command mode:

  cleartool *subcommand* [*options/args*]

- Interactive mode:

  cleartool [**–e**] [**–status**]

  `cleartool>` *subcommand* `[` *options/args* `]`
  .
  .
  .
  `cleartool>` **quit**

- Display version information for ClearCase or ClearCase LT, the kernel, and **cleartool**:

  cleartool –ver·sion

- Display version information for ClearCase or ClearCase LT, the kernel, **cleartool**, and the UNIX libraries or ClearCase or ClearCase LT Windows DLLs that **cleartool** uses:

  cleartool –VerAll

cleartool

DESCRIPTION

cleartool is the primary command-line interface to ClearCase and ClearCase LT version-control and configuration management software. It has a rich set of subcommands that create, modify, and manage the information in VOBs and views.

cleartool SUBCOMMANDS

Each **cleartool** subcommand is described in its own reference page, but not all subcommands are available in ClearCase LT. See individual reference pages to determine which commands are available in ClearCase LT.

annotate
apropos
catcr
catcs
cd
chactivity
chbl
checkin
checkout
checkvob
chevent
chflevel
chfolder
chmaster
chpool
chproject
chstream
chtype
chview
cptype
deliver
describe
diff
diffbl
diffcr
dospace
edcs
endview
file
find
findmerge
get
getcache
getlog
help
hostinfo
ln
lock
ls
lsactivity
lsbl
lscheckout
lsclients
lscomp
lsdo
lsfolder
lshistory
lslock
lsmaster
lspool
lsprivate
lsproject
lsregion
lsreplica
lssite
lsstgloc
lsstream
lstype
lsview
lsvob
lsvtree
man
merge
mkactivity
mkattr
mkattype
mkbl
mkbranch
mkbrtype
mkcomp
mkdir
mkelem
mkeltype
mkfolder
mkhlink
mkhltype
mklabel
mklbtype
mkpool
mkproject
mkregion
mkstgloc
mkstream
mktag
mktrigger
mktrtype
mkview
mkvob
mount
mv
protect
protectvob
pwd
pwv
quit
rebase
recoverview
reformatview
reformatvob
register
relocate
rename
reqmaster
reserve
rmactivity
rmattr
rmbl
rmbranch
rmcomp
rmdo
rmelem
rmfolder
rmhlink
rmlabel
rmmerge
rmname
rmpool
rmproject
rmregion
rmstgloc
rmstream
rmtag
rmtrigger
rmtype
rmver
rmview
rmvob
setactivity
schedule
setcache
setcs
setplevel
setsite
setview
shell
space
startview
umount
uncheckout
unlock
unregister
unreserve
update
winkin

GETTING HELP

cleartool provides several online help facilities for its subcommands:

- **Syntax summary —** To display a syntax summary for an individual subcommand, use the **help** subcommand or the **–help** option:

cleartool help	*(syntax of all subcommands)*
cleartool help mklabel	*(syntax of one subcommand)*
cleartool mklabel –help	*(syntax of one subcommand)*

- **Reference pages — cleartool** has its own interface to the UNIX **man(1)** command and Windows Help Viewer. Enter **cleartool man** *command-name* to display the reference page for a command.

  Reference pages are also accessible from the Windows help system's main contents. On UNIX systems, type **hyperhelp main.hlp** to view the main contents.

  See the **man** and **hyperhelp** reference pages for more information.

USAGE OVERVIEW

You can use **cleartool** in either single-command mode or interactive mode. A single **cleartool** command can be invoked from a UNIX shell or the Windows command interpreter using this syntax:

cleartool *subcommand* [*options-and-args*]

If you want to enter a series of subcommands, enter the **cleartool** command with no arguments. This places you at the interactive mode prompt:

```
cleartool>
```

You can then issue any number of subcommands (simply called "commands" from now on), ending with **quit** to return to the shell or command interpreter. You can continue **cleartool** commands onto additional lines as follows:

- On UNIX systems, use the backslash (\)
- On Windows systems, use the caret (^)

Interactive Mode Options

These options apply to interactive mode:

- **–e** causes **cleartool** to exit upon encountering an error.
- **–status** returns the status (**0** or **1**) of each **cleartool** subcommand executed.

Command Options

Command options may appear in any order, but all options must precede any nonoption arguments (typically, names of files, versions, branches, and so on). If an option is followed by an additional argument, such as **–branch /main/bugfix**, there must be white space between the option string and the argument. If the argument itself includes space characters, it must be enclosed in quotes.

If a non-option argument begins with a hyphen (**–**) character, you may need to precede it with a double-hyphen argument, to prevent it from being interpreted as an option:

cleartool rmtype -lbtype -- -temporary_label-

Command Abbreviations and Aliases

Many subcommand names and option words can be abbreviated. A subcommand's syntax summary indicates all valid abbreviations. For example:

–pre·decessor

This means that you can abbreviate the option to the minimal **–pre**, or to any intermediate spelling: **–pred**, **–prede**, and so on.

For option words, the minimal abbreviation is always three characters or fewer.

A few **cleartool** commands have a built-in command alias. For example, **checkin**'s alias is **ci**; **checkout**'s alias is **co**. These commands are equivalent:

cleartool checkin test.c

cleartool ci test.c

ARGUMENTS IN cleartool COMMANDS

Arguments in **cleartool** commands specify objects—either file-system objects (which may or may not be in a VOB) or non-file-system VOB objects. File-system objects are elements, versions, VOB symbolic links, derived objects, view-private directories, and view-private files. File-system objects also include files, UNIX symbolic links, and directories that have been loaded into a snapshot view. Examples of arguments that specify file-system objects:

cleartool ls .
cleartool mkelem new_doc
cleartool checkin -nc ..\src\main.h

Non-file-system VOB objects include types (attribute, branch, element, hyperlink, label, replica, trigger), pools, hyperlinks, replicas, and VOBs. Examples of arguments that specify non-file-system VOB objects:

cleartool lock brtype:v2_release
cleartool describe vob:/vobs/smg_tmp
cleartool mkhltype tested_by

The sections *File-System Objects* and *Non-File-System VOB Objects* give more details about specifying objects.

NOTE: If a nonoption argument begins with a hyphen (–), you may need to precede it with a double-hyphen argument to prevent it from being interpreted as an option.

Use of Slashes and Backslashes on Windows Systems

Slashes (/) and backslashes (\) can be used interchangeably in pathnames in **cleartool** commands. For example, the following command is legal on a Windows host:

`z:\myvob>` **cleartool ls /srcvob/util.c**

File-System Objects

To specify a file-system object as an argument, you can use either a full or relative pathname. In many cases, you can also use these variants: a view-extended pathname (full or relative) or a version-extended pathname (full or relative).

On UNIX systems, a full pathname begins with a slash; for example:

`/usr/src/project`	*(full pathname)*
`/usr/bin/cc`	*(full pathname)*
`/view/jpb/usr/src/project/test.c`	*(view-extended full pathname)*
`/usr/src/project@@/main/3/test.c/main/bugfix/4`	*(version-extended full pathname)*

On Windows systems, a full pathname begins with an optional drive letter and a backslash; for example:

`c:\users\smg\test\test.c`	*(full pathname (non-VOB object))*
`e:\myvob\src\main.c`	*(full pathname to VOB object — e: is assigned to a view)*
`\myvob\src\main.c`	*(full pathname to VOB object; also called an absolute VOB pathname, because it begins with a VOB-tag (\myvob); only legal if current drive is assigned to a view; also used in config specs)*
`m:\smg_view\myvob\src\main.c`	*(view-extended full pathname (VOB object); the M: drive constitutes 'view-extended namespace')*
`e:\vob_src\view_priv_dir\view_priv_file`	*(full pathname (view-private file))*
`\myvob\src\main.c@@\main\3`	*(version-extended full pathname)*

NOTE: In general, you perform ClearCase and ClearCase LT on Windows operations in a view context, on a drive assigned with the Windows **net use** command or by clicking **Tools→Map Network Drive** in Windows Explorer. It is rare to work directly on **M:**, the default dynamic-views drive. However, it is common to use view-extended pathnames that include the **M:\view-tag** prefix.

On UNIX systems, a relative pathname does not begin with a slash or an implied slash (for example, ~*user*). For example:

`test.c`	*(relative pathname)*
`../lib`	*(relative pathname)*
`motif/libX.a`	*(relative pathname)*
`../../beta_vu/usr/src/project`	*(view-extended relative pathname)*
`test.c@@/main/4`	*(version-extended relative pathname)*

On Windows systems, a relative pathname does not begin with a backslash. For example:

`test.c`	*(relative pathname)*
`..\lib`	*(relative pathname)*
`tcp\libw.lib`	*(relative pathname)*
`test.c@@\main\4`	*(version-extended relative pathname)*

NOTE: On Windows systems, pathnames relative to another drive (for example, **c:\lib\util.o** when **c:** is not the current drive) are not supported.

For both full and relative pathnames:

- The standard operating system pathname of an element implicitly references the version selected by the current view.
- A view-extended pathname references the version of the element selected by the specified view.
- A version-extended pathname directly references a particular version in an element's version tree.

For more information, see the **version_selector** and **pathnames_ccase** reference pages.

NOTE: On Windows systems, although the ClearCase MVFS uses case-insensitive lookup by default, **cleartool** itself is case-sensitive.

Non-File-System VOB Objects

In **cleartool** commands, you specify non-file-system VOB objects (VOBs, types, pools, hyperlinks, and replicas) with object selectors.

Object selectors identify non-file-system VOB objects with a single string:

[*prefix*:]*name*[@*vob-selector*]

where

prefix

Identifies the kind of object. The prefix is optional if the context of the command implies the kind of object. For example,

cleartool mkbrtype brtype:v3_bugfix

is equivalent to

cleartool mkbrtype v3_bugfix

If a context does not imply any particular kind of object, **cleartool** assumes that a *name* argument with no prefix is a pathname. For example, the command **cleartool describe ddft** describes a filesystem object named **ddft** but **cleartool describe pool:ddft** describes the **ddft** pool.

If the name of a file-system object looks like a *prefix***:***name* argument, you must use the **–pname** option to identify it. (In the **mkhlink** command, the options **–fpname** and **–tpname** serve the same function.) For example, to describe a file named **lbtype:L**, enter this command:

cleartool describe –pname lbtype:L

name

The name of the object. See the section *Object Names* for the rules about composing names.

vob-selector

VOB specifier. The default is the current working directory, unless the reference page specifies otherwise. Specify *vob-selector* in the form [**vob:**]*pname-in-vob* (for some commands, the **vob:** prefix is required; this is noted in the reference page)

pname-in-vob	Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

Object Names

In object-creation commands, you must compose the object name according to these rules:

- It must contain only letters, digits, and the special characters underscore (_), period (.), and hyphen (-). A hyphen cannot be used as the first character of a name.
- It must not be a valid integer or real number. (Be careful with names that begin with “0x”, “0X”, or “0”, the standard prefixes for hexadecimal and octal integers.)
- It must not be one of the special names “ **.** “, “ **..** “, or “ **...** “.

Windows imposes a limit of 260 characters on object names. Consult your operating system documentation for more information about the maximum length of object names, keeping in mind that in a multiplatform environment, object names must conform to the rules of the most restrictive platform.

PROCESSING OF VOB SYMBOLIC LINKS

In general, **cleartool** commands do not traverse VOB symbolic links; rather, they operate on the link objects themselves. For example:

- You cannot check out a VOB symbolic link, even if it points to an element.
- A **describe** command lists information on a VOB symbolic link object, not on the object to which it points.
- A **mklabel –recurse** command walks the entire subtree of a directory element, but it does not traverse any VOB symbolic links it encounters.

UNIX COMMAND-LINE PROCESSING

In single-command mode, the **cleartool** command you enter is first processed by the UNIX shell. The shell expands file-name patterns and environment variables, and it interprets quotes and other special characters. **cleartool** processes the resulting argument list directly, without any further interpretation.

In interactive mode, **cleartool** itself interprets the command line similarly, but not identically, to the UNIX shells:

Line continuation	A **\\<NL>** sequence is replaced by a **<SPACE>** character.
Character escape	The two-character sequence **** *special-char* suppresses the special meaning of the character.
Single-quoting	Allows white-space characters and other special characters to be included in command argument. Within a single-quoted string (**’** ... **’**), a double-quote character (**"**) has no special meaning, and **\\’** is replaced by **’**.
Double-quoting	Allows white-space characters and other special characters to be included in command argument. Within a double-quoted string (**"** ... **"**), **\\"** is replaced by **"**, and **\\’** is replaced by **’**.
Commenting	Command lines that begin with a number sign (**#**) are ignored.
Wildcards	Filename patterns (including *, ?, and so on) that are not enclosed in quotes are expanded as described in the **wildcards_ccase** reference page. These patterns are also supported in config specs and, except for ellipsis (...), by the UNIX shells. (The meaning of ellipsis is slightly different in config specs; see the **config_spec** reference page.)

In interactive mode, **cleartool** does not expand environment variables and does not perform command substitution.

WINDOWS COMMAND-LINE PROCESSING

Single-Command Mode

In single-command mode, the **cleartool** command you enter is processed first by the Windows command interpreter and C run-time library, then by **cleartool**:

1. The standard command interpreter, **cmd.exe**, expands environment variables, but does no special processing for file-name patterns, quotes, or other special characters (including the asterisk (***)** and question mark (**?)** characters, which are expanded by individual commands).
2. The C run-time library does interpret quotes, stripping each pair and passing its contents through to **cleartool** as a single argument. (To pass a quote character through to **cleartool**, escape it with the backslash (\).)
3. **cleartool** processes the resulting argument list directly, without any further interpretation.

Some third-party shells perform additional command-line processing before passing the argument list through to **cleartool**. All descriptions and examples of **cleartool** command usage assume the standard **cmd.exe** interpreter.

Interactive Mode

In interactive mode, **cleartool** itself interprets the command line; it recognizes various special characters and constructs:

Line continuation (^)	A **^<NEWLINE>** sequence is replaced by a **<SPACE>** character.
Character escape (\)	The two-character sequence **** *special-char* suppresses the special meaning of the character.
Single-quoting (‘ ’)	Allows white-space characters and other special characters to be included in command argument. Within a single-quoted string (**’** ... **’**), a double-quote character (**"**) has no special meaning, and **\ ’** is replaced by **’**.
Double-quoting (" ")	Allows white-space characters and other special characters to be included in command argument. Within a double-quoted string (**"** ... **"**), **\\"** is replaced by **"**, and **\ ’** is replaced by **’**.
Commenting (#)	Command lines that begin with a number sign (**#**) are ignored.
Wildcards	Filename patterns (including *, ?, and so on) that are not enclosed in quotes are expanded as described in the **wildcards_ccase** reference page. These patterns are also supported in config specs. (The meaning of ellipsis is slightly different in config specs; see the **config_spec** reference page.)

In interactive mode, **cleartool** does not expand environment variables.

OBJECT LOCKING

ClearCase and ClearCase LT provide temporary access control through explicit locking of individual objects with the **lock** command. When an object is locked, it cannot be modified by anyone (except those explicitly excluded from the lock).

cleartool command descriptions list the locks that can prevent a command from being executed, even if you have the necessary permissions. For example, the **chtype** command lists three locks that would prevent you from changing an element type:

```
VOB, element type, pool (non-directory elements only)
```

This means that **chtype** would fail if the VOB containing the element were locked, if the element's type were locked (such as the **text_file** type), or if the storage pool containing the (nondirectory) element were locked.

EXIT STATUS

If you exit **cleartool** by entering a **quit** command in interactive mode, the exit status is 0. The exit status from single-command mode depends on whether the command succeeded (zero exit status) or generated an error message (nonzero exit status).

Note that for the **diff** command, success means finding no differences.

UNIX FILES

ccase-home-dir/**doc/man/whatis**	whatis file
ccase-home-dir/**doc/man/whatis.aux**	auxiliary whatis

SEE ALSO

comments, **fmt_ccase**, **pathnames_ccase**, **permissions**, **profile_ccase**, **version_selector**, **wildcards_ccase**, *Administrator's Guide*

clearviewupdate

Updates elements in a snapshot view

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX
Windows

SYNOPSIS

clearviewupdate [**–ws** *pname*] [**–pname** *pname...*]

DESCRIPTION

The **clearviewupdate** command provides a graphical user interface to the **update** command. For one or more loaded elements, **clearviewupdate** does the following:

- Reevaluates the config spec to select a versions of loaded elements in the VOB, and loads them if they differ from the currently loaded element versions
- Unloads the file or directory from the view if a loaded element is no longer visible (that is, a new directory version doesn't have an entry for the element). To unload a directory element, ClearCase and ClearCase LT
  - Recursively delete all loaded elements
  - Rename the directory to *directory-name***.unloaded** if necessary, thus preserving all view-private files and view-private directories.
- If the version in the snapshot view is different from the version in the VOB selected by the config spec, copies the version selected by the config spec into the view. The version in the view can be different if, for example, the selected version in the VOB is newer, or if a label is attached to the selected version in the VOB, but not to the version in the view

clearviewupdate does not apply to files or directories that are checked out to the current view.

If **clearviewupdate** cannot access a VOB (perhaps due to problems in the network), any elements from that VOB remain loaded, but are put in a special state (`rule unavailable`).

The **clearviewupdate** command accounts for the fact that VOB elements specified by your config spec may change while an update is in progress. To avoid loading an inconsistent set of element versions, **clearviewupdate** ignores versions that meet both of the following criteria:

- The version is selected by a config spec rule that specifies the **LATEST** version label.
- The version was checked in after the moment the update operation began.

clearviewupdate also accounts for the fact that the system clocks on different hosts may not be synchronized.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE VIEW TO BE UPDATED. *Default:* The current snapshot view.

–ws *pname*
Specifies the snapshot view to be updated.

SPECIFYING THE ELEMENTS TO BE UPDATED. *Default:* The elements specified by the view's config spec.

–pname *pname...*
Specifies individual files to be updated in the snapshot view.

EXAMPLES

- Invoke **clearviewupdate** without specifying any view or elements to be updated.

  cmd-context **clearviewupdate**

- Update **foo.c** in the current view.

  cmd-context **clearviewupdate –pname foo.c**

SEE ALSO

checkin, **checkout**, **config_spec**, **edcs**, **get**, **findmerge**, **ln**, **merge**, **mkdir**, **mkelem**, **mv**, **rmname**, **setcs**, **uncheckout**, **update**

clearvobadmin

Starts the VOB administration browser

APPLICABILITY

Product	Command Type
ClearCase	command

Platform
UNIX

SYNOPSIS

clearvobadmin [**–reg·ion** *region-name*]

DESCRIPTION

The **clearvobadmin** command starts the VOB Admin Browser, which you can use to perform various VOB operations—create, mount, lock, remove, and so on.

OPTIONS AND ARGUMENTS

SPECIFYING A NETWORK REGION. *Default:* Displays VOB-tags registered in the local host's network region.

–reg·ion *region-name*
Displays VOB-tags registered in ClearCase network region *region-name*. (Use **lsregion** to list region names.)

SEE ALSO

lsvob, **mkvob**

comments

Event records and comments

APPLICABILITY

Product	Command Type
ClearCase	general information
ClearCase LT	general information
Attache	general information

Platform
UNIX
Windows

DESCRIPTION

Each change to a VOB (checkin of new version, attaching of a version label, and so on) is accompanied by the creation of an event record in the VOB database. Many commands allow you to annotate the event records they create with a comment string. Commands that display event record information (**describe**, **lscheckout**, **lshistory**, **lslock**, **lspool**, **lsreplica**, and **lstype**) show the comments, as well. See the **fmt_ccase** reference page for a description of the report-writing facility built into these commands.

All commands that accept comment strings recognize the same options:

–c·omment *comment-string*

Specifies a comment for all the event records created by the command. The comment string must be a single command-line token; typically, you must quote it.

–cfi·le *comment-file-pname*

Specifies a text file whose contents are to be placed in all the event records created by this command.

NOTE: A final line-terminator in this file is included in the comment.

In Attache, the text file must be on the local host. Specifying a relative pathname for *comment-file-pname* begins from Attache's startup directory, not the working directory, so a full local pathname is recommended. For a file in DOS format, any final line-terminator is included in the comment.

–cq·uery
Prompts for one comment, to be placed in all the event records created by the command.

–cqe·ach
For each object processed by the command, prompts for a comment to be placed in the corresponding event record.

–nc·omment
(no additional comment) For each object processed by the command, creates an event record with no user-supplied comment string.

A **–cq** or **–cqe** comment string can span several lines; to end a comment, enter an EOF character at the beginning of a line, typically by pressing CTRL-D (UNIX) or CTRL-Z and RETURN (Windows), or typing a period and pressing RETURN. For example:

cmd-context **checkout main.c**

```
Checkout comments for "main.c":
```

This is my comment; the following line terminates the comment.

.

```
Checked out "main.c" from version "\main\3"
```

The **chevent** command revises the comment string in an existing event record. See the **events_ccase** reference page for a detailed discussion of event records.

Specifying Comments Interactively

cleartool can reuse a previously specified comment as the default comment. If the environment variable CLEARCASE_CMNT_PN specifies a file, that file is used as a comment cache:

- When a **cleartool** subcommand prompts for a comment, it offers the current contents of the file **$CLEARCASE_CMNT_PN** (UNIX) or **%CLEARCASE_CMNT_PN%** (Windows) as the default comment.

  Exception: If an element's **checkout** record includes a comment, that comment is the default for **checkin**, not the contents of the comment cache file.

- When a user interactively specifies a comment string to a **cleartool** subcommand, it updates the contents of CLEARCASE_CMNT_PN with the new comment. (The comment cache file is created if necessary.)

  NOTE: A comment specified noninteractively (for example, with the command **cleartool mkdir –c "test files"**), does not update the comment cache file.

The value of CLEARCASE_CMNT_PN can be any valid pathname. Using a simple file name (for example, **.ccase_cmnt**) can implement a comment cache for the current working directory; different directories then have different **.ccase_cmnt** files. Using the full pathname

$HOME/.ccase_cmnt (UNIX) or **%HOME%\.ccase_cmnt** (Windows) implements a cache of the individual user's comments, across all VOBs.

If environment variable CLEARCASE_CMNT_PN is not defined in a **cleartool** process, a default comment is supplied only in certain situations:

- Any comment specified by the user when checking out an element becomes the default comment for checking in that same element.
- When the user checks in a directory element, the default comment is a set of program-generated comments describing the directory-level changes.

CUSTOMIZING COMMENT HANDLING

Each command that accepts a comment string has a comment default, which takes effect if you enter the command without any comment option. For example, the **checkin** command's comment default is **–cqe**, so you are prompted to enter a comment for each element being checked in. The **ln** command's comment default is **–nc**: create the event record without a comment.

You can customize comment handling with a user profile file, **.clearcase_profile**, in your home directory (in Attache, on your helper host). For example, you can establish **–cqe** as the comment default for the **ln** command. See the **profile_ccase** reference page for details.

SEE ALSO

Reference pages for individual commands

config_ccase

ClearCase and ClearCase LT configuration

APPLICABILITY

Product	Command Type
ClearCase	data structure
ClearCase LT	data structure

Platform
UNIX
Windows

SYNOPSIS

Most ClearCase configuration options can be set using a cleartool command or a ClearCase GUI. Only a few require you to edit a text file (on UNIX) or a Windows registry key as described here.

DESCRIPTION—UNIX SYSTEM FILES

Files in /var/adm/atria

Anyone can read the information in this directory, but only the **root** user can modify it.

license.db

(License server host only) The license database file, which defines a set of ClearCase licenses. See the *Administrator's Guide*.

Files in /var/adm/atria/config

Anyone can read and write information in this directory to configure the local host.

alternate_hostnames

If a host has two or more network interfaces (two or more separate lines in the **/etc/hosts** file or the hosts NIS map), you must create a file with this name on that host to record its multiple entries. For example, suppose that the **/etc/hosts** file includes these entries:

```
159.0.10.16 widget sun-005 wid
159.0.16.103 widget-gtwy sun-105
```

In this case, the **alternate_hostnames** file should contain:

```
widget
widget-gtwy
```

Note that only the first hostname in each hosts entry need be included in the file. The file must list each alternative host name of a separate line. There is no commenting facility; all lines are significant. If a host does not have multiple network interfaces, this file should not exist at all.

automount_prefix

By default, **automount(1M)** mounts directories under /**tmp_mnt**. If another location is used for a host's automatic mounts (for example, you use **automount –M**, or you use an alternative auto-mount program), you must specify it in file **/var/adm/atria/automount_prefix**. For example, if your automatic mounts take place within directory **/autom**, create an automount_prefix file containing this line:

```
/autom
```

bldserver.control

Controls the way in which a host is used during parallel builds. See *Building Software*.

license_host

(Required for each ClearCase host) Contains the name of the host that acts as the ClearCase license server host for the local host.

snapshot.conf

Configuration information for semi-live VOB backup. See the **vob_snapshot** reference page for more information.

admin.conf

Controls whether this host's Clearcase properties can be changed by a user logged into a remote host.

db.conf

Configuration information for the file, **vista.tjf**, which is a journal of VOB updates. This file can grow large, especially as a result of such operations as **rmver**, **rmview**, **rmtype**, and **scrubber**. To limit the size of **vista.tjf**, create the file **db.conf** (in **/var/adm/atria/config**) and add the line

–journal_file_limit *bytes*

where *bytes* may not be less than 5000000. Setting this value too low may degrade performance.

Files in /var/adm/atria/rgy

Anyone can read these files but only the **root** user can modify them to configure the local host.

rgy_hosts.conf

Specifies this host's registry server and backup registry server hosts.

rgy_region.conf
Specifies this host's registry region.

DESCRIPTION—WINDOWS REGISTRY KEYS AND FILES

Values in Registry Key HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase*CurrentVersion*

If necessary, you may modify these values directly with a Windows Registry Editor.

Registry Key	Description
snapshot	Configuration information for semi-live VOB backup. See the **vob_snapshot** reference page for more information.
LockMgrCommandLine	Startup options for the **lockmgr**. See the *Administrator's Guide* for more information.
DomainMappingEnabled	Enables this host to use domain mapping (to support ClearCase access by users from multiple Windows NTdomains) as described in the *Administrator's Guide*.

Files in *ccase-home-dir*\var\config

Anyone can read or write to this directory.

db.conf
Configuration information for the file, **vista.tjf**, which is a journal of VOB updates. This file can grow large, especially as a result of such operations as **rmver**, **rmview**, **rmtype**, and **scrubber**. To limit the size of **vista.tjf**, create the file **db.conf** (in *ccase-home-dir*\var\config) and add the line

–journal_file_limit *bytes*

where *bytes* may not be less than 5000000. Setting this value too low may degrade performance.

SEE ALSO

Administrator's Guide, *Building Software*

config_spec

Rules for selecting versions of elements to appear in a view

APPLICABILITY

Product	Command Type
ClearCase	data structure
ClearCase LT	data structure

Platform
UNIX
Windows

SYNOPSIS

- Standard Rule:

  scope *pattern* *version-selector* [*optional-clause*]

- Create Branch Rule:

  mkbranch *branch-type-name* [**–override**]
  ...
  [**end mkbranch** [*branch-type-name*]]

- Time Rule:

  time *date-time*
  ...
  [**end time** [*date-time*]]

- File-Inclusion Rule:

  include *config-spec-pname*

- Load Rule (for snapshot views):

  load *pname* ...

DESCRIPTION

A view's config spec (configuration specification) contains an ordered set of rules for selecting versions of elements. The view's associated **view_server** process populates a view with versions by evaluating the config spec rules. For more information about **view_server**, see the *Administrator's Guide*.

In a *dynamic view*, version selection is dynamic. Each time a reference is made to a file or directory element—either by ClearCase software or by standard programs—the **view_server** uses the config spec to select a particular version of the element. (In practice, a variety of caching techniques and optimizations reduce the computational requirements.)

In a snapshot view, users invoke an **update** operation to select versions from the VOB.

UCM config specs are different from those for base ClearCase and base ClearCase LT in that their rules are generated, not user-specified—read *UCM CONFIG SPECS* before reading any other section of this reference page.

Config Spec Storage / Default Config Spec

Each view is created with a copy of the default config spec, **default_config_spec**:

element * CHECKEDOUT	*(For any element, select the checked out version, if any)*
element * /main/LATEST	*(otherwise, select most recent version on the main branch)*

Modifying this file changes the config spec that newly created views receive, but does not affect any existing view.

An individual view's config spec is stored in its view storage directory, in two forms:

- **Source format** — The user-visible version, **config_spec**, contains only the series of config spec rules.
- **Compiled format** — A modified version, **.compiled_spec**, includes accounting information. This version is created and used by the **view_server** process.

Do not modify either of these files directly; instead, use the commands listed below. Different views' config specs are independent: they may contain the same set of rules, but changing one view's config spec never affects any other view.

Commands for Maintaining Config Specs

Commands for manipulating config specs:

catcs	Lists a view's config spec.
setcs	Makes a specified file a view's config spec.
edcs	Revises the current config spec of a view.
update –add_loadrules	Adds load rules to the config spec of a snapshot view while updating the view.

HOW A CONFIG SPEC SELECTS VERSIONS

The set of elements considered for version selection is different for the two kinds of views:

- In a dynamic view, all elements in VOBs mounted on the current host are considered for version selection.
- In a snapshot view:
  - If you are updating a loaded element, the behavior is the same as in a dynamic view and the selected version is loaded into the view.
  - If you are not updating and the element is loaded, the selection from the last **update** is used.
  - If the element isn't loaded at all, the behavior is the same as in a dynamic view.

For each element, the following procedure determines which version, if any, is in the view.

1. The view's associated **view_server** process tries to find a version of the element that matches the first rule in the config spec:
   - If such a version exists, that version is in the view.
   - If multiple versions match the rule, an error occurs, and no version of the element is in the view. ClearCase and ClearCase LT commands that access the element print errors like this one:

     cleartool: Error: Trouble looking up element "ht.c" in directory ".".

     Standard commands that access the element print errors like this one:

     The request could not be performed because of an I/O device error.
   - If no version matches the first rule, the search continues.
2. If no matching version was found for the first rule, the **view_server** tries to find a version that matches the second rule.
3. The **view_server** continues in this way until it finds a match or until it reaches the last rule.

Order Is Important

Because the rules in a config spec are processed in order, varying the order may affect version selection. For example, suppose this rule appears near the beginning of a config spec:

```
element * /main/LATEST
```

Any subsequent rules in the config spec will never be used, because the rule will always provide a match; every element has a most-recent version on its **main** branch.

NOTE: The order in which the load rules for a snapshot view are specified is not important.

CHECKEDOUT Rule for Snapshot Views

The config spec for a snapshot view must contain **element * CHECKEDOUT** as the first element rule.

Failure to Select Any Version

If no version of an element matches any rule in the config spec:

- In a dynamic view:
  - The element's data is not accessible through the view. The operating system listing command and other standard programs print a `not found` error when attempting to access the element.
  - The ClearCase/ClearCase LT **ls** command lists the element with a `[no version selected]` annotation. You can specify the element in commands that access the VOB database only, such as **describe**, **lsvtree**, and **mklabel**.
- In a snapshot view, the element will not be loaded.

View-Private Files

A view's config spec has no effect on the private objects in a view, such as view-private files, links, directories; or, in the case of a dynamic view, derived objects. View-private objects are always accessible.

Exception: (Dynamic views only) If a config spec lacks a **CHECKEDOUT** rule, the view-private file that is a file element's checked-out version is not visible. See *Special Version Selectors* below.

OVERALL SYNTAX GUIDELINES

Each config spec rule must be contained within a single physical text line; you cannot use a backslash (UNIX), caret (Windows), or other line continuation character to continue a rule onto the next line. Multiple rules can be placed on a single line, separated by semicolon (**;**) characters.

Lines that begin with a number sign (**#**) are comments.

Extra white space (SPACE, TAB, vertical-tab, and form-feed) characters are ignored, except within the version selector. If a version selector includes white space, enclose it in single quotes.

If a load rule specifies a file or directory name that includes one or more SPACE characters, you must enclose the entire pathname in either single-quotes (') or double quotes (").

In general, VOBs, views, and the ClearCase and ClearCase LT tools that access them are *case-sensitive*. Therefore, config spec rules must use case-correct pathnames.

You can use slashes (**/**) or backslashes (****) as pathname separators in pathname patterns and version selectors unless you are sharing the config spec between UNIX and Windows hosts. In that case, you must use slashes. (See *SHARING CONFIG SPECS BETWEEN UNIX AND WINDOWS HOSTS*.)

SHARING CONFIG SPECS BETWEEN UNIX AND WINDOWS HOSTS

Windows and UNIX clients can share config specs, which are portable between the two operating systems. That is, clients on both systems, using views whose storage directories reside on either kind of host, can set and edit the same set of config specs. However, Windows and UNIX network regions often use different VOB-tags to register the same VOBs. Only single-component VOB-tag names, like **\src2vob**, are permitted on Windows clients; multiple-component VOB-tags, like **/vobs/src/proj1**, are common on UNIX. When the VOB-tags diverge between regions, config spec element rules that use full pathnames (which include VOB-tags) are resolvable (at config spec compile time) only by hosts in the applicable network region. This implies a general restriction regarding shared config specs: a given config spec must be compiled only by hosts on one operating system or the other—the operating system for which full pathnames in element rules make sense. That is, a config spec with full pathnames can be shared across network regions, even when VOB-tags disagree, but it must be compiled in the right place.

This restriction does not apply if any of the following are true:

- The config spec's element rules use relative pathnames only, which do not include VOB-tags.
- Shared VOBs are registered with identical, single-component VOB-tags in both Windows and UNIX network regions. (The VOB-tags **\r3vob** and **/r3vob** are logically identical, differing only in their leading slash characters.)
- The config spec does not include any load rules or element rules.

Config Spec Compilation

An in-use config spec exists in both text file and compiled formats (both of which are visible in the view's storage directory). A config spec in its compiled form is portable. The restriction is that full VOB pathnames in element rules must be resolvable at compile time. A config spec is compiled if a client executes either of these **cleartool** commands: **edcs** or **setcs –current**. Therefore, if a client on the "wrong" operating system recompiles a config spec with one of these commands, the config spec becomes unusable by any client using that view. If this happens, simply recompile the config spec on the "right" operating system.

A sample element rule that could be problematic:

```
element /vob_p2/src/*    /main/rel2/LATEST
```

If the VOB is registered with VOB-tag **\vob_p2** on a Windows network region, but with VOB-tag **/vobs/vob_p2** on a UNIX network region, only Windows clients can compile the config spec.

Pathname Separators

When writing config specs to be shared by Windows and UNIX clients, use the slash (**/**), not the backslash (\), as the pathname separator in pathname patterns and version selectors. ClearCase and ClearCase LT on Windows can parse either separator in pathnames; ClearCase and ClearCase LT on UNIX recognizes **/** only.

STANDARD RULES

A standard version-selection rule takes this form:
scope *pattern* *version-selector* [*optional-clause*]

The following subsections describe these components.

Scope

The scope specifies that the rule applies to all elements, or restricts the rule to a particular type of element.

element
The rule applies to all elements.

element –file
The rule applies to *file* elements only. This includes any element created with a **mkelem** command that omits **–eltype directory** (or a user-defined element type derived from *directory*).

element –directory
The rule applies to *directory* elements only. This includes any element created with **mkdir** or **mkelem –eltype directory** (or a user-defined element type derived from *directory*).

element –eltype *element-type*
The rule applies only to elements of the specified element type (predefined or user-defined). This mechanism is not hierarchical: if element type **aaa** is a supertype of element type **bbb**, the scope **element –eltype aaa** does not include elements whose type is **bbb**. To specify multiple element types, you must use multiple rules:

element –eltype aaa * RLS_1.2
element –eltype bbb * RLS_1.2

Selecting Versions of VOB Symbolic Links. There is no VOB symbolic link scope. A VOB symbolic link is cataloged (listed) in one or more versions of a directory element. The link appears in a view if both of these conditions are true:

- One of those directory versions is selected by the view's config spec.
- The config spec includes any element rule, even a **–none** rule.

Pattern

A pathname pattern, which can include any ClearCase/ClearCase LT wildcard (see the **wildcards_ccase** reference page for a complete list). For example:

*

Matches all element pathnames; does not match recursively.

***.c**

Matches all element pathnames with a **.c** extension.

src/util.c

Matches any element named **util.c** that resides in any directory named **src**.

/vobs/project/include/util.h

Matches one particular element.

src/.../util.c

Matches any element named **util.c** that resides anywhere within the subtree of a directory named **src** (including in **src** itself).

src/.../*.[ch]

Matches all elements with **.c** and **.h** extensions located in or below any directory named **src**.

src/...

Matches the entire directory tree (file elements and directory elements) starting at any directory named **src**.

NOTE: In non-config-spec contexts, the **...** pattern matches directory names only.

Restrictions:

- A view-extended pathname pattern is not valid.
- A relative pathname pattern must start below the VOB-tag (VOB mount point, VOB root directory). For example, if the VOB-tag is **/vobs/project**, **project/include/utility.h** is not a valid pattern.
- A full pathname pattern must specify a location at or beneath a valid VOB-tag. For example, if the VOB-tag is **/vobs/project**, then **/vobs/project/...** and **/vobs/project/include/...** are both valid.

  The **setcs** or **edcs** command fails if it encounters an invalid location in any config spec rule:

  ```
  cleartool: Error: No registered VOB tag in path: "..."
  ```
- VOB symbolic links are not valid in pathname patterns.
- On Windows systems, patterns can be specified using either backslashes (\) or slashes (/).

Version Selector

You can use a version label, version-ID, or any other standard version selector. See the **version_selector** reference page for a complete list. Some examples follow:

/main/4
Version 4 on an element's **main** branch.

REL2
The version to which version label **REL2** has been attached. An error occurs if more than one version of an element has this label.

.../mybranch/LATEST
The most recent version on a branch named **mybranch**; this branch can occur anywhere in the element's version tree.

/main/REL2
The version on the **main** branch to which version label **REL2** has been attached.

{created_since(yesterday)}
The version that has been created since yesterday. An error occurs if more than one version satisfies this query. Because all queries are evaluated at run time, the value **yesterday** is always interpreted relative to the day that the query is executed.

{QA_Level>3}
The version to which attribute **QA_Level** has been attached with a value greater than 3. An error occurs if more than one version satisfies this query.

.../mybranch/{QA_Level>3}
The most recent version on a branch named **mybranch** satisfying the attribute query.

Standard version selectors cannot select checked-out versions in a config spec rule. (They can in other contexts, such as the **find** command.) Instead, you must use the special version selector, **CHECKEDOUT**, described below.

Special Version Selectors. The following special version selectors are valid only in a config spec rule, not in any other version-selection context:

CHECKEDOUT
Matches the checked-out version of an element, if this view has a pending checkout. It doesn't matter where (on which branch of the element) the checkout occurred; there is no possibility of ambiguity, because only one version of an element can be checked out to a particular view.

This special version selector actually matches the checked-out version object in the VOB database, which is created by the **checkout** command.

For file elements, standard commands access the view-private file created by **checkout** at the same pathname as the element.

–config *do-pname* [**–select** *do-leaf-pattern*] [**–ci**]
This special version selector replicates the configuration of versions used in a particular

clearmake build. It selects versions listed in one or more configuration records associated with a particular derived object: the same set of versions that would be listed by a **catcr –flat** command. See the **catcr** reference page for explanations of the specifications that follow the **–config** keyword.

When you set or edit a config spec, the **view_server** resolves the *do-pname* with respect to the view's preexisting config spec, not on the basis of any preceding rules in the config spec being evaluated.

If the configuration records list several versions of the same element, the most recent version is selected to appear in the view. In such cases, a warning message is displayed when the config spec is set.

–none

Generates an ENOENT (`No such file or directory`) error when a standard UNIX operating system program references the element. For dynamic views:

- No error occurs when an operating system listing command lists the element's entire parent directory; the element is included in such a listing. This also applies to other **readdir** situations, such as expansion of wildcard characters and **emacs** file name completion.
- An error occurs when an operating system listing command names the element explicitly (perhaps after wildcard expansion), or whenever the name is processed with (UNIX) **stat(2)**: in an **ls –F** command, when the entire directory is listed with **ls –l**, and so on.
- The **cleartool ls** command always lists the element, annotating it with `no version selected`.
- In ClearCase and ClearCase LT commands, the element's standard pathname refers to the element itself. (**–none** suppresses the transparency mechanism—translation of an element's standard pathname into a reference to a particular version.)

–error

Like **–none**, except that the annotation generated by the **cleartool ls** command is `error on reference`.

Optional Clause

Some config spec rules can include an additional clause, which modifies the rule's meaning.

–time *date-time*

Modifies the meaning of the special version label **LATEST**: the rule selects from a branch the last version that was created before a particular time. The *date-time* argument is specified in one of the standard formats:

*date***.***time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit *date*, the default is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want to resolve the time to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, Greenwich Mean Time (GMT) is used. (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

The creation times of the versions on the branch are looked up in their `create version` event records. (No error occurs if you use a **–time** clause in a rule that does not involve the version label **LATEST**; the clause has no effect.)

The **–time** clause in a particular rule overrides any general time rule currently in effect. (See *TIME RULES* on page 261.)

NOTE: A **–time** clause must precede any other optional clauses and may not include any query language constructs.

Examples:

/main/LATEST –time 10–Jul.19:00	Most recent version on main branch, as of 7 P.M. on July 10.
.../bugfix/LATEST –time yesterday	Most recent version on a branch named **bugfix** (which can be at any branching level), as of the beginning of yesterday (12 A.M.).
/main/bugfix/LATEST –time Wed.12:00	Most recent version on subbranch **bugfix** of the **main** branch, as of noon on the most recent Wednesday.
–time 5–Dec.13:00	December 5, at 1 P.M.
–time 11:23:00	Today, at 11:23 A.M.
–time 12–jun–99	June 12, 1999, at 00:00 A.M.
–time now	Today, at this moment.
–time 9-Aug.10:00UTC	August 9, at 10 A.M. GMT.

The *date/time* specification is evaluated when you set or edit the config spec, and whenever the **view_server** process is started (for example, with **startview** or **setview** (dynamic views only)). Thus, the meaning of a relative specification, such as **today**, may change over time. However, the *date/time* is not evaluated at run time. Therefore if you last performed one of the commands listed above four days ago, the meaning of a relative specification, such as **today**, has the value of the date four days ago, not the value of the date today.

–nocheckout
Disables checkouts of elements selected by the rule.

–mkbranch *branch-type-name*
Implements the auto-make-branch facility. When a version selected by this rule is checked out:

- A branch of type *branch-type-name* is created at that version.
- Version 0 on the new branch is checked out, instead of the version that was originally selected.

(This is a slight oversimplification. See *Multiple-Level Auto-Make-Branch* on page 259.) A **mkelem** command invokes the auto-make-branch facility if the config spec includes a **/main/LATEST** rule with a **–mkbranch** clause.

Restrictions: You cannot use **–mkbranch** in combination with **–none** or **–error**.

Multiple-Level Auto-Make-Branch

A config spec can include a cascade of auto-make-branch rules, causing **checkout** to create multiple branching levels at once. **checkout** keeps performing auto-make-branch until version 0 on the newly created branch is not selected by a rule with a **–mkbranch** clause; then, it checks out that version. For example:

```
(1)   element * CHECKEDOUT
(2)   element * .../br2/LATEST
(3)   element * .../br1/LATEST -mkbranch br2
(4)   element * MYLABEL -mkbranch br1
(5)   element * /main/LATEST
```

```
(1)   element * CHECKEDOUT
(2)   element * ...\br2\LATEST
(3)   element * ...\br1\LATEST -mkbranch br2
(4)   element * MYLABEL -mkbranch br1
(5)   element * \main\LATEST
```

If you check out an element in a view that currently selects the version labeled **MYLABEL**:

1. A branch of type **br1** is created at the **MYLABEL** version (Rule (4)).
2. Rule (3) now selects the newly created version **.../br1/0**, so a branch of type **br2** is created at that version.
3. Version **.../br1/br2/0** is checked out. The checked-out version has the same contents as the **MYLABEL** version, and is selected by Rule (1). When you edit and check in a new version, **.../br1/br2/1**, the view will select it with Rule (2).

CREATE BRANCH RULES

A create branch rule takes the following form:

mkbranch *branch-type-name* [**–override**]
<*config spec lines*>
[**end mkbranch** [*branch-type-name*]]

This rule is similar to the **–mkbranch** clause; use it when you want to add a **–mkbranch** clause to many lines in a complex config spec.

mkbranch *branch-type-name* [**–override**]

Attaches an implicit **–mkbranch** *branch-type-name* clause to all element rules between **mkbranch** and **end mkbranch** (or the end of the file) that do not have a **–mkbranch** clause or include the **CHECKEDOUT** version selector.

Specifying **–override** will override any explicit **–mkbranch** clauses or **mkbranch** rules within the scope and replace them with **–mkbranch** *branch-type-name*. Use **–override** if you do not want multilevel branch creation.

end mkbranch [*branch-type-name*]

Ends the **mkbranch** *branch-type-name* rule. If **end mkbranch** is omitted, the rule is ended at the end of the config spec. The *branch-type-name* argument is optional, but if you include it, it must match the branch type specified with the **mkbranch** rule.

mkbranch and **end mkbranch** rules may be nested. For example:

```
element * .../branch2/LATEST
mkbranch branch2

element * .../branch1/LATEST
mkbranch branch1

element * /main/LATEST

end mkbranch branch1
end mkbranch branch2
```

Checking out **foo.c** creates **foo.c@@/main/branch1/branch2/CHECKEDOUT**. This is a multiple-level **mkbranch**.

TIME RULES

A time rule takes this form:

time *date-time*
[**end time** [*date-time*]]

It is analogous to the **–time** clause. A time rule modifies the meaning of the special version label **LATEST** in subsequent rules, with the following exceptions:

- An optional **–time** clause in a particular rule overrides any general time rule currently in effect.
- A subsequent time rule cancels and replaces an earlier one.

Use **end time** to limit the effect of a time rule to a certain range. The *date-time* argument is optional with **end time**, but if you include it, it must match the *date-time* argument specified with the **time** rule.

The *date-time* specification is evaluated when you set or edit the config spec, and whenever the **view_server** process is started (for example, with **startview** or **setview** (dynamic views only)). Thus, the meaning of a relative specification, such as **today**, may change over time. However, the *date-time* is not evaluated at run time. So if you last performed one of the commands listed above four days ago, the meaning of a relative specification, such as **today**, has the value of the date four days ago, not the value of the date today.

Time rules may be nested. They may not include any query language constructs.

FILE-INCLUSION RULES

A file-inclusion rule takes this form:

include *config-spec-pname*

The argument specifies a text file containing one or more config spec rules (possibly other **include** rules). Include files are reread on each execution of **setcs** and **edcs**. A file-inclusion rule must be the last rule in a line. For example,

include *config-spec-pname*

and

time *date-time***; include** *config-spec-pname*

are both valid.

LOAD RULES

Load rules define which elements are loaded (copied) into a snapshot view (by contrast, element rules define which version of an element is selected). A load rule takes this form:

load *pname ...*

The argument specifies one or more file or directory elements. Naming a directory element implies the directory and all elements below the directory. Naming a file element specifies that element only. Wildcarding is not supported: you must explicitly specify all elements to be loaded.

More than one load rule can appear in a config spec; you must have at least one to see any files in a snapshot view. (Load rules in the config spec of a dynamic view are ignored.)

Load rules can be positioned anywhere in a config spec, and their order is irrelevant.

An element can be selected by more than one load rule without causing an error.

Links are treated as follows:

- On UNIX systems, hard VOB links are followed; symbolic links are copy-created.
- On Windows systems VOB links (both symbolic links and hard links) are followed.

In all cases, the link target is copied into the snapshot view at the location in which the link appeared.

UCM CONFIG SPECS

UCM config specs are unlike config specs for base ClearCase and base ClearCase LT in that they are generated by **mkstream** and regenerated from time to time by **chstream** and **rebase**. You may edit UCM config specs only as follows:

- To add custom element-selection rules.
- To add custom load rules for snapshot views.

Only custom rules that are correctly delimited are preserved when a UCM config spec is regenerated.

NOTE: Never use UCM-generated rules in config specs for base ClearCase or base ClearCase LT.

Custom Element-Selection Rules

Add custom element-selection rules only between the custom element delimiters, as follows:

```
#UCMCustomElemBegin - DO NOT REMOVE - ADD CUSTOM ELEMENT RULES AFTER THIS LINE
rule
.
.
.
#UCMCustomElemEnd - DO NOT REMOVE - END CUSTOM ELEMENT RULES
```

The typical use for custom element selection is to add an **include** rule that enables the UCM view to see the contents of base ClearCase or base ClearCase LT VOBs. See *FILE-INCLUSION RULES*.

Custom Load Rules

Add custom load rules after the custom load delimiter, as follows:

```
#UCMCustomLoadBegin - DO NOT REMOVE - ADD CUSTOM LOAD RULES AFTER THIS LINE
rule
.
.
.
```

See *LOAD RULES* for more information.

EXAMPLES

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Include a standard set of rules to be used by every user on a particular project.

  include /proj/cspecs/v1_bugfix_rules

- Modify the meaning of “most recent” to mean “as of 7 P.M. on July 10.”

  ```
  time 10-Jul.19:00
      element \atria\lib\* ...\new\LATEST
      element * \main\LATEST
  end time
  ```

- Select version 3 on the **main** branch of a particular header file.

  element /usr/project/include/utility.h /main/3

- Select the most recent version on the **main** branch for all elements with a **.c** file-name extension.

  element *.c \main\LATEST

- Select the most recent version on the **bugfix** branch.

  element * .../bugfix/LATEST

- Select versions of elements from a particular development branch, or with a related label.

  ```
  element * CHECKEDOUT           (If no checked-out version, select latest
  element * ...\maint\LATEST     version on the 'maint' branch, which
                                 may or may not be a direct subbranch
                                 of main)
  element * BL2.6                (Else, select version labeled 'BL2.6'
                                 from any branch)
  ```

element * \main\LATEST

- Select versions of C language source files (**.c** file extension) based on the value of an attribute. A config spec such as this may be used by a developer to select versions of files for which he is responsible.

element * CHECKEDOUT	
element –file *.c /main/{RESPONSIBLE=="jpb"}	*(For any '.c' file, select latest version on main branch for which 'jpb' is responsible)*
element –file /project/utils/.../*.c /main/BL2.6	*(Else, select version labeled BL2.6 on main branch from /project/utils directory, or any of its subdirectories)*
element * /main/LATEST	

- Use the **–mkbranch** qualifier to create a new **BL3** branch automatically. Create the branch off the version labeled **BL2.6**, or the latest version on the **main** branch if no version is labeled **BL2.6**.

element * CHECKEDOUT	*(If no version is checked out, select*
element * ...\bl3_bugs\LATEST	*latest version on 'bl3_bugs' branch)*
element -file * BL2.6 –mkbranch bl3_bugs	*(Else, select version labeled 'BL2.6' and create 'bl3_bugs' branch on checkout)*
element -file * \main\LATEST –mkbranch bl3_bugs	*(Else, select latest version on main branch and create new branch on checkout)*

- Same as above, but use a **mkbranch** rule.

element * CHECKEDOUT
element * .../bl3_bugs/LATEST
mkbranch bl3_bugs
element -file * BL2.6
element -file * /main/LATEST
end mkbranch bl3_bugs

- Select the version labeled **REL3** for all elements, preventing any checkouts to this view:

element * REL3 –nocheckout

- Select the most recent version on the **bug_fix_v1.1.1** branch, making sure that this branch is a subbranch of **bug_fix_v1.1**, which is itself a subbranch of **bug_fix_v1**.

element * CHECKEDOUT
element * bug_fix_v1.1.1\LATEST
element * ...\bug_fix_v1.1\LATEST –mkbranch bug_fix_v1.1.1
element * ...\bug_fix_v1\LATEST –mkbranch bug_fix_v1.1
element * \main\LATEST –mkbranch bug_fix_v1

When a user checks out an element for which none of these branches yet exists, a cascade of auto-make-branch activity takes place:

`Z:\myvob>` **cleartool checkout -nc .**

```
Created branch "bug_fix_v1" from "." version "\main\0".
Created branch "bug_fix_v1.1" from "." version "\main\bug_fix_v1\0".
Created branch "bug_fix_v1.1.1" from "." version
"\main\bug_fix_v1\bug_fix_v1.1\0".
Checked out "." from version
"\main\bug_fix_v1\bug_fix_v1.1\bug_fix_v1.1.1\0".
```

- Modify the previous config spec to create branch **bug_fix_v2** off an existing branch rather than creating multiple subbranches.

  element * CHECKEDOUT
  mkbranch bug_fix_v2 –override
  element * .../bug_fix_v1.1.1/LATEST
  element * .../bug_fix_v1.1/LATEST –mkbranch bug_fix_v1.1.1
  element * .../bug_fix_v1/LATEST –mkbranch bug_fix_v1.1
  element * /main/LATEST –mkbranch bug_fix_v1
  end mkbranch bug_fix_v2

- For a snapshot view, select the most recent version on the **main** branch. Use load rules to select in the **applets** VOB all elements below the **\cmdlg** directory and one specific element in the **\testdlg** directory.

  element * CHECKEDOUT
  element *... \main\LATEST
  load \applets\cmdlg
  load \applets\testdlg\opendlg.h

UNIX FILES

/usr/atria/default_config_spec
*view-storage-directory***/config_spec**
*view-storage-directory***/.compiled_spec**

WINDOWS FILES

*ccase-home-dir***\default_config_spec**
*view-storage-directory***\config_spec**
*view-storage-directory***\.compiled_spec**

SEE ALSO

catcs, **checkout**, **checkin**, **edcs**, **ls**, **mkbranch**, **setcs**, **setview**, **version_selector**, **csh(1)**, *Administrator's Guide*

cptype

Makes a copy of an existing type object.

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

cptype [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery**
| **–cqe·ach** | **–nc·omment**] [**–rep·lace**]
existing-type-selector new-type-selector

DESCRIPTION

The **cptype** command creates a new type object (for example, a label type or attribute type) that is a copy of an existing type object. The existing and new objects can be in the same VOB, or in different VOBs. The copy can have the same name as the original only if you are making the copy in a different VOB.

The original and copy do not retain any connection after you execute **cptype**. They are merely two objects with the same properties, and perhaps even the same name.

EXCEPTION: Global types are handled differently. For more information, see the *Administrator's Guide*.

ClearCase—Ordinary Types and AdminVOB Hierarchies

When you copy an ordinary type object to a VOB that is part of an **AdminVOB** hierarchy, ClearCase determines whether the new type name is already defined as a global type in the administrative VOB of the copy's destination VOB. If it is, **cptype** fails with an explanatory message. When this is the case, you can do one of the following things:

- Specify a different name for the copy

- Try using the original type object in the VOB where you wanted to make the copy

Handling of Supertypes

The **cptype** command recursively copies the supertypes of the original type to the copy's destination VOB.

Firing of mktype Triggers

When you copy a type, the **cptype** command fires any **mktype** triggers attached to the destination VOB.

RESTRICTIONS

Identities: This command has the same restrictions as the type-object creation commands (**mk***object***type**).

Locks: An error occurs if one or more of these objects are locked: VOB containing the new object. With **–replace**, an error occurs if the type object being replaced is locked.

Mastership: (Replicated VOBs) The replica containing the original type must master that type.

OPTIONS AND ARGUMENTS

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, preserving the comment associated with the original type. Any new comment you specify is appended to the preserved comment. (The file **.clearcase_profile** defines default commenting behavior; you can also edit comments using **chevent**.)

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
Overrides the default with the option you specify. See the **comments** reference page.

REPLACING AN EXISTING TYPE OBJECT. *Default:* An error occurs if *new-type-selector* already exists.

–rep·lace
Replaces the definition of *new-type-selector* with the definition of *existing-type-selector*. An error occurs if *existing-type-selector* and *new-type-selector* have the same definition. If you specify **–c** or **–cfile** with **–replace**, the comment appears in the event record for the modification (displayed with **lshistory –minor**); it does not replace the object's creation comment (displayed with **describe**). Use **chevent** to change a creation comment.

SPECIFYING THE EXISTING AND NEW TYPE OBJECTS. *Default:* None.

existing-type-selector
new-type-selector
The name of an existing type object, and a name for the new copy. Specify *existing-type-selector* in the form *type-kind***:***type-name*[*@vob-selector*] and *new-type-selector* in the form [*type-kind*]**:***type-name*[*@vob-selector*]

type-kind	One of	
	attype	Attribute type
	brtype	Branch type
	eltype	Element type
	hltype	Hyperlink type
	lbtype	Label type
	trtype	Trigger type
type-name	Name of the type object See the **cleartool** reference page for rules about composing names.	
vob-selector	VOB specifier Specify *vob-selector* in the form [**vob:**]*pname-in-vob*	
	pname-in-vob	Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, <u>*cmd-context*</u> represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, <u>*cmd-context*</u> represents the interactive **cleartool** prompt. In Attache, <u>*cmd-context*</u> represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Make a copy of a label type object, in the same VOB.

  <u>*cmd-context*</u> **cptype lbtype:RE1.3 REL1.4**

- Copy a branch type object, to create an object with the same name in a different VOB.

cmd-context **cptype -c "copied from source VOB" ^
brtype:proj_test3.7@\projsrc proj_test3.7@\projtest**

- Replace the definition of the trigger type **label_it** with the description of **label_it** from another VOB.

  cmd-context **cptype –replace trtype:label_it@/vobs/stage label_it@/vobs/dev**

SEE ALSO

describe, **lstype**, **mkhltype**, **profile_ccase**, *Administrator's Guide*

credmap

Display username/UID and group name/GID on Windows NT and UNIX

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
Windows

SYNOPSIS

*ccase-home-dir***etc****utils****credmap** *remote-host*

DESCRIPTION

Use the **credmap** utility to check that your Windows NT user and group assignments match those on the UNIX side. **credmap** returns the following values:

- The Windows NT username and the primary group name used by ClearCase/ClearCase LT
- The UNIX user ID and primary group ID

A mismatch returns a UNIX user ID or primary group ID of `–2`; in this case, you should:

1. Run the **id** command on the UNIX host to determine the correct UNIX user and group IDs.
2. Make the proper adjustment to either the UNIX user account information or the Windows NT security account information.

NOTE: Mismatches can also occur in the group ID list; to prevent permission problems, the group list for Windows NT users should include all groups known to the UNIX VOBs.

See the *Administrator's Guide* for more information.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

remote-host

UNIX host running the NFS daemon or authentication server.

EXAMPLES

- Check Windows NT user **anne**'s user and group IDs against their counterparts on UNIX host **saturn**:

```
c:\Program Files\Rational\ClearCase> \etc\utils\credmap saturn
Identity on local Windows NT system:
  User: anne (0x1003f2)
  Primary group: user (0x1003ff)
  Groups:
    Administrators (0x20220)
    Domain Users (0x100201)

Identity on host "saturn":
  User ID: 1149 (0x47d)
  Primary group ID: 20 (0x14)
  Group ID list:
    -2 (0xfffffffe)
```

Run the **id** command on a UNIX system to verify that the UNIX User ID and Primary group ID values correspond with UNIX user **anne** and group **user**:

```
% id
uid=1149(anne) gid=20(user)
```

SEE ALSO

creds, *Administrator's Guide*

creds

Display user and group information

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
Windows

SYNOPSIS

*ccase-home-dir***etc****utils****creds** *options*

DESCRIPTION

The **creds** utility displays user and group information for the currently logged-in user for the local Windows NT host. The value of the user's `Primary Group` is affected by the **CLEARCASE_PRIMARY_GROUP** environment variable. If this EV is set, **creds** displays its value, which is used only for ClearCase/ClearCase LT processing. If this EV is not set, **creds** displays the user's Windows NT primary group.

creds also displays the user's ClearCase/ClearCase LT privilege status:

`You have ClearCase administrative privileges`	You are a member of the ClearCase group.
`You do not have ClearCase administrative privileges`	You are not a member of the ClearCase group.

You do not have to be in the ClearCase group to use ClearCase. However, members of this group have more privileges to create and modify ClearCase objects. See the **permissions** reference page for more information.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

By default, **creds** displays the credentials for the current user. Lookup is done on the local machine.

Some ClearCase and ClearCase LT operations cannot evaluate group membership information for more than 32 groups per user. If the user is a member of more than 32 groups, **creds** lists only the 32 groups used by all ClearCase and ClearCase LT operations.

-w

Lists all groups of which the user is a member. If the user is a member of more than 32 groups, this list will include some groups that some ClearCase and ClearCase LT operations do not consider when determining the user's identity.

–h

Show options for this command.

EXAMPLES

- Display credentials for the current user.

```
c:\Program Files\Rational\ClearCase> \etc\utils\creds
Login name:    jeri
UID:           0x10040a
Primary group: user (0x100432)
Groups:
user (0x100432)
doc (0x303eb)
Administrators (0x20220)
Users (0x20221)
Domain Users (0x100201)

Current user is ClearCase privileged
```

SEE ALSO

credmap, **permissions**, *Administrator's Guide*

CtCmd

Perl extension for ClearCase and ClearCase LT

APPLICABILITY

Product	Command Type
ClearCase	Perl module
ClearCase LT	Perl module

Platform
UNIX
Windows

SYNOPSIS

use CtCmd;

.

.

.

DESCRIPTION

CtCmd::exec() takes either a string or an array as an input argument, and returns a three-element Perl array as output, as follows:

- The first element is a status bit containing **0** on success, **1** on failure.
- The second element is a scalar string, if any, corresponding to **stdout**.
- The third element contains any error message corresponding to output on **stderr**.

If the first element has any content, the second element is empty, and vice versa.

CtCmd–>new() may be used to create an instance of **CtCmd**. There are three possible construction variables:

CtCmd–>new(outfunc=>0,errfunc=>0,debug=>1);

Setting **outfunc** or **errfunc** to **0** disables the standard output and error handling functions. Output goes to **stdout** or **stderr**. The size of the output array is reduced correspondingly.

CtCmd

Cleartool Command Support

The **CtCmd** module supports all **cleartool** commands, but not the **cleartool** options **–ver** and **–verall**. It also supports all format strings for command output (see **fmt_ccase**).

Exit Status

Upon the return of

class method exec: ($a,$b,$c)=CtCmd::exec(*cleartool_command***);**

or

instance method exec: $x=CtCmd–new; ($a,$b,$c)=$x–>exec(*cleartool_command***);**

the first returned element, **$a**, contains the status of *cleartool_command* (**0** on success, **1** on failure).

Upon the return of

instance method exec: $x=CtCmd–new; $a=$x–>exec(*cleartool_command***);**

that is, where the return value is a scalar, the status is available as

$status=$x–>status();

and scalar **$a** contains output corresponding to **stdout** or **stderr**, as appropriate.

Use of the –e Option inCommand-Line Invocation of CtCmd::exec()

The Perl interpreter, when invoked with its script argument from the command line with the **–e** option, may interpret any unquoted explicit list element of the form **/–[a–zA–Z]/** (hyphen followed by single uppercase or lowercase alphabetical character) as an argument to the Perl interpreter rather than as part of the script argument.

For example, the following succeeds:

```
$perl -e '$i="A";
use CtCmd;
@aa=CtCmd::exec("lsproj","-s",-invob,"/var/tmp/bills_1_pvob");
for(@aa){s/\n//g;print $i++,"",$_,"\n"};'

A   0

B   Ranhattan       Yanhattan       Fanhattan

C
```

but the following fails:

```
$perl -e '$i="A";
use CtCmd;
@aa=CtCmd::exec("lsproj",-s,-invob,"/var/tmp/bills_1_pvob");
for(@aa){s/\n//g;print $i++,"",$_,"\n"};'
```

```
A   1
B
C   noname: Error: project not found: "". noname: Error: project not
    found: "-invob". noname: Error: t not found: "/var/tmp/bills_1_pvob"
```

This condition only affects command-line invocation of **CtCmd::exec()**. Programmatic invocation of **CtCmd::exec()** is not affected.

RESTRICTIONS

No restrictions apply.

EXAMPLE

```
use CtCmd;

@aa=CtCmd::exec("ls /vobs/public");

$status = $aa[0];

$stdout = $aa[1];

@aa=CtCmd::exec("ls /nowheresville");

$error = $aa[2];

die $error if $status;

my $inst = CtCmd->new();

my $pvob="\@/var/tmp/bills_1_pvob";

my ($status,$stream)=$inst->exec(des,-fmt,"%Ln,dbid:2390".$pvob);

($status,my $stream, my $err)=
$inst->exec(deliver,-to,wjs_integ_manhattan_y,-stream,$stream.$pvob,-complete,-force);

die $err if $inst->status();
```

SEE ALSO

cleartool, **fmt_ccase**, **perl(1)**

deliver

Delivers changes in a source stream to the target stream within or across projects

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- Deliver changes in the source stream using the graphical user interface:

  deliver –g·raphical [**–str·eam** *stream-selector*] [**–to** *target-view-tag*] [**–tar·get** *stream-selector*]
  [**–q·uery** | **–qal·l**]

- Cancel or obtain the status of a deliver operation in progress:

  deliver { **–can·cel** | **–sta·tus** [**–l·ong**] } [**–str·eam** *stream-selector*]

- Preview a deliver operation:

  deliver –pre·view [**–s·hort** | **–l·ong**] [**–str·eam** *stream-selector*] [**–to** *target-view-tag*]
  [**–tar·get** *stream-selector*] [**–act·ivities** *activity-selector* ... | **–bas·elines** *baseline-selector* ...]

- Deliver changes in the source stream:

  deliver [**–str·eam** *stream-selector*] [**–to** *target-view-tag*] [**–tar·get** *stream-selector*]
  [**–act·ivities** *activity-selector* ... | **–bas·elines** *baseline-selector* ...] [**–com·plete**]
  [**–gm·erge** | **–ok**] [**–q·uery** | **–abo·rt** | **–qal·l**] [**–ser·ial**] [**–f·orce**]

- Resume or complete work on a deliver operation:

  deliver { **–res·ume** | **–com·plete** } [**–str·eam** *stream-selector*] [**–gm·erge** | **–ok**]
  [**–q·uery** | **–ab·ort** | **–qal·l**] [**–ser·ial**] [**–f·orce**]

DESCRIPTION

The **deliver** command lets you deliver work from a source stream to a target stream in the same or a different project. There may be several steps to delivering work:

- Previewing the changes to be delivered
- Identifying the activities, stream, or baselines you want to deliver
- Resolving merge conflicts
- Testing and building work in the target stream
- Completing a deliver operation, which checks in new versions and records other information

If a deliver operation is interrupted through a system interrupt or user action, you must explicitly resume or cancel it.

In general, it is good practice to check in all work to your source stream before beginning a deliver operation.

You may not deliver when a rebase operation is in progress.

The Integration Activity

The deliver operation creates a UCM activity called the integration activity, which records a change set for the deliver operation. The activity name is of the form *deliver.stream-name.date-stamp*. When the deliver operation begins, the integration activity becomes the current activity for the target view in use.

One-Step Deliver Operation

You can deliver your work in one step by specifying the **–complete** and **–force** options. The **–force** option suppresses prompting for user input during the deliver operation. The **–complete** option causes the deliver operation to continue to completion after the merge phase. Use this feature carefully to avoid the possibility of delivering merged files that may not compile.

Handling of Elements of Non-default Merge Types

In a deliver operation, automatic merging is the default behavior for elements, unless **user** or **never** merge types were set for the elements. For information about setting merge behavior for an element type, see **mkeltype**.

For elements of the **user** merge type, you must invoke your own tool to merge from the source stream version to the target view. The deliver operation checks out the elements, but you are responsible for merging and drawing merge arrows. Until merge arrows are drawn, the deliver operation will not proceed to the completion phase.

A deliver operation ignores versions from the source stream for elements of the **never** merge type. If all other versions are properly merged, the deliver operation will be able to proceed to the completion phase. No merging or merge arrows are required.

Setting Policies

UCM includes policies that you can set to affect the deliver operation. You can set policies at project or stream creation with **mkproject** or **mkstream**, or at a later time with **chproject** or **chstream**. For information about project and stream policies, see the reference pages for **mkproject** and **mkstream**.

Delivering Changes with MultiSite

The **deliver** command determines whether the target stream and source stream are mastered at different replicas. If they are, a remote deliver operation is put into effect. The source stream is assigned a **posted** status.

In a remote deliver operation, the source stream must be a development stream. If you want to deliver changes from an integration stream, you can do so by delivering to a development stream on the same replica, and then delivering from the development stream to the remotely mastered stream.

After the stream is in the **posted** state, the deliver operation can be continued only by someone working at the target stream's replica. Generally, this is the team's project integrator. Also, after it is posted, the deliver operation can be canceled only by a user at the replica where the target stream resides.

The option **–to** *target-view-tag* is not accepted when starting a remote deliver operation. The remote post goes to the default target stream or an alternate target stream, if specified. The **deliver –status** command reports on any remote deliver operation in progress for the specified stream. The project integrator can then cancel or continue the deliver operation, using the **–cancel** option to halt it, or the **–resume** or **–complete** options to continue it. The project integrator must specify the target view for the deliver operation.

While the remote deliver is in progress, you can create activities and perform checkins and checkouts for your source stream, but you cannot perform any of the following operations:

- Add, remove, or create baselines
- Add or remove components
- Rebase the source stream
- Post another deliver operation

RESTRICTIONS

Identities: No special identity required.

Locks: An error occurs if one or more of these objects are locked: the source stream, the UCM project VOB.

Mastership: (Replicated VOBs only) Your current replica must master the source stream.

OPTIONS AND ARGUMENTS

STARTING THE GRAPHICAL USER INTERFACE: *Default*. command line interface.

–g·raphical
Starts the graphical user interface for **deliver**.

SPECIFYING THE SOURCE AND DESTINATION FOR THE DELIVER OPERATION. *Default:* The default source is the stream attached to the current view. The default destination is the parent stream of the source stream. For an integration stream, the default destination is the default deliver target if one is set, using either a view attached to the parent stream owned by the current user, or the view used by the last default deliver operation performed on the source stream.

–stre·am *stream-selector*
Specifies a stream that is the source for the deliver operation.

stream-selector is of the form [**stream:**]*stream-name*[*@vob-selector*], where *vob-selector* specifies the stream's project VOB.

–to *target-view-tag*
Specifies a view attached to the deliver target stream in the same project or in a different project. This option is not accepted for a remote-post deliver operation.

SPECIFYING AN ALTERNATIVE DELIVER TARGET. *Default:* For development streams, the parent stream of the source stream. For integration streams, the default deliver stream set by **mkstream** or **chstream**.

–tar·get *stream-selector*
Specifies the deliver target stream in the same or a different project.

Using **–to** *target-view-tag* alone is sufficient to start a deliver operation to the target stream. If **–to** and **–target** are used together, the view must be attached to the stream. If one default view is attached to the target stream and owned by the user, using **–target** *stream-selector* alone can start a deliver operation. Otherwise, a target view must be specified.

stream-selector is of the form [**stream:**]*stream-name*[*@vob-selector*], where *vob-selector* specifies the stream's project VOB.

CANCELING A DELIVER OPERATION. *Default:* None.

–can·cel
Halts a deliver operation in progress, returning the source and destination streams to

their states before the deliver operation began. However, this option cannot undo a deliver operation after the completion phase has begun.

Also use **–cancel** when a deliver operation is interrupted (as by CTRL+C) or when it encounters an external error or condition that requires more information.

OBTAINING THE STATUS OF A DELIVER OPERATION. *Default:* None.

–sta·tus

Displays the status of a deliver operation. You are informed whether a deliver operation for the specified stream is in progress, whether the delivery is to a local stream or a remote stream, and, in the case of a remote deliver, whether the posted changes have been merged with the target stream.

PREVIEWING THE RESULTS OF A DELIVER OPERATION. *Default:* For each activity to be delivered, displays the owner and activity-selector. For each baseline to be delivered, displays the owner and baseline-selector.

–pre·view

Shows activities or baselines that would be delivered if you were to execute the deliver operation for the specified stream. These are any activities or baselines that have changed since the last deliver operation from this stream. Use **–preview** only when no deliver operation is in progress for the stream.

CONTROLLING OUTPUT VERBOSITY. *Default:* Varies according to the kind of output that the options described here modify. See the descriptions of **–status** and **–preview**.

–l·ong

As a modifier of **–status** or **–preview**, displays a list of versions that may require merging, in addition to the default information displayed by **–status** or **–preview**.

–s·hort

Modifies the **–preview** option. (This option does not modify the default **–preview** output.)

SELECTING ACTIVITIES TO DELIVER. *Default:* Delivers all activities in the stream that have changed since the last deliver operation from the stream.

–act·ivities *activity-selector, ...*

Specifies a list of activities to deliver. The list must be self-consistent: the activities specified must not depend on the inclusion of any unspecified activities. For example, activity A2 is dependent on activity A1 if both contain versions of the same element and A2 contains a later version than does A1.

Additionally, any activities that have been included in baselines but not delivered must also be delivered if changes for that component are in the specified activities. If the list of activities you specify is incomplete, the operation fails and lists the additional required activities.

activity-selector is of the form [**activity:**]*activity-name*[*@vob-selector*] where *vob-selector* specifies the activity's project VOB.

SELECTING BASELINES TO DELIVER. *Default:* Delivers all activities in the stream that have changed since the last deliver operation from the stream.

–bas·elines *baseline-selector*, ...

Specifies a list of baselines to deliver. The baselines must be created in the source stream or from the source stream's foundation. A development stream can deliver activities or baselines, but an integration stream can deliver only baselines.

RESUMING A DELIVER OPERATION. *Default:* None.

–res·ume

Resumes a deliver operation from the point at which it was suspended.

COMPLETING A DELIVER OPERATION. *Default:* None.

–com·plete

Completes a deliver operation. Verifies that changes being delivered have been merged with versions in the deliver target stream and that merge conflicts have been resolved. Checks in resulting new versions to the target stream and records that the deliver operation has been made. If merge conflicts exist, the deliver operation is suspended.

Use this option to complete a deliver operation or to resume a suspended deliver operation. To complete a deliver operation, you must specify this option; checking in merged versions to the target view alone does not complete the deliver operation.

When used for a deliver operation in progress, this option implies the **–resume** option; that is, **deliver –complete** reports any merges that are still required and attempts to resolve them.

MERGING. *Default:* Merging works as automatically as possible, prompting you to make a choice when two or more contributors differ from the base contributor. For general information, see the **findmerge** reference page.

–gm·erge

Performs a graphical merge for each element that requires it. This option does not remain in effect after a deliver operation is interrupted.

–ok

Pauses for verification on each element to be merged, allowing you to process some elements and skip others. This option does not remain in effect after a deliver operation is interrupted.

–q·uery

Turns off automated merging for nontrivial merges and prompts you for confirmation before proceeding with each change in the from-versions. Changes in the to-version are accepted unless a conflict exists. This option does not remain in effect after a deliver operation is interrupted.

–abo·rt

Cancels a merge if it is not completely automatic. This option does not remain in effect after a deliver operation is interrupted.

–qal·l

Turns off all automated merging. Prompts you for confirmation before proceeding with each change. This option does not remain in effect after a deliver operation is interrupted.

–ser·ial

Uses a serial format when reporting differences among files. Differences are presented in a line-by-line comparison with each line from one contributor, instead of in a side-by-side format. This option does not remain in effect after a deliver operation is interrupted.

CONFIRMATION STEP. *Default:* Prompts for user input.

–f·orce

Suppresses prompting for user input during the course of a deliver operation. The **–force** option does not remain in effect if the deliver operation is interrupted. You must include it again on the command line when you restart the deliver operation with **–resume** or **–complete**. The merge options for **deliver** are not affected by the **–force** option.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form

/vobs/*vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Start a deliver operation using command defaults.

  cmd-context **deliver -to webo_integ**

```
Changes to be DELIVERED:
   FROM: stream "chris_webo_dev"
   TO: stream "integration"
Using integration view: "webo_integ".
Do you wish to continue with this deliver operation?  [no] yes
Needs Merge "/view/webo_integ/vobs/webo_modeler/design/foo" [(automatic)
to /main/integration/1 from /main/integration/chris_webo_dev/1 (base also
/main/integration/1)]
Checked out "/view/webo_integ/vobs/webo_modeler/design/foo" from version
"/main/integration/1".

Attached activities:
activity:deliver.chris_webo_dev.20000606.160519@/vobs/webo_pvob  "deliver
chris_webo_dev on 06/06/00 16:05:19."

Needs Merge "/view/webo_integ/vobs/webo_modeler/design/foo" [to
/main/integration/CHECKEDOUT from /main/integration/chris_webo_dev/1 base
/main/integration/1]

Trivial merge: "/view/webo_integ/vobs/webo_modeler/design/foo" is same as
base "/view/webo_integ/vobs/webo_modeler/design/foo@@/main/integration/1".

Copying
"/view/webo_integ/vobs/webo_modeler/design/foo@@/main/integration/chris_we
bo_dev/1" to output file.
Moved contributor "/view/webo_integ/vobs/webo_modeler/design/foo" to
"/view/webo_integ/vobs/webo_modeler/design/foo.contrib".
Output of merge is in "/view/webo_integ/vobs/webo_modeler/design/foo".
Recorded merge of "/view/webo_integ/vobs/webo_modeler/design/foo".

Deliver has merged
FROM: stream "chris_webo_dev"
TO: stream "integration"
Using integration view: "webo_integ".
Build and test are necessary in integration view "webo_integ"
to ensure that the merges were completed correctly.  When build and
test are confirmed, run "cleartool deliver -complete".
```

- Complete a deliver operation that is in progress.

  cmd-context **deliver –complete**

```
Resume deliver
   FROM: stream "chris_webo_dev"
   TO: stream "integration"
Using integration view: "webo_integ".
Do you wish to continue with this deliver operation?  [no] yes
Are you sure you want to complete this deliver operation?  [no] yes
Deliver has completed
   FROM: stream "chris_webo_dev"
   TO: stream "integration"
Using integration view: "webo_integ".
```

- Check the status of a deliver operation.

  cmd-context **deliver –status**

```
Deliver operation in progress on stream "stream:chris_webo_dev@\webo_pvob"
   Started by "ktessier" on "14-Jun-00.16:07:46"
   Using integration activity "deliver.chris_webo_dev.20000614.160746".
   Using view "webo_integ".
   Activities will be delivered to stream "stream:integration@\webo_pvob".
Development Stream Baselines:
baseline:deliverbl.chris_webo_dev.20000614.160746.129@\webo_pvob
Activities:
activity:fix_copyright@\webo_pvob
activity:update_date@\webo_pvob
activity:fix_defect_215@\webo_pvob
```

- Cancel a deliver operation that is in progress.

  cmd-context **deliver –cancel**

```
Cancel deliver
   FROM: stream "chris_webo_dev"
   TO: stream "integration"
Using integration view: "webo_integ".
Are you sure you want to cancel this deliver operation?  [no] yes
Private version of "/view/webo_integ/vobs/webo_modeler/design/add_proc"
saved in "/view/webo_integ/vobs/webo_modeler/design/add_proc.keep".
Deliver of stream "chris_webo_dev" canceled.
```

SEE ALSO

checkin, **checkout**, **chproject**, **chstream**, **findmerge**, **mkeltype**, **rebase**, **setactivity**

describe

Describes an object

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command
MultiSite	multitool subcommand

Platform
UNIX
Windows

SYNOPSIS

- ClearCase, Attache, and MultiSite only—Describe objects in graphical format:

  des·cribe –g·raphical { *object-selector* | *pname* } ...

- ClearCase LT on Windows only—Describe objects in graphical format:

  des·cribe –g·raphical { *object-selector* | *pname* } ...

- Describe objects:

  des·cribe [**–local**] [**–l·ong** | **–s·hort** | **–fmt** *format-string*]
  [**–ala·bel** { *label-type-selector*[,...] | **–all** }]
  [**–aat·tr** { *attr-type-selector*[,...] | **–all** }]
  [**–ahl·ink** { *hlink-type-selector*[,...] | **–all** }]
  { [**–cvi·ew**] [**–ver·sion** *version-selector* | **–anc·estor**]
  [**–ihl·ink** { *hlink-type-selector*[,...] | **–all** }]
  [**–pre·decessor**] [**–pna·me**] *pname* ...
  | **–typ·e** *type-selector* ...
  | **–cact**
  | *object-selector* ...
  }

DESCRIPTION

The **describe** command lists information about VOBs and the objects they contain. For example:

- Attributes and/or version labels attached to a particular version
- Hyperlinks attached to a particular object
- Predecessor version of a particular version
- Views that have checkouts.
- Views that have unshared derived objects in a particular VOB (**describe –long vob:**) (ClearCase and Attache dynamic views only)
- Family feature level of a VOB or the replica feature level of a MultiSite VOB replica. This information is of interest only to MultiSite users.

describe produces several kinds of listings:

- **File-system data —** Provides information on elements, branches, versions, derived objects, and VOB symbolic links.

  The description of an element (for example, **describe hello.h@@**) includes a listing of the storage pools to which the element is currently assigned. (See **mkpool** and **chpool** for more information.)

  A version's description includes the version-ID of its predecessor version.

  An ordinary derived object is listed with `derived object` in its header. A derived object that has been checked in as a version of an element (DO version) is listed with `derived object version` in its header.

- **Type object** — Provides information on a VOB's type objects (for example, on a specified list of label types). This form of the command displays the same information as **lstype –long**.
- **Hyperlink object —** Provides information on a hyperlink object.
- **Storage pool —** Provides information on a VOB's source, derived object, and cleartext storage pools. This form of the command displays the same information as **lspool** –**long**.
- **VOB object —** Provides information on the object that represents the VOB itself. This includes such information as its storage area, creation date, owner, and related views.
- **VOB replica —** Provides information on the object that represents a VOB replica, including the replica's master replica, host, mastership request setting, feature level, and TCP connectivity to the current replica. For more information on replicas, see the *Administrator's Guide*.

- **UCM objects** — Provides information on UCM objects: activities, baselines, components, folders, projects, and streams. This form of the command displays information similar to that displayed by the UCM commands **lsactivity –long**, **lsbl –long**, **lscomp –long**, **lsfolder –long**, **lsproject –long**, and **lsstream –long**.

Access Control Information

For an MVFS object, **describe** lists the object's protections. For information on access control, see the *Administrator's Guide* and the reference pages for **protect** and **protectvob**.

Unavailable Remote VOB

File-system objects can be hyperlinked to objects in another VOB. If the other VOB is currently unavailable (perhaps it has been unmounted), **describe** tries to be helpful:

```
cleartool: Error: Unable to locate versioned object base with object id:
 "51023fa9.68b711cc.b358.08:00:69:02:1d:c7".
.
.
.
Hyperlinks:
@183@/usr/proj /usr/proj/elem2.c@@/main/2 -> <object not available>
```

Versions Without Data

The description of a version can include the annotation `[version has no data]`. A file element version can be created without data, using **checkin –cr**; an existing version's data can be removed with **rmver –data**.

Hyperlink Inheritance

By default, a version inherits a hyperlink attached to any of its ancestor versions, on the same branch or on a parent branch. Inherited hyperlinks are listed only if you use the **–ihlink** option.

A hyperlink stops being passed down to its descendents if it is superseded by another hyperlink of the same type, explicitly attached to some descendent version. You can use a null-ended hyperlink (from-object, but no to-object) as the superseding hyperlink to effectively cancel hyperlink inheritance.

DOs in Unavailable Views

NOTE: Derived objects may be present only in ClearCase and Attache dynamic views.

describe maintains a cache of tags of inaccessible views. For each view-tag, **describe** records the time of the first unsuccessful contact. Before trying to access a view, **describe** checks the cache. If the view's tag is not listed in the cache, **describe** tries to contact the view. If the view's tag is listed in the cache, **describe** compares the time elapsed since the last attempt with the time-out period specified by the **CCASE_DNVW_RETRY** environment variable. If the elapsed time is greater than the time-out period, **describe** removes the view-tag from the cache and tries to contact the view again.

The default timeout period is 60 minutes. To specify a different time-out period, set CCASE_DNVW_RETRY to another integer value (representing minutes). To disable the cache, set CCASE_DNVW_RETRY to 0.

Objects in Replicated VOBs

The **describe** command shows additional information for objects in MultiSite replicated VOBs:

- For objects that have mastership, **describe** shows the master replica of the object.

  NOTE: If the object is a local instance of a global type and you do not specify **–local**, **describe** shows the master replica of the global type.

- For attribute types, hyperlink types, and label types, **describe** shows the instance mastership of the type (whether the type's mastership can be shared by multiple replicas).
- For branches and branch types, **describe** shows the mastership request setting of the object. This setting controls whether users at other sites can request mastership of the object.

For more information about replicated VOBs, see the *Administrator's Guide*.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

DESCRIBING OBJECTS GRAPHICALLY. *Default:* Describes objects in nongraphical form.

–g·raphical
: Starts a browser that describes objects.

DESCRIBING LOCAL COPIES OF GLOBAL TYPES. *Default:* **describe** displays information about the global type object for the specified *object-selector*.

–local
: Displays information for the local copy of the specified *object-selector*. For more information about global types, see the *Administrator's Guide*.

REPORT FORMAT. *Default:* Lists the object's name and some additional information.

–l·ong
: Expands the listing. With **vob:**, for example, lists all views that have checkouts or unshared derived objects associated with the specified VOB. This listing includes the UUIDs of those views, which can be used with **rmview**.

–s·hort

Lists only an object's pathname. The effect is slightly different when used in combination with **–alabel**, **–aattr**, **–ahlink**, **–ihlink**, or **–predecessor**.

–fmt *format-string*

Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this report-writing facility.

DESCRIBING OBJECTS IN OTHER VIEWS. *Default:* If you use a view-extended pathname to specify an object in (or as seen through) another view, **describe** lists that view's name for the object:

```
version: "M:\gamma\vob1\project\src\util.c"
```

–cvi·ew

Lists an object using the current view's name for it.

```
version: "/usr/project/src/all_utils.c"
```

This option is useful when different views select different directory versions, in which elements have been renamed.

–cact

Describes the current activity for your view.

EXCERPTING DESCRIPTION INFORMATION. *Default:* **describe** lists the predecessor (if the object is a version), and reports on all of the object's version labels, attributes, and hyperlinks. With one or more of the following options, the report includes the extended pathname of the object and the requested information only—for example, only a listing of the predecessor version and version label.

–ala·bel { *label-type-selector*[,...] | **–all** }
–aat·tr { *attr-type-selector*[,...] | **–all** }
–ahl·ink { *hlink-type-selector*[,...] | **–all** }
–ihl·ink { *hlink-type-selector*[,...] | **–all** }
–pre·decessor

Specify one or more of these options to excerpt information from the overall description of an object. A list of names of type objects must be comma-separated, with no white space; you can use the special keyword **–all** to specify all types of a particular kind.

If you combine **–fmt** with any of these options, **describe** uses the *format-string* to construct and display the object's extended pathname.

For the *type-selector* arguments, use one of the type selectors shown in the *object-selector* description.

If you specify **–short** as well, the listing is restricted even further.

–predecessor Only the version-ID of the predecessor version is listed.

–alabel	Only the version labels are listed.
–aattr	Only the attribute values are listed.
–ahlink	The listing includes the pathnames of the objects hyperlinked to *pname*, annotated with → (listed object is the to- object) or ← (listed object is the from-object).

For example:

```
-> M:\gamma\vob1\proj\include\db.c@@\main\52
<- M:\gamma\vob1\proj\bin\vega@@\main\5
```

Inherited hyperlinks are not included in this listing.

–ihlink The listing includes the hyperlinks inherited by *pname*, which must specify a version. Pathnames of the from-object and to- object are listed, one of which is an ancestor of *pname*, or is *pname* itself. (That is, **–ihlink** also includes hyperlinks that are attached to *pname* itself.)

SPECIFYING THE OBJECTS TO BE DESCRIBED. *Default:* **describe** expects at least one argument that names an element, branch, version, VOB link, derived object, or hyperlink (*pname*, *DO-name*, or *hlink-selector*). You can use **–version** or **–ancestor** to control the way *pname* arguments are interpreted.

[**–pna·me**] *pname* ...

One or more pathnames, indicating objects to be described: elements, branches, versions, or derived objects. If *pname* has the form of an object selector, you must include the **–pname** option to indicate that *pname* is a pathname.

- A standard or view-extended pathname to an element specifies the version selected by the view.
- A standard or view-extended pathname to a derived object specifies the DO in the view.
- An extended pathname specifies an element, branch, version, or derived object, different from the one selected by the view. For example:

`foo.c`	*(version of foo.c selected by current view)*
`foo.o`	*(derived object foo.o built in or winked in to current view)*
`/view/gamma/usr/proj/src/foo.c`	*(version of foo.c selected by another view; however, the current view must select some version of foo.c)*
`M:\gamma\vob1\proj\src\foo.o`	*(derived object foo.o built in another view)*
`foo.c@@/main/5`	*(version 5 on main branch of foo.c)*
`foo.o@@11-Nov.09:19.219`	*(derived object, specified by DO-ID)*

`foo.c@@\REL3`	*(version of foo.c with version label REL3; however, the view must select some version of foo.c)*
`foo.c@@`	*(the element foo.c)*
`foo.c@@/main`	*(the main branch of element foo.c)*

For versions, **–version** overrides these interpretations of *pname*.

–ver·sion *version-selector*

(For use with versions only) For each *pname*, describes the version specified by *version-selector*. This option overrides both version selection by the view and version-extended naming. See the **version_selector** reference page for syntax details.

–anc·estor

(For use with elements and versions only) Describes the closest common ancestor version of all the *pname* arguments, which must all be versions of the same element. See the **merge** reference page for a information about closest common ancestors.

–typ·e *type-selector* ...

Lists information about the type objects specified by the *type-name* arguments. If there are multiple types with the same name (for example, a label type and a hyperlink type are both named **REL3**), all of them are listed. Use one of the *type-selectors* shown in the description of the *object-selector* argument.

object-selector ...

One or more *object-selectors*, indicating objects to be described. Specify *object-selector* in one of the following forms:

vob-selector	[**vob:**]*pname-in-vob* *pname-in-vob* can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted). It cannot be the pathname of the VOB storage directory.
attribute-type-selector	**attype:***type-name*[@*vob-selector*]
branch-type-selector	**brtype:***type-name*[@*vob-selector*]
element-type-selector	**eltype:***type-name*[@*vob-selector*]
hyperlink-type-selector	**hltype:***type-name*[@*vob-selector*]
label-type-selector	**lbtype:***type-name*[@*vob-selector*]
trigger-type-selector	**trtype:***type-name*[@*vob-selector*]
pool-selector	**pool:***pool-name*[@*vob-selector*]
hlink-selector	**hlink:***hlink-id*[@*vob-selector*]
oid-obj-selector	**oid:***object-oid*[@*vob-selector*]

The following object selectors are valid only if you use MultiSite:

replica-selector	**replica:***replica-name*[@*vob-selector*]
replica-uuid-selector	**oid:***replica-oid***@replicauuid:***replica-oid*
	NOTE: The replica you specify must be located at your current site.

The following object selectors are valid only if you use UCM:

activity-selector	**activity**:*activity-name*[@*vob-selector*]
baseline-selector	**baseline:***baseline-name*[@*vob-selector*]
component-selector	**component:***component-name*[@*vob-selector*]
folder-selector	**folder:***folder-name*[@*vob-selector*]
project-selector	**project:***project-name*[@*vob-selector*]
stream-selector	**stream***:stream-name*[@*vob-selector*]

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Describe the version of element **msg.c** selected by your view.

  cmd-context **describe msg.c**

```
version "msg.c@@/main/3"
 created 08-Dec-98.12:12:55 by Chuck Jackson (test user) (cj.dvt@oxygen)
 Element Protection:
   User : sgd      : r--
   Group: user     : r--
   Other:          : r--
 element type: c_source
 predecessor version: /main/2
 Labels:
  REL6
  REL1
```

- Describe a branch of an element, specifying it with an extended pathname.

  cmd-context **describe util.c@@\main\rel2_bugfix**

```
branch "util.c@@\main\rel2_bugfix"
 created 08-Dec-98.12:15:40 by Bev Jackson (test user) (bev.dvt@oxygen)
 branch type: rel2_bugfix
 Element Protection:
   User : sgd      : r--
   Group: user     : r--
   Other:          : r--
 element type: text_file
 branched from version: \main\31
```

- Describe the label type **REL3**.

  cmd-context **describe lbtype:REL3**

```
label type "REL3"
 created 08-Dec-98.12:13:36 by Bev Jackson (test user) (bev.dvt@oxygen)
 owner: bev
 group: dvt
 scope: this VOB (ordinary type)
 constraint: one version per branch
```

- Create a **Tested** attribute type and apply the attribute to the version of element **util.c** selected by your current view. Then, use **describe** to display the newly applied attribute value, and use the **–fmt** option to format the output.

  cmd-context **mkattype -nc -default '"TRUE"' Tested**

  cmd-context **mkattr -default Tested util.c** *cmd-context* **describe -aattr -all -fmt "Name: %Xn\nType of object: %m\n" util.c**

```
Name: util.c@@/main/CHECKEDOUT
Type of object: version
 Attributes:
 Tested = "TRUE"
```

- Describe **ddft**, the current VOB's default derived object storage pool.

cmd-context **describe pool:ddft**

```
pool "ddft"
 created 15-Dec-98.09:34:00 by jenny.adm@oxygen
 "Predefined pool used to store derived objects."
 owner: jenny
 group: dvt
 kind: derived pool
 pool storage global pathname "\\oxygen\users\vb_store\tut\tut.vbs\d\ddft"
 maximum size: 0 reclaim size: 0 age: 96
```

- Describe how the current view names an element that is named **hello.mod** in the **jackson_fix** view.

cmd-context **describe -cview /view/jackson_fix/usr/hw/src/hello.mod**

```
version "/usr/hw/src/hello.c@@/main/4"
 created 08-Dec-98.12:16:29 by Chuck Jackson (test user) (cj.dvt@oxygen)
 Element Protection:
   User : sgd      : r--
   Group: user     : r--
   Other:          : r--
 element type: text_file
 predecessor version: /main/3
```

- Describe the VOB containing the current working directory. List views with checkouts or unshared derived objects in that VOB.

cmd-context **describe -long vob:.**

```
versioned object base "\hw"
 created 15-Dec-98.09:34:00 by jenny.adm@oxygen
 "VOB dedicated to development of "hello, world" program"
 VOB family feature level: 2
 VOB storage host:pathname "oxygen:c:\users\vb_store\tut\tut.vbs"
 VOB storage global pathname "\\oxygen\users\vb_store\tut\tut.vbs"
 VOB ownership:
   owner jackson
   group dvt
 VOB holds objects from the following views:
   oxygen:\vb_store\tut\old.vws [uuid
249356fe.d50f11cb.a3fd.00:01:56:01:0a:4f]
```

- Describe a hyperlink.

  cmd-context **describe hlink:Merge@516262@/vobs/proj**

```
hyperlink "Merge@516262@/vobs/proj"
 created 14-Jul-98.16:43:35 by Bill Bo (bill.user@uranus)
 Merge@516262@/vobs/proj /vobs/proj/lib/cvt/cvt_cmd.c@@/main/v1.1_port/8
->
 /vobs/proj/lib/cvt/cvt_cmd.c@@/main/71
```

- Describe a derived object in the current view.

  cmd-context **describe -cview util.o**

```
derived object "util.o@@11-Apr.12:03.33"
 created 11-Apr-98.12:03:33 by Anne Duvo (anne.dev@oxygen)
 references: 2 (shared)
 derived pool: ddft
 => saturn:/usr/anne/views/anne_main.vws
 => oxygen:/usr/jackson/views/jackson_proj2.vws
```

- Describe a derived object in the current working directory.

  cmd-context **describe util.obj**

```
derived object "util.o@@11-Apr.12:03.33"
 created 11-Apr-98.12:03:33 by Anne Duvo (anne.dev@oxygen)
 references: 2 (shared)
 derived pool: ddft
 => saturn:\users\anne\views\anne_main.vws
 => oxygen:\users\jackson\views\jackson_proj2.vws
```

- For a particular element, list its name, element type, attached triggers, and cleartext and source pools.

  cmd-context **describe –fmt ^**
  "%n\n\t%[type]p\n\t%[triggers]p\n\t%[pool]Cp,%[pool]p\n" file.txt@@

```
file.txt@@
   text_file
   (CI_TRIG, CO_TRIG)
   cdft,sdft
```

- For a branch type in a replicated VOB, list the master replica of the branch type.

  cmd-context **describe –fmt "%n\t%[master]p\n" brtype:main**

```
main    lex@/vobs/dev
```

- For the current VOB, list the OID, replication status, MS-DOS text mode setting, and creation comment.

  cmd-context **describe –fmt "%On\n%[vob_replication]p\n%[msdostext_mode]p\n%c" ^ vob:.**

  ```
  46cf5bfd.240d11d3.a37e.00:01:80:7b:09:69
  unreplicated
  disabled
  storage of header files
  ```

- Describe the local copy of global label type **REL6**.

  cmd-context **describe –local lbtype:REL6**

  ```
  label type "REL6"
    created 28-Jul-99.14:00:26 by Suzanne Gets (smg.user@neon)
    "Automatically created label type from global definition in VOB
  "/vobs/admin"."
    owner: smg
    group: user
    scope: this VOB (local copy of global type)
    constraint: one version per element
    Hyperlinks:
      GlobalDefinition -> lbtype:REL6@/vobs/admin
  ```

- Describe the current VOB, its hyperlinks being of particular interest.

  cmd-context **describe –long vob:.**

  ```
  versioned object base "/vobs/doc"
   created 07-Nov-91.16:46:28 by ratl.user
   "ClearCase documentation VOB."
   VOB family feature level: 1
   VOB storage host:pathname "mercury:\usr3\vobstorage\doc_vob"
   VOB storage global pathname "\\mercury\usr3\vobstorage\doc_vob"
   VOB ownership:
    owner 4294967294
    group user
   Hyperlinks:
    AdminVOB -> vob:\vobs\admin
  ```

  This VOB has a hyperlink named **AdminVOB**; it points from the current VOB to the VOB **vob:\vobs\admin**. If it were pointing to the current VOB, the listing would show

  ```
   Hyperlinks:
    AdminVOB <- vob:\vobs\admin
  ```

  Now describe the hyperlink **AdminVOB**.

  cmd-context **describe hltype:AdminVOB**

```
hyperlink type "AdminVOB"
created 07-Nov-91.16:46:28 by ratl.user
"Predefined hyperlink type used to link a VOB to another VOB with
administrative information."
owner: 4294967294
group: user
scope: this VOB (ordinary type)
```

- List the name, master replica, and host of a replica by specifing its OID. (Note that the **oid:** object-selector must be entered on a single line.)

  cmd-context **describe –fmt "%n\n%[master]p\n%[replica_host]p\n" \ oid:87f6265f.72d911d4.a5cd.00:01:80:c0:4b:e7@replicauuid:87f6265f.72d911d4.a5cd.00:01:80:c0:4b:e7**

  ```
  boston_hub
  boston_hub@/vobs/dev
  minuteman
  ```

SEE ALSO

chflevel, **chpool**, **fmt_ccase**, **lsactivity**, **lsbl**, **lscomp**, **lsdo**, **lshistory**, **lspool**, **lsproject**, **lsstream**, **lstype**, **merge**, **mkpool**, **protect**, **protectvob**, **rmview**, **version_selector**

diff

Compares versions of a text-file element or a directory

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and ClearCase LT—Display differences graphically:

  diff –g·raphical [**–tin·y**] [**–hst·ack** | **–vst·ack**] [**–pre·decessor**]
  [**–opt·ions** *pass-through-opts*] *pname* ...

- ClearCase and ClearCase LT on UNIX—Display differences nongraphically:

  diff [**–tin·y** | **–win·dow**] [**–ser·ial_format** | **–dif·f_format** | **–col·umns** *n*]
  [**–opt·ions** *pass-through-opts*] [**–pre·decessor**] *pname* ...

- ClearCase and ClearCase LT on Windows—Display differences nongraphically:

  diff [**–ser·ial_format** | **–dif·f_format** | **–col·umns** *n*]
  [**–opt·ions** *pass-through-opts*] [**–pre·decessor**] *pname* ...

- Display differences graphically:

  diff –g·raphical [**–tin·y**] [**–hst·ack** | **–vst·ack**] [**–pre·decessor** | **–vie·w**]
  [**–opt·ions** *pass-through-opts*] *pname* ...

- Attache—Display differences nongraphically:

  diff [**–ser·ial_format** | **–dif·f_format** | **–col·umns** *n*]
  [**–opt·ions** *pass-through-opts*] [**–pre·decessor** | **–vie·w**] *pname* ...

DESCRIPTION

ClearCase and ClearCase LT

The **diff** command calls an element-type-specific program (the compare method for the type) to compare the contents of two or more file elements, or two or more directory elements. Typically, the files are versions of the same file element.

You can also use this command to compare ordinary text files.

diff uses the type manager mechanism to determine how to compare the specified objects. For more information, see the **type_manager** reference page.

Attache

The **diff** command compares the contents of two or more file elements or two or more directory elements. Typically, the files are versions of the same file element. For a file **diff**, any locally-referenced files are used as contributors; any nonlocal files are downloaded temporarily. **diff** presumes that all files are text files, using the built-in textual **diff** and **merge** compare methods, and bypassing the type manager mechanism. For more information, see the **type_manager** reference page.

TEXT FILE COMPARISON REPORT FORMAT

Each difference is reported as one or more pairwise differences. For example, if three contributors all differ from the base contributor in a particular section, **diff** lists the file1-file2 difference, followed by the file1-file3 difference, followed by the file1-file4 difference.

Side-by-Side File Comparison Report Style

The default file-comparison report begins with a file summary, which lists all the input files and their assignments as `file 1`, `file 2`, and so on. If no differences are detected among the files, this listing is replaced by the message `Files are identical`.

The remainder of the report is a series of pairwise differences, each of which is preceded by a descriptive header line:

```
********************************                          (file summary)
<<< file 1: util.c@@/main/1
>>> file 2: util.c@@/main/3
********************************
----------[after 15]-----|-------[inserted 16]-------     (header)
                        -| char *s;                       (difference)
                         |-
---------[changed 18]----|-----[changed to 19-21]----     (header)
return ctime(&clock);    | s = ctime(&clock);             (difference)
                        -| s[ strlen(s)-1 ] = '\0';
                         | return s;
                         |-
```

The **–quiet** and **–diff_format** options suppress the file summary. The **–headers_only** option suppresses the differences, listing the header lines only.

Header Lines. Each header line indicates which text lines in the input files were changed, and how they were changed. The words describe the change in terms of how the first file was changed to produce the second file. Header lines can have the following formats, where each of *A*, *B*, and so on may be a single line number (for example, **46**) or a range (for example, **256–290**):

`------------[after` *A*`]------------|------------[inserted` *B*`]-------------`

Insertion of one or more lines. *B* indicates where the inserted lines occur in the second file. *A* indicates the corresponding point in the first file.

`-----------[deleted` *C*`]-----------|--------------[after` *D*`]--------------`

Deletion of one or more lines. *C* indicates which lines from the first file were deleted. *D* indicates the corresponding point in the second file.

`--------[deleted/moved` *C*`]--------|----------[after` *D* `now` *B*`]-----------`

Deletion of one or more lines from the first file, to which there corresponds an insertion of the same lines in the second file. Typically, this indicates that a range of lines was moved from one location to another; see `inserted/moved` below. *C* indicates where the lines were deleted from the first file; *B* indicates the location where these same lines were inserted in the second file. *D* indicates the point in the second file that corresponds to *C*.

`-----------[changed` *X*`]-----------|------------[changed to` *Y*`]-----------`

One or more lines changed in place. *X* indicates which lines in the first file were changed. *Y* indicates where the replacement lines occur in the second file.

`----------[after` *A* `was` *C*`]--------|----------[inserted/moved` *B*`]--------`

Insertion of one or more lines in the second file, to which there corresponds a deletion of the same lines from the second file. Typically, this indicates that a range of lines was moved from one location to another; see `deleted/moved` above. *B* indicates where the lines were inserted in the second file; *C* indicates where these same lines were deleted from the first file; *A* indicates the point in the first file that corresponds to *B*.

Differences. diff can report a difference in several ways. When comparing files, its default is to list corresponding lines side by side, and possibly truncated.

A plus sign (+) at the end of a difference line indicates that it has been truncated in the report. To see more of such lines, you can increase the report width using the **–columns** option (or, in ClearCase and ClearCase LT on UNIX, using the **–tiny** option. The minus signs (–) along the vertical separator line indicate the endpoints of the groups of differing lines. They help to distinguish empty lines in the input files from blank space in command output.

Other File Comparison Report Styles

The **–serial_format** option causes the differences to be reported as entire lines, in above-and-below format instead of side-by-side format. For example:

```
-----[after 15 inserted 16]-----
>    char *s;
-----[18 changed to 19-21]-----
<    return ctime(&clock);
---
>    s = ctime(&clock);
>    s[ strlen(s)-1 ] = ' ';
>    return s;
```

The **–diff_format** option causes both the headers and differences to be reported in the style of the UNIX **diff** utility, writing a list of the changes necessary to convert the first file being compared into the second file, as follows:

- The first number (or range of numbers in the form *n*,*n*) indicates line numbers in the first file being compared.
- The second value is one of the following: **a d c**. These indicate whether lines are to be added deleted, or changed.
- The second number (or range of numbers) indicates line numbers in the second file being compared.

When **diff** compares multiple files, it adds file-identification annotations to the **diff**-style headers.

The **–graphical** option displays differences graphically.

DIRECTORY-COMPARISON ALGORITHM AND REPORT FORMAT

For a comparison of directory versions, **diff** uses a directory-element-specific compare method, whose report format is very similar to the one described in *Side-by-Side File Comparison Report Style* on page 301.

Kinds of Directory Entries

A version of a VOB directory can contain several types of entries:

- **File Elements —** Reported by **diff** as the element's name (in this directory version), the element's creation time, and the username of the element's creator. For example:

  ```
  obj2 12-Aug.14:00 akp
  ```

  NOTE: Multiple VOB hard links to the same element will have the same creator and creation time, but different names.

- **Directory Elements —** Reported by **diff** in the same way as file elements, except that a slash (UNIX) or backslash (Windows) is appended to the element name. For example:

  ```
  sub6/ 13-Aug.15:00 akp
  ```

- **VOB Symbolic Links —** Reported by **diff** as the link's name (in this directory version), followed by – and the text (contents) of the link; the link's creation time; and the username of the element's creator. For example:

  ```
  doctn -> ..\vob1\doctn 13-Aug.08:44 akp
  ```

How Differences Are Reported

The **diff** report is a series of differences, each of which focuses on one directory entry. A difference can be a simple addition or deletion; it can also involve the renaming of an existing object, or the reuse of an existing name for another object. The following examples illustrate the various possibilities.

```
----------------------------------------|---------------[ added ]-------------
                                       -| obj2 12-Aug.14:00 akp
```

An object named **obj2** was added (**mkelem**, **mkdir**, or **ln**) in the second version of the directory.

```
---------------[ removed ]--------------|-----------------------------
obj5 12-Aug.14:00 akp                   |-
```

An object named **obj5** was removed (**rmname**) in the second version of the directory.

```
---------------[ renamed ]--------------|--------[ renamed to ]-------
obj3 12-Aug.14:00 akp                   | obj3.new 12-Aug.14:00 akp
```

An object named **obj3** was renamed (**mv**) to **obj3.new** in the second version of the directory.

```
-------------[ old object ]-------------|--------[ new object ]-------
obj4 12-Aug.14:04 akp                   | obj4 19-Oct.17:10 akp
```

In the second version of the directory, an object named **obj4** was removed (**rmname**) and another object was created with that same name.

```
----------[ old link text ]-------------|------[ new link text ]----
doc -> ../vob1/doc 13-Aug.08:44 akp     | doc -> ../vb/doc 19-Sep.21:01 akp
```

(Special case of the preceding example) In the second version of the directory, a VOB symbolic link named `doc` was removed (**rmname**) and another VOB symbolic link was created with that same name.

```
---------------[ renamed ]--------------|---[ renamed to ]------------
obj4 12-Aug.14:01 akp                   | obj1 12-Aug.14:01 akp
---------------[ removed ]--------------|-----------------------------
obj1 12-Aug.14:00 akp                   |-
```

These two differences show that in the second version of the directory, an object named **obj1** was removed and another object was renamed from **obj4** to **obj1**.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

In ClearCase and ClearCase LT, with the exception of **–predecessor** and **–options**, **diff** options are the same as those of **cleardiff**.

REPORTING DIFFERENCES GRAPHICALLY. *Default:* Reports differences in nongraphical form and uses the default display font.

–g·raphical [**–tin·y**]
: Displays differences graphically. With **–tiny**, a smaller font is used to increase the amount of text displayed in each display pane.

NOTE: When comparing files of type html on UNIX systems, if the machine on which you execute **diff –graphical** is not the machine on which you run your HTML browser, your browser may not be able to find the pathname to the files being compared.

USING A SEPARATE WINDOW. *Default:* Sends output to the current window.

–tin·y
: Same as **–window**, but uses a smaller font in a 165-character difference window.

–win·dow
: Displays output in a separate difference window, formatted as with **columns 120**. Type an operating system interrupt character (typically, **CTRL+C**) in the difference window to close it. The **diff** command returns immediately, not waiting for the difference window to be closed.

CONTRIBUTOR PANE STACKING. *Default:* Each of the two or more contributors being compared or merged is displayed in a separate subwindow, or contributor pane. By default, these panes are displayed, or stacked, horizontally (side by side), with the base contributor on the left.

–hst·ack
: Displays the contributor panes horizontally (the default behavior).

–vst·ack
: Stacks the contributor panes vertically, with the base contributor at the top.

OUTPUT FORMAT. *Default:* Reports differences in the format described in *How Differences Are Reported* on page 304.

–ser·ial_format

Reports differences with each line containing output from a single file, instead of in a side-by-side format.

–dif·f_format

Reports both headers and differences in the same style as the UNIX **diff** utility (see *Other File Comparison Report Styles* on page 303), and suppresses the file summary from the beginning of the report.

–col·umns *n*

Establishes the overall width of a side-by-side report. The default width is 80; only the first 40 or so characters of corresponding difference lines appear. If *n* does not exceed the default width, this option is ignored.

COMPARISON OF A VERSION WITH ITS PREDECESSOR. *Default:* None.

–pre·decessor

Effectively converts the first *pname* argument into two names: (1) the predecessor version of *pname* in the version tree; (2) *pname* itself. If *pname* specifies a checked-out version, the predecessor is the version from which it was checked out.

An error occurs if the *pname* does not specify a version:

```
cleartool: Error: Not a vob object: "myfile.c".
```

(ATTACHE) SPECIFYING THE BASIS OF THE COMPARISON. *Default:* None.

–vie·w

Supports file elements only. Converts the first *pname* argument into two names: (1) the version of *pname* selected by the view; (2) *pname* itself. If *pname* is not already present in the workspace, using **–view** results in a comparison of *pname* with itself.

PASSING THROUGH OPTIONS TO THE COMPARE METHOD. *Default:* Does not pass any special options to the underlying compare method (in ClearCase and ClearCase LT, typically, the **cleardiff** program).

–opt·ions *pass-through-opts*

Specifies one or more compare method options that are not directly supported by **diff**.

Use quotes if you are specifying more than one pass-through option; **diff** must see them as a single command-line argument. For example, this command passes through the **–quiet** and **–blank_ignore** options:

cmd-context **diff –options "–qui –b" –pred util.c**

For descriptions of the options valid in ClearCase and ClearCase LT, see the **cleardiff** reference page. Attache accepts the following pass-through options:

–hea·ders_only
–qui·et (mutually exclusive)

–headers_only lists only the header line of each difference. The differences themselves are omitted.

–quiet suppresses the file summary from the beginning of the report.

–b·lank_ignore

Ignores extra white space characters in text lines: leading and trailing white space is ignored altogether; internal runs of white-space characters are treated like a single **SPACE** character.

SPECIFYING THE DATA TO BE COMPARED. *Default:* None.

pname ...

One or more pathnames, indicating the objects to be compared: versions of file elements, versions of directory elements, or any other files. If you don't use **–predecessor** or **–view**, you must specify at least two *pname* arguments.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- (Attache) Compare the version of **foo.c** selected by the current view with the version in the current workspace.

  cmd-context **diff –view foo.c**

- (ClearCase and ClearCase LT) Compare the version of a file element in the current view with the version in another view.

  cmd-context **diff util.c /view/jackson_old/usr/hw/src/util.c**

- (Attache) Compare the version of a file element in the current workspace with an older version.

  cmd-context **diff util.c util.c\@@\main\1**

- Compare the version of **foo.c** in the current view with its predecessor version.

  cmd-context **diff –predecessor foo.c**

- Compare your unreserved checkout of **hello.c** with the latest checked-in version on the **main** branch.

  cmd-context **diff hello.c hello.c@@\main\LATEST**

- Compare three files: the version of **msg.c** selected by the current view, its predecessor version, and **msg.SAVE** in your home directory.

  cmd-context **diff –pre msg.c $HOME/msg.SAVE**

- In a separate 132-column window, compare the version of **util.c** in the current view with a version on the **rel2_bugfix** branch.

  cmd-context **diff –window –columns 132 util.c util.c@@/main/rel2_bugfix/LATEST**

- (ClearCase and ClearCase LT) Start the Diff Merge Tool to compare the version of **util.c** in the current view with a version on the **rel2_bugfix** branch.

  cmd-context **diff –graphical util.c util.c@@\main\rel2_bugfix\LATEST**

- (Attache) Use **diff –graphical** to compare two files in different local directories. (This command must be entered on a single line.)

  cmd-context **diff –graphical \users\jed\jed_ws\vob_des\source\test.c \users\jpb\my_proj\test_NEW.c**

SEE ALSO

attache_command_line_interface, **diffcr**, **merge**, **type_manager**, **xclearcase**, **xcleardiff**

diffbl

Compares the contents of baselines or streams

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- Display differences between two baselines or streams nongraphically:

  diffbl [**–act·ivities**] [**–ver·sions**] [–**bas·elines**] [–**fir·st_only**] [–**nre·curse**]
  { *baseline-selector1* | *stream-selector1* }
  { *baseline-selector2* | *stream-selector2* }

- Display differences between the specified baseline and its predecessor baseline nongraphically:

  diffbl –pre·decessor [**–act·ivities**] [**–ver·sions**] *baseline-selector*

- Display differences between two baselines graphically:

  diffbl –g·raphical *baseline-selector1 baseline-selector2*

- Display differences between the specified baseline and its predecessor baseline graphically:

  diffbl –g·raphical –pre·decessor *baseline-selector*

DESCRIPTION

The **diffbl** command compares the contents of two baselines or streams and displays any differences it finds. You can choose to see differences in terms of activities, versions, and for composite baselines, members.

diffbl

You can use the **diffbl** command to compare a baseline and a stream, a baseline and a baseline, or a stream and a stream. When specifying a stream, all baselines in the stream are used in the comparison as well as any changes in the stream that are not yet captured in a baseline.

The **diffbl** command must be issued from a view context to display versions.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE INFORMATION TO DISPLAY. *Default:* **–activities**.

–act·ivities
Displays differences in terms of activities.

–ver·sions
Displays differences in terms of versions.

–**bas·elines**
For composite baselines, displays differences in terms of member baselines. When used with the **–activities** option, displays each member baseline followed by activities in the member baseline.

–fir·st_only
Shows only those changes that appear in the first object specified for the comparison.

–nre·curse
For composite baselines, displays differences between the named baselines themselves and not between their members.

COMPARING A VERSION WITH ITS PREDECESSOR. *Default:* None.

–pre·decessor
Displays differences between the specified baseline and its immediate predecessor.

REPORTING DIFFERENCES GRAPHICALLY. *Default:* Reports differences in nongraphical form.

–g·raphical
Displays differences graphically. This applies only to baselines, not streams.

SELECTING THE OBJECTS TO COMPARE. *Default:* None.

baseline-selector1
stream-selector1
Specifies objects to use in the comparison.

baseline-selector is of the form: [**baseline:**]*baseline-name*[@*vob-selector*] and *vob* is the baseline's UCM project VOB. *stream-selector* is of the form: [**stream:**]*stream-name*[@*vob-selector*] and *vob* is the stream's UCM project VOB.

baseline-selector2
stream-selector2

Specifies objects to use in the comparison.

baseline-selector is of the form: [**baseline:**]*baseline-name*[@*vob-selector*] and *vob* is the baseline's UCM project VOB. *stream-selector* is of the form: [**stream:**]*stream-name*[@*vob-selector*] and *vob* is the stream's UCM project VOB.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Compare activities in two streams:

  cmd-context **diffbl stream:java_int stream:java_dev**

  ```
  << deliver.java_dev.19990917.140443 "deliver java_dev on 09/17/99
  14:04:43."
  ```

  ```
  << deliver.java_dev.19990917.141046 "deliver java_dev on 09/17/99
  14:10:46."
  ```

- Compare baselines in two streams:

  cmd-context **diffbl -ver stream:java_int stream:java_dev**

  ```
  << /vobs/parser/myfile.c@@/main/java_int/2
  ```

```
<< /vobs/parser/myfile.c@@/main/java_int/3
```

- Compare a baseline with its predecessor baseline:

  cmd-context **diffbl -predecessor test_sum.D001207.124@net/acrolein/export/home/lli/vobs/lli_test_sum**

```
>> deliver.lli_test_sum.20001113.165650 "deliver lli_test6_sum on 11/13/00
16:56:50."

>> test_act_00795 "test act"
```

SEE ALSO

chbl, **lsbl**, **mkbl**, **rmbl**

diffcr

Compares configuration records created by **clearmake**, **omake**, or **clearaudit**

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

diffcr [**–r·ecurse** | **–fla·t**] [**–sel·ect** *do-leaf-pattern*] [**–ci**] [**–typ·e** { **f** | **d** | **l** } ...]
[**–ele·ment_only**] [**–vie·w_only**] [**–cri·tical_only**] [**–nam·e** *tail-pattern*]
[**–wd**] [**–nxn·ame**] [**–fol·low**] [**–l·ong** | **–s·hort**] *do-pname-1 do-pname-2*

DESCRIPTION

The **diffcr** command compares the configuration records (CRs) of two derived objects. A CR is produced by **clearmake**, **clearaudit**, and **omake** when they finish executing a build script in a dynamic view. By comparing CRs, you can determine these differences:

- Versions of MVFS objects used as sources or produced during the build (includes elements and other objects whose pathnames are under a VOB mount point)
- Versions of non-MVFS objects that appeared as makefile dependencies during the build (explicit dependencies declared in the makefile)
- The total number of times an object was referenced during a build, and the first target in which that object was referenced
- Build options (which can come from the command line, the operating system environment, the makefile itself, and so on)
- The build script executed
- Noncritical differences, such as the date/time of the build, dynamic view name, or host name

NOTE: Not all of this information is available from configuration records of DOs created by **clearaudit**.

The *do-pname* arguments specify the derived objects to be compared. You can specify a derived object in these ways:

- Use a derived-objectID (DO-ID), which identifies a derived object created in any dynamic view. A DO-ID takes the following form:

  DO-pname@@*creation_date*.*creation_time*.*id-number*

  For example:

  `myprog.o@@11-Nov.17:39.3871`

  To display a derived object's DO-ID, use **lsdo**.

- Use a standard pathname, which identifies a DO created in the current dynamic view. For example, **myprog.obj**.
- Use a view-extended pathname, which identifies a DO created in another dynamic view. For example, **/view/jpb/usr/src/myprog.o**.

You can compare a nonshareable DO in your view to a nonshareable DO created in another view, but you must use a view-extended pathname to specify the DO in the other view.

diffcr supports the same filter and report style options as the **catcr** command. This means that you can restrict the comparison to particular subtargets of the *do-pname*s, control which objects appear in the listing, select how pathnames are displayed, and expand the listing to include comments and other supplementary information. See the **catcr** reference page for additional information.

DOs in Unavailable Views

diffcr maintains a cache of tags of inaccessible views. For each view-tag, the command records the time of the first unsuccessful contact. Before trying to access a view, the command checks the cache. If the view's tag is not listed in the cache, the command tries to contact the view. If the view's tag is listed in the cache, the command compares the time elapsed since the last attempt with the timeout period specified by the CCASE_DNVW_RETRY environment variable. If the elapsed time is greater than the timeout period, the command removes the view-tag from the cache and tries to contact the view again.

The default timeout period is 60 minutes. To specify a different timeout period, set CCASE_DNVW_RETRY to another integer value (representing minutes). To disable the cache, set CCASE_DNVW_RETRY to 0.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

COMPARING DIFFERENCES IN SUBTARGETS. *Default:* **diffcr** compares the CRs for *do-pname-1* and *do-pname-2* only, not for any of their subtargets.

–r·ecurse
Compares the CRs of the two specified derived objects, and their common subtargets. Each pair of CRs is compared separately. By default, a recursive comparison does not descend into DO versions; use **–ci** to override this.

–fla·t
Similar to **–recurse**, but consolidates the CRs for each *do-pname-n* into a single list, with no duplicates, and then compares the lists. The report includes file system objects only; no headers, variables and options, or build scripts. It also includes the total number of times each object was referenced during the build, and the first target in which that object was referenced (`First seen in target`).

–sel·ect *do-leaf-pattern*
Starts the comparison at the subtargets of *do-pname* that match *do-leaf-pattern* (which can include pattern-matching characters; see the ClearCase **wildcards_ccase** or Attache **wildcards** reference page). This option is useful for focusing on a particular object (for example, object module **hello.obj**) that was built as part of a larger object (for example, executable **hello.exe**).

–ci (for use with **–recurse** or **–flat** only)
Descends into the CRs of DO versions that were used as build sources.

SPECIFYING KINDS OF OBJECTS TO DISPLAY. *Default for UNIX systems:* **diffcr** reports on all objects in the CRs, which may include source files, directories, and symbolic links; derived objects; makefiles; view-private files, and non-MVFS objects that were explicitly declared as dependencies.

Default for Windows systems: **diffcr** reports on all objects in the CRs, which may include source files and directories; derived objects; makefiles; view-private files, and (for builds performed with **omake**) non-MVFS objects that were explicitly declared as dependencies.

–typ·e { **f** | **d** | **l** } ...
Lists file-system objects of a particular kind: files (**f**) directories (**d**), or links (**l**). The default value varies with the report style: normal and short listings (**–short**) default to **f**; long listings (**–long**) default to **fdl**. You may specify multiple kinds of objects by grouping them into a single argument; **–type fd**, for example.

–ele·ment_only
Lists versions of elements only, including checked-out versions. This option excludes from the listing derived objects (except DO versions), view-private files and directories, UNIX symbolic links, and non-MVFS objects.

–vie·w_only
Lists view-private objects only, including checked-out versions of elements. If you specify this option along with **–element_only**, the listing includes only checked-out versions of elements.

–cri·tical_only
Excludes from the listing any differences in objects marked as "noncritical" in the CR. Objects with that property typically have it because the user specified them as dependents of the **.NO_DO_FOR_SIBLING** special target in a **clearmake** or an **omake** makefile.

–nam·e *tail-pattern*
Considers the entry for a file system object only if its final pathname component matches the specified pattern. See the ClearCase **wildcards_ccase** or Attache **wildcards** reference page for a list of pattern-matching characters.

CONTROLLING REPORT APPEARANCE. *Default:* **diffcr** reports, in three sections, on MVFS objects, variables and options, and the build script. The report uses full pathnames, and it omits comments and directory versions.

–wd
Lists pathnames relative to the current working directory, rather than as full pathnames.

–nxn·ame
Lists simple pathnames for MVFS objects, rather than version-extended pathnames or DO-IDs.

–fol·low
Lists the link targets only (that is, suppresses printing of the link text).

–l·ong
Expands the report to include the kinds of objects in the CR, and comments. For example, an object may be listed as a version, a directory version, or derived object (see **ls –long** for a complete list). Comments indicate whether an object is in a makefile, a referenced derived object, or a new derived object.

–s·hort
Restricts the report to file-system objects only (omits header information, variables and options, and build scripts).

SPECIFYING THE DERIVED OBJECTS. *Default:* None.

do-pname-1, *do-pname-2*
Standard pathnames and/or DO-IDs of two derived objects to be compared. Either or both can be a DO version.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Compare the CRs of two derived objects built at the name **bgrs**. Use **lsdo** to determine the DO-ID of the derived object that is not visible in the current dynamic view.

  cmd-context **lsdo -zero bgrs**

  ```
  11-Dec.15:24 "bgrs@@11-Dec.15:24.1487"
  11-Dec.12:05 "bgrs@@11-Dec.12:05.1256"
  ```

  cmd-context **diffcr –flat bgrs bgrs@@11-Dec.12:05.1956**

  ```
  < Reference Time 11-Dec-98.15:23:52, this audit started 11-Dec-98.15:23:59
  > Reference Time 11-Dec-98.12:02:39, this audit started 11-Dec-98.12:04:52
  < View was oxygen:/usr/jones/views/main.vws [uuid
  66e68edc.471511cd.ac55.08:00:2b:33:ec:ab]
  > View was oxygen:/usr/jones/views/r1_fix.vws [uuid
  8b468fd0.471511cd.aca5.08:00:2b:33:ec:ab]
  ----------------------------
  ----------------------------
  MVFS objects:
  ----------------------------
  < /vobs/docaux/bgr/sun4/bgrs@@11-Dec.15:24.1987
  > /vobs/docaux/bgr/sun4/bgrs@@11-Dec.12:05.1956
  ----------------------------
  < /vobs/docaux/bgr/sun4/bugs.o@@11-Dec.15:23.1981
  > /vobs/docaux/bgr/sun4/bugs.o@@11-Dec.12:03.1902
  ----------------------------
  < /vobs/docaux/bgr/sun4/bugsched.o@@11-Dec.15:23.1984
  > /vobs/docaux/bgr/sun4/bugsched.o@@11-Dec.12:04.1953
  ```

  The comparison shows that the builds used different versions of the object modules **bugs.o** and **bugsched.o**.

- Compare the same two derived objects again, this time including the CRs of all subtargets.

  cmd-context **diffcr -flat bgrs.exe bgrs.exe@@11-Dec.12:05.1956**

```
----------------------------
MVFS objects:
----------------------------
----------------------------
< First seen in target "bugs.obj"
< 1 \vob1\docaux\bgr\bugs.c@@\main\2 <11-Dec-98.15:22:53>
> First seen in target "bugs.obj"
> 1 \vob1\docaux\bgr\bugs.c@@\main\1 <19-Dec-97.11:49:54>
----------------------------
< First seen in target "bugsched.obj"
< 1 \vob1\docaux\bgr\bugsched.c@@\main\2 <11-Dec-98.15:23:04>
> First seen in target "bugsched.obj"
> 1 \vob1\docaux\bgr\bugsched.c@@\main\1 <19-Dec-97.11:50:07>
----------------------------
< First seen in target "bgrs.exe"
< 1 \vob1\docaux\bgr\bgrs.exe@@11-Dec.15:24.1987
> First seen in target "bgrs.exe"
> 1 \vob1\docaux\bgr\bgrs.exe@@11-Dec.12:05.1956
----------------------------
< First seen in target "bgrs.exe"
< 2 \vob1\docaux\bgr\bugs.obj@@11-Dec.15:23.1981
> First seen in target "bgrs.exe"
> 2 \vob1\docaux\bgr\bugs.obj@@11-Dec.12:03.1902
----------------------------
< First seen in target "bgrs.exe"
< 2 \vob1\docaux\bgr\bugsched.obj@@11-Dec.15:23.1984
> First seen in target "bgrs.exe"
> 2 \vob1\docaux\bgr\bugsched.obj@@11-Dec.12:04.1953
```

The integer at the beginning of an entry indicates the number of times the object was referenced during the build. The `first seen in target` message indicates the first target rebuild in which the object was referenced.

- For the same two derived objects as in the preceding examples, compare the file element versions used to build subtarget **bugsched.o**. Report the differences in short format.

  cmd-context **diffcr -short -select bugsched.o -type f -element_only \
  bgrs bgrs@@11-Dec.12:05.1956**

  ```
  ----------------------------
  < /vobs/docaux/bgr/bugsched.c@@/main/2
  > /vobs/docaux/bgr/bugsched.c@@/main/1
  ```

- Compare two builds of program **main**, listing only those entries that involve the files **src/prog.c**, **include/prog.h**, and **bin/prog.o**.

  cmd-context **diffcr -recurse -name 'prog.[cho]' main1 main2**

- Compare two builds of program **main**, listing only those entries that involve the file **src\prog.c**

  <u>*cmd-context*</u> **diffcr -recurse -name prog.c main1 main2**

SEE ALSO

catcr, **clearaudit**, **clearmake**, **ls**, **lsdo**, **make**, **rmdo**, **wildcards**, **wildcards_ccase**

dospace

Reports on VOB disk space used for shared derived objects

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- Report VOB disk space used for shared derived objects:

  dospace [**–upd·ate**] [**–sin·ce** *date-time*] [**–bef·ore** *date-time*] [**–ref·erences** {**0**,**1**,**n**}]
  [**–top** *number* | **–a·ll**] [**–siz·e** *size*] [**–reg·ion** *network-region*] [**–poo·l** *pool-name*] *vob-tag ...*

- Report disk space in raw format:

  dospace [**–upd·ate**] **–dum·p** [**–reg·ion** *network-region*] *vob-tag...*

- Generate and cache data on disk space used for local VOBs:

  dospace –gen·erate [**–scr·ub** *days*] [*vob-tag ...*]

DESCRIPTION

The **dospace** command displays data on VOB disk space used for shared derived objects. The report shows which views refer to shared DOs and shows how much disk space is used by the shared DOs that each view refers to.

You can use this information to identify views that should no longer refer to the DOs. Removing references to the DOs in those views can allow the disk space used by the DOs in the VOB to be reclaimed. When a DO no longer appears in any view, the **scrubber** utility, usually run as a periodically scheduled job, can remove the DO and its associated storage from the VOB.

The **dospace** command has a number of options that determine the range of derived objects and views it reports. By default, the command displays total disk space used for DOs in the specified VOBs and then lists the top 10 views in order of disk space used by DOs that appear in those views.

By default, **dospace** uses previously generated, cached data. The **–update** option generates fresh data and updates the cache before displaying the report.

The **–generate** option is intended for use by scheduled jobs. By default, the ClearCase scheduler periodically runs **dospace** with the **–generate** option to generate and cache data on disk space used by derived objects for all local VOBs. See the **schedule** reference page for information on describing and changing scheduled jobs.

RESTRICTIONS

Identities: For **–generate**, you must have one of the following identities:

- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)

Locks: No locks apply.

Mastership: (Replicated VOBs only) No mastership restrictions.

Other: With **–update**, you must have **Change** or **Full** access in the ClearCase scheduler ACL on the host where each VOB storage directory resides. See the **schedule** reference page.

OPTIONS AND ARGUMENTS

SPECIFYING CREATION TIMES OF DOS IN THE REPORT. *Default:* No restrictions.

–sin·ce *date-time*
Restricts the report to DOs that were last accessed at or after *date-time*. For the format of *date-time*, see the **lshistory** reference page.

–bef·ore *date-time*
Restricts the report to DOs that were last accessed before *date-time*. For the format of *date-time*, see the **lshistory** reference page.

SPECIFYING REFERENCE COUNTS OF DOS IN THE REPORT. *Default:* Reports disk space used by DOs with reference counts greater than or equal to 1.

–ref·erences {**0**,**1**,**n**}
Restricts the report to DOs with the specified reference counts. The reference count is the number of views in which a DO appears. A value of **n** reports on DOs that appear in 2 or more views. Each time the DO is winked in to another view, its reference count is incremented. Each time the DO is deleted from a view, its reference count is decremented.

SPECIFYING NUMBER OF VIEWS IN THE REPORT. *Default:* Lists top 10 views in space used.

–top *number* | **–a·ll**
Restricts the report to the top *number* of views (default 10) in amount of VOB disk space

used by DOs that appear in those views. The **–all** option reports all views that have references to DOs in the specified VOBs, subject to any restrictions imposed by other options.

SPECIFYING MINIMUM DISK SPACE USED BY EACH VIEW IN THE REPORT. *Default:* No restrictions.

–siz·e *size*
Restricts the report to those views each of which accounts for at least *size* kilobytes of VOB disk space used by DOs that appear in that view.

SPECIFYING THE STORAGE POOL. *Default:* All derived object pools.

–poo·l *pool-name*
Restricts disk space data in the report to the derived object storage pool whose name is *pool-name*.

SPECIFYING THE VOB. *Default:* For the **–generate** option, all local VOBs. Otherwise, no default; you must specify a VOB-tag.

vob-tag ...
One or more VOB-tags specifying the VOBs to report. Each VOB-tag must be valid in the region specified by **–region**.

–reg·ion *network-region*
Specifies the network region in which each *vob-tag* resides. The default is the region of the local host.

REPORTING RAW DATA. *Default:* Formatted report data.

–dum·p
Reports data in a raw form, with no filtering options (such as **–since** or **–size**) applied. This form is intended for use by user-created scripts or other programs that do advanced data analysis or formatting.

DISPLAYING AND CACHING UP-TO-DATE DATA. *Default:* Uses cached data.

–upd·ate
Computes and caches data on DO disk space usage at the time the command is issued, instead of using cached data, and then displays a report. The computation can take a significant amount of time.

GENERATING, CACHING, AND SCRUBBING DATA. *Default:* None.

–gen·erate
Computes and caches data on DO disk space used at the time the command is issued but does not display a report. The VOB storage directories for all specified VOBs must reside on the local host. If no *vob-tag* argument is specified, the command generates data for all

VOBs on the local host. The computation can take a significant amount of time. This option is intended to be used by periodic jobs run by the ClearCase scheduler.

–scr·ub *days*

Deletes cached records of data on DO disk space used that are older than the specified number of *days*. A value of **–1** deletes all cached records other than the one generated by the current invocation of the command, if any. This option is intended to be used in conjunction with the **–generate** option by periodic jobs run by the ClearCase scheduler. The default scheduled job specifies a value of **–1** for the **–scrub** option.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Report the top 10 views in disk space used by DOs each of which appears in only one view.

  cmd-context **dospace –references 1 /projects/bigapp**

  ```
  K Bytes Date.Time          VOB
   141284 25-Jul-99.01:51:53 /projects/bigapp
  K Bytes View
    11026 dave
     7722 sue_bl4
     7500 cindy_v3.2.1
     7224 release
     6263 jack_build
     5159 terry_v4
     4871 v3.2.1.win -region rd_windows
     4855 birdseye
     4268 v3.2.win -region rd_windows
     3903 andrea_40
  ```

- Report disk space used by DOs last accessed before June 1, 1999, and list all views that account for at least 3 MB each of DO space.

  cmd-context **dospace –before 01-Jun-1999 –all –size 3000 \bigapp**

```
Using 02-May-99.01:45:06 ("3 months") for before date.time.
      01-Jun-99.01:45:06 ("2 months") is next before bucket.
 K Bytes Date.Time         VOB
 159562 01-Aug-99.01:51:53 \bigapp
 K Bytes View
   28734 v3.1.1_pfm.re
   28734 v3.1.1_pfm.bld
   19208 v3.1.1.bld
   19197 v3.1.1.re
   14169 dave
    7501 cindy_v3.2.1
    7231 release
    4866 birdseye
    4102 v3.2.win
    3987 kim_v3.2
    3714 v3.2_pfm.bld
    3587 sue_mtypes
    3538 proj_v3.nt
    3386 [uuid: f532a8f2.afb711d2.a4e2.00:01:72:31:df:c2]
           machine1:c:\users\jack\jack_v3.3
```

- Generate and cache data on disk space used by DOs in a VOB and then report the top 10 views in disk space used.

  cmd-context **dospace –update /projects/bigapp**

```
Job is running on remote host ("server1"), waiting for it to finish.
..........
Job completed successfully on remote host ("server1").
 K Bytes Date.Time          VOB
  371523 01-Aug-99.01:45:06 /projects/bigapp
 K Bytes View
   30428 v3.2.1.bld
   30329 v3.1.1_rev.bld
   30329 v3.1.1_rev.re
   29986 joe_v4.0_merge
   29905 v3.2.1.re
   29888 v3.2_rev.bld
   28152 v3.2_rev.re
   26549 winapp_v4.0.bl4.nt -region rd_windows
   25670 v3.2.1.bld
   25663 v3.2.1.re
```

- Report disk space used by DOs in raw format.

  cmd-context **dospace –dump \smallapp**

```
start time: 932879789
run time: 14
bucket 0: 86400 1 day
bucket 1: 259200 3 days
bucket 2: 604800 1 week
bucket 3: 1209600 2 weeks
bucket 4: 2592000 1 month
bucket 5: 5270400 2 months
bucket 6: 7862400 3 months
bucket 7: 10540800 4 months
bucket 8: 13132800 5 months
bucket 9: 15811200 6 months
bucket 10: 23673600 9 months
bucket 11: 31536000 1 year
bucket 12: 47347200 1.5 years
bucket 13: 63158400 2 years
bucket 14: 94694400 3 years
bucket 15: 0 more than 3 years
pool: ddft 73e0001b.747d11cb.a0ea.08:00:09:25:75:d8
all: 0 1 0 0 0 0 0 0 92321 0 0 0 0 0 0 0 0 0
all: 0 -2 0 0 0 96198 0 0 0 27647 0 0 0 0 0 0 0 0
view: 59f62178.580511d2.af73.00:01:80:9a:19:fe machine2
  C:\views\app2.1_win.vws
view: 0 -2 0 0 0 96198 0 0 0 27647 0 0 0 0 0 0 0 0
view: 8054b303.63ba11d2.bcef.00:01:80:90:ae:6d machine8
  D:\users\mary\vws\mary_test.vws
view: 0 -2 0 0 0 96198 0 0 0 27647 0 0 0 0 0 0 0 0
view: e7ca6e7a.990811d2.a1dc.00:01:80:85:f3:f5 machine4
  C:\views\jeff\jeff_build.vws
view: 0 -2 0 0 0 96198 0 0 0 0 0 0 0 0 0 0 0 0
view: ac6db1f1.bd4811d2.b314.00:01:80:a2:f2:c3 machine7
  D:\users\sue\views\app_axp.vws
view: 0 1 0 0 0 0 0 0 92321 0 0 0 0 0 0 0 0 0
```

SEE ALSO

schedule, **space**, *Building Software*

edcs

Edits the config spec of a view

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and Attache—Edit the config spec of a dynamic view:

  edcs [**–tag** *view-tag*] [*file*]

- ClearCase, ClearCase LT, and Attache—Edit the config spec of a snapshot view:

  edcs [*file*]

DESCRIPTION

For information on specifying config spec content, see the **config_spec** reference page. As a rule, you never edit a UCM config spec; see the **config_spec** reference page for exceptions to this rule.

This command does not require a product license.

General Information—The Edit Session

The **edcs** command revises a view's config spec by invoking a text editor on an existing config spec (ClearCase and ClearCase LT) or by downloading a config spec into a temporary file and invoking a text editor on an existing config spec (Attache). The config spec can be one of the following:

- The view's current config spec
- A text file that you want to edit and then make the view's config spec. (If you don't need to edit the file, use **setcs**.)

ClearCase on UNIX—The Edit Session

In ClearCase, if the working directory view differs from the set view (established by the **setview** command), **edcs** displays a warning message and uses the working directory view. The text editor invoked by **edcs** is specified by the environment variable WINEDITOR (first choice), VISUAL (second choice), or EDITOR (third choice). If none of these EVs is set, **vi(1)** is invoked.

When you invoke **edcs** without specifying the *file* argument, ClearCase and ClearCase LT run the text editor from the **/tmp** directory. When you end the editing session and set the config spec, the config spec is stored in the view storage directory. However, if you perform other operations from within the text editor, ClearCase and ClearCase LT execute those operations from **/tmp**, not from the directory where you invoked **edcs**. For example, in a **vi** editing session, the command **:w prev-cs** copies the contents of the config spec to a file named **prev-cs** and stores **prev-cs** in **/tmp**.

ClearCase and ClearCase LT on Windows—The Edit Session

In ClearCase and ClearCase LT, the text editor invoked by **edcs** is specified by the VISUAL environment variable; if this EV is not set, **edcs** uses the value of the EDITOR environment variable. If neither EV is set, **edcs** invokes the **Notepad**.

After the Edit Session

At the end of the edit session, there is a confirmation step. For dynamic views, this prompt is:

`Set config spec for view "`*view-tag*`"? [yes]`

For snapshot views, this prompt is:

`Set config spec and load snapshot view "`*view-tag*`"? [yes]`

If you answer **yes**:

- In ClearCase and ClearCase LT, the modified config spec is set as the view's config spec. (In a snapshot view, there is an additional confirmation step if the edits to the config spec cause elements to be unloaded from the snapshot view.)

  NOTE: In a snapshot view, setting the config spec initiates an **update -noverwrite** operation and generates an update logfile with the default name and location (see the **update** reference page for information on this logfile). To execute this command on the command line, you must be in or under the root directory of the snapshot view.

- In Attache, the changed file is uploaded and a remote **setcs** command is issued.

If you answer **no**, the command is canceled; the view retains its current config spec.

ClearCase, ClearCase LT, and Attache on UNIX—Export View Config Specs

If you modify the config spec of a view that is being exported for non-ClearCase access, make sure that all users who may currently have the view mounted for that purpose unmount and remount the view. Unmounting and remounting the view ensures access to the correct set of files as specified in the updated config spec.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE VIEW. *Default:* Edits and sets a config spec for the current view.

–tag *view-tag*

The view-tag of any dynamic view; the view need not be active.

NOTE: To edit the config spec of a snapshot view, you must be in that view. However, in a snapshot view, you can use this option to edit the config spec of a dynamic view.

SPECIFYING THE CONFIG SPEC FILE. *Default:* Edits the view's current config spec.

file

The pathname of a file to be used as input to the edit session.

- (ClearCase and ClearCase LT) If the file does not exist, **edcs** creates it.
- (Attache) If the file does not exist locally, **edcs** downloads it if it exists remotely, or creates it locally if it does not exist. The named file is saved both locally and remotely. Both the local and remote temporary files are deleted after the file has been uploaded.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Edit the config spec of the current view.

  cmd-context **edcs**

- Edit the config spec of the dynamic view with the view-tag **jackson_fix**.

  cmd-context **edcs –tag jackson_fix**

- Use a text file named **cspec_rel3** as input to an edit session, producing a new config spec for the current view.

  cmd-context **edcs cspec_rel3**

SEE ALSO

attache_command_line_interface, **attache_graphical_interface**, **catcs**, **config_spec**, **lsview**, **mktag**, **setcs**, **setview**, **update**

endview

Deactivates a view

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

endview [**–ser·ver**] *view-tag*

DESCRIPTION

Deactivates the specified view. The exact behavior varies according to the kind of view.

We recommend against deactivating a view for the purpose of backing it up.

Dynamic View

The **endview** command deactivates the dynamic view. It removes all references to the view from the MVFS on the current host. The *view-tag* disappears from the MVFS directory (on Windows this directory is, by default, **M:**\). The **–server** option terminates the view's **view_server** process. Without **–server**, **endview** does not affect the view's availability from computers other than the current one. For information about **view_server**, see the *Administrator's Guide*.

If a Windows drive was assigned to the view, the drive is marked `unavailable` in the drive table (see the Windows **net use** command) and can be reused.

In a mixed UNIX/Windows environment, any NFS drive that was mounted to support access to the view storage directory gets unmounted (assuming no other active views or VOBs require it).

In Attache, if you execute **endview** for a view associated with the current workspace, the command fails with the message

```
Cannot stop the view associated with the current workspace.
```

CAUTION: Processes set to or associated with a view are stranded if you deactivate that view without exiting the processes. This can cause MVFS activities to fail. To recover from this situation, use **startview** or **setview** to restart the view on all computers that were using it, or kill the processes manually with the UNIX **kill** command or the Windows Task Manager. To avoid this situation, follow these guidelines:

- Before running **endview** (without **–server**) on your computer, exit all processes on your computer that are set to or associated with the view. This includes any processes started by other users. If the processes are running when you deactivate the view, they will be stranded.
- Before running **endview –server**, exit all processes set to or associated with the view. This includes processes on other computers and/or processes started by other users.

Snapshot View

When issued with the **-server** option, **endview** ends the view's **view_server** process on the host where the view-storage directory resides. Any ClearCase or ClearCase LT command issued from the view-storage directory restarts the snapshot view's **view_server** process.

endview used without the **-server** option has no effect on a snapshot view.

RESTRICTIONS

With the **–server** option, you must have permission to modify the view.

OPTIONS AND ARGUMENTS

STOPPING THE VIEW SERVER. *Default:* **endview** does not stop the view's **view_server** process; the view remains accessible from other network hosts.

–ser·ver

Terminates the view's **view_server** process.

SPECIFYING THE VIEW TO DEACTIVATE. *Default:* None. You must supply a view-tag.

view-tag

Deactivates view *view-tag* if *view-tag* specifies a dynamic view. If *view-tag* specifies a snapshot view, **-server** must also be specified to deactivate the view.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Deactivate the dynamic view **r3_main** on the local host. Do not terminate the **view_server** process.

  cmd-context **endview r3_main**

- Same as previous example, but terminate the **view_server** process, making the view temporarily unavailable to all hosts.

  cmd-context **endview –server r3_main**

- Stop and restart the dynamic view **alh_main**, which is currently assigned to drive **w:**.

  cmd-context **endview alh_main**

  cmd-context **startview alh_main**

  net use

  ```
  ...
  Unavailable W:    \\view\alh_main       ClearCase Dynamic Views
  ...
  ```

  net use w: \\view\alh_main

  ```
  Command completed successfully.
  ```

- Deactivate the snapshot view r4_main.

  cmd-context **endview –server r4_main**

SEE ALSO

startview, *Administrator's Guide*

env_ccase

Environment variables

APPLICABILITY

Product	Command Type
ClearCase	general information
ClearCase LT	general information
MultiSite	general information
Attache	general information

Platform
UNIX
Windows

DESCRIPTION

This reference page describes the environment variables (EVs) used by ClearCase, ClearCase LT, MultiSite, and Attache commands, programs, utilities, and software installation scripts. It also describes standard UNIX EVs that are particularly important for ClearCase, ClearCase LT, MultiSite, and Attache. Descriptions of the EVs are organized as follows:

- EVs common to UNIX and Windows
- UNIX-only EVs
- Windows-only EVs

NOTE: This reference page does not describe all environment variables. Omitted are the EVs used by triggers and by the **find** commands; see the **mktrtype**, **find**, and **findmerge** reference pages for descriptions.

Environment Variables Common to UNIX and Windows

CCASE_ABE_PN (or **CLEARCASE_ABE_PN**)

The full pathname with which **clearmake** invokes the audited build executor (**abe**) on a local or remote host during a parallel build. For more information about **abe**, see *Building Software*.

Default: ccase-home-dir/**bin/abe**

CCASE_AUDIT_TMPDIR (or **CLEARCASE_BLD_AUDIT_TMPDIR**)

Sets the directory where **clearmake** and **clearaudit** create temporary build audit files. If this variable is not set or is set to an empty value, **clearmake** creates these files in the directory specified by the **TMPDIR** (UNIX) or **TMP** (Windows) environment variable.

NOTE: On UNIX systems, **clearmake** creates these files in the **/tmp** directory if neither EV is set.

All temporary files are deleted when **clearmake** exits. If the value of **CCASE_AUDIT_TMPDIR** is a directory under a VOB-tag, **clearmake** prints an error message and exits.

NOTE: Multiple build clients can use a common directory for audit files. Names of audit files are unique because **clearmake** names them using both the PID of the **clearmake** process and the host name of the machine on which the process is running.

Default for UNIX: **/tmp**

Default for Windows: None

CCASE_AUTO_DO_CI

Checks in DOs checked out by **clearmake -c** or **omake -C** unless the build of the corresponding target fails or the automatic checkout of the DO or a sibling DO fails. Checkout comments are preserved. The **checkin** is invoked with the **-ptime** option to preserve the DO's modification time.

Default: Undefined

CCASE_BLD_HOSTS

Specifies one or more build hosts on which **clearmake** must build targets. For more information, see *Building Software*.

Default: Undefined.

CCASE_BLD_NOWAIT

Turns off **clearmake**'s sleep-check cycle during a build. When this environment variable is set, **clearmake** does not check for a VOB-lock (or wait for the VOB to be unlocked).

CCASE_BLD_VOBS

A list of VOB-tags—separated with a space, tab, comma, colon (UNIX), or semicolon (Windows)—to be checked for lock status during a build. If a VOB on this list is locked, **clearmake** goes into a sleep-check cycle.

CCASE_CONC (or **CLEARCASE_BLD_CONC**)

Sets the concurrency level in a **clearmake** build. This EV takes the same values as the **–J** option. Specifying a **–J** option on the **clearmake** command line overrides the setting of this EV.

Default: None.

CCASE_DNVW_RETRY

Specifies time-out period, in minutes, for **clearmake**, **omake**, **catcr**, **describe**, or **lsdo** to wait before trying to contact an inaccessible view listed in its cache. To disable the cache, set **CCASE_DNVW_RETRY** to 0. For more information, see the **clearmake** or **omake** reference pages.

Default: 60 minutes.

CCASE_HOST_TYPE (or **CLEARCASE_BLD_HOST_TYPE**)

Determines the name of the build hosts file to be used during a parallel build (**–J** option): file **.bldhost.$CCASE_HOST_TYPE** in your home directory. (Your home directory is determined by examining the password database.) See the **clearmake** reference page. For information about **bldhost**, see *Building Software*.

Specifying a **–B** option on the command line overrides the setting of this EV.

C Shell Users: Set this EV in your **.cshrc** file, not in your **.login** file. The parallel build facility invokes a remote shell, which does not read the **.login** file.

CCASE_HOST_TYPE can also be coded as a make macro.

Default: None.

CCASE_MAKE_CFG_DIR (or **CLEARCASE_MAKE_CONFIG_DIR**)

In a makefile read by **clearmake**, expands to the full pathname of the **clearmake** configuration directory in the ClearCase installation area—typically *ccase-home-dir***/config/clearmake** (UNIX) or *ccase-home-dir***\config\clearmake** (Windows).

CCASE_MAKE_COMPAT (or **CLEARCASE_MAKE_COMPAT**)

Specifies one of **clearmake**'s compatibility modes. This EV takes the same values as **clearmake**'s **–C** option. Specifying **–C** on the command line overrides the setting of this EV.

Default: None.

CCASE_MAKEFLAGS

Provides an alternative or supplementary mechanism for specifying **clearmake** command options. **CCASE_MAKEFLAGS** can contain the same string of key letters used for command-line options, except that options that take arguments are not allowed. Options on the **clearmake** command line override the setting of this environment variable if there is a conflict.

clearmake uses either **CCASE_MAKEFLAGS** or **MAKEFLAGS**, but not both. If **CCASE_MAKEFLAGS** is set, **clearmake** uses it. If **CCASE_MAKEFLAGS** is not set, **clearmake** looks for **MAKEFLAGS**.

NOTE: CCASE_MAKEFLAGS is useful if you use multiple **make** programs. In this case, putting options that are specific to **clearmake** in the **MAKEFLAGS** environment variable causes problems for the other **make** programs.

Default: None.

CCASE_OPTS_SPECS (or **CLEARCASE_BLD_OPTIONS_SPECS**)
A list of pathnames—separated by colons (UNIX) or semicolons (Windows)— each of which specifies a BOS file to be read by **clearmake**. You can use this EV instead of specifying BOS files on the **clearmake** command line with one or more **–A** options.

Default: Undefined.

CCASE_SHELL_FLAGS (or **CLEARCASE_BLD_SHELL_FLAGS**)
Specifies **clearmake** command options to be passed to the subshell program that executes a build script command.

Default for UNIX: **–e**

Default for Windows: None

CCASE_SHELL_REQUIRED
Forces **clearmake** to execute build scripts in the shell program you specify with the **SHELL** macro. To make **clearmake** execute builds scripts in the shell program, set this EV to **TRUE**. To allow **clearmake** to execute build scripts directly, unset the EV.

Default: **clearmake** executes build scripts directly.

CCASE_VERBOSITY (or **CLEARCASE_BLD_VERBOSITY**)
An integer that specifies the **clearmake** message logging level, as follows:

`1`	Equivalent to **–v** on the command line
`2`	Equivalent to **–d** on the command line
`0` or `undefined`	Equivalent to standard message logging level

If you also specify **–v** or **–d** on the command line, the higher value prevails.

Default: `0`

CCASE_WINKIN_VIEWS
A list of whitespace-separated view tags. If this environment variable is set, **clearmake** winks in only derived objects that were built in one of the specified views.

CLEARAUDIT_SHELL
The program that **clearaudit** runs in an audited shell. You must set this environment variable to the program's full pathname; for example:

/bin/csh or **/usr/home/myscript** (UNIX)

\windows\system32\cmd.exe or **\users\anne\bin\myscript** (Windows)

Default on UNIX: **clearaudit** runs the program specified by the **SHELL** environment variable or, if **SHELL** is undefined, a Bourne shell (**/bin/sh**).

Default on Windows: **clearaudit** runs the program specified by the **COMSPEC** environment variable or, if **COMSPEC** is undefined, **cmd.exe**.

See also: **SHELL**.

CLEARCASE_AVOBS

A list of VOBs to process when you use the **–avobs** option in the **find**, **findmerge**, **lscheckout**, **lshistory**, or **rmview** commands. If this EV is not set, specifying **–avobs** invokes the command on all VOBs mounted on the host. If there are many such VOBs, the command can take a long time to complete.

Specify **CLEARCASE_AVOBS** as a list of VOB-tags separated by, commas, white-space, colons (UNIX), or semicolons (Windows).

Default: None.

CLEARCASE_CMNT_PN

The pathname of the file in which **cleartool** and **multitool** cache the most recent user-supplied comment. Defining/removing this EV enables/disables comment caching.

CLEARCASE_OBSO_SYN

Detects instances of the obsolete option-argument style of specifying an object (see the **cleartool** reference page).

If you set this environment variable to the value `WARN`, it issues warnings when it detects obsolete syntax. When set to `SILENT`, it silently accepts obsolete syntax. When set to `FAIL`, it issues errors when it detects obsolete syntax.

Default: `FAIL`.

CLEARCASE_PERLLIB

Set to 1 to force **ccperl** to ignore the **PERLLIB** environment variable if it is set.

Default: Undefined. **ccperl** will look for **perl** libraries in the directory specified by the **PERLLIB** environment variable if it is set. Otherwise, it will look for **perl** libraries in *ccase-home-dir*/*platform***/lib/perl5**, where *platform* is a ClearCase platform mnemonic such as **sun5**.

CLEARCASE_PROFILE

The file containing your ClearCase or ClearCase LT user profile, which includes rules that determine the comment option default for one or more **cleartool** and **multitool** commands. This setting must be a full pathname.

Default: For ClearCase and ClearCase LT, **.clearcase_profile** in your home directory (in Attache, on your helper host)

CLEARCASE_TAB_SIZE

Specifies the tab width for output produced by **cleardiff**, **xcleardiff**, and source lines listed by the **annotate** command.

Default: `8`

CLEARCASE_TRACE_TRIGGERS

A flag variable: if defined with a nonzero value, it causes all triggers to behave when they fire as if they were defined with the **–print** option. See the **mktrtype** reference page.

Default: Undefined.

CLEARCASE_VOBLOCKWAIT

(MultiSite only) Specifies the number of minutes for **syncreplica** to keep retrying exports or imports when the VOB is locked. During that time, **syncreplica** retries the write operation every minute. If the time elapses and the VOB is still locked, **syncreplica** exits with an error. For more information, see the **syncreplica** reference page in the *Administrator's Guide* for Rational ClearCase MultiSite.

EXPORT_REPLACE_CHAR

A character used by the **clearexport_*** utilities to replace invalid characters in exported label and branch names.

Default: **.** (period character)

EXPORT_REPLACE_COMM

A character string used in the data file created by **clearfsimport** as the comment for `create version` event records.

Default: `made from flat file`.

EXPORT_REPLACE_STRING

A character string used by the **clearexport_*** utilities to replace an invalid string in exported labels and branch names. This environment variable is used if the exporter cannot replace invalid characters with the **EXPORT_REPLACE_CHAR** EV.

Default: `REPLACED`

HOME

UNIX systems—Not used. ClearCase and ClearCase LT programs determine your home directory by reading the password database, not by using this environment variable.

Windows systems—User's home directory; must be set for ClearCase and ClearCase LT to work correctly. **HOME** is used to search for various resources, including file typing information (see also the **cc.magic** reference page). **HOME** must be a full pathname, including drive specification. For example, **C:\users\anne** is a legal value; **\users\anne** is not.

NOTE : Windows sets two variables, **HOMEDRIVE** and **HOMEPATH**, which combine to specify the current user's home directory as supplied by the Administrator when the user account was created. So, unless **HOMEDRIVE** and **HOMEPATH** have been reassigned, **HOME** can be set to %**HOMEDRIVE**%%**HOMEPATH**%.

MAGIC_PATH

A colon (UNIX) or semicolon (Windows) separated list of directories to be searched for magic files. Various ClearCase and ClearCase LT programs consult magic files to perform file-typing on file system objects. See the **cc.magic** reference page.

*Default for UNIX: home-directory***/.magic:$ATRIAHOME/config/magic**

*Default for Windows: home-directory***\.magic;***ccase-home-dir***\config\magic**

MAKEFLAGS

Provides an alternative (or supplementary) mechanism for specifying **clearmake** command options. **MAKEFLAGS** can contain the same string of keyletters used for command-line options, except that options that take arguments are not allowed. Options on the **clearmake** command line override the setting of this environment variable if there is a conflict.

See also the description of the **CCASE_MAKEFLAGS** environment variable.

Default: None.

SHELL

The default shell program to be run by various commands and programs, including the **shell** and **setview** commands. On UNIX, the **clearaudit** utility uses the value of **SHELL** if the environment variable **CLEARAUDIT_SHELL** is undefined.

Default for UNIX: Set by your shell program.

Default for Windows: Not set by most Windows shells; some shells that are ported from UNIX (for example, Hamilton **csh**, MKS korn **sh**, etc.) may set it.

TZ

Time zone for the host. If the **TZ** environment variable is set to a value different from the time maintained by the operating system, the **TZ** time rather than the system time is used. In this case, file creation and change dates can be in error, and config specs do not work as expected.

UNIX-Only Environment Variables

ATRIAHOME

Installation directory for ClearCase, ClearCase LT, and MultiSite software. Set this EV before running the **install_release** script to specify a nonstandard installation location. On such hosts, users' shell startup scripts should use **$ATRIAHOME/bin** to specify the pathname of the ClearCase, ClearCase LT, or MultiSite executables.

Default: **/usr/atria**

ATRIA_NO_BOLD

A flag variable: if defined with a nonzero value, it suppresses generation of bold characters in **cleartool** and **clearmake** output.

Default: Undefined.

BITMAP_PATH

Bitmap file search path. The icons that an **xclearcase** directory browser displays for file system objects are stored in bitmap files. It searches in directories on this colon-separated search path for such bitmap files. See the **cc.icon** reference page.

Default: home-directory/**.bitmaps:$ATRIAHOME/config/ui/bitmaps**
See also **ICON_PATH**.

CCASE_BLD_UMASK (or **CLEARCASE_BLD_UMASK**)

Sets the **umask**(1) value to be used for files created from a **clearmake** build script. It may be advisable to have this EV be more permissive than your standard umask—for example, **CCASE_BLD_UMASK** = `2` where umask = `22`. The reason to create DOs that are more accessible than other files is winkin: a winked-in file retains its original ownership and permissions. For example, when another user winks in a file that you originally built, the file is still owned by you, is still a member of your principal group, and still has the permissions with which you created it. You can use the standard **chmod** command to change the permissions of a DO after you create it, and these permissions remain in effect while the DO is unshared. However, for a shared DO, you may need to use the standard **chmod** and **protect –chmod** to set appropriate permissions.

If you are using a tool that ignores **umask** (and hence **CCASE_BLD_UMASK**) settings and you want winkins to work correctly, you have to use **chmod** on the file in your build script to give it write permissions if the tool creates the file without these permissions.

CLEARCASE_BLD_UMASK can also be coded as a make macro.

NOTE: If you want to use **CCASE_BLD_UMASK**, do not set your **umask** value in your shell startup file. If you set the **umask** value in your startup file, the **umask** value is reset to its original value when clearmake starts a shell to run the build script. Setting **CCASE_BLD_UMASK** in your startup file has no effect.

Default: Same as current umask.

CLEARCASE_DBG_GRP

Set this variable to a nonzero value to force **xclearcase** to print debugging information when executing button and menu commands in the graphical interface.

Default: None.

CLEARCASE_MSG_PROTO

Enables one-way message forwarding between ClearCase or ClearCase LT and an interprogram messaging system. This feature enables ClearCase and ClearCase LT to notify the messaging system that an operation succeeded (for example, a checkout) without going through an encapsulator. One-way message forwarding succeeds only if all programs involved have the same value for the **DISPLAY** environment variable.

For more information, see the **softbench_ccase** reference page.

Default: None. Supported values: `SoftBench`. See also: **DISPLAY, WINEDITOR**.

CLEARCASE_ROOT

(Set by the **setview** command; do not set this variable yourself.) The full pathname of the root directory of a set view process, which is a process created by the **setview** command. For example, the command **setview bugfix** creates a shell in which **CLEARCASE_ROOT** is set to `/view/bugfix`.

Default: Not set in a process that was not created by **setview**.

DISPLAY

The X Window System display to use for ClearCase and ClearCase LT GUI utilities (and all other X applications). If you are using an interprogram messaging system, all your tools must have the same **DISPLAY** value.

Default: Undefined.

EDITOR
VISUAL

The pathname of a text editor. The **edcs** subcommand invokes the editor specified by the environment variable **WINEDITOR** (first choice), or **VISUAL** (second choice), or **EDITOR** (third choice). **xclearcase** invokes the editor specified by the environment variable **WINEDITOR** (first choice) or **EDITOR** (second choice). See also: **WINEDITOR**.

Default: **vi**

GRP_PATH

A colon-separated list of files and directories to be searched for group files when you start **xclearcase**.

Default: home-directory/**.grp:***ccase-home-dir*/**config/ui/grp**

ICON_PATH

A colon-separated list of directories to be searched for icon files. **xclearcase** directory browsers use the bitmap images in such files as icons for file system objects. See the **cc.icon** reference page.

Default: home-directory/**.icon:$ATRIAHOME/config/ui/icon**
See also: **BITMAP_PATH**.

MANPATH

A colon-separated list of directories in which the UNIX **man(1)** command searches for reference pages. (The **cleartool man** and **multitool man** commands do not use **MANPATH**, but always search in **$ATRIAHOME/doc/man**.)

Default: Varies with operating system.

PATH

The standard UNIX program search path. To access ClearCase or ClearCase LT executables, change your search path to include directory **$ATRIAHOME/bin**.

Default: Set by your shell program; typically modified in shell startup script.

NOTE: Do not specify an MVFS path as a value for **PATH**.

SCHEMESEARCHPATH

A colon-separated list of directories to be searched for scheme files, which contain X Window System resource settings. See the **schemes** reference page.

Default: **/usr/lib/X11/%T/%N%:$ATRIAHOME/config/ui/%T/%N%S**

TERM

The kind of terminal for which output is to be prepared. Certain **cleartool** commands produce output that use special terminal capabilities. For example, **catcr** uses boldface to highlight information in a configuration record. To see bold characters in an **xterm**, set **TERM** to **xterm**, and provide a bold font with the X Toolkit option **–fb**, or with the X resource **xterm*boldFont**. To prevent the control characters that enable bold from appearing in an **emacs** shell, set **TERM** to **emacs** in your **emacs** startup script, or set **ATRIA_NO_BOLD**.

Default: None; typically set in shell startup script.

WINEDITOR

An X Window System text editor application (for example, **xedit(1)**), which is invoked by **xclearcase** on a browser item. If **WINEDITOR** is undefined, **xclearcase** creates a terminal window, and runs the program specified by the **EDITOR** environment variable. If neither of these variables are defined, no editor is invoked.

Default: None.

Windows-Only Environment Variables

ATRIAHOME (environment variable)

Installation directory for ClearCase, ClearCase LT, and MultiSite; same as Windows Registry value **HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\ProductHome**. This EV is set by the **albd_server** when it runs a **schedule** request. For more information, see the **schedule** reference page. For information about **albd_server**, see the *Administrator's Guide*.

NOTE: You can create **ATRIAHOME** as a user or system environment variable, but the **albd_server** will overwrite it when it runs a **schedule** request.

Default: Directory in which you installed ClearCase or ClearCase LT (the installation default is **c:\Program Files\Rational\ClearCase**)

CLEARCASE_PRIMARY_GROUP

Specifies which of the user's groups ClearCase should consider the primary group. Overrides the Windows primary group assignment for ClearCase operations. This environment variable must be a per-user variable (not a system variable) and must be set to a group that already appears in the user's group list.

CLEARCASE_GROUPS

Specifies a list of up to 32 groups that ClearCase will consider first when determining or displaying which groups a user belongs to. Users who are members of more than 32 groups can set this environment variable to designate a subset of those groups that ClearCase will consider when evaluating the user's group membership. The value of this variable is a semicolon-separated list of groups to which the user belongs. The list should not include the group that is specified in the environment variable **CLEARCASE_PRIMARY_GROUP.**

CMAKE_PNAME_SEP

Sets the pathname separator for pathnames constructed by **clearmake**. This variable can be set in the makefile, in a BOS file, on the command line, or as an environment variable.

Default: If this variable is not set or is set to any other value than a slash (**/**) or a backslash (****), **clearmake** uses a backslash (\) as the pathname separator.

COMSPEC

The default command shell program to be run by various ClearCase or ClearCase LT commands and programs, including the **shell** command, and the **clearaudit** utility (if the environment variable **CLEARAUDIT_SHELL** is undefined).

Default: Set by Windows to **cmd.exe**

EDITOR
VISUAL

The pathname of a text editor. The **edcs** subcommand invokes the editor specified by the environment variable **VISUAL** (first choice), or **EDITOR** (second choice). If the pathname contains spaces, you enclose the pathname in quotes. For example:

```
"c:\Program Files\tools\editor.exe"
```

Default: **notepad**

PATH

The standard executable program search path. The Install Wizard adds the directory *ccase-home-dir***bin** to your search path.

Default: Set from the system and user path values in the Windows registry.

NOTE: Do not specify an MVFS path as a value for **PATH**.

ProductHome (Windows NT Registry key value)

Installation directory for ClearCase and ClearCase LT software. This value is stored in the Windows registry key **HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion**.

Default: **C:\Program Files\Rational\ClearCase**

SEE ALSO

mktrtype, **find**, and **findmerge** for information about other environment variables

events_ccase

Operations and event records

APPLICABILITY

Product	Command Type
ClearCase	general information
ClearCase LT	general information
Attache	general information
MultiSite	general information

Platform
UNIX
Windows

DESCRIPTION

Nearly every operation that modifies the VOB creates an event record in the VOB database. For example, if you create a new element, attach a version label, or lock the VOB, an event record marks the change.

Event records are attached to specific objects in VOB databases. Thus, each object (including the VOB object itself) accumulates a chronological event history, which you can display with the command **lshistory**.

In addition, you can do the following:

- Customize event history reports with **lshistory –fmt**; see the **fmt_ccase** reference page.
- Scrub minor event records from the VOB database to save space; see the **vob_scrubber** reference page.
- Assign triggers to many event-causing operations (**mkelem**, **checkout**, and **mklabel**, for example); see the **mktrtype** reference page.
- Change the comment stored with an event; see the **chevent** reference page.

Contents of an Event Record

An event record stores information for various operations:

obj-name	The object(s) affected
obj-kind	The kind of object (file element, branch, or label type, for example)
user-name	The user who changed the VOB database
host-name	The client host from which the VOB database was changed
operation	The operation that caused the event (usually a **cleartool** command like **checkout** or **mklabel**)
date-time	When the operation occurred (reported relative to the local time zone)
event-kind	A description of the event, derived from a combination of the `operation` and `obj-kind` fields
comment	A text string that is generated by ClearCase or ClearCase LT, provided by *user-name*, or a combination of both

VOB Objects and Event Histories

The following kinds of VOB-database objects have event histories, which you can display with **lshistory**:

VOB
VOB storage pool
Element
Branch
Version
VOB symbolic link
Hyperlink
Derived Object (no creation event)
Replica
Type
- Attribute type
- Branch type
- Element type
- Hyperlink type
- Label type
- Trigger type

Each time an object from any of these categories is created, it begins its own event history with a creation event. (Derived objects are an exception; ClearCase stores a DO's creation time in its config record, not in an event record.) As time passes, some objects—VOBs and elements, in particular—can accumulate lengthy event histories.

Do not confuse type objects (created with **mkattype**, **mkbrtype**, **mkeltype**, **mkhltype**, **mklbtype**, and **mktrtype**) with the instances of those types (created with **mkattr**, **mkbranch**, **mkelem**, **mkhlink**, **mklabel**, and **mktrigger**). The type objects are VOB-database objects, with their own event histories. Individual branches, elements, and hyperlinks are also VOB-database objects.

However, individual attributes, labels, and triggers are not VOB-database objects and, therefore, do not have their own event histories. Their create and delete events (**mkattr**/**rmattr**, **mklabel**/**rmlabel**, and **mktrigger**/**rmtrigger**) are recorded on the objects to which these metadata items are attached.

Operations that Cause Event Records to be Written

The following kinds of operations cause event records to be written to the VOB database:

- Create or import a new object.
- Destroy (remove) an object.
- Check out a branch.
- Modify or delete version data.
- Modify a directory version's list of names.
- Attach or remove an attribute, label, hyperlink, or trigger.
- Lock or unlock an object.
- Change the name or definition of a type or storage pool.
- Change a branch or element's type.
- Change an element's storage pool.
- Change the protections for an element or derived object.

Table 7 lists event-causing operations as you may see them in **lshistory** output that has been formatted with the **–fmt** option's **%o** (operation) specifier. Note that most operations correspond exactly to **cleartool** subcommands.

Key to Table 7

Symbol	**Meaning**
M	Causes a minor event (see **lshistory –minor**)
T	Can have a trigger (see **mktrtype**)
S	Resulting event records can be scrubbed (see **vob_scrubber**)
C	Generates a comment (see the **comments** reference page)

Table 7 Operations That Generate Event Records

Operation that Generates the Event Record	Notes (see key above)				Commands that Always Cause the Operation	Commands that May Cause the Operation	Object to Which Event Record Is Attached
checkin		T			**checkin**, **mkelem**, **mkbranch**	**clearimport**, **relocate**	Newly created version
checkout		T			**checkout**	**clearimport**, **findmerge**, **mkelem**, **mkbranch**, **relocate**	Checked-out branch (event deleted automatically at checkin or uncheckout)
chmaster		T		C	**chmaster**, **reqmaster** (**reqmaster** is not triggerable)		Object whose mastership was changed
chpool	M		S	C	**chpool**		Element
chtype	M	T	S	C	**chtype**		Element or branch
exportsync				C	**syncreplica –export**		Replica
import						**clearimport**	Imported element or type
importsync				C	**syncreplica –import**		Replica
lnname	M	T	S	C	**ln**, **ln –s**, **mkelem**, **mkdir**, **mv**	**relocate**	Directory version
lock		T	S	C	**lock**	(Various)	Locked object (type, pool, VOB, element, or branch)
mkattr	M	T	S	C	**mkattr**	**clearimport**, **mkhlink**, **relocate**	Element, branch, version, hlink, or VOB symlink
mkbranch		T			**mkbranch**, **mkelem**	**checkout**, **clearimport**, **relocate**	New branch
mkelem		T		C	**mkelem**, **mkdir**	**clearimport**, **relocate**	New element

Table 7 Operations That Generate Event Records

Operation that Generates the Event Record	Notes (see key above)				Commands that Always Cause the Operation	Commands that May Cause the Operation	Object to Which Event Record Is Attached
mkhlink	M	T	S	C	**mkhlink**	**clearimport**, **findmerge**, **merge**, **relocate**	Hyperlink object and from-object, and for bidirectional hyperlinks, to-object (unless cross-VOB hyperlink)
mklabel	M	T	S	C	**mklabel**	**clearimport**, **relocate**	Version
mkpool					**mkpool**		Storage pool object
mkreplica					**mkreplica**		Replica
mkslink		T			**ln –s**	**clearimport**, **relocate**	Directory version
mktrigger	M	T	S		**mktrigger**	**relocate**	Element
mktype		T			**mk**type**	**clearimport**, **relocate**	Newly created type object
mkvob					**mkvob** (causes numerous creation events), **mkreplica –import**		VOB
modpool	M		S	C	**mkpool –update**		Storage pool
modtype	M		S	C	**mk**type –replace**		Type object
protect	M		S	C	**protect**		Element or DO
reconstruct	M		S			**checkvob –fix**	Element
reformatvob					**reformatvob**		VOB
rename (pool)	M			C	**rename**		Storage pool
rename (type)	M	T		C	**rename**		Type object
reserve	M	T			**reserve**		Checked-out version

Table 7 Operations That Generate Event Records

Operation that Generates the Event Record	Notes (see key above)				Commands that Always Cause the Operation	Commands that May Cause the Operation	Object to Which Event Record Is Attached
rmattr	M	T	S		**rmattr**		(See **mkattr**)
rmbranch		T	S	C	**rmbranch**		Parent branch
rmelem		T	S	C	**rmelem**	**relocate**	VOB
rmhlink	M	T	S	C	**rmhlink**, **rmmerge**		From-object, to-object (unless cross-VOB, unidirectional), VOB
rmlabel	M	T	S		**rmlabel**		Version
rmname	M	T	S	C	**rmname**, **rmelem**, **mv**		Directory version(s)
rmpool			S	C	**rmpool**		VOB
rmtrigger	M	T	S		**rmtrigger**		Element
rmtype		T	S	C	**rmtype**		VOB
rmver	M	T	S	C	**rmver**	**checkvob –fix**	Element
unlock		T	S		**unlock**	(various)	Unlocked object
unreserve	M	T			**unreserve**		Checked-out version

Operations and Triggers

Each of the following superoperations represents a group of the above event-causing operations. See **mktrtype** for information on how to use the following keywords to write triggers for groups of operations.

MODIFY_TYPE	MODIFY_DATA
MODIFY_ELEM	MODIFY_MD

Table 7 omits the triggerable operations **uncheckout** and **chevent**; as these operations do not cause event records to be stored in the VOB database.

Event Visibility

This section describes where, directly or indirectly, you may encounter event record contents. The following commands include event history information in their output, which can be formatted with the **–fmt** option:

describe	**lshistory**
lsactivity	**lslock**
lsbl	**lspool**
lscheckout	**lsproject**
lscomp	**lsreplica**
lsdo	**lsstream**
lsfolder	**lstype –long**

Comments and Event Records

The set of ClearCase and ClearCase LT commands named in Table 7 matches almost exactly the set of commands that accept user comments as input. (**reformatvob**, which takes no comment, is the only exception.) When you supply comments to a ClearCase or ClearCase LT command, your comment becomes part of an event record.

Some **cleartool** commands create a comment even if you do not provide one. These generated comments describe the operation in general terms, such as “modify metadata” or “create directory element.” User comments, if any, are appended to generated comments. For a complete description of comment-related command options and comment processing, see the **comments** reference page.

SEE ALSO

chevent, **cleartool**, **comments**, **fmt_ccase**, **lshistory**, **mktrtype**, **vob_scrubber**

exports_ccase

List of VOBs to be accessed by non-ClearCase hosts

APPLICABILITY

Product	Command Type
ClearCase	data structure

Platform
UNIX

SYNOPSIS

- Exports table entry:

  Solaris: *VOB-tag* [**–ro**] [**–ro=***client*[:*client*]...] [**–rw**] [**–rw=***client*[:*client*]...] [**–anon=***uid*] [**–root=***host*[:*host*]...]

  All other platforms: *VOB-tag* [*options*] [*netgroup* | *hostname*] ...

- Standard options:

  Solaris: (none)

  All other platforms: **ro**, **rw**, **anon**, **root**, **access**

- Default options:

  Solaris: **–rw**

  All other platforms: **rw, anon=nobody**

DESCRIPTION

A host that has not installed ClearCase can still access any VOB, using NFS. Refer to the *Administrator's Guide* for the procedure for setting up non-ClearCase access to a VOB.

EXAMPLES

- A ClearCase host, **saturn**, wants to export a VOB mounted at **/vobs/proj**, as seen through view **beta**. This line exports the VOB to all hosts in the network:

  /view/beta/vobs/proj

  A non-ClearCase host soft-mounts the VOB with this file system table entry:

saturn:/view/beta/vobs/proj /ccase_vobs/proj nfs rw,noauto,soft,timeo=300,retrans=10 0 0

Another host hard-mounts the VOB with this file-system table entry:

saturn:/view/beta/vobs/proj /vobs/proj nfs rw,hard 0 0

- Export a VOB to netgroup **pcgroup**, and also to individual host **newton**.

  /view/beta/vobs/proj –w=pcgroup:newton *(Solaris)*
  /view/beta/vobs/proj –access=pcgroup:newton *(all other architectures)*

NOTES

The ClearCase installation procedure creates a template **/etc/exports.mvfs**, all of whose lines are commented out (on all platforms except Digital UNIX).

FILES

AIX 4, MP-RAS	**/etc/exports.mvfs** **/etc/rc.atria**
Digital UNIX	**/sbin/init.d/atria**
HP-UX 10, HP-UX 11	**/etc/exports.mvfs** **/sbin/init.d/atria**
Solaris, IRIX, Reliant UNIX, UnixWare	**/etc/exports.mvfs** **/etc/init.d/atria**

SEE ALSO

cleartool, **export_mvfs**, **init_ccase**, *Administrator's Guide*

Architecture-specific man pages:

AIX 4, HP-UX 10, HP-UX 11, IRIX, MP-RAS:	**exports(4)**, **fstab(4)**, **netgroup(4)**
Digital UNIX:	**exports(4)**, **fstab(4)**, **netgroup(4)**, **showmount(8)**, **mountd(8)**
Reliant UNIX:	**exports(4)**, **dfstab(4)**, **vfstab(4)**, **share(1M)**
Solaris, UnixWare:	**dfstab(4)**, **vfstab(4)**, **share(1M)**, **netgroup(4)**

export_mvfs

Exports and unexports VOBs to NFS clients (non-ClearCase access)

APPLICABILITY

Product	**Command Type**
ClearCase	command

Platform
UNIX

SYNOPSIS

- Digital UNIX platforms:

  ccase-home-dir/**etc/export_mvfs –a** [**–i**] [**–u** [**–r**]] [**–v**] [**–I** *exportid*] [**–o** *options*] [*pname*]
  or
  ccase-home-dir/**etc/export_mvfs** [**–a**] [**–i**] [**–u** [**–r**]] [**–v**] [**–I** *exportid*] [**–o** *options*] *pname*

- All other platforms:

  ccase-home-dir/**etc/export_mvfs** [**–a**] [**–i**] [**–u** [**–r**]] [**–v**] [**–I** *exportid*] [**–o** *options*] [*pname*]

DESCRIPTION

The **export_mvfs** command enables access by non-ClearCase-family-products by making a local VOB available for mounting over the network by hosts on which ClearCase is not installed. This command is the ClearCase counterpart of the **exportfs(1M)** (HPUX-10, HPUX-11, IRIX, AIX-4, UnixWare) or **share(1M)** (Solaris 2, MP-RAS, Reliant UNIX) command for file systems of type MVFS.

RESTRICTIONS

Identities: You must be **root**.

Locks: No locks apply.

Mastership: (Replicated VOBs) No mastership restrictions.

OPTIONS AND ARGUMENTS

With no options or arguments, **export_mvfs** lists the VOBs currently exported by the host (except on Digital UNIX).

–a
: (All) Exports all pathnames listed in **/etc/exports.mvfs**; with **–u**, unexports all currently exported VOBs.

–i
: Ignores the options in **/etc/exports.mvfs**. By default, **export_mvfs** consults **/etc/exports.mvfs** for the options associated with each pathname to be exported.

–u [**–r**]
: Unexports the specified pathnames. With **–a**, unexports all currently exported VOBs. With **–r**, removes view/export-ID mapping from the MVFS but leaves the view active.

–v
: (Verbose) Displays each pathname as it is exported or unexported.

–I *exportid*
: Specifies an export-ID for temporary NFS export of a view. Start with export-ID 4095 and work down. (View export IDs that are assigned automatically start at 1 and work up.)

  NOTE: Do not use this option for views that already have export-IDs assigned in the ClearCase registry. Use **lsview –long** *view-tag* to determine whether a view has an assigned export-ID.

–o *options*
: A comma-separated list of optional characteristics for the pathnames being exported. See the **exports_ccase** reference page for the supported options.

pname
: A view-extended pathname to the VOB-tag (mount point) of the VOB to be exported.

FILES

/etc/exports.mvfs
/usr/atria/etc/export_mvfs

SEE ALSO

exports_ccase, **init_ccase**, **mkvob**, **exportfs(1M)**, **exports(4)**, **share(1M)**, *Administrator's Guide*

file

Displays the element type ClearCase or ClearCase LT would use for a file

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

file [**–invob** *pname*] [**–all**] *pname*...

DESCRIPTION

The **file** command is similar to the UNIX **file(1)** command, which determines the file type of a specified file. **cleartool file** displays the element type ClearCase or ClearCase LT would use for the specified file if the file were converted to an element.

file uses the following process to find the element type:

1. Search magic files for the first rule that matches the file’s type.

   For more information on magic files, file-typing, and the search path for magic files, see the **cc.magic** reference page.

2. Compare the element types in the rule with the element types in a particular VOB.

   By default, **file** uses the VOB containing the view-private file. If the file is not in a VOB, the command uses the VOB containing the current working directory.

3. Display the first element type in the rule that exists in the VOB.

   file processes the element types in the rule from left to right. (In a magic file rule, element types are listed from most to least specific.) For example, with a rule like the following:

   ```
   txt document text_file : -printable & -name "*.[tT][xX][tT]" ;
   ```

file first looks for an element type named **txt** and displays it if it exists in the VOB. If **txt** doesn't exist in the VOB, **file** looks for an element type named **document** and displays it if it exists. If **document** doesn't exist, **file** displays the **text_file** element type.

For information about creating new element types in a VOB, see the **mkeltype** reference page.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

–invob *vob-pname*
: Compares the potential element types against the list of element types in the specified VOB.

–all
: Skips the comparison with the list of element types in the VOB and prints every element type in the magic file rule.

EXAMPLES

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Display the element type that would be used for a view-private HTML file.

  cleartool file foo.html
  ```
  foo.html: html
  ```

- List all possible element types for a view-private HTML file.

  cleartool file –all foo.html
  ```
  foo.html: html_source html web_file source text_file
  ```

- Display the element type that would be used if the file were converted to an element in the VOB **/vobs/dev**.

  cleartool file –invob /vobs/dev foo.html
  ```
  foo.html: html_source
  ```

- Display the element type that would be used if the file were converted to an element in the VOB **\dev**.

  cleartool file –invob \dev foo.html
  ```
  foo.html: html_source
  ```

UNIX FILES

ccase-home-dir/**config/magic/default.magic**

WINDOWS FILES

*ccase-home-dir***config****magic****default.magic**

SEE ALSO

cc.magic, **mkelem**, **mkeltype**, **type_manager**

find

Uses a pattern, query, or expression to search for objects

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- Find objects visible in the directory structure seen in the current view:
  find *pname* ... *selection-options action-options*
- Find all objects in the VOB:
  find [*pname*...] **–a·ll** [**–vis·ible** | **–nvi·sible**] *selection-options action-options*
- Find objects throughout all mounted VOBs:
  find –avo·bs [**–vis·ible** | **–nvi·sible**] *selection-options action-options*
  selection-options:
  - **–nam·e** *pattern*
  - **–dep·th** | **–nr·ecurse** | **–d·irectory**
  - **–cvi·ew**
  - **–use·r** *login-name*
  - **–gro·up** *group-name*
  - **–typ·e** { **f** | **d** | **l** } ...
  - **–fol·low**
  - **–nxn·ame**
  - **–ele·ment** *query*
  - **–bra·nch** *query*

–ver·sion *query*
–kin·d { *object_selector* | **all** }
ClearCase and ClearCase LT *action-options* (at least one required, multiple allowed):
–pri·nt
–exe·c *command-invocation*
–ok *command-invocation* ...
Attache *action-options* (at least one required, multiple allowed except with **–get**):
–get [–compress] [–ove·rwrite |**–nov·erwrite] [–pti·me] [–log** *pname*]
–pri·nt
–exe·c *command-invocation*
–ok *command-invocation* ...

DESCRIPTION

The **find** command starts with a certain set of objects, selects a subset of the objects, and then performs an action on the subset. The selected objects can be elements, branches, versions, or VOB symbolic links. The action can be to list the objects, or to execute a command on each object, either conditionally or unconditionally.

Typically, you start with all objects in a directory tree as seen in your view. You can also start with all objects in one or more VOBs, regardless of their visibility in a particular view.

NOTE: The **find** command is similar to the UNIX **find(1)** command. Only a limited set of the standard **find** options are supported; the way that commands are invoked on selected objects (**–exec** and **–ok** options) differs from **find(1)**.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE STARTING SET OF OBJECTS. *Default:* None. You must specify one of the following:

- One or more elements, using *pname* arguments
- One or more VOBs, using the **–all** option
- All mounted VOBs, using the **–avobs** option

NOTE: Processing all of a VOB's elements using **–all** or **–avobs** is an order of magnitude faster than going through its entire directory tree by specifying the VOB's root directory as a *pname* argument. With these options, the order in which elements are processed and/or reported is very different from directory-tree order.

pname ...

One or more file and/or directory elements. **find** starts with the elements, branches, and versions that are part of the specified file elements and the subtrees under the specified directory elements.

In Attache, arguments of the form @*pname* can be used to add the contents of the local file *pname* as pathname arguments. The pathname arguments can contain wildcards, and must be listed in the file one per line, or also be of the form @*pname*. Specifying a relative pathname for @*pname* begins from Attache's startup directory, not the working directory, so a full local pathname is recommended.

–a·ll

With *pname* arguments, modifies the meaning of each argument to specify its entire VOB, not just a single file or directory. Without any *pname* arguments, specifies the VOB containing the current working directory.

NOTE: When you use **find –all**, only one instance of an element is reported, even if there is one or more VOB hard links that point to the element. Either the element name or one of the VOB hard links is displayed.

–avo·bs

By default, **find** starts with all the elements, branches, and versions in all the VOBs mounted on the local host or on the helper host (Attache). A snapshot view issues a warning if all mounted VOBS have not been loaded into the view. This option depends on the MVFS, and so has no meaning for snapshot-view-only hosts.

If the CLEARCASE_AVOBS EV is set to a colon-separated list of VOB-tags (in UNIX; in Windows, list items must be separated by semicolons), this set of VOBs is used instead. In Attache, this environment variable must be set in the helper process.

CONSIDERING OBJECTS THAT ARE NOT CURRENTLY VISIBLE. *Default:* All elements in the VOB are included, whether or not they are visible in the view.

–vis·ible

Includes only those elements, along with their branches and versions, that are visible (have a standard pathname) in the view.

–nvi·sible

Includes only those elements, along with their branches and versions, that are not visible (do not have a standard pathname) in the view.

SELECTING ELEMENTS USING STANDARD CRITERIA. The following options use the specified criteria to select subsets of objects.

–nam·e *pattern*

Selects the subset of objects whose element names match the specified file-name pattern. *pattern* must be a leaf name. (See the **wildcards_ccase** (ClearCase and ClearCase LT) or **wildcards** (Attache) reference page.)

–dep·th

Causes directory entries to be processed before the directory itself.

–nr·ecurse

For each directory element, selects the objects in the element itself, and in the file and directory elements within it, but does not descend into its subdirectories.

–d·irectory

For each directory, examines only the directory itself, not the directory or file elements, or VOB symbolic links it catalogs.

–cvi·ew

Modifies the set of objects selected by the **–element**, **–branch**, and **–version** queries (if any):

If you did not specify **–version**, replaces each element and branch with the version that is currently in the view. (No substitution is performed on VOB symbolic links.)

If you did specify **–version**, further restricts the subset to versions that are currently in the view.

–use·r *login-name*

Selects only those objects in the subset of elements owned by user *login-name*.

–gro·up *group-name*

Selects only those objects in the subset of elements belonging to group *group-name*.

–typ·e f
–typ·e d
–typ·e l

Selects the subset of objects of a certain kind: file elements (**f**), directory elements (**d**), or VOB symbolic links (**l**). To include multiple kinds of objects, group the key letters into a single argument (**–type fd**), or use multiple options (**–type f –type d**).

–fol·low

Traverses VOB symbolic links during the walk of the directory tree.

USE OF EXTENDED PATHNAMES. *Default:* **find** submits the objects it selects to the specified action using extended pathnames, such as **foo.c@@** (element), **foo.c@@/main** (branch), or **foo.c@@/main/5** (version).

–nxn·ame

Removes the extended naming symbol (by default, @@) and any subsequent version-ID or branch pathname from the name of each selected object. Duplicate names that result from this transformation are suppressed. In effect, this option transforms extended names into standard operating system names (in Attache, for the helper host); it also transforms names of branches or versions into names of elements. In Attache, this *selection-option* is always applied when the **–get** action is used.

SELECTING ELEMENTS USING QUERIES. The options in this section select a subset of objects using the VOB query language, which is described in the **query_language** reference page. You can use these options in any combination. They are always applied in this order, successively refining the set of selected objects: first **–element**, then **–branch**, then **–version**. The result of applying one or more of these options is a set of objects at the finest level of granularity level: all versions if you used **–version**, or else all branches if you used **–branch**, or else all elements if you used **–element**. If you use none of these options, the set includes elements and VOB symbolic links. There is no way to use a query to select a set of VOB symbolic links.

–ele·ment *query*
: Selects element objects using a VOB query; all of a selected element's branches and versions are also selected. Using this option with a **brtype** query makes **find –all** much faster in a large VOB where the specified branch type exists on a relatively small number of elements.

–bra·nch *query*
: From the set of objects that survived the element-level query (if any), selects branch objects using a VOB query; all of a selected branch's versions are also selected.

–ver·sion *query*
: From the set of objects that survived the element-level and branch-level queries (if any), selects version objects using a VOB query.

SPECIFYING THE ACTION. *Default:* None. You must specify an action to be performed on the selected objects. You can specify a sequence of several actions, using two **–exec** options, or **–exec** followed by **–print**, and so on. In Attache, you cannot specify **–get** with any of the other actions.

(Attache) **–get [–compress] [–ove·rwrite | –nov·erwrite] [–pti·me] [–log** *pname* **]**
: Causes files matching a query to be downloaded to the workspace. See the **get** reference page for explanations of the specifications following the **–get** keyword.

–pri·nt
: Lists the names of the selected objects, one per line.

–exe·c *command-invocation*
: UNIX and Windows—Execute the specified command (in Attache, on the helper host) once for each selected object.

Windows—If you invoke a command built in to the Windows shell (for example, **cd**, **del**, **dir**, or **copy**), you must invoke the shell with **cmd /c**. For example:

–exec "cmd /c copy %CLEARCASE_PN% %HOME%"

If a path within *command-invocation* contains spaces, you must embed it in quotation marks. For example, in **cleartool** single-command mode (note the backslash used to escape the second quotation mark):

–exec "cmd /c copy %CLEARCASE_PN% \"c:\find results""

In **cleartool** interactive mode (no escape character needed):

–exec 'cmd /c copy %CLEARCASE_PN% "c:\find results"'

–ok *command-invocation*
For each selected object, displays a confirmation prompt; if you respond **yes**, executes the specified command (in Attache, on the helper host).

When using the **–exec** or **–ok** command invocation, do not use braces ({ }) to indicate a selected object, or use a quoted or escaped semicolon to terminate the command. Instead, enter the entire command as a quoted string; use one or more of these environment variables to reference the selected object:

CLEARCASE_PN
Pathname of selected element or VOB symbolic link

CLEARCASE_XN_SFX
Extended naming symbol (default: @@)

CLEARCASE_ID_STR
Branch pathname of a branch object (**\main\rel2_bugfix**); version-ID of a version object (**\main\rel2_bugfix\4**); null for an element

CLEARCASE_XPN
Full version-extended pathname of the selected branch or version (concatenation of the three preceding variables)

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- List all file elements in and below the current working directory.

  cmd-context **find . –type f –print**
  ```
  ./Makefile@@
  ./hello.c@@
  ./hello.h@@
  ./msg.c@@
  ./util.c@@
  ```
  This listing includes the extended naming symbol. The **–nxname** option suppresses this symbol.

- List all objects owned by user **smg** throughout all mounted VOBs.

  cmd-context **find –avobs –user smg –print**
  ```
  \work_vob\hw\util.c@@
  \work_vob\hw\hello.c@@
  \smg_tmp\bin@@\main\6\misc\main\3\text@@
  \smg_tmp\bin@@\main\6\misc\main\3\Makefile@@
  \smg_tmp\bin@@\main\6\misc\main\3\test.c@@
  ...
  ```

- List the version labeled **REL1** for each element in or below the current working directory.

  cmd-context **find . –version "lbtype(REL1)" –print**
  ```
  .@@/main/1
  ./Makefile@@/main/1
  ./hello.c@@/main/2
  ```

- Excluding any elements that do not have both labels, list all versions in the current VOB labeled either **REL1** or **REL2** but not both.

  cmd-context **find –all –element '{lbtype_sub(REL1) && lbtype_sub(REL2)}' ^
  –version '{(lbtype(REL1) && ! lbtype(REL2)) || ^
  (lbtype(REL2) && !lbtype(REL1))}' –print**
  ```
  \dev\testfile.txt@@\main\43
  \dev\testfile.txt@@\main\68
  \dev\util.c@@\main\50
  \dev\util.c@@\main\58
  ...
  ```

- (ClearCase and ClearCase LT) List each header file (***.h**) for which some version is labeled **REL2** or **REL3**.

  cmd-context **find . –name '*.h' –element 'lbtype_sub(REL2) \
  || lbtype_sub(REL3)' –print**
  ```
  ./hello.h@@
  ```

- (Attache) Download to your workspace each header file (***.h**) for which some version is labeled **REL2** or **REL3**. Note that the wildcard pattern ***.h** must be enclosed in quotes so that it is not expanded.

  <u>*cmd-context*</u> **find . –name '*.h' –element 'lbtype_sub(REL2) || lbtype_sub(REL3)' –get**

- (Attache) Download to your workspace each source file (***.c**) created since yesterday. The set of files brought across can easily be restricted further.

  <u>*cmd-context*</u> **find . –name '*.c' –avobs –version '{created_since(yesterday)}' –get**

- List all versions that have a **QAed** attribute with the string value `"Yes"`.

  <u>*cmd-context*</u> **find . –version 'QAed == "YES"' –print**

  ```
  .\Makefile@@\main\2
  .\hello.c@@\main\4
  .\hello.h@@\main\1
  .\util.c@@\main\2
  .\util.c@@\main\rel2_bugfix\1
  ```

- List the standard name of each element that has (or contains a branch or version that has) a **BugNum** attribute with the value `189`.

  <u>*cmd-context*</u> **find . –nxname –element ’attr_sub(BugNum,==,189)’ –print**
  ```
  ./hello.c
  ```

- For each element that has had a merge from the **rel2_bugfix** branch to the **main** branch, archive the current version of the element to your home directory.

  <u>*cmd-context*</u> **find . –element merge(\main\rel2_bugfix,\main) ^**
  –exec 'cmd /c copy %CLEARCASE_PN% %HOME%'

- For each element that has had a merge from the **rel2_bugfix** branch to the **main** branch, archive the current version of the element to a **tar(1)** file in your home directory (in Attache, on the helper host).

  ClearCase and ClearCase LT:

  <u>*cmd-context*</u> **find . –element "merge(/main/rel2_bugfix, /main)" **
  –exec 'echo $CLEARCASE_PN >> /tmp/filelist'

  % tar –cvf $HOME/rel2bugmerge.tar `cat /tmp/filelist`

  % rm /tmp/filelist

  Attache:

  <u>*cmd-context*</u> **find . –element merge(/main/rel2_bugfix,/main) **
  –exec 'cmd /c copy $CLEARCASE_PN $HOME'

- If any element's most recent version on the main branch is missing label **REL3**, label it.

  cmd-context **find . –version 'version(\main\LATEST) && ! lbtype(REL3)' ^**
  –exec 'cleartool mklabel –replace REL3 %CLEARCASE_XPN%'

- Attach a **Testing** attribute with string value `"Done"` to all versions labeled **REL2**. Note that the double-quote characters that surround the string value must themselves be escaped or quoted:

  ClearCase and ClearCase LT:

  cmd-context **find . –ver 'lbtype(REL2)' **
  –exec 'cleartool mkattr Testing \"Done\" $CLEARCASE_XPN'

  Attache:

  cmd-context **find .–ver lbtype(REL2) –exec 'cleartool mkattr Testing "Done" $CLEARCASE_XPN'**

- Conditionally delete all branches of type **experiment**.

  ClearCase and ClearCase LT:

  cmd-context **find . –branch brtype(experiment) ^**
  –ok 'cleartool rmbranch –force %CLEARCASE_XPN%'

  Attache:

  cmd-context **find . –branch brtype(experiment) –ok 'cleartool rmbranch –force %CLEARCASE_XPN%'**

- Change all elements currently using storage pool **my_cpool** to use pool **cdft** instead.

  ClearCase:

  cmd-context **find . –all –element 'pool(my_cpool)' **
  –exec 'cleartool chpool cdft $CLEARCASE_PN'

  Attache:

  cmd-context **find . –all –element pool(my_cpool) –exec 'cleartool chpool cdft $CLEARCASE_PN'**

- Obsolete elements that are no longer visible.

  cmd-context **find . –all –nvisible –exec 'cleartool lock –obsolete %CLEARCASE_PN%'**

- List merges (recorded by hyperlinks of type **Merge**) involving versions located at the ends of branches named **gopher**.

  ClearCase and ClearCase LT:

cmd-context **find . –version ’version(.../gopher/LATEST)’ –print \ –exec 'cleartool describe –short –ahlink Merge $CLEARCASE_XPN'**

```
.@@/main/gopher/1
-> /vob1/proj/src@@/main/146
./base.h@@/main/gopher/1
-> /vob1/proj/src/base.h@@/main/38
./main.c@@/main/gopher/1
-> /vob1/proj/src/main.c@@/main/42
```

Attache:

cmd-context **find . –version version(...\gopher\LATEST) –print –exec 'cleartool describe –short –ahlink Merge $CLEARCASE_XPN'**

```
.@@\main\gopher\1
-> \vob1\proj\src@@\main\146
.\base.h@@\main\gopher\1
-> \vob1\proj\src\base.h@@\main\38
.\main.c@@\main\gopher\1
-> \vob1\proj\src\main.c@@\main\42
```

- In the current directory and its subdirectories, list element versions that are on the branch **main_dev** and that were created in May of this year and that are not the **LATEST** versions.

  cmd-context **find . –version "{brtype(main_dev) && created_since(30-Apr) && (! created_since(31-May)) && (! version(\main\main_dev\LATEST))}" -print**

SEE ALSO

describe, **ls**, **query_language**, **wildcards**, **wildcards_ccase**

findmerge

Searches for versions that require a merge /optionally perform merge

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase, ClearCase LT—Search for versions:

  findm·erge { *pname* ... | [*pname* ...] **–a·ll** | **–avo·bs** | *activity-selector* ... **–fcs·ets** }
  { **–fta·g** *view-tag* | **–fve·rsion** *version-selector* | **–fla·test** }
  [**–dep·th** | **–nr·ecurse** | **–d·irectory**] [**–fol·low**] [**–vis·ible**]
  [**–use·r** *login-name*] [**–gro·up** *group-name*] [**–typ·e** { **f** | **d** | **fd** }]
  [**–nam·e** *pattern*] [**–ele·ment** *query*]
  [**–nze·ro**] [**–nba·ck**] [**–why·not**] [**–log** *pname*]
  [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* | **–cq·uery** | **–cqe·ach** | **–nc·omment**]
  [**–unr·eserved**] [**–q·uery** | **–abort** | **–qal·l**] [**–ser·ial**]
  { **–pri·nt** [**–l·ong** | **–s·hort** | **–nxn·ame**]
  | **–mer·ge** | **–okm·erge** | **–g·raphical** | **–gm ·erge** | **–okg·merge**
  | **–exe·c** *command-invocation*
  | **–ok** *command-invocation*
  | **–co**
  } ...

- Attache—Search for versions:

  findm·erge { *pname* ... | [*pname* ...] **–a·ll** | **–avo·bs** }
  { **–fta·g** *view-tag* | **–fve·rsion** *version-selector* | **–fla·test** }
  [**–dep·th** | **–nr·ecurse** | **–d·irectory**] [**–fol·low**] [**–vis·ible**]

findmerge

[**–use·r** *login-name*] [**–gro·up** *group-name*] [**–typ·e** { **f** | **d** | **fd** }]
[**–nam·e** *pattern*] [**–ele·ment** *query*]
[**–nze·ro**] [**–nba·ck**] [**–why·not**] [**–log** *pname*]
[**–c·omment** *comment* | **–cfi·le** *comment-file-pname* | **–cq·uery** | **–cqe·ach** | **–nc·omment**]
[**–unr·eserved**] [**–ser·ial**]
{ **–pri·nt** [**–l·ong** | **–s·hort** | **–nxn·ame**]
| **–mer·ge –abort** | **–okm·erge –abort** | **–g·raphical** | **–gm·erge** [**–qal·l**] | **–okg·merge**
| **–exe·c** *command-invocation*
| **–ok** *command-invocation*
| **–co**
} ...

DESCRIPTION

For one or more versions, the **findmerge** command determines whether a merge is required from a specified version to the version in your view, then executes one or more actions:

- Listing the versions that require a merge
- Performing the required merges, checking out versions as necessary; in Attache, the merges are done locally
- Performing an arbitrary command

findmerge works as follows:

1. It considers a set of versions, which you specify using syntax similar to that of the **find** command.
2. For each of these versions, **findmerge** examines the relationship between the version in your view and the version specified by the **–ftag**, **–fversion**, or **–flatest** option. It determines whether a merge is required from that other version to your view's version.
3. **findmerge** then performs the actions you specify with **–print**, **–exec**, and/or the various **–merge** variants. (In Attache, if merges are performed, the resulting merged files are left in your workspace.
4. (Attache) Local directories are not updated after a directory merge; the user must issue **get** commands to update merged directories.

Keep in mind that nontrivial merge capability is guaranteed to work only for versions whose type manager implements the **merge** or **xmerge** methods. See the **type_manager** reference page for more information.

Using findmerge with UCM Activities

To use **findmerge** with UCM activities, you specify one or more activities and the option **–fcsets**. (The *activity-selector* arguments must precede the **–fcsets** option.) Each version listed in a change set becomes the from-version in a merge operation. As always, the to-version is the one in your view.

(ClearCase and ClearCase LT) For other versions—for example, those of type **file**—the type manager may or may not be able to merge the data in the versions that **findmerge** identifies. For some versions, you may need to perform the merge manually, as follows:

1. Check out the version
2. Incorporate data from the version on another branch into your checked-out version, using a text editor or some other tool
3. Connect the appropriate versions with a merge arrow, using **merge –ndata**

(Attache) For some versions, namely, those not containing text, you need to perform the merge manually, as follows:

1. Check out the version
2. Incorporate data from the version on another branch into your checked-out version, using a text editor or some other tool
3. Connect the appropriate versions with a merge arrow, using **merge –ndata**

Deferring Merges with the -print Option

If you specify **–print** as the action (and you do not also specify any of the merge actions), **findmerge** does not actually perform any merges. Instead, it shows what merge activity would be required:

```
Needs Merge "proj.c" [to /main/41 from /main/v2_plus/6 base /main/v2_plus/3]
Log has been written to "findmerge.log.16-Nov-98.17:39:18"

.
.
.
```

In addition, it writes a set of shell commands to perform the required merges to a log file. In Attache, the log file is in the view:

```
cleartool findmerge proj.c@@\main\41 -fver \main\v2_plus\6 -log nul -merge
-cqe

.
.
.
```

At some later point, you can execute the commands in the log file—all at once, or a few at a time. In Attache, the commands in the log file can be executed only on the helper host.

If the directory version from which you are merging contains new files or subdirectories, **findmerge –print** does not report on those files or directories until you merge the directory versions. Therefore, you may want to run **findmerge** twice: once to merge the directory versions, and again with the **–print** option to report which files need to be merged. You can then cancel the checkout of the directories if you do not want to save the directory merge.

Incomplete Reporting of Required Merges

Under some circumstances, **findmerge –print** does not detect all the required merges (that is, all the merges that **findmerge –merge** would perform). This occurs if one or more directory merges are required, but are not performed.

By default, **findmerge** merges a directory before determining merge requirements for the versions cataloged within the directory. Thus, if merging directory **srcdir** makes a newly created file version, **patch.c**, appear, **findmerge** proceeds to detect that **patch.c** itself also needs to be merged. But if the only specified action is **–print**, then **findmerge** can determine only that **srcdir** must be merged; it cannot determine that **patch.c** must also be merged.

This incomplete reporting also occurs in these situations:

- You decline to merge a directory when prompted by the **–okmerge** or **–okgmerge** option.
- You specify **–depth**, which causes the versions cataloged in a directory to be processed before the directory version itself.
- You use **–directory** or **–nrecurse** to suppress processing of the versions cataloged in a directory.
- You use **–type f**, which suppresses processing of directory versions.

You can use the following procedure to guarantee that the log file produced by **findmerge –print** includes all the required file-level merges within the directory tree under **srcdir**:

1. Actually perform all the directory-level merges:

   cmd-context **findmerge srcdir –type d –merge**

2. Generate a log file containing the **findmerge** commands required for files within the merged directory hierarchy:

   cmd-context **findmerge srcdir –type f –print**

In ClearCase and ClearCase LT, executing the log file produces results identical to entering the single command **findmerge srcdir –merge**.

findmerge Algorithm

The **findmerge** command uses one of two algorithms to locate and examine versions. When the number of versions to be examined is below a certain threshold (approximately 100), **findmerge** uses the algorithm that uses the VOB's metadata. When the number of versions exceeds the threshold, **findmerge** uses the algorithm that requires walking through the VOB's directory structure. The directory walkthrough method is slower than the metadata method.

RESTRICTIONS

If the specified action involves checking out and/or merging files, the restrictions for the **checkout** and **merge** commands apply.

OPTIONS AND ARGUMENTS

SPECIFYING THE VERSIONS TO BE CONSIDERED. *Default:* None.

pname ...

One or more file and/or directory versions; only the specified file versions and the subtrees under the specified directory versions are considered.

In Attache, arguments of the form @*pname* can be used to add the contents of the local file *pname* as pathname arguments. The pathname arguments can contain wildcards (see the **wildcards** reference page), and must be listed in the file one per line, or also be of the form @*pname*. Specifying a relative pathname for @*pname* begins from Attache's startup directory, not the working directory, so a full local pathname is recommended.

–all
pname ... **–all**

Appending **–all** to a *pname* list (or an @*pname* list in Attache) causes all the versions in the VOB containing the *pname* to be considered, whether or not they are visible in your view. By itself, **–all** specifies the top-level directory of the VOB containing the current working directory.

findmerge performs additional work after processing the VOB directory tree if you use **–all** or **–avobs** in combination with **–ftag**; in this case, it issues a warning message for each version that does not appear in the to-view, but does appear in the from-view.

–avo·bs

Considers all versions in the VOBs active (mounted) on the local host in ClearCase or on the helper host in Attache. (If environment variable **CLEARCASE_AVOBS** is set to a list of VOB-tags, this set of VOBs is used instead (separate VOB tags in the list by colons (UNIX) or semicolons (Windows)). In Attache, the environment variable must be set in the helper process.)

activity-selector ...

One or more UCM activities. Specify *activity-selector* in the form **activity:***activity-name*[@*vob-selector*]. You must specify the **–fcsets** option immediately following this argument.

SPECIFYING THE FROM–VERSION. *Default:* None. You must use one of these options to specify another version of each version, to be compared with the version in your view.

–fta·g *view-tag*

Compare with the version in the view with the version in the view specified by *view-tag*.

view-tag may not specify a snapshot view. A version of the same version is always used, even if the version has a different name in the other view.

–fve·rsion *version-selector*
Compare with the version specified by the version-selector. A version selector involving a branch type, for example, **.../branch1/LATEST**, is optimized for selecting the set of versions to consider and performs better than other types of queries. In the case where the branch exists only on a relatively small number of versions in the VOB, this option performs much better than other types of queries.

–fla·test
(Consider only versions that are currently checked out.) Compare with the most recent version on the branch from which your version was checked out. This option is useful with versions for which you have unreserved checkouts: if one or more new versions have been checked in by other users, you must merge the most recent one into your checked-out version before you can perform a checkin.

–fcs·ets
Consider all the versions in the change set of each specified *activity-selector* argument.

NARROWING THE LIST OF VERSIONS TO BE CONSIDERED. Use the following options to select a subset of the versions specified by *pname* arguments and the **–all** or **–avobs** option.

–dep·th
Causes directory entries to be processed before the directory itself.

–nr·ecurse
For each directory version, considers the file and directory versions within it, but does not descend into its subdirectories.

–d·irectory
For each directory, considers only the directory itself, not the directory or file versions, or VOB symbolic links it catalogs.

–fol·low
Causes VOB symbolic links to be traversed.

–use·r *login-name*
Considers only those versions owned by user *login-name*.

–gro·up *group-name*
Considers only those versions belonging to group *group-name*.

–typ·e f
–typ·e d
–typ·e fd
Considers file versions only (**f**), directory versions only (**d**), or both (**fd**).

–nam·e *pattern*

Considers only those versions whose leaf names match the specified file-name pattern. (See the **wildcards_ccase** or **wildcards** (Attache) reference page.)

–ele·ment *query*

Considers only those versions that satisfy the specified query (same as the ClearCase/ClearCase LT/Attache **find** command). A simple branch query, for example, **brtype(br1)**, is optimized for selecting the set of versions to consider and performs better than other types of queries. When the branch exists only on a relatively small number of versions in the VOB, this option performs much better than other types of queries.

SPECIAL VERSION TREE GEOMETRY: MERGING FROM VERSION 0. If a merge is required from a version that happens to be version 0 on its branch, **findmerge**'s default behavior is to perform the merge and issue a warning message:

```
Element "util.c" has empty branch [to \main\6 from \main\br1\0]
```

More often, **findmerge** determines that no merge is required from a zeroth version; it handles this case as any other no-merge-required case.

The following option overrides this default behavior.

–nze·ro

Does not perform a merge if the from-contributor is version 0 on its branch. This gives you the opportunity to delete the empty branch, and then perform a merge from the version at which the branch was created.

SPECIAL VERSION TREE GEOMETRY: MERGE BACK-AND-OUT TO SUBBRANCH. **findmerge** flags this special case with a warning message:

```
Element "msg.c" requests merge to /main/12 backwards on same branch from
/main/18
```

This situation arises in these cases:

- You are merging from a parent branch to a subbranch.
- For a particular version, no subbranch has been created yet.
- Your config spec selects a version of that version using a **–mkbranch** config spec rule.

In this case, **findmerge**'s default behavior is to perform the merge by checking out the version (which creates the subbranch at the to-version), then overwriting the checked-out version with the from-version.

The following option overrides this default behavior.

–nba·ck

Does not perform the merge in the case described earlier. It may be appropriate to simulate the merge by moving the version label down to the from-version. Note, however, that this alternative leaves the version without a subbranch, which may or may not be desirable.

VERBOSITY OF MERGE ANALYSIS. By default, **findmerge**:

- Silently skips versions that do not require a merge.
- (If you use **–all** or **–avobs** in combination with **–ftag**) Issues a warning message if your config spec does not select any version of a version, but the config spec of the view specified with **–ftag** does. (For example, this occurs when a new version has been created in the from-view.)

The following options override this behavior.

–why·not
For each version that does not require a merge, displays a message explaining the reason. This is especially useful when you are merging between views whose namespaces differ significantly.

–vis·ible
Suppresses the warning messages for versions that are not visible in the current view.

LOGGING OF MERGE ANALYSIS. *Default:* A line is written to a merge log file (in Attache, in the working directory in the view) for each version that requires a merge. The log takes the form of a UNIX shell script or Windows batch file that can be used to perform, at a later time (in Attache, on the helper host), merges that are not completed automatically (see **–print** and **–abort**, for example). A number sign (#) at the beginning of a line indicates that the required merge was performed successfully. The log file's name is generated by **findmerge** and displayed when the command completes.

NOTE: In Attache, the log file can be executed later, but the results may differ since copies of versions in the workspace are not taken into account. The commands can be cut and pasted from this log into the Attache command window, which will work correctly.

–log *pname*
Creates *pname* as the merge log file (in Attache, on the helper host), instead of selecting a name automatically. To suppress creation of a merge log file, use

–log /dev/null (UNIX)

or

–log NUL (Windows)

SPECIFYING CHECKOUT COMMENTS. *Default:* When **findmerge** checks out versions in order to perform merges, it prompts for a single checkout comment (**–cq**). You can override this behavior with your **.clearcase_profile** file. See the **comments** reference page. Edit comments with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
Overrides the default with the option you specify. See the **comments** reference page.

AFFECTING TYPE OF CHECKOUT. *Default:* If the **findmerge** action performs a **checkout**, it is a reserved **checkout**.

–unr·eserved
Performs any **findmerge** checkouts as unreserved checkouts.

MERGE OPTIONS. If you have **findmerge** actually perform merges, you can specify the following options, which work exactly as they do in the **merge** command. (In ClearCase and ClearCase LT, **–abort** and **–qall** are mutually exclusive.)

–q·uery (ClearCase and ClearCase LT)
Turns off automatic merging for nontrivial merges and prompts you to proceed with every change in the from-versions. Changes in the to-version are accepted unless a conflict exists.

–abo·rt
Cancels a merge if it is not completely automatic. In Attache, **–abort** is required with **–merge** and **–okmerge**.

–qal·l
Turns off automated merging. In Attache, turns off automated merging when in graphical mode. Prompts you to determine whether you want to proceed with each change.

–ser·ial
Reports differences with each line containing output from one contributor, instead of in a side-by-side format.

ACTIONS TO BE PERFORMED ON THE SELECTED VERSIONS. *Default:* None.

–pri·nt [**–l·ong** | **–s·hort** | **–nxn·ame**]
Lists the names of the versions that require a merge. The default listing includes the version-IDs of the to-versions and from-versions, and that of the base contributor (common ancestor):

```
Needs Merge "Makefile" [to \main\7 from \main\br1\1 base \main\6]
```

Specifying **–short** reduces the listing to version-extended pathnames of the to- and from-versions:

```
Makefile@@/main/7 Makefile@@/main/br1/1
```

Specifying **–long** adds to the default listing a description (**describe** command output) of the from-version:

```
Needs Merge "Makefile" [to /main/7 from /main/br1/1 base /main/6]
version "Makefile@@/main/br1/1"
  created 09-Nov-98.11:18:39 by Allison K. Pak (akp.user@neptune)
  element type: text_file
  predecessor version: /main/br1/0
```

Specifying **–nxname** reduces the listing to just the standard pathname of the version:

```
.\Makefile
```

–mer·ge –abort (**–abort** only with Attache)
–okm·erge –abort (**–abort** only with Attache)
–g·raphical
–gm·erge
–okgm·erge

(Valid only for versions whose type manager implements the **merge** method. See the **type_manager** reference page for more information.) Performs a merge for each version that requires it.

Three kinds of interfaces can be used: the **–merge** option performs a character-oriented merge, the **–graphical** option invokes the Merge Manager, and the **–gmerge** option invokes the graphical merge utility. All these actions attempt to check out the to-version, if it is not already checked out to your view. In Attache, only noninteractive merges are allowed in character mode; interactive merges must be done in graphical mode.

The **ok** variants pause for verification on each version, thus allowing you to process some versions and skip others.

SPECIAL CASE: Specifying **–merge –gmerge** causes **findmerge** to perform a character-oriented merge in **–abort** mode; if the merge aborts (because it could not proceed completely automatically), the interactive graphical merge tool is invoked.

–exe·c *command-invocation*
–ok *command-invocation*

Runs the specified command for each selected version. **findmerge** does not perform a checkout operation when either of these options is specified. With **–ok**, **findmerge** pauses for verification on each version, thus allowing you to process some versions and skip others.

Like the **find** command, **findmerge** sets the following variables in the specified command's environment:

CLEARCASE_PN	Pathname of version
CLEARCASE_XN_SFX	Extended naming symbol (default: @@)
CLEARCASE_ID_STR	Version-ID of to-version
CLEARCASE_XPN	Version-extended pathname of to-version

CLEARCASE_F_ID_STR	Version-ID of from-version
CLEARCASE_FXPN	Version-extended pathname of from-version
CLEARCASE_B_ID_STR	Version-ID of base contributor version

Windows—If you invoke a command built in to the Windows shell (for example, **cd**, **del**, **dir**, or **copy**), you must invoke the shell with **cmd /c**. For example:

–exec 'cmd /c copy %CLEARCASE_PN% %HOME%'

If a path within *command-invocation* contains spaces, you must embed it in quotation marks. For example, in **cleartool** single-command mode (note the backslash used to escape the second quotation mark):

–exec "cmd /c copy %CLEARCASE_PN% \"c:\findmerge results""

In **cleartool** interactive mode (no escape character needed):

–exec 'cmd /c copy %CLEARCASE_PN% "c:\findmerge results"'

–co

Attempts to check out the destination if it is not already checked out to your view. May be used as part one of a two-pass invocation of **findmerge**, where the second part uses an option such as **–exec**.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- (ClearCase and ClearCase LT) Compare a source file version in your current view to a version on another branch. Log the results of the comparison, but do not perform the merge. (If a merge is required, the log file stores a command that performs the merge.)

  cmd-context **findmerge msg.c –fversion /main/rel2_bugfix/LATEST –print**
  ```
  Needs Merge "msg.c" [to /main/2 from /main/rel2_bugfix/1 base /main/1]
  A 'findmerge' log has been written to "findmerge.log.04-Feb-99.10:01:23"
  ```

cat findmerge.log.04-Feb-99.10:01:23

```
cleartool findmerge msg.c@@/main/2 -fver /main/rel2_bugfix/1 -log
/dev/null -merge
```

- (Attache) Compare a source file version in your current workspace or view to a version on another branch. (If the file does not exist in your workspace, the version in your view is used.) Log the results of the comparison, but do not perform the merge. (If a merge is required, the log file on the helper host stores a command that performs the merge, but only from the helper host.)

  cmd-context **findmerge msg.c –fversion \main\rel2_bugfix\LATEST –print**

```
Needs Merge "msg.c" [to \main\2 from \main\rel2_bugfix\1 base \main\1]
A 'findmerge' log has been written to "findmerge.log.04-Feb-99.10:01:23"
```

  cmd-context **shell type findmerge.log.04-Feb-99.10:01:23**

```
cleartool findmerge msg.c@@/main/2 -fver /main/rel2_bugfix/1 -log
/dev/null -merge
```

- (ClearCase and ClearCase LT) For the current directory subtree, compare all versions visible in the current view against the versions in another view. Print a list of versions that require merging, but do not perform the merge. For versions where no merge is required, explain why.

  cmd-context **findmerge . –ftag rel2_bugfix_view –whynot –print**

```
No merge ".\Makefile" [\main\3 descended from \main\2]
No merge ".\cm_add.c" [element not visible in view rel2_bugfix_view]
No merge ".\hello.c" [to \main\4 from version zero \main\rel2_bugfix\0]
. . .
A 'findmerge' log has been written to "findmerge.log.04-Feb-99.11:00:59"
```

  type findmerge.log.04-Feb-99.11.00.59

```
cleartool findmerge .\msg.c@@\main\2 -fver \main\rel2_bugfix\1 -log nul
-merge
```

- (Attache) For the current directory subtree, compare all versions visible in the current view against the versions in another view. Print a list of versions that require merging, but do not perform the merge. For versions where no merge is required, explain why.

  cmd-context **findmerge . –ftag rel2_bugfix_view –whynot –print**

```
No merge "./Makefile" [/main/3 descended from /main/2]
No merge "./cm_add.c" [element not visible in view rel2_bugfix_view]
No merge "./hello.c" [to /main/4 from version zero /main/rel2_bugfix/0]

.
.
.

A 'findmerge' log has been written to "findmerge.log.04-Feb-99.11:00:59"
```

shell cat findmerge.log.04-Feb-99.11:00:59

```
cleartool findmerge ./msg.c@@/main/2 -fver /main/rel2_bugfix/1 -log
/dev/null -merge
```

- (ClearCase and ClearCase LT) For the current directory subtree, compare versions visible in the current view against versions on another branch, and perform any required merges. The resulting log file annotates all successful merges with a number sign (`#`).

cmd-context **findmerge . –fversion \main\rel2_bugfix\LATEST –merge**

```
Needs Merge ".\util.c" [to \main\3 from \main\rel2_bugfix\2 base
\main\rel2_bugfix\1]
Comment for all listed objects:
Merge from rel2_bugfix branch.
.

Checked out "util.c" from version "\main\3".
********************************
<<< file 1: M:\view1\george_fig_hw\src\util.c@@\main\rel2_bugfix\1
>>> file 2: .\util.c@@\main\rel2_bugfix\2
>>> file 3: .\util.c
********************************
-------[changed 7-8 file 1]--------|------[changed to 7-12 file 3]-----
    if (user_env)                  | if (user_env) {
      return user_env;               | if ( strcmp(user_env,"root") == +

.
.
.

Moved contributor ".\util.c" to ".\util.c.contrib".
Output of merge is in ".\util.c".
Recorded merge of ".\util.c".
A 'findmerge' log has been written to "findmerge.log.24-Mar-99.13.23.05"
```

type findmerge.log.24-Mar-99.13.23.05

```
#cleartool findmerge .\util.c@@\main\3 -fver \main\rel2_bugfix\2 -log nul
-merge -c "Merge from rel2_bugfix branch."
```

- (Attache) For the current directory subtree, compare versions visible in the current view against versions on another branch, and perform any required merges.

cmd-context **findmerge . –fversion \main\rel2_bugfix\LATEST –merge –abort**

```
Needs Merge ".\util.c" [to \main\3 from \main\rel2_bugfix\2 base
\main\rel2_bugfix\1]
Comment for all listed objects:
Merge from rel2_bugfix branch.
.
```

```
Checked out "util.c" from version "\main\3".
********************************
<<< file 1: M:\view1\george_fig_hw\src\util.c@@\main\rel2_bugfix\1
>>> file 2: .\util.c@@\main\rel2_bugfix\2
>>> file 3: .\util.c
*******************************
-------[changed 7-8 file 1]--------|------[changed to 7-12 file 3]-----
    if (user_env)                  | if (user_env) {
      return user_env;             | if ( strcmp(user_env,"root") == +
.
.
.
Output of merge is in ".\util.c".
Recorded merge of ".\util.c".
A 'findmerge' log has been written to "findmerge.log.24-Mar-99.13.23.05"
```

cmd-context **shell type findmerge.log.24-Mar-99.13.23.05**

```
#cleartool findmerge .\util.c@@\main\3 -fver \main\rel2_bugfix\2 -log nul
-merge -c "Merge from rel2_bugfix branch."
```

- As in the previous commands, merge from another branch. This time, if any merge cannot be completed automatically (two or more contributors modify the same line from the base contributor), start the graphical merge utility to complete the merge.

  cmd-context **findmerge . –fversion /main/rel2_bugfix/LATEST –merge –gmerge**

- (ClearCase and ClearCase LT) For the current directory subtree, compare all versions visible in the current view to versions on another branch. Follow any VOB symbolic links. Log the results of the comparison, but do not perform the merge. The current directory contains a symbolic link to the **beta** directory. The **findmerge** command follows that link and determines that version 1 of **foo.c** on the **bugfix** branch should be merged with version 4 on the main branch.

  cmd-context **findmerge . –fol –fversion \main\bugfix\LATEST –print**

```
Needs Merge "\usr2\home\ktessier\testvobs\testvob\testdir\beta\foo.c"
[to \main\4 from \main\bugfix\1 base \main\3]
Needs Merge ".\update [to \main\4 from \main\bugfix\1 base \main\2]
Log has been written to "findmerge.log.02-Jul-99.14:07:49".
```

- (ClearCase and ClearCase LT) For the current directory subtree, compare all versions visible in the current view to versions on another branch. Do not consider versions contained in any of the current directory's subdirectories. Log the results of the comparison, but do not perform the merge. The first invocation of **findmerge** detects no required merges in the current directory version or the file and directory versions it contains. Invoking **findmerge** from the subdirectory **source** detects a required merge. Invoking **findmerge** without the **–nr** option at **source**'s parent directory also detects the required merge because **findmerge** descends into **source**.

cmd-context **findmerge . –nr –fversion main/bugfix/LATEST –print**

```
% cd source
```

cmd-context **findmerge . –nr –fversion main/bugfix/LATEST –print**

```
Needs Merge "./update" [to /main/4 from /main/bugfix/1 base /main/2]
Log has been written to "findmerge.log.02-Jul-99.14:17:15".
```

- (ClearCase and ClearCase LT) For the current directory only, compare the directory version visible in the current view to a version on another branch. Do not consider the versions contained in those directories. Log the results of the comparison but do not perform the merge. The **findmerge** command discovers that the version of the directory on the **rel1** branch contains a file that is not in the version of the directory visible in the current view; therefore, version 2 of the directory on the **rel1** branch should be merged with version 2 on the main branch. Because the **findmerge** command specifies **–dir**, it does not evaluate this file for merging.

  cmd-context **findmerge . –dir –fversion \main\rel1\LATEST –print**

  ```
  Needs Merge "." [(automatic) to \main\2 from \main\rel1\2 (base also
  \main\2] Log has been written to "findmerge.log.03-Jul-99.15:30:46".
  ```

- Invoke the Merge Manager from the command line and complete the merge using the Merge Manager.

  cmd-context **findmerge . -fver .../mybranch/LATEST -graphical**

SEE ALSO

find, **merge**, **update**, **xcleardiff**, **find(1)**

fmt_ccase

Format strings for command output

APPLICABILITY

Product	Command Type
ClearCase	general information
ClearCase LT	general information
Attache	general information

Platform
UNIX
Windows

SYNOPSIS

- **–fmt** option syntax (used in various reporting commands: **annotate**, **describe**, **lshistory**, **lscheckout**, and so on):

  –fmt "*format-string***"**

  format-string is a character string, composed of alphanumeric characters, conversion specifications, and escape sequences. It must be enclosed in double quotes (**"**).

Conversion specifications:

%a Attributes (modifiers: **N**, **S**, [**attr-name**])
%c Comment string (modifiers: **N**)
%d Date (modifiers: **S**, **V**, **N**, **A**, **MA**, **BA**, **OA**)
%e Event description
%f Checked-out version information (modifiers: **R**, **T**, [*text*])
%h Host name
%i Indent level (modifier: [*indent-level*])
%l Labels (modifiers: **C**, **N**)
%m Object kind (version, derived object, and so on) (modifiers: **K**)
%n Name of object (modifiers: **D**, **E**, **L**, **O**, **S**, **PS**, **PV**, **V**, **X**)
%o Operation kind (checkin, lock, mkelem, and so on)
%p Property value (modifiers: [*property*], **C**, **D**, **O**, **S**, **T**)

%[*c*]t	Starting column number (modifiers: **N**, **S**, **T**)
%u	User/group information associated with the object's creation event (modifiers: **F**, **G**, **L**); see also **%[owner]p** and **%[group]p**.
%%	% character

Escape sequences:

\n	**<NL>**
\t	**<TAB>**
\'	Single quote
\\	Literal (uninterpreted) backslash
nnn	Character specified by octal code

DESCRIPTION

Many ClearCase, ClearCase LT, and Attache commands read information from a VOB database, format the data, and send it to standard output. (In most cases, the information is stored in event records, written by the command when it creates or modifies an object in a VOB. See the **events_ccase** reference page.) Some of these commands have a **–fmt** option, which you can use to format simple reports on VOB contents. Note that **–fmt** is a mutually exclusive alternative to the **–short** and **–long** options.

The following example shows how output-formatting options affect an **lshistory** command.

cmd-context **lshistory -since 1-Feb util.c**

```
10-Feb.11:21 anne create version "util.c@@/main/rel2_bugfix/1"
 "fix bug: extra NL in time string"
10-Feb.11:21 anne create version "util.c@@/main/rel2_bugfix/0"
10-Feb.11:21 anne create branch "util.c@@/main/rel2_bugfix"
```

cmd-context **lshistory -short -since 1-Feb util.c**

```
util.c@@/main/rel2_bugfix/1
util.c@@/main/rel2_bugfix/0
util.c@@/main/rel2_bugfix
```

cmd-context **lshistory -fmt "\tElement: %-13.13En Version: %Vn\n" -since 1-Feb util.c**

```
Element: util.c Version: /main/rel2_bugfix/1
Element: util.c Version: /main/rel2_bugfix/0
Element: util.c Version: /main/rel2_bugfix
```

(A **\t** escape sequence tabs output to the next tab stop. Tab stops occur at eight-character intervals, except as described in the **annotate** reference page.)

CONVERSION SPECIFICATIONS

A conversion specification identifies a particular data item to display and specifies its display format.

fmt_ccase

Syntax

%[*min*][.*max*][*MODIFIER* [, ...]]*keyletter*

The conversion specification format closely resembles that of the C-language function **printf()**:

- Percent sign (%)
- Optionally, a minimum and/or maximum field display width specifier, of the form *min***.***max* (see *Specifying Field Width* on page 395)
- Optionally (for some conversion specs), one or more modifier characters (uppercase) that specify one or more variants, and/or, a bracket-enclosed parameter (see the **%*a*** conversion specification)
- A key letter (lowercase), which indicates the kind of data to display

Unlike **printf()** specifiers, conversion specifications are not replaced by arguments supplied elsewhere on the command line; they are replaced automatically by **cleartool** or Attache, usually with field values extracted from event records.

These are the conversion specifications:

%**a**

All attached attributes. Attributes are listed as *attr-name=value* pairs. These pairs are enclosed in parentheses and separated by a comma-space combination (**,SPACE**). Variants:

%**Na**	No commas. Suppress the parentheses and commas in attribute list output; separate multiple attributes with spaces only.
%**Sa**	Value only. Display attribute values only (rather than *attr=value*)
%**[***attype***]a**	This attribute only. Display only the specified attribute, if it has been attached to the object

%**c**

Comment string. The user-supplied or system-generated comment stored in an event record. A newline character is appended to the comment string for display purposes only. Variant:

%**Nc**	No newline. Do not append a newline character to the comment string.

%**d**

Date/Time. The time stamp of the operation or event, in *date***.***time* format. Variants:

%**Sd**	(Short) Date only.
%**Vd**	(Very long) Day of week, date, and time.
%**Nd**	(Numeric) Date and time in numeric form — *yyyymmdd***.***time* (*time* reported in 24-hour format).

%Ad	Age in days.
%MAd	Age in months.
%BAd	Age as a bar graph (longer bars for more recent events). A bar graph is drawn as a sequence of 0-5 number signs (`#`), representing the elapsed time since the reported operation as follows:

`#####`	Less than a week
`####`	Less than a month
`###`	Less than three months
`##`	Less than six months
`#`	Less than a year
	More than a year

%OAd	Age as a bar graph (longer bars for older events). A bar graph is drawn as a sequence of 0-5 number signs (`#`), representing the elapsed time since the reported operation as follows:

`#####`	More than a year
`####`	Less than a year
`###`	Less than six months
`##`	Less than three months
`#`	Less than a month
	Less than a week

%e

Event kind; a brief description of the event. The event kind is derived from an event record's name, object kind, and operation kind fields. Sample event kinds:

```
create version
create branch
make hyperlink "Merge" on version
make label "REL2" on version
lock branch type
```

%f

Checked-out version information — For an element checked out to your view, the version ID of the checked-out element; for an element that is not checked out to your view, displays nothing. Variants:

%Rf	Checkout status — `reserved` or `unreserved`.
%Tf	View tag — the view-tag of the view that checked out the element.
%[*text*]f	Text — Displays *text* as a prefix to the version ID.

%h

Name of the host where the event originated (the host on which the user **%u** was running that user caused the event). The host name is as reported by **uname(2)** (UNIX) or as stored in the **ComputerName** key in the Windows Registry (Windows).

For a VOB replica, **%h** displays the name of the host at which the **mkreplica –export** command that created the replica was entered. For the original replica in a family, this is the host where the original VOB was located when the first **mkreplica –export** command was entered.

%l

Labels — For versions, all attached labels; the null string otherwise. Labels are output as a comma-separated list, enclosed in parentheses. A **<SPACE>** character follows each comma. Variants:

%Cl Max labels — Specify the maximum number of labels to display with the *max-field-width* parameter (see *Specifying Field Width* on page 395). If there are more labels, `"..."` is appended to the output. If no *max-field-width* is specified, the maximum default value is 3.

%Nl No commas — Suppress the parentheses and commas in label list output; separate labels with spaces only.

%m

Object kind — The kind of object involved in the operation. For example:

```
file element
branch
version
stream
derived object
branch type
label type
```

Variant:

%Km Object selector kind — For example, **brtype** or **lbtype**. For more information about object selectors, see the **cleartool** reference page.

%n

Name of object — For a file system object, the extended pathname (including the version ID for versions, and the DO ID for derived objects); for a type object, its name. Variants:

%Dn Database identifier (DBID) — The unique database identifier of the object.

%En Element name — For a file system object, its standard file or element name, or its pathname; for a type object, its name.

%Ln Leaf name — For any named object, its simple name. The terminal node of a pathname. This modifier can be combined with others.

%On Object identifier (OID) — The unique identifier of a VOB object.

%Sn Short name — For a version, a short form of the version ID: *branch-pathname*/*version-number*. For other objects, the null string.

%PSn Predecessor Short name — For a version, a short form of the predecessor version's version ID: *branch-pathname*/*version-number*. For other objects, the null string.

%Vn Version ID — For a version or derived object, the version ID; for other objects, the null string.

%PVn Predecessor Version ID — For a version, the predecessor version's version ID; for other objects, the null string.

%Xn Extended name — Same as default **%n** output, but for checked-out versions, append the .extension `@@\`*branch-pathname*`\CHECKEDOUT`. For non-file-system objects, prints the object selector. For more information about object selectors, see the **cleartool** reference page.

%o

Operation kind — The operation that caused the event to take place; commonly, the name of a **cleartool** subcommand or an Attache command. For example:

```
mkelem
mklabel
checkin
checkout
```

See the **events_ccase** reference page for a complete list of operations and the commands that cause them.

%[*p*]p

Property value — Displays the value of the property specified in square brackets. The following tables list variants and the objects to which they apply. For ClearCase and ClearCase LT variants, see Table 8. For UCM variants, see Table 9. For MultiSite variants, see Table 10.

Table 8 Variants for ClearCase and ClearCase LT Objects

Variant	Applies to	Description
%[name]p	All objects	Same as **%n**, including variants.
%[object_kind]p	All objects	Kind of object. For example: `version`, `file element`, `directory element`, `versioned object base`, `replica`, `branch type`, and so on.

Table 8 Variants for ClearCase and ClearCase LT Objects

Variant	Applies to	Description
%[locked]p	All objects that can be locked	Lock status of the object: `locked`, `unlocked`, or `obsolete`.
%[activity]p	Versions	Activity whose change set contains the specified version.
%[version_predecessor]p	Versions	Version ID (branch pathname and version number) of the version's predecessor version.
%[type]p	Versions, elements	Name of version or element's element type (see **type_manager** for a list of element types); not to be confused with the object kind (for which the conversion specification is **%m**).
%[triggers]p	Elements	List of trigger types attached to element. Does not list all-element triggers. The list is displayed in the following format: (*trtype*, *trtype*, *trtype*, ...)
%[triggers]Np	Elements	Suppresses parentheses and commas.
%[pool]p	Elements, shared derived objects	For an element, name of source pool. For a shared DO, name of DO pool.
%[pool]Cp	Elements	Name of cleartext pool.
%[pool]Dp	Shared derived objects	Name of derived object pool.
%[pool]Sp	Elements	Name of source pool.
%[DO_kind]p	Derived objects	Kind of derived object: `shared`, `unshared`, `non-shareable`.
%[DO_ref_count]p	Derived objects	Reference count for derived object.
%[slink_text]p	VOB symbolic links	Target of symbolic link, as displayed by **cleartool ls**.

Table 8 Variants for ClearCase and ClearCase LT Objects

Variant	Applies to	Description
%[slink_text]Tp	VOB symbolic links	Target of symbolic link, after link is traversed.
%[type_scope]p	Metadata object types	Object type's scope. • `ordinary` means that use of the type is limited to the current (or specified) VOB • `global` means that the VOB is an administrative VOB and the type can be used in any client VOB of the admin VOB or in any client VOB of a lower-level Admin VOB within an Admin VOB hierarchy • `local copy` means that the type has been copied to the VOB from the Admin VOB that contains the master version of the type's definition
%[type_constraint]p	Branch types, label types	Constraint on type object: `one version per element` or `one version per branch`.
%[trigger_kind]p	Trigger types	Kind of trigger type: `element trigger`, `all element trigger`, `type trigger`.
%[msdostext_mode]p	VOBs	State of MS-DOS text mode setting for VOB: `enabled` or `disabled`.
%[group]p		Group name.
%[owner]p (Windows only)		Login name of the object's current owner.
%[owner]Fp (UNIX only)		Login name of the objects' current owner. The optional **F** argument lists the owner's full name.

The variants in Table 9 apply only to UCM objects.

Table 9 Variants for UCM Objects

Variant	Applies to	Description
%[crm_record_id]p	UCM activities	A ClearQuest record ID.

Table 9 Variants for UCM Objects

Variant	Applies to	Description
%[stream]p	UCM activities	The stream containing the activity
%[versions]p	UCM activities	Space separated list of versions in activity's change set
%[component]p	UCM baselines	The component associated with the baseline
%[depends_on]p	UCM baselines	The baselines that the composite baseline directly depends on
%[depends_on_closure]p	UCM baselines	All of the baselines in the full dependence graph of a composite baseline
%[label_status]p	UCM baselines	The label status of a baseline: full, incremental, or unlabeled
%[root_dir]p	UCM components	The root directory for the component
%[contains_folders]p	UCM folders	Subfolders of the folder
%[contains_projects]p	UCM folders	Projects contained by the folder
%[folder]p	UCM folders	The parent folder for the folder
%[def_rebase_level]p	UCM projects	The promotion level required of a baseline before it can be used as the source of a rebase operation
%[folder]p	UCM projects	The parent folder for the project
%[istream]p	UCM projects	The project integration stream
%[mod_comps]p	UCM projects	The modifiable components for a project
%[rec_bls]p	UCM projects	The recommended baselines of a project's integration stream
%[activities]p	UCM streams	Activities that are part of the stream
%[config_spec]p	UCM streams	Config spec of object
%[def_deliver_tgt]	UCM streams	The default stream that the stream will deliver to
%[dstreams]p	UCM streams	The child streams of an integration stream or a development stream

Table 9 Variants for UCM Objects

Variant	Applies to	Description
%[found_bls]p	UCM streams	The foundation baselines for the stream
%[latest_bls]p	UCM streams	Latest baseline in each component in a stream's configuration
%[project]p	UCM streams	The project the stream is part of
%[rec_bls]p	UCM streams	The recommended baselines of an integration stream or a parent development stream
%[views]p	UCM streams	Views attached to the stream

The variants in Table 10 apply only to objects in replicated VOBs (ClearCase MultiSite product).

Table 10 Variants for Replicated Objects

Variant	Applies to	Description
%[master]p	All objects that have mastership	Name of object's master replica
%[master]Op	All objects that have mastership	OID of object's master replica

Table 10 Variants for Replicated Objects

Variant	Applies to	Description
%[reqmaster]p	Replicas, branch types, branches	Request for mastership status of the object. For a replica: • `disabled` means that requests for mastership are not enabled in the replica • `enabled` means that requests for mastership are enabled in the replica For a branch type: • `denied for all instances` means that requests for mastership of any instance of the branch type are denied • `allowed for all instances` means that requests for mastership of any instance of the branch type are allowed (unless mastership requests for the specific branch are denied) • `denied for branch type` means that requests for mastership of the branch type are denied • `allowed for branch type` means that requests for mastership of the branch type are denied For a branch: • `denied` means that requests for mastership of the branch are denied • `allowed` means that requests for mastership of the branch are allowed
%[type_mastership]p	Attribute types, hyperlink types, label types	Kind of mastership of the type: `shared` or `unshared`.
%[replica_name]p	VOBs	Replica name of the specified VOB.
%[vob_replication]p	VOBs	Replication status of VOB: `replicated` or `unreplicated`.
%[replica_host]p	Replicas	Name of replica host.

%[*c*]t

Starting column number — Starts printing at the column number specified in square brackets. An overflow condition exists if the current position on the line is beyond the starting column number. By default, when an overflow condition occurs, the **%t** directive is ignored. Variants:

%[*c*]Nt When an overflow condition occurs, print a newline and resume printing at the starting column number.

%[*c*]St When an overflow condition occurs, print one space before printing the next value.

%[*c*]Tt When an overflow condition occurs, print a tab before printing the next value.

%u

Login name of the user associated with the event. Variants:

%Fu Full name of the user. This information is taken from the password database.

%Gu Group name of the user.

%Lu Login name and group of the user, in the form *user* **.** *group*.

%%

Percent character (%).

Specifying Field Width

A conversion specification can include an optional field width specifier, which assigns a minimum and/or maximum width, in characters, to the data field display. For example, the conversion specifier **%10.15Lu** will display, for each output line, the user's login name and group with a minimum of 10 characters (space padded if necessary) but not more than 15.

Usage rules:

- A single number is interpreted as a minimum width.
- To supply only a maximum width, precede the number with a decimal point (for example, **%.10En**) or with a zero and decimal point (**%0.10En**).
- To specify a constant display width, set the minimum and maximum widths to the same value (**%20.20c**).
- Values smaller than the specified minimum width are right-justified (padded left). A negative minimum width value (**%–20.20c**) left-justifies short values.
- Values longer than the specified maximum width are truncated from the right. A negative maximum width value (**%15.–15Sn**) truncates long values from the left.

- A maximum width specifier has special meaning when used with the **%Cl** specifier. For example, **%.5Cl** prints a version's first five labels only, followed by `"..."`.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Format the output from **lsco –cview**.

  cmd-context **lsco -cview -fmt "\t%-10.10n (from %8.8PVn) %d %u\n"**

  ```
  util.c (from /main/23) 24-Jun-99.14:12:48 anne
  main.c (from /main/46) 23-Jun-99.18:42:33 anne
  msg.c (from ugfix/11) 23-Jun-99.10:45:13 anne
  msg.h (from bugfix/3) 22-Jun-99.14:51:55 anne
  ```

- Format the event history of a file element. (The command line, including the quoted format string, constitutes a single input line. The input line below is broken to improve readability. Spaces are significant.)

  cmd-context **lshistory -fmt "OBJ-NAME: %-20.20n\n USER: %-8.8u\n DATE: %d\\n OPERATION:\t%-12.12o\n OBJ-TYPE:\t%-15.15m\n EVENT:\t%e\n COMMENT: %c\n" util.c**

  ```
  OBJ-NAME: util.c@@main3
      USER: anne
       DATE: 10-May-99.09:24:38
         OPERATION: checkin
         OBJ-TYPE: version
         EVENT: create version
       COMMENT: fix bug r2-307
  ```

```
OBJ-NAME: util.c@@main2
     USER: anne
     DATE: 10-May-99.09:09:29
       OPERATION: checkin
       OBJ-TYPE: version
       EVENT: create version
    COMMENT: ready for code review
```

- Describe a checked-out element, **util.c**.

  cmd-context **describe -fmt "\tVer:\t%f\n\tPrefix:\t%[MY TEXT]%f\n\t Status:\t%Rf\n\tView:\t%Tf\n" util.c**

  ```
          Ver:    /main/23
          Prefix: MY TEXT/main/23
          Status: reserved
          View:   eba_view
  ```

- Display the type of a file element.

  cmd-context **describe -fmt "Type: %[type]p\n" util.c@@**

  ```
  Type: text_file
  ```

- Display the target of a symbolic link and the target after the link is traversed.

  cmd-context **describe –fmt "%n\t%[slink_text]p\t%[slink_text]Tp\n" link1.txt**

  ```
  link1.txt     file.txt     ..\dev\file.txt
  ```

- Display the master replica of all label types in a VOB replica.

  cmd-context **lstype –fmt "Label type: %n\tMaster: %[master]p\n" –kind lbtype**

  ```
  Label type: BACKSTOP    Master: evanston@/vobs/tromba
  Label type: CHECKEDOUT  Master: evanston@/vobs/tromba
  Label type: LATEST      Master: evanston@/vobs/tromba
  Label type: V3.4        Master: paris@/vobs/tromba
  ```

- Display the name of an element, using tabular format. The command is a single input line; line breaks are added for readability.

  cmd-context **describe -fmt**
  "%[4]tName:%[6]t%[name]p\n
  %[4]tName:%[6]Nt%[name]p\n
  %[4]tName:%[6]St%[name]p\n
  %[4]tName:%[6]Tt%[name]p\n" util.c

  ```
     Name:util.c@@\main\30
     Name:
       util.c@@\main\30
     Name: util.c@@\main\30
     Name:        util.c@@\main\30
  ```

- Mimic the output from **lshistory –long**. Note the use of single quotes to enclose the format string, which includes literal double quotes.

  cleartool lshistory -fmt '%d   %Fu (%u@%h)\n %e "%n"\n "%Nc"\n' util.c

  ```
  11-May-99.09:24:38     Anne Duvo (anne@neptune)
    create version "util.c@@/main/3"
    "fix bug r2-307"
  10-May-99.09:09:29     Ravi Singha (ravi@mercury)
    create version "util.c@@/main/2"
    "ready for code review"
  .
  .
  .
  ```

- Mimic the output from **lshistory –long**. Note that in **cleartool** single-command mode, backslashes (\) are used to escape double quotes in the format string.

  cleartool lshistory -fmt "%d   %Fu (%u@%h)\n %e \"%n\"\n \"%Nc\"\n" util.c

  ```
  11-May-99.09:24:38     Anne Duvo (anne@neptune)
    create version "util.c@@\main\3"
    "fix bug r2-307"
  10-May-99.09:09:29     Ravi Singha (ravi@mercury)
    create version "util.c@@\main\2"
    "ready for code review"
  .
  .
  .
  ```

- Describe the element **main.c** in detail. This example illustrates many of the conversion specifications (but does not use field width specifiers). Again, the command is a single input line; line breaks are added for readability.

  cmd-context **describe -fmt "Name (default): %n\n**
  Element name: %En\n
  Leaf name: %Ln\n
  Short name: %Sn\n
  Predecessor short name: %PSn\n
  Version ID: %Vn\n
  Predecessor version ID: %PVn\n
  Extended name: %Xn\n
  Attributes: %a\n
  Attr values only: %Sa\n
  Attrs without commas or parens: %Na\n
  This attr only: %[Tested]a\n
  Comment: %c
  Date/Time: \tdefault: %d\n
  \t\tshort: %Sd\n

\t\tlong: %Vd\n
Age in days: %Ad\n
Age in months: %MAd\n
Age graph (long = new): %BAd\n
Age graph (long = old): %OAd\n
Host: %h\n
Labels: %Cl\n
Labels without commas or parens: %Nl\n
Object kind: %m\n
Operation kind: %o\n
Event kind: %e\n
User (default): %u\n
Full user name: %Fu\n
Group name: %Gu\n
Long name: %Lu\n\n" main.c

```
Name (default): main.c@@/main/34
Element name: main.c
Leaf name: 34
Short name: /main/34
Predecessor short name: /main/33
Version ID: /main/34
Predecessor version ID: /main/33
Extended name: main.c@@/main/34
Attributes: (Tested="yes", QAlevel=4, Responsible="anne")
Attr values only: ("yes", 4, "anne")
Attrs without commas or parens: Tested="yes" QAlevel=4 Responsible="anne"
This attr only: (Tested="yes")
Comment: still needs QA
Date/Time: default: 30-Jul-99.15:02:49
 short: 30-Jul-99
 long: Tuesday 07/30/99 15:02:49
Age in days: 42
Age in months: 1
Age graph (long = new): ####
Age graph (long = old): ##
Host: neptune
Labels: (Rel3.1C, Rel3.1D, Rel3.1E)
Labels without commas or parens: Rel3.1C Rel3.1D Rel3.1E
Object kind: version
Operation kind: checkin
Event kind: create version
User (default): anne
Full user name: Anne Duvo
Group name: dev
Long name: anne.dev
```

SEE ALSO

annotate, **cleartool**, **describe**, **events_ccase**, **lsactivity**, **lsbl**, **lscheckout**, **lscomp**, **lsdo**, **lsfolder**, **lshistory**, **lslock**, **lspool**, **lsproject**, **lsreplica**, **lsstream**, **lstype**, **reqmaster**, **type_manager**

get

In ClearCase and ClearCase LT, copies a specified version of a file element into a snapshot view. In Attache, downloads files to an Attache workspace

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and ClearCase LT:

  get –to *dest-pname pname*

- Attache:

  get [**–r·ecurse**] [**–compress**] [**–ove·rwrite** | **–nov·erwrite**] [**–pti·me**] [**–to** *pname*]
  [**–log** *pname*] *pname*...

DESCRIPTION

ClearCase and ClearCase LT

Use the **get** command to copy a specified version of a file element into your snapshot view. You must issue the **get** command from the root directory of a snapshot view or any directory below it.

You can use this command as follows:

- To read versions of file elements that are not selected by the view's config spec, either because the element is not specified by a load rule, or because you want to see a version of a loaded element other than the one currently in the view. You cannot perform ClearCase or ClearCase LT operations on these nonloaded versions copied into your view with the **get** command.

- To get an updated copy of a version currently loaded in your view.

The **get** command copies only file elements into a view.

Attache

This command downloads the specified files to the workspace.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

ClearCase and ClearCase LT

SPECIFYING THE DESTINATION FILE NAME. *Default*: None.

dest-pname
Specifies a pathname for the version. If you do not specify a directory name, the file is copied into the current directory. By requiring a destination pathname, the **get** command is prevented from overwriting any version already loaded into your view.

SPECIFYING THE FILE TO COPY. *Default:* None.

pname
Specifies the version of the file element to copy into the view. Use a version-extended pathname to copy a version other than the one currently loaded in the view. Specifying a pathname that contains a symbolic link causes the link target to be downloaded.

Attache

SPECIFYING THE FILES TO BE DOWNLOADED. *Default:* None.

pname...
Specifies the files, directories, and/or links to be downloaded. Downloading a pathname containing a symbolic link, downloads a copy of the file or directory the link points to, rather than the link itself. Wildcard patterns are expanded with reference to the view. In addition, arguments of the form @*pname* can be used to add the contents of the local file *pname* as pathname arguments. The pathname arguments can contain wildcards (see the **wildcards** reference page), and must be listed in the file one per line, or also be of the form @*pname*. Specifying a relative pathname for @*pname* begins from Attache's startup directory, not the working directory, so a full local pathname is recommended.

SPECIFYING HOW THE FILES ARE TO BE DOWNLOADED. *Default:* When a directory is specified, its file contents are downloaded. If a destination file that is identical in contents with the source file already exists, it is not overwritten. If an existing destination file is read-only and differs from the source, it is always overwritten. If the destination file exists and is writable, an overwrite query is issued.

–ove·rwrite

Suppresses the query and causes all writable files to be overwritten.

–nov·erwrite

Suppresses the query and causes no writable file to be overwritten.

–to *pname*

Specifies a destination file name or directory. If the specified destination is a directory, it becomes a prefix for each downloaded file name. If the specified destination is a file, or does not exist, then only one source argument can be specified, and it must be a file.

–pti·me

Causes the last-modified time stamp of the destination file to be set to that of the source file. **–ptime** has no effect on directories.

–compress

Causes files to be compressed while being uploaded and uncompressed after the transfer to improve performance over slow communications lines.

NOTE: On Windows, the default behavior for this option can be set with the **Preferences** command on the **Options** menu.

HANDLING OF DIRECTORY ARGUMENTS. *Default:* For each *pname* that specifies a directory element, **get** downloads the contents of that directory, but not the contents of any of its subdirectories.

NOTE: This includes directories in the version-extended namespace, which represent elements and their branches. For example, specifying **foo.c@@\main\bug403** as an argument downloads the contents of that branch: all the versions on the branch, providing the resulting filenames are valid on your client host.

–r·ecurse

Includes files from the entire subtree below any subdirectory included in the top-level listing. Directories are created as necessary and the current directory is taken into account if relative patterns are given.

SPECIFYING A FILE TRANSFER LOG. *Default:* None.

–log

Specifies a log file for the operation. The log file lists the workspace-relative pathname of each file transferred by the Attache **get** command, as well as an indication of any errors that occur during the operation. Log file pathnames are absolute, not relative to the current workspace root.

Each line in a log file is a comment line, except for the names of files which were not transferred. Log files, therefore, can be used as indirect files to redo a file transfer operation.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

ClearCase and ClearCase LT

- Copy the version loaded in the view into the current directory.

  cmd-context **get –to foo.c.temp foo.c**

- Copy **/dev/hello_world/foo.c@@/main/2** into the current directory.

  cmd-context **get –to foo.c.temp /dev/hello_world/foo.c@@/main/2**

- Copy **\dev\hello_world\foo.c@@\main\2** into the **C:\build** directory.

  cmd-context **get –to C:\build\foo.c.temp \dev\hello_world\foo.c@@\main\2**

Attache

- Download all files in the current directory to the current Attache workspace, keeping their original time stamps. Do not overwrite any files writable already in the current workspace.

  cmd-context **get –noverwrite –ptime ***

- Download to the current Attache workspace the file **file.c** in the current directory and rename it **tempfile.c**.

  cmd-context **get –to tempfile.c file.c**

- Download to the current Attache workspace a more recent version of the file **file.c** in the current directory and compress while transferring.

  cmd-context **get –compress file.c**
  `Overwrite existing file /jed_ws/my_vob_tag/src/file.c? [no]` **y**

- Download to the current Attache workspace all of the files and subdirectories beneath the directory **my_dir**.

  cmd-context **get –r my_dir**

- Download to the current Attache workspace all of the files specified in **c:\users\jed\get_file**, and do not overwrite writable files. (Note that the output from the **wshell** command appears in a separate window.)

  <u>*cmd-context*</u> **wshell type c:\users\jed\get_file**
  ```
  \proj\src\*.c
  \proj\include\*.h
  ```
  <u>*cmd-context*</u> **get –noverwrite @c:\users\jed\get_file**

SEE ALSO

checkin, **checkout**, **config_spec**, **update**, **version_selector**

getcache

Displays cache information

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- Display/reset statistics for a view:

  getcache –vie·w [**–a·ll** | **–s·hort**] [**–reset**] { **–cvi·ew** | *view-tag* }

- Display the default cache size for the current host:

  getcache –vie·w –hos·t

- Display the site-wide default size for view caches:

  getcache –vie·w –sit·e

- ClearCase dynamic views and Attache—Display cache information for the MVFS:

  getcache –mvfs [**–s·hort**] [**–reset**]

DESCRIPTION

The **getcache** command displays cache information for a view. In ClearCase dynamic views and and Attache, you can also use **getcache** to get information on the multiversion file system (MVFS). View cache information includes cumulative statistics about the number of operations performed and the size of each of the view's caches. (You can specify the total size for the view's caches with the **setcache** command; that total is allocated among the individual caches.) **getcache** can also reset the view or MVFS statistics with the **–reset** option.

NOTE: Two sets of statistics are kept for a view: the set displayed by **–all**, which is reset only when the **view_server** is restarted (with **endview –server** or by rebooting); and the normal set, which you can zero with the **–reset** option.

With the **–host** option, **getcache** displays the default size of the view cache for the current host. With the **–site** option, **getcache** displays the site-wide default size for view caches. With the **–mvfs** option, **getcache** displays information about a host's MVFS caches, which are used to optimize file-system performance. (For more information on optimizing performance, see the chapters on performance tuning in the *Administrator's Guide*.)

getcache can sometimes reports cache use greater than 100%. The **Object cache**, in particular, can show usage exceeding 100% because other objects, including cache objects, can reference the **Object cache**. Usually, this means the **Object cache** size is too small compared to the sizes of other caches.

RESTRICTIONS

getcache -mvfs -reset requires **root** privileges on UNIX and local administrator privileges on Windows. No other restricitons apply

OPTIONS AND ARGUMENTS

SPECIFYING THE CACHE INFORMATION TO DISPLAY. *Default:* None.

–vie·w

Displays cache information for a single view.

–vie·w –hos·t

Displays the default cache size for the current host. If this value has not been set, **getcache** displays the following message:

```
No host-wide default view cache size is known.
```

This value is stored in the **view_cache_size** file (in **/var/adm/atria/config/** (UNIX) or *ccase-home-dir***var\config** (Windows)) and is set with **setcache –view –host**.

–vie·w –sit·e

Displays the site-wide default size for view caches. If this value has not been set, **getcache** displays the following message:

```
No site-wide default view cache size is known.
```

This value is stored in the ClearCase or ClearCase LT site config registries and is set with **setcache –view –site** or **setsite**.

–mvfs

Displays cache information for the MVFS. These values are set with **setcache –mvfs**.

SPECIFYING HOW MUCH INFORMATION TO DISPLAY. *Default:* With **–view**, displays statistics gathered since the last reset and the current cache sizes. With **–mvfs**, displays current cache sizes/utilizations and advice on cache sizing.

–a·ll
: Displays view statistics since the time that the **view_server** was started. These statistics are not reset when you execute **getcache –reset**.

–view –s·hort
: Displays only cache sizes.

–mvfs –s·hort
: Displays only cache sizes and utilizations.

RESETTING VIEW STATISTICS. *Default:* The counters for the normal set of view statistics keep running.

–reset
: With the **-view** option, displays current statistics, resets them to zero, and prints a summary to the view log. With the **-mvfs** option, displays current statistics, then resets them to zero.

SPECIFYING A VIEW. *Default:* None.

–cvi·ew
: Displays or resets statistics for the current view.

view-tag
: Specifies the view whose statistics are displayed or reset.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Display cache information for the **cep_dev** view:

  cmd-context **getcache –view cep_dev**

```
Lookup cache:    29% full,   1121 entries ( 56.8K),  15832 requests,  75% hits
Readdir cache:    4% full,     24 entries ( 36.5K),   4159 requests,  83% hits
Fstat cache:     31% full,    281 entries (105.1K),  55164 requests, 100% hits
Object cache:    26% full,   1281 entries (176.6K),  40626 requests,  72% hits
Total memory used for view caches: 375.0Kbytes
The current view server cache limits are:
Lookup cache:             201312 bytes
Readdir cache:            838860 bytes
Fstat cache:              352296 bytes
Object cache:             704592 bytes
Total cache size limit:  2097152 bytes
```

- Display cache information for the MVFS:

  cmd-context **getcache –mvfs**

```
Mnodes: (active/max)    1043/4096 (25.464%)
Mnode freelist:         885/900 (98.333%)
Cltxt freelist:         41/819 (5.006%)


DNC:   Files:           219/800 (27.375%)
       Directories:     108/200 (54.000%)
       ENOENT:          106/400 (26.500%)


RPC handles:            2/5 (40.000%)


Attribute cache miss summary (for tuning suggestions, see the
documentation for administering ClearCase):
Attribute cache total misses:             1215        (100.00%)
Close-to-open (view pvt) misses:           434        ( 35.72%)
Generation (parallel build) misses:          0        (  0.00%)
Cache timeout misses:                      297        ( 24.44%)
Cache fill (new file) misses:                1        (  0.08%)
Event time (vob/view mod) misses:              484         ( 39.84%)
```

SEE ALSO

mkview, **mvfscache**, **setcache**, **setsite**, *Administrator's Guide*

getlog

Displays UNIX log files or Windows log entries

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and Attache—Display logs graphically:

  getlog –g·raphical [**–hos·t** *hostname* | **–cvi·ew** | **–tag** *view-tag* | **–vob** *pname-in-vob*]

- ClearCase and Attache—Display logs nongraphically:

  getlog [**–las·t** [*#_lines*] | **–fu·ll** | **–sin·ce** *date-time* | **–aro·und** *date-time* [*#_minutes*]]
  [**–hos·t** *hostname* | **–cvi·ew** | **–tag** *view-tag* | **–vob** *pname-in-vob*]
  { **–a·ll** | *log-name ...* }

- ClearCase and Attache—Display the current set of logs:

  getlog –inq·uire [**–hos·t** *hostname*]

- ClearCase LT on UNIX—Display logs:

  getlog [**–las·t** [*#_lines*] | **–fu·ll** | **–sin·ce** *date-time* | **–aro·und** *date-time* [*#_minutes*]]
  { **–a·ll** | *log-name ...* }

- ClearCase LT on Windows—Display logs graphically:

  getlog –g·raphical

- ClearCase LT on Windows—Display logs nongraphically:

getlog [**–las**·**t** [#*_lines*] | **–fu**·**ll** | **–sin**·**ce** *date-time* | **–aro**·**und** *date-time* [#*_minute*s]]
{ **–a**·**ll** | *log-name* ... }

- ClearCase LT—Display the current set of logs:

getlog –inq·**uire**

DESCRIPTION

The **getlog** command displays extracts from one or more log files (UNIX) or log entries (Windows). Run **getlog –inquire** to return a list of the available logs.

NOTE: If the host for which you are trying to view logs is having problems, you may have to

- UNIX—Go to the log directory (**/var/adm/atria/log**) and use standard commands to look at the log files
- Windows—Open the Windows NT Event Viewer to view the log entries

Attache—Using getlog

When a command fails because of an error on the helper host, if the helper is still active, you can use **getlog** on the client to fetch the helper error messages.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

DISPLAYING ENTRIES GRAPHICALLY. *Default:* Entries are displayed in the current command window.

–g·**raphical**
Starts a log browser.

SPECIFYING LOG EXTRACTS. *Default:* Displays the last 10 lines for the specified logs (equivalent to specifying **–last 10**).

–las·**t** [#*_lines*]
Gets the last #*_lines* lines for the specified logs. The default value of #*_lines* is 10.

–fu·**ll**
Gets the complete contents of the specified logs.

–sin·**ce** *date-time*
–aro·**und** *date-time* [#*_minute*s]
The **–since** option gets log entries made since *date-time*. The **–around** option gets log entries #*_minute*s minutes either side of *date-time*. The default value of #*_minute*s is 10. If you specify either of these options for an unformatted log file, **getlog** prints an error. (**getlog –inquire** indicates unformatted logs.)

The *date-time* argument can have any of the following formats:

*date***.***time* | *date* | *time* | **now**
where:

date	:=	*day-of-week* | *long-date*
time	:=	*h*[*h*]**:***m*[*m*][**:***s*[*s*]] [**UTC** [[**+** | **-**]*h*[*h*][**:***m*[*m*]]]]
day-of-week	:=	**today** | **yesterday** | **Sunday** | ... | **Saturday** | **Sun** | ... | **Sat**
long-date	:=	*d*[*d*]**–***month*[**–**[*yy*]*yy*]
month	:=	**January** | ... | **December** | **Jan** | ... | **Dec**

Specify *time* in 24-hour format, relative to the local time zone. If you omit the time, the default value is **00:00:00**. If you omit *date*, the default is **today**. If you omit the century, year, or a specific date, the most recent one is used. Specify **UTC** if you want to resolve the time to the same moment in time regardless of time zone. Use the plus (+) or minus (-) operator to specify a positive or negative offset to the UTC time. If you specify **UTC** without hour or minute offsets, Greenwich Mean Time (GMT) is used. (Dates before January 1, 1970 Universal Coordinated Time (UTC) are invalid.)

SPECIFYING WHICH HOST'S LOGS TO DISPLAY. *Default for ClearCase and Attache:* Get the current host's log files. These options are not applicable to ClearCase LT, where the host is always the ClearCase LT server host.

–hos·t *hostname*
Gets logs from *hostname*.

–cvi·ew
Gets logs from the current view's view host.

–tag *view-tag*
Gets logs from the view server host of the specified view.

–vob *pname-in-vob*
Gets logs from the VOB server host of the specified VOB.

SPECIFYING THE LOGS TO DISPLAY. *Default:* None. You must specify one or more log names, **–all** to view all logs, or **–inquire** to return the list of available logs.

–inq·uire
Returns the list of available logs, which can vary with the host's installed product set and configuration. Unformatted logs are annotated with the string `(unformatted)`.

–a·ll
Displays every available log file.

log-name ...
Specifies one or more logs of interest. Use **–inquire** to list valid log names.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Return the list of available logs.

  cmd-context **getlog –inquire**

```
     vobsnap ClearCase VOB snapshot log
      vobrpc ClearCase vobrpc_server log
vob_scrubber ClearCase vob scrubber log (unformatted)
         vob ClearCase vob_server log
        view ClearCase view_server log
...
```

- Display the last 10 lines of the current host's **vob_server** log. For information about **vob_server**, see the *Administrator's Guide*.

cmd-context **getlog vob**

```
==========================================================================
Log Name: vob                   Hostname: superior   Date: 24-Feb-99.16:51:02
Selection: Last 10 lines of log displayed
--------------------------------------------------------------------------
02/22/99 00:52:52 vob_server(183): debug=0, verbose=0, dont_fsync=0, check_for_nulls=0,
wait_time=500ms
02/22/99 00:52:52 vob_server(183): UID: 1049610 GID: 1049650
02/22/99 00:52:52 vob_server(183): using C:\USERS\vobs\ms_test.vbs, on host: superior
02/22/99 00:52:51 vob_server(183): addr = 0, port= 1247
02/22/99 00:50:37 vob_server(167): debug=0, verbose=0, dont_fsync=0, check_for_nulls=0,
wait_time=500ms
02/22/99 00:50:37 vob_server(167): UID: 1049610 GID: 1049650
02/22/99 00:50:37 vob_server(167): using C:\USERS\vobs\smg_bld.vbs, on host: superior
02/22/99 00:50:36 vob_server(167): addr = 0, port= 1201
02/19/99 01:10:47 vob_server(166): debug=0, verbose=0, dont_fsync=0, check_for_nulls=0,
wait_time=500ms
02/19/99 01:10:47 vob_server(166): UID: 1049610 GID: 1049650
=============================================================================
```

- Display host **saturn**'s view log entries within 20 minutes of 4:00 P.M. on August 24.

cmd-context **getlog –host saturn –around 24-Aug.16:00 20 view**

```
===========================================================================
Log Name: view                  Hostname: saturn      Date: 25-Aug-99.17:28:57
Selection: Lines between 24-Aug-99.15:40:00 and 24-Aug-99.16:20:00 displayed
---------------------------------------------------------------------------
08/24/99 16:02:26 view_server(4904): Using 2097152 bytes of cache
08/24/99 16:02:26 view_server(4904): Db initialized
08/24/99 16:02:26 view_server(4904): Db initialized
08/24/99 16:02:26 view_server(4904): View server addr = 0, port= 35200
08/24/99 16:02:25 view_server(4904): View server addr = 0, port= 40227
===========================================================================
```

UNIX FILES

/var/adm/atria/log/*

SEE ALSO

Administrator's Guide

help

Displays help on command usage, or (on Attache) a reference page

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command
MultiSite	multitool subcommand

Platform
UNIX
Windows

SYNOPSIS

h·elp [*command-name*]

command-name **–h·elp**

DESCRIPTION

This command does not require a product license.

ClearCase, ClearCase LT, MultiSite

The **help** command displays a usage message for all **cleartool** or **multitool** subcommands, or for one particular subcommand. You can also use **help** as a command option. For example:

cmd-context **lsco –help**

Attache

When used as a command name, this command is a synonym for **man**.

help formats and displays the specified on-line reference page. For local and hybrid commands, or if your helper is running on a UNIX host, you can use any valid command abbreviation or alias. For example:

cmd-context **help uncheckout** *(full command name)*

cmd-context **help uncheck** *(abbreviation – local or hybrid command or UNIX)* *cmd-context* **help unco** *(alias – local or hybrid command or UNIX)*

With no arguments, **help** displays the Attache overview reference page. **man** is a synonym for **help**.

When used as a **–help** command option, it displays a usage message for one particular command. You can use any valid command abbreviation or alias for any command. For example:

cmd-context **lsco –help**

```
Usage: lscheckout | lsco [-long | -short | -fmt format] [-cview]
                        [-brtype branch-type] [-me | -user login-name]
                        [-recurse | -directory | -all | -avobs ] [-areplicas]
                        [pname ...]
```

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

ClearCase, ClearCase LT, MultiSite

SPECIFYING A SUBCOMMAND. *Default:* Displays syntax summaries for all **cleartool** or **multitool** subcommands, grouped by function (not alphabetically).

command-name **–h·elp**
h·elp *command-name*
Displays the syntax summary for one **cleartool** or **multitool** subcommand.

Attache

SPECIFYING THE REFERENCE PAGE. *Default:* Displays the overview reference page for Attache.

command_name
The name (or abbreviation, or alias) of an Attache local or hybrid command; or the name of any other reference page.

GETTING USAGE INFORMATION. *Default:* None.

command-name **–h·elp**
Displays the syntax summary for one Attache command.

EXAMPLES

- (Attache) Display the reference page for the **checkout** command.

  cmd-context **help checkout**

- (ClearCase, ClearCase LT, MultiSite) Display a usage message for the **checkout** command.

  cmd-context **help checkout**

```
Usage: checkout | co [-reserved | -unreserved] [-out dest-pname | -ndata]
                     [-branch branch-pname | -version]
                     [-c comment | -cfile pname | -cq | -cqe | -nc] pname ...
```

- Display a usage message for the **checkout** command using the **–help** option.

  cmd-context **checkout -help**

```
Usage: checkout | co [-reserved | -unreserved] [-out dest-pname | -ndata]
                     [-branch branch-pname | -version]
                     [-c comment | -cfile pname | -cq | -cqe | -nc] pname ...
```

- (Attache) Display the Attache overview reference page.

  cmd-context **help**

- (ClearCase and ClearCase LT on UNIX) Display a usage message for all **cleartool** commands, and redirect the output to a file for future reference.

  cmd-context **help > cleartool_cmd_summary**

SEE ALSO

attache_command_line_interface, **man**

hostinfo

Displays configuration data for one or more hosts

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

hostinfo [**–l·ong**] [**–pro·perties** [**–ful·l**]] [*hostname* ...]

DESCRIPTION

For one or more hosts, the **hostinfo** command displays basic system and ClearCase or ClearCase LT configuration data. When you run **hostinfo** on an Attache client host, **hostinfo** displays information about the ClearCase helper host.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

REPORT FORMAT. *Default:* **hostinfo** displays a one-line report for each host.

–l·ong

Expands the listing to include the host's ClearCase registry region, registry server host, and license server host.

–pro·perties

Reports the following properties of the host:

- Backup registry host
- Registry interoperability region

- MVFS scaling factor and some cache sizes
- Release number and build creation date for all installed ClearCase Product Family products

With the **–full** option, reports the following additional properties for UNIX hosts:

- Installation model (for example, standard or full)
- ClearCase or ClearCase LT installation directory (*ccase-home-dir*)
- Date and time of ClearCase or ClearCase LT installation
- Release area from which ClearCase or ClearCase LT was installed
- Names of all installed components

SPECIFYING HOSTS. *Default:* **hostinfo** displays only local host information.

hostname ...
Specifies one or more network hosts.

EXAMPLES

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Display condensed information about the local host, **mercury**.

  cmd-context **hostinfo**

  ```
  mercury: ClearCase 4.0 (SunOS 5.6 Generic_105181-15 sun4u)
  ```

- Display expanded information about remote host **neptune**.

  cmd-context **hostinfo –long neptune**

  ```
  Client: neptune
    Product: ClearCase 4.0
    Operating system: Windows NT 4.0 (build 1381) Service Pack 5
    Hardware type: Pentium
    Registry host: saturn
    Registry region: devel
    License host: venus
  ```

- Display properties of host **neptune**.

  cmd-context **hostinfo –properties neptune**

```
neptune: ClearCase 4.0 (SunOS 5.6 Generic_105181-15 sun4u)
  Backup registry host: uranus
  Scaling factor to initialize MVFS cache sizes: 1
  MVFS cache sizes:
    Free mnodes: 1800
    Free mnodes for cleartext files: 1800
    File names: 1600
    Directory names: 400
    Names not found: 1600
    RPC handles: 10
  Installed product: ClearCase version 4.0 (Tue Jul 13 11:59:49 EDT 1999)
  Installed product: MultiSite version 4.0 (Tue Jul 13 11:59:49 EDT 1999)
```

SEE ALSO

clearlicense, *Administrator's Guide*

hyperhelp

Displays a help file

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command
MultiSite	command

Platform
UNIX

SYNOPSIS

- Start a HyperHelp viewer:

  hyperhelp [*helpfile*] [**–display**]

- Display version information:

  hyperhelp –version

DESCRIPTION

Most printed documentation for ClearCase, ClearCase LT and MultiSite is provided in online form as hypertext Windows-style help files. The **hyperhelp** command starts a HyperHelp help viewer, in which you can view and navigate the help files.

The **hyperhelp** executable is installed in *ccase-home-dir*/**bin** (*ccase-home-dir* is the directory in which ClearCase or ClearCase LT is installed) and reads help files installed in *ccase-home-dir*/**doc/hlp**.

NOTE: You can also view hypertext reference pages with the **man –graphical** command.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

STARTING A HELP VIEWER. *Default:* The HyperHelp viewer starts up on the current display and does not display a help file.

helpfile

Displays *helpfile* (one of the **.hlp** files in *ccase-home-dir*/**doc/hlp**) in a HyperHelp viewer.

–display

Opens the HyperHelp viewer on a different display.

DISPLAYING VERSION INFORMATION. *Default:* None.

–version

Displays the release number of the HyperHelp viewer and information about your environment.

EXAMPLES:

- Open the Version Tree Browser help file.

  hyperhelp vtree.hlp

- Open the History Browser help file on a different display.

  hyperhelp clearhistory.hlp –display neon

FILES

ccase-home-dir/**bin/hyperhelp**
ccase-home-dir/**doc/hlp/*.hlp**

SEE ALSO

man

import

Creates an element (if one does not exist) corresponding to each selected file or directory in a workspace subtree

APPLICABILITY

Product	Command Type
Attache	command

Platform
UNIX
Windows

SYNOPSIS

import [**–ci**] [**–exc·lude** *exclude-pname*] [**–lca·se**]
[**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**]
pname ...

DESCRIPTION

The **import** command creates an element corresponding to each file or directory name in the workspace, if the element does not already exist in the VOB. For each file in the specified subtree, **import** checks to see whether an element of that name exists in the VOB. If not, **import** invokes **mkelem** on the file.

RESTRICTIONS

For checked-out directories, restrictions are the same as for **checkout**. For created elements, restrictions are the same as for **mkelem**.

OPTIONS AND ARGUMENTS

CHECKOUT OF THE NEW ELEMENT. *Default:* **mkelem** checks out the new element. The file in the workspace becomes the checked-out version of the element. If **–ci** option is not specified, the local files are made writable.

–ci

Creates the new element and version **\main\0**, checks out the element, then uploads and checks in a new version containing the data in the workspace file. Local files that correspond to successfully checked-in versions are read-only.

SPECIFYING WHICH FILES TO EXCLUDE. *Default:* None.

–exc·lude *exclude-pname*

Specifies a local file containing patterns of files to be excluded during the import. This **exclude-pname** contains a list of file names, one or more per line, separated by tabs or spaces. The file name arguments may contain standard wildcard patterns. Specifying a relative pathname for *exclude-pname* begins from Attache's startup directory, not the working directory, so a full local pathname is recommended.

CREATING NEW ELEMENT NAMES IN LOWERCASE. *Default:* In Windows 95 or Windows NT, the default is to create the filename as it exists in the workspace. In Windows 3.x, the default is to create all new element names in all lower-case by default.

–lca·se

Specifies that all new element names are created in all lower-case. When the **–lcase** option is used and **import** converts the fiele name to lowercase, the file in the workspace is also renamed to match the new element.

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–cqe**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**

Overrides the default with the option you specify. See the **comments** reference page.

SPECIFYING THE FILES TO BE IMPORTED. *Default:* None.

pname...

Specifies the files to be imported. Wildcard patterns apply to the workspace contents; / (slash) denotes the workspace root. For example, **/vobs/gui/*.c** refers to all of the **.c** files in the **/vobs/gui** subdirectory of the workspace. Either slashes or backslashes can be used.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, <u>*cmd-context*</u> represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, <u>*cmd-context*</u> represents the interactive **cleartool** prompt. In Attache, <u>*cmd-context*</u> represents the workspace prompt.

- Import all **.c** files in your current working directory

  <u>*cmd-context*</u> **import *.c**

- Import all files excluding executables and dynamically linked libraries from the directory **foo**. The file **ex_file** contains the text "***.exe *.dll**."

  <u>*cmd-context*</u> **import –exclude c:\users\jed\ex_file foo***

- Import all **.c** files in the workspace directory **src** and check them in after they are created

  <u>*cmd-context*</u> **import –ci src/*.c**

- Import all files in the workspace directory **foo,** creating **foo** in the VOB if it does not exist.

  <u>*cmd-context*</u> **import foo**

SEE ALSO

attache_command_line_interface, **checkout**, **mkelem**, **wildcards**

init_ccase

Startup/shutdown script

APPLICABILITY

Product	Command Type
ClearCase	command
ClearCase LT	command

Platform
UNIX

SYNOPSIS

AIX 4, MP-RAS	**/etc/rc.atria** { **start** | **stop** }
Digital UNIX, HP-UX 10, HP-UX 11	**/sbin/init.d/atria** { **start** | **stop** }
Solaris, IRIX, Reliant UNIX, UnixWare	**/etc/init.d/atria** { **start** | **stop** }

DESCRIPTION

The shell script listed in the *SYNOPSIS* section is invoked at system startup and shutdown. It can also be executed as a shell command.

CLEARCASE AND CLEARCASE LT STARTUP

When invoked with the argument **start** (or without an argument), the script performs initialization as follows:

- Start the Location Broker Daemon, **albd_server**.
- Start the database lock manager process, **lockmgr**.
- (On Solaris, AIX 4, Digital UNIX, UnixWare) Dynamically loads the MVFS (multiversion file system) into the operating system kernel.
- Initialize the viewroot directory (default name **/view**).
- Mount public VOBs listed in storage registry. If the network is partitioned into multiple network regions, only the VOBs that have public VOB-tags in the local host's region are mounted.

- Export VOBs through particular views to enable access by non-ClearCase hosts; the list of VOBs to be exported is read from the ClearCase file **/etc/exports.mvfs** (all platforms except Digital UNIX) or **/etc/exports** (Digital UNIX).

Startup Retry Loop

The startup script resides outside the host's ClearCase/ClearCase LT installation area. It calls on another script, which resides *inside* the installation area, to do the actual startup processing. If this other script, *ccase-home-dir***/etc/atria_start**, cannot be accessed, the startup script enters a retry loop. (This can occur if the ClearCase/ClearCase LT installation area is located on a remote host, and that host is currently unavailable.)

In its retry loop, the startup script tries periodically to invoke *ccase-home-dir***/etc/atria_start**.The retries continue indefinitely; if you want to terminate the loop, remove the flag file **/tmp/ClearCase.retrying**.

The Viewroot Mount Command

The startup script runs a standard **mount** command to mount the **viewroot** directory as a file system of type MVFS. This **mount** command is architecture-specific:

Architecture	Command
AIX 4	**mount –v mvfs –o rw,viewroot** *hostname***:/view /view**
Digital UNIX	**mount –t mvfs –o –o=rw,–o=viewroot /view /view**
IRIX, MP-RAS	**mount –t mvfs –o rw,viewroot /view /view**
Reliant UNIX, HP-UX 10, HP-UX 11	**mount –F mvfs –o rw,viewroot /view /view**
Solaris, UnixWare	**mount –F mvfs –o rw,viewroot** *hostname***:/view /view**

You can change the extending naming symbol by appending a string to the argument that follows the **–o** option:

,xnsuffix=*symbol*	*(all platforms except Digital UNIX)*
,–o=xnsuffix=*symbol*	*(Digital UNIX)*

This specifies a character string to be used on the local host as the ClearCase extended naming symbol. By default, the string @@ is used. Be careful: this option affects the local host only; other hosts may use the default extended naming symbol or another symbol specified with this mount option.

You can specify a directory other than **/view** as the viewroot. Whatever directory you specify (for example, **/ccasevu**) must exist at system startup time. Note that you must specify this directory name twice in the **mount** command.

Mounting the viewroot directory enables use of ClearCase views on the local host. When a view is activated (by **startview**, **setview**, or **mktag**), its view-tag is entered into the viewroot directory.

For example, activating a view whose view-tag is **gamma** creates the directory entry **/view/gamma**. See the **pathnames_ccase** reference page for a discussion of view-extended pathnames that use such directory entries.

A mounted viewroot directory is not actually an on-disk directory. Rather, it is a data structure maintained in main memory by the MVFS code loaded into the operating system kernel. The viewroot directory's list of view-tags is lost whenever ClearCase operation on the local host is stopped (including an operating system shutdown).

The viewroot directory cannot be exported and cannot be mounted by any other host. Each ClearCase host must have its own viewroot directory.

SHUTDOWN

When invoked with the argument **stop**, the script shuts down ClearCase/ClearCase LT:

- Unexport any view/VOB combinations that were exported through **/etc/exports.mvfs** to enable non-ClearCase access
- (On Solaris, Digital UNIX, Reliant UNIX) Kills all user processes that are using the MVFS (multiversion file system)
- Unmount all VOBs
- Kill the **vob_server** processes for VOBs whose storage directories are on the local host
- Kill the **albd_server** process, which also causes **view_server**, **db_server**, and **vobrpc_server** processes to exit
- Kill the **lockmgr** process
- (On Solaris, AIX 4, Digital UNIX, UnixWare) Unloads the MVFS from the operating system kernel
- Unmount the viewroot directory

SEE ALSO

exports_ccase, **mount_ccase**, **pathnames_ccase**, *Administrator's Guide*

ln

Creates VOB hard link or VOB symbolic link

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- Create one link:

  ln [**–s·link**] [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery**
  | **–cqe·ach** | **–nc·omment**]
  [**–nco** [**–f·orce**]] *pname target-pname*

- Create one or more links in a specified directory:

  ln [**–s·link**] [**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery**
  | **–cqe·ach** | **–nc·omment**]
  [**–nco** [**–f·orce**]] *pname* [*pname* ...] *target-dir-pname*

DESCRIPTION

The links created with the **ln** command (VOB symbolic links or VOB hard links) are cataloged in directory versions, in the same way as elements. By default, a link can be created in a directory only if that directory is checked out. A VOB link becomes visible to those using other views only after you have checked in the directory in which you created the link. (**ln** appends an appropriate default checkin comment to the directory version.)

In a snapshot view, this command executes the **update** command for elements affected by the link operation.

VOB SYMBOLIC LINKS

A VOB symbolic link (created if you use the **–slink** option) is a separate, unversioned object. It contains a character string, the link text, in the form of a pathname. You can attach attributes and hyperlinks—but not version labels—to a VOB symbolic link.

You cannot check out a VOB symbolic link. To revise a VOB symbolic link, check out its directory, remove the link with **rmname**, create a new link, and check in the directory. (Note that if you use the **–nco** option, the checkout and checkin steps are not required.)

VOB Symbolic Links in UNIX

We recommend that you use relative VOB symbolic links instead of absolute VOB symbolic links. Absolute VOB symbolic links require you to use absolute pathnames from the VOB-tag level; if the VOB mount point should change, the link becomes invalid.

VOB Symbolic Links in Windows

VOB symbolic links that point to files outside the ClearCase MVFS are not supported by the Windows operating system. Although the **ln** command creates the link, the link does not appear in a standard directory listing; it is displayed only by the **cleartool ls** command. (This is true for all symbolic links that do not point to a valid MVFS pathname.)

We recommend that you use relative VOB symbolic links instead of absolute VOB symbolic links. Absolute VOB symbolic links require you to use absolute pathnames from the view-tag level and are therefore valid only in the view in which they were created.

NOTE: Although an absolute VOB symbolic link that includes the view-tag at the beginning works when you are in the view, an absolute VOB symbolic link pointing to a pathname that begins with a VOB-tag (for example, **cleartool ln \my_vob\file my_link**) does not work.

VOB HARD LINKS

A VOB hard link (created if you omit the **–slink** option) is an additional name for an existing element. We recommend that you use VOB symbolic links instead of VOB hard links whenever possible.

VOB Hard Links in UNIX

In UNIX, you cannot make a VOB hard link to a derived object, but you can make additional UNIX system hard links (created with **ln(1)**) to a derived object. The links are visible in your view, but are not part of the VOB. For more information, see *Building Software*.

VOB Hard Links in Windows

When you check out a VOB hard link (that is, check out the element it names), all the other names for the element are listed by (**cleartool**) **ls** as `checkedout but removed` and do not appear in Windows Explorer. The element is checked out, but there are no view-private files having the other names. The command **lscheckout –all** lists the checked-out element only once.

After you check in the element or cancel the checkout (using **uncheckout**), the other names for the element are listed by (**cleartool**) **ls** as `disputed checkout, checkedout but removed` and do not appear in Windows Explorer. To update the state of the other names, use the **setcs –current** command.

VOB Hard Links and Directory Merges

The **merge** and **findmerge** commands can merge both file elements and directory elements. Merging versions of a directory element can involve creating a VOB hard link to a directory or removing a VOB hard link from a directory:

- Working on a subbranch, a user checks out the directory **/src**, and then uses **mkdir** to create directory element **/testing** within **/src**, or uses **rmname** to remove **/testing** from **/src**.
- When the subbranch is merged back into the main branch, a hard link named **testing** is made in (or removed from) a main-branch version of **src**, referencing the directory element already cataloged in the subbranch version.

ClearCase, ClearCase LT and Attache allow creation of hard links to directories only in this directory-merge context: the two links (both named **testing** in the example above) must occur in versions of the same directory element (**/src** in the example above).

VOB Hard Links in Snapshot Views

In a snapshot view, a VOB hard link is a copy of the element to which it points.

RECOVERING A REMOVED ELEMENT

You can use **ln** to recover an element that you mistakenly removed from a VOB directory with **rmname**. See the **rmname** reference page for details. Note that you cannot use **ln** to link elements that are in the **lost+found** directory.

RESTRICTIONS

Identities: No special identity is required if you checked out the directory. To use the **–nco** option, you must have one of the following identities:

- VOB owner
- **root** (UNIX)
- Member of the ClearCase group (ClearCase on Windows)
- Local administrator of the ClearCase LT server host (ClearCase LT on Windows)

Locks: An error occurs if one or more of these objects are locked: VOB.

Mastership: (Replicated VOBs only) No mastership restrictions.

OPTIONS AND ARGUMENTS

TYPE OF LINK. *Default:* VOB hard links.

–s·link

Creates VOB symbolic links.

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–nc**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**

Overrides the default with the option you specify. See the **comments** reference page.

CREATING A LINK IN A CHECKED-IN DIRECTORY VERSION. *Default:* None: you must check out a directory to create a link in it.

–nco [**–f·orce**]

Prompts for confirmation, then creates the link in the checked-in directory version that you specify. Use the **–force** option to suppress the confirmation prompt.

NOTE: You cannot use **–nco** in a replicated VOB.

SPECIFYING THE ELEMENT TO WHICH THE LINK POINTS. *Default:* None.

pname ...

Specifies an existing element to which a link is to be created. Each *pname* must be a standard or view-extended pathname. For VOB hard links, each *pname* must specify an existing element that is not a VOB symbolic link and that resides in the same VOB as the link being created. For VOB soft links, *pname* need not reside in the same VOB as the link to it.

SPECIFYING THE NAME OF THE LINK. *Default:* None.

target-pname

A pathname that specifies the name for the VOB link to *pname*. An error occurs if an object already exists at *target-pname*. For a VOB hard link, *pname* and *target-pname* must reside in the same VOB; this restriction does not apply to VOB symbolic links.

target-dir-pname

The pathname of an existing directory element in the same VOB as the *pname* argument. **ln** creates a new link in this directory for each preceding *pname* argument.

NOTE: This form of the command is intended for the creation of VOB hard links. If you use this form to create VOB symbolic links, make sure the links do not point to themselves. For example, the following command creates circular links:

cleartool ln –s file.txt dir1

```
Link created: "dir1/file.txt".
```

cd dir1

ls –l

```
lrwxrwxrwx  1 smg  user     8 May 12 13:36 file.txt -> file.txt
```

The following command creates symbolic links that are not circular:

cleartool ln –s ../file.txt .

```
Link created: "./file.txt".
```

cd dir1

ls –l

```
lrwxrwxrwx  1 smg  user     8 May 12 13:36 file.txt -> ../file.txt
```

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Create a VOB hard link, **hw.c**, as another name for element **hello.c**.

  cmd-context **ln hello.c hw.c**

  ```
  Link created: "hw.c".
  ```

- Create a VOB symbolic link, **messages.c**, pointing to **msg.c**.

  cmd-context **ln -slink msg.c messages.c**

  ```
  Link created: "messages.c".
  ```

- Create a group of hard links in the **subd** directory for all **.h** files in the current working directory.

cmd-context **ln *.h subd**

```
Link created: "subd/hello.h".
Link created: "subd/msg.h".
Link created: "subd/util.h".
```

- As a member of the ClearCase group (ClearCase), or logged in at the ClearCase LT server host as the local administrator (ClearCase LT), create a VOB symbolic link in the checked-in directory version **\vobs_hw@@\main\3** that points to **hello.c** in the current working directory.

  cmd-context **ln -slink -nco hello.c ..\vobs_hw@@\main\3\hello.c**

  ```
  Modify checked-in directory version “\vobs_hw@@\main\3”? [no] yes
  Link created: “..\vobs_hw@@\main\3\hello.c”.
  ```

SEE ALSO

describe, **ln (1)**, **mv**, **rename**, **update**

lock

Locks an object

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

lock [**–rep·lace**] [**–nus·ers** *login-name*[,...] | **–obs·olete**]
[**–c·omment** *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**]
{ [**–pna·me**] *pname* ...
| *object-selector* ...
}

DESCRIPTION

The **lock** command creates a lock on an entire VOB, or on one or more file-system objects, type objects, or VOB storage pools. A lock on an object disables operations that modify the object; a lock has no effect on read operations, such as **lshistory**. (Exception: see the *Storage Pool Lock: Cleartext Pool* section.)

The VOB does not need to be mounted for you to lock type objects, storage pools, or the VOB itself. However, you need a view context (and therefore a mounted VOB if you're using a dynamic view) to lock elements or versions.

The following sections describe the several kinds of locks.

VOB Lock

Locking an entire VOB disables all write operations to that VOB and forces a database checkpoint by causing a state flush. A typical application is locking a VOB to prevent it from being modified during backup.

You must lock a VOB before backing it up, and you cannot use the **–nusers** option. With **–nusers**, it is possible that the VOB will be modified during the backup, and the **nuser** lock does not perform a database checkpoint.

NOTE: Locking a VOB does not lock its **cleartext** storage pools, because this would prevent read access to **text_file**, **compressed_text_file**, and **binary_delta_file** elements. (For example, it would prevent a locked VOB from being backed up.) To completely lock a VOB, you must also lock its cleartext pools, using one or more **lock pool:** commands. You may want to do this to move a cleartext pool.

Type Lock

In general, locking a type object disables these kinds of operations:

- Operations that create, delete, or modify instances of the type
- Operations that delete or modify the type object itself (for example, renaming it)

The following sections describe how these general rules apply to the different kinds of type objects.

- **Element Type.** If an element type is locked, you cannot:
  - Use it in an **rmtype** or **rename** command
  - Create an element of that type with **mkelem** or **mkdir**
  - Change an existing element to that type with **chtype**
  - Modify the element's version tree with **checkout**, **checkin**, or **mkbranch**
- **Branch Type.** If a branch type is locked, you cannot:
  - Use it in an **rmtype**, **rename**, or **mkbrtype –replace** command
  - Create a branch of that type with **mkbranch**
  - Rename (that is, change the type of) an existing branch to or from that type with **chtype**
  - Modify the branch with **checkout** or **checkin**
  - Cancel a checkout using **uncheckout**
  - Attach a label using **mklabel**
  - Remove a label using **rmlabel** or **mklabel -replace**

  You can create a subbranch at any version on a locked branch, using **mkbranch**. (Creating a subbranch does not modify the branch itself.)
- **Label Type.** If a label type is locked, you cannot:
  - Use it in an **rmtype**, **rename**, or **mklbtype –replace** command

  - Attach or remove a version label of that type with **mklabel** or **rmlabel** (This includes moving a label from one version to another with **mklabel –replace**.)
- **Attribute Type.** If an attribute type is locked, you cannot:
  - Use it in an **rmtype**, **rename**, or **mkattype –replace** command
  - Attach or remove an attribute of that type with **mkattr** or **rmattr** (This includes moving an attribute from one version to another with **mkattr –replace**.)
- **Hyperlink Type.** If a hyperlink type is locked, you cannot:
  - Use it in an **rmtype**, **rename**, or **mkhltype –replace** command
  - Create or remove a hyperlink of that type with **mkhlink** or **rmhlink**
- **Trigger Type.** If a trigger type is locked, you cannot:
  - Use it in an **rmtype**, **rename**, or **mktrtype –replace** command
  - (If created with **mktrtype –element**) Create or remove a trigger of that type with **mktrigger** or **rmtrigger**

  In general, locking a trigger type does not inhibit triggers of that type from firing. Exception: trigger firing is inhibited if a trigger type created with **mktrtype –element –all**, **mktrtype –ucm –all**, or if **mktrtype –type** is made obsolete (using **lock –obsolete**).

Storage Pool Lock

Locking a VOB storage pool inhibits commands that create or remove the pool's data containers. It also prevents the pool's scrubbing parameters from being modified with **mkpool –update**. The following sections describe how this principle applies to the different kinds of storage pools.

- **Source Pool.** If a source storage pool is locked, you cannot:
  - Create an element that would be assigned to that pool, with **mkelem** or **mkdir.** (A new element inherits its pool assignments from its parent directory element.)
  - Change an existing element's pool assignment to/from that pool, with **chpool**.
  - Change an element's element type with **chtype**, if the change would require recreation of source data containers (for example, changing from type **file** to type **text_file**).
  - Check in a new version of an element assigned to that pool.
  - Create or remove a branch of an element assigned to that pool, with **mkbranch** or **rmbranch.**
  - Remove a version of an element assigned to that pool, or remove the element itself, with **rmver** or **rmelem.**
- **Derived Object Pool.** If a derived object storage pool is locked:

- **clearmake** cannot winkin a previously unshared derived object in a directory assigned to that pool. (The invocation of **promote_server** to copy the data container from view-private storage to the derived object storage pool fails.)
- **scrubber** cannot remove data containers from the pool.
- An **rmdo** command fails for a derived object whose data container is in that pool.

- **Cleartext Pool.** If a cleartext storage pool is locked:
  - An attempt to read a version of an element assigned to that pool may fail. (It fails if a new cleartext data container for that version would have been created and cached in the cleartext pool.)

LOCKING OR UNLOCKING GLOBAL TYPES

Locking or unlocking a global type or one of its local copies locks or unlocks the global type and all local copies. For more information, see the *Administrator's Guide*.

OBSOLETE OBJECTS

An object becomes obsolete if it is processed with a **lock –obsolete** command. An obsolete type object or obsolete storage pool is not only locked, but is also invisible to certain forms of the **lstype**, **lslock**, **lspool**, and **lsvtree** commands. An obsolete VOB or obsolete VOB object is no different from one with an ordinary lock. You can change an object's status from obsolete to locked by using a **lock –replace** command:

cmd-context **lock –obsolete brtype:test_branch** *(make a branch type obsolete)*

```
Locked branch type "test_branch".
```

cmd-context **lock –replace brtype:test_branch** *(change the branch type to 'just locked')*

Similarly, you can use a **lock –replace** command to make a locked object obsolete.

REMOVING LOCKS

The **unlock** command removes a lock from an object, reenabling the previously prohibited operations.

RESTRICTIONS

Identities:

Kind of Object to be Locked	Identity Required for ClearCase, ClearCase LT, and Attache on UNIX	Identity Required for ClearCase and Attache on Windows	Identity Required for ClearCase LT on Windows
Type object	Type owner, VOB owner, **root**	Type owner, VOB owner, member of the ClearCase group (ClearCase)	Type owner, VOB owner, local administrator of the ClearCase LT server host
Storage pool	VOB owner, **root**	VOB owner, member of the ClearCase group	VOB owner, local administrator of the ClearCase LT server host
VOB	VOB owner, **root**	VOB owner, member of the ClearCase group	VOB owner, local administrator of the ClearCase LT server host
Element	Element owner, VOB owner, **root**	Element owner, VOB owner, member of the ClearCase group	Element owner, VOB owner, local administrator of the ClearCase LT server host
Branch	Branch creator, element owner, VOB owner, **root**	Branch creator, element owner, VOB owner, member of the ClearCase group	Branch creator, element owner, VOB owner, local administrator of the ClearCase LT server host

Locks: An error occurs if one or more of these objects are locked: the VOB containing the object.

Mastership: (Replicated VOBs only) With **–obsolete**, your current replica must master the object.

OPTIONS AND ARGUMENTS

REPLACING AN EXISTING LOCK. *Default:* None.

–rep·lace

Uses a single atomic transaction to replace an existing lock with a new lock. (If you use two commands to **unlock** the object and then **lock** it again, there is a short interval during which the object is unprotected.)

You can use this option to change a object's status from just locked to obsolete.

SPECIFYING THE DEGREE OF LOCKING. *Default:* Locks an object to all users, but does not make the object obsolete.

–nus·ers *login-name* [,...]
: Allows the specified users to continue using the object, which becomes locked to all other users. The list of user names must be comma-separated, with no white space.

–obs·olete
: Locks an object for all users, and also makes it obsolete.

EVENT RECORDS AND COMMENTS. *Default:* Creates one or more event records, with commenting controlled by your **.clearcase_profile** file (default: **–nc**). See the **comments** reference page. Comments can be edited with **chevent**.

–c·omment *comment* | **–cfi·le** *comment-file-pname* |**–cq·uery** | **–cqe·ach** | **–nc·omment**
: Overrides the default with the option you specify. See the **comments** reference page.

SPECIFYING THE OBJECTS TO BE LOCKED. *Default:* The final arguments are assumed to be the names of elements and/or branches. To lock another kind of object, you must use an object-selector prefix.

When locking type objects and storage pools, the command processes objects in the VOB containing the current working directory. To lock an entire VOB, you must specify a VOB.

[**–pna·me**] *pname* ...
object-selector ... (mutually exclusive)
: One or more names, specifying the objects to be locked. To lock an element, you can specify the element itself (for example, **foo.c@@**) or any of its versions (for example, **foo.c** or **foo.c@@/RLS1.3**).To lock a branch, use an extended pathname (for example, **foo.c@@\main\rel2_bugfix**). If *pname* has the form of an object selector, you must use the **–pname** option to indicate that *pname* is a pathname.

Specify *object-selector* in one of the following forms:

vob-selector	**vob:***pname-in-vob* *pname-in-vob* can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted). It cannot be the pathname of the VOB storage directory.
attribute-type-selector	**attype:***type-name*[@*vob-selector*]
branch-type-selector	**brtype:***type-name*[@*vob-selector*]
element-type-selector	**eltype:***type-name*[@*vob-selector*]
hyperlink-type-selector	**hltype:***type-name*[@*vob-selector*]

label-type-selector	**lbtype:***type-name*[@*vob-selector*]
trigger-type-selector	**trtype:***type-name*[@*vob-selector*]
pool-selector	**pool:***pool-name*[@*vob-selector*]
oid-obj-selector	**oid:***object-oid*[@*vob-selector*]

The following object-selectors apply to the UCM usage model (see **NOTE**):

activity-selector	**activity**:*actvity-name*[@*vob-selector*]
baseline-selector	**baseline**:*baseline-name*[@*vob-selector*]
component-selector	**component**:*component-name*[@*vob-selector*]
folder-selector	**folder**:*folder-name*[@*vob-selector*]
project-selector	**project**:*project-name*[@*vob-selector*]
stream-selector	**stream**:*stream-name*[@*vob-selector*]

NOTE: In UCM object selectors, *@vob-selector* refers to a UCM project VOB.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Lock three label types for all users.

  cmd-context **lock lbtype:REL1 lbtype:REL1.1 lbtype:REL2**

  ```
  Locked label type "REL1".
  Locked label type "REL1.1".
  Locked label type "REL2".
  ```

- Obsolete a branch type.

  cmd-context **lock -obsolete brtype:rel2_bugfix**

  ```
  Locked branch type "rel2_bugfix".
  ```

- Lock the VOB containing the current working directory.

  cmd-context **lock vob:.**

  ```
  Locked versioned object base "/usr/hw".
  ```

- Lock the **test** branch type for all users except **gomez** and **jackson**.

  <u>*cmd-context*</u> **lock –nusers gomez,jackson brtype:test**

  ```
  Locked branch type "test".
  ```

- Lock elements with a **.c** extension for all users. Then try to check out one of the locked elements.

  <u>*cmd-context*</u> **lock *.c**

  ```
  Locked file element "hello.c".
  Locked file element "msg.c".
  Locked file element "util.c".
  ```

  <u>*cmd-context*</u> **checkout –nc msg.c**

  ```
  cleartool: Error: Lock on file element prevents operation "checkout".
  cleartool: Error: Unable to check out "msg.c".
  ```

SEE ALSO

unlock

ls

Lists VOB-resident objects, elements loaded into a snapshot view, and view-private objects in a directory

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

ls [**–r·ecurse** | **–d·irectory**] [**–l·ong** | **–s·hort**] [**–vob·_only** | **–vie·w_only**]
[**–nxn·ame**] [**–vis·ible**] [*pname* ...]

DESCRIPTION

The **ls** command lists VOB-resident objects, elements loaded into a snapshot view, and view-private objects in a directory.

The default listing includes this information:

- The name of each element cataloged in the current directory, with the version-ID of the particular version in the view. Also included is the version selector part of the config spec rule that selects this version. In a snapshot view, you see the message `<rule info unavailable>` if **ls** encounters errors
- The name of each view-private object in the current directory
- In a dynamic view, the name of each derived object (DO) visible in the view, along with its unique DO-ID

The listing for an element or a derived object in a dynamic view may also include an annotation that indicates an unusual or noteworthy state. For example, the listing for an element that has been checked out to your view identifies the version that was checked out:

```
hello.c@@/main/CHECKEDOUT from /main/4                Rule: CHECKEDOUT
```

Annotations Common to All View Types

The following annotations may appear when you issue **ls** from any type of view:

`eclipsed`
: No version of the element is selected because a view-private object with the same name exists in your view. Typical occurrence: you create a view-private file in your view; then an element with the same pathname is created in another view. In your view, an **ls –vob_only** shows the element to be eclipsed.

`eclipsed by checkout`
: (Appears only when you use the **–vob_only** option) No version from the element's version tree is selected, because the element has been checked out in this view, and a checked-out version always eclipses all checked-in versions.

`checkedout but eclipsed`
: The element has been checked out in this view, but there is no **CHECKEDOUT** config spec rule; thus, the checked-out version is not visible in the view.

`checkedout but removed`
: The element was checked out in this view, but the view-private file was subsequently removed. You may have deletd the file. ClearCase, ClearCase LT, and Attache remove it (in effect) when you check out a file with **checkout –out**, or when you check out a DO version.

  NOTE: If a file element has several names, by virtue of one or more VOB hard links, checking out the element under one name causes all the *other* names to be listed with this annotation. (The element is checked out, but there are no view-private files with the other names.)

`no version selected`
: The element is not selected by any config spec rule, or is selected by a **–none** config spec rule.

`error on reference`
: The element is selected by an **–error** config spec rule.

UNIX-Only Annotations Common to All View Types

`view-->vob hard link`
: The object is a view-private (operating system level) hard link to an object in VOB storage.

Annotations Specific to Dynamic Views

The following annotations may appear when you issue **ls** from a dynamic view:

`no config record`
: (Shareable derived objects only) The derived object's data container is still stored in the view, but the derived object in the VOB database (and, typically, its associated configuration record) have been deleted by **rmdo**. This can occur only in the view in which the derived object was originally built.

`disputed checkout`
: The element is considered to be checked out by the **view_server** but is not so indicated in the VOB database (or vice versa). This can occur during the short interval in which a checkin or checkout is in progress. For information about **view_server**, see the *Administrator's Guide*.

`removed with white out`
: The derived object was winked in by, and is still referenced by, the current view, but it has been forcibly removed from the VOB database with **rmdo**. The derived object is not recoverable.

Annotations Specific to Snapshot Views

The following annotations may appear when you issue **ls** from a snapshot view:

`not loaded`
: The element is not loaded into the snapshot view. Either there are no load rules specifying the element, or the version-selection rules do not select any version of the element.

`loaded but missing`
: A version of the element was loaded into the view, but you have deleted or renamed the file in the view. To copy the version back into the view, use the **cleartool get** command (note that generates a hijacked file) or update the snapshot view, specifying the pathname to the missing file.

`hijacked`
: The version in the view was modified without being checked out.

`overridden`
: The element is loaded in the snapshot view, but its file type is not the same as the corresponding object in the VOB; or the element is not loaded in the snapshot view, but an object with the same name exists in the view.

`special selection`
: The version you checked in (and, hence, the version currently in the view) is not the version that the config spec selects from the VOB. For more information, refer to the section, *Actions Taken in the View* on page 73, in the **checkin** reference page.

`nocheckout`
: The version hijacked in the view is no longer the version the config spec selects from the

VOB. To prevent losing changes in the version selected by the config spec, you cannot check out the hijacked file. To check in your modifications, you must fix the hijack condition:

1. Rename the hijacked file and update the file.
2. Check out the version from which you hijacked the file.
3. Copy your hijacked file over the checked-out version.
4. Merge from the current version to your checked-out version.

You can now check in your version.

(You can use the graphical **update** tool to do the checkout and merge operations.)

`deleted version`

The version currently in the view has been removed from the VOB (for example, by the **rmver** command). Use the **update** command to copy a valid version into the view.

Elements Suppressed from the View

The listing includes elements selected with **–none** and **–error** config spec rules, and elements that are not selected by any config spec rule. UNIX commands, such as **ls(1)** and **cat(1)**, return `not found` errors when accessing such elements. You can specify such elements in commands that access the VOB database only, such as **describe**, **lsvtree**, and **mklabel**.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. *Default:* For each *pname* that specifies a directory element, **ls** lists the contents of that directory, but not the contents of any of its subdirectories.

NOTE: This includes directories in version-extended namespace, which represent elements and their branches. For example, specifying **foo.c@@/main/bug403** as an argument lists the contents of that branch: all the versions on the branch.

–r·ecurse

Includes a listing of the entire subtree below any subdirectory included in the top-level listing. VOB symbolic links are not traversed during the recursive descent.

–d·irectory

Lists information on a directory itself, rather than its contents.

REPORT FORMAT. *Default:* The default report format is described in the *The default listing includes this information:* section.

–l·ong

For each object, lists the config spec rule matching the object, and classifies each object.

The classification can be one of: version, directory version, file element, directory element, view-private object, derived object, derived object version, or symbolic link. For each derived object, **ls –long** indicates whether the DO is nonshareable, unshared, promoted, or shared.

–s·hort
Restricts the listing of each entry to its version-extended pathname only.

–nxn·ame
Lists simple pathnames instead of version-extended pathnames.

VOB/VIEW RESTRICTION. *Default:* The listing includes both objects in VOB storage and objects in view storage.

–vob·_only
Restricts the listing to objects in the VOB storage, including versions of elements and VOB links. This may also add some entries to the listing: those for the underlying elements that are eclipsed by checked-out versions.

–vie·w_only
Restricts the listing to objects that belong logically to the view: view-private files, view-private directories, and view-private links; checked-out versions; and all derived objects visible in the view.

NOTE: Checked-out directories are listed by **–vob_only**, and not by **–view_only**.

NOTE: Derived objects visible in the view are listed by **–view_only** (and not **–vob_only**), regardless of whether they are (or ever have been) shared.

–vis·ible
Restricts the listing to objects visible to the operating system listing command.

SPECIFYING THE OBJECTS TO BE LISTED. *Default:* The current working directory (equivalent to specifying "." as the *pname* argument). If you don't specify any other options, all files and links in the current working directory are listed; all subdirectory entries are listed, but not the contents of these subdirectories.

pname ...
Restricts the listing to the specified files, directories, and/or links. *pname* may be a view- or VOB-extended pathname to list objects that are not in the view, regardless of whether the view is a snapshot view or a dynamic view (see **pathnames_ccase**).

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In some examples, output is wrapped for clarity.

- List the VOB-resident objects and view-private objects in the current working directory.

  cmd-context **ls**

```
Makefile@@/main/3                                    Rule: /main/LATEST
bug.report
cm_add.c@@/main/0                                    Rule: /main/LATEST
cm_fill.c@@/main/0                                   Rule: /main/LATEST
convolution.c@@/main/CHECKEDOUT from /main/0         Rule: CHECKEDOUT
edge.sh
hello@@24-Mar.11:32.418
hello.c@@/main/CHECKEDOUT from /main/4               Rule: CHECKEDOUT
hello.h@@/main/CHECKEDOUT from /main/2               Rule: CHECKEDOUT
hello.o@@24-Mar.11:32.412
hw.c@@/main/4                                        Rule: /main/LATEST
include@@/main/CHECKEDOUT                            Rule: CHECKEDOUT
```

- List the objects in the current working directory, with annotations.

  cmd-context **ls -long**

```
version               Makefile@@\main\3       Rule: element * \main\LATEST
view private object   bug.report
version               cm_add.c@@\main\0       Rule: element * \main\LATEST
derived object (unshared)   hello@@24-Mar.11:32.418
version               hello.h@@\main\CHECKEDOUT from \main\2
                                              Rule: element * CHECKEDOUT
derived object (unshared)   hello.o@@24-Mar.11:32.412
directory version     include@@\main\CHECKEDOUT
                                              Rule: element * CHECKEDOUT
symbolic link         messages.c --> msg.c
version               msg.c@@\main\1          Rule: element * \main\LATEST
view private object   util.c.contrib
```

- List only the view-private objects in the current working directory.

  cmd-context **ls -view_only**

```
bug.report
hello@@24-Mar.11:32.418
hello.c@@/main/CHECKEDOUT from /main/4        Rule: CHECKEDOUT
hello.h@@/main/CHECKEDOUT from /main/2        Rule: CHECKEDOUT
hello.o@@24-Mar.11:32.412
msg.o@@23-Mar.20:42.379
util.c@@/main/CHECKEDOUT from /main/4          Rule: CHECKEDOUT
util.o@@24-Mar.11:32.415
```

- List the contents of the directory in extended namespace that corresponds to the **main** branch of element **util.c**.

  cmd-context **ls util.c@@\main**

```
util.c@@\main\0
util.c@@\main\1
util.c@@\main\2
util.c@@\main\3
util.c@@\main\CHECKEDOUT
util.c@@\main\LATEST
util.c@@\main\REL2
util.c@@\main\REL3
util.c@@\main\rel2_bugfix

view private object   util.c.contrib
```

- List the directory version that is visible in the current view:

  cmd-context **ls –directory –vob_only**

```
.@@/main/CHECKEDOUT from /main/4        Rule: CHECKEDOUT
```

SEE ALSO

checkout, **config_spec**, **lsprivate**, **lsvtree**, **pathnames_ccase**, **uncheckout**

lsactivity

Lists information about UCM activities

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- List information about UCM activities:

  lsact·ivity [**–s·hort** | **–l·ong** | **–fmt** *format-string* |
  –anc·estor [**–fmt** *format-string*] [**–dep·th** *depth*]]
  [**–obs·olete**] [**–inv·ob** *vob-selector* | **–in** *stream-selector-name* |
  –cac·t | [**–cac·t**] **–vie·w** *view-tag* | **–cvi·ew** | *activity-selector* ...]

- List activities or baselines that contributed to the change set of an integration activity:

  lsact·ivity –contrib *activity-selector*

DESCRIPTION

The **lsactivity** command lists information about UCM activities.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING OUTPUT FORMAT *Default*: A one-line summary of the activity.

–s·hort
: Displays only the name of each activity.

–l·ong
: Displays a detailed description of each activity.

–fmt *format-string*
Displays information in the format specified by *format-string*. See the **fmt_ccase** reference page.

–anc·estor [**–fmt** *format-string*] [**–dep·th** *depth*]
Displays the containing stream, project, and folder for one or more activities. For information about the **–fmt** option, see the **fmt_ccase** reference page. The **–depth** option sets the number of levels displayed. The *depth* argument must be a positive integer.

LISTING OBSOLETE ACTIVITIES. *Default*: Lists only nonobsolete activities.

–obs·olete
Includes obsolete activities in the listing. Obsolete activities are those that have been processed with **lock –obsolete**.

SPECIFYING THE ACTIVITY. *Default:* **–cview**.

–inv·ob *vob-selector*
Displays a list of all activities in the specified project VOB.

–in *stream-selector*
Displays a list of all activities in the specified stream.

–cac·t
Displays information for the current activity.

–**vie·w** *view-tag*
For the specified view, displays a list of all activities in its stream.

–cvi·ew
For the current view, displays a list of all activities in its stream.

activity-selector ...
Specifies one or more activities to list.

You can specify an activity as a simple name or as an object selector of the form [**activity**]**:***name@vob-selector*, where *vob-selector* specifies a project VOB (see the **cleartool** reference page). If you specify a simple name and the current directory is not a project VOB, then this command assumes the activity resides in the project VOB associated with the stream attached to the current view. If the current directory is a project VOB, then that project VOB is the context for identifying the activity.

LISTING CONTRIBUTING ACTIVITIES OR BASELINES FOR AN INTEGRATION ACTIVITY. *Default*: Displays information about the named activity.

–contrib *activity-selector*
Displays activities or baselines that contributed to the change set of an integration

activity, which is created by a deliver or rebase operation. An error occurs if the specified activity is not an integration activity.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

Display detailed information for an activity.

cmd-context **lsactivity -l fix_copyright**

```
activity "fix_copyright"
06-Jun-00.15:49:23 by Ken Tessier (ktessier.user@mymachine)
owner: ktessier
group: user
stream: chris_webo_dev@/vobs/webo_pvob
title: Fix copyright text
change set versions:
/vobs/webo_modeler/design/add_proc@@/main/chris_webo_dev/1
/vobs/webo_modeler/design/foo@@/main/integration/chris_webo_dev/1
```

- Display a short description of the current activity. This is the currently set activity for the view from which the command was issued.

  cmd-context **lsact -cact**

```
06-Jun-00.17:16:12  update_date  ktessier   "Update for new date
convention"
```

SEE ALSO

chactivity, **lock**, **mkactivity**, **rmactivity**

lsbl

Lists information about a baseline

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- List baseline information per stream or component or by promotion level:

  lsbl [**–s·hort** | **–l·ong** | **–fmt** *format-string* | **–tre·e**]
  [**–lev·el** *promotion-level* | [**–ltl·evel** *promotion-level*] [**–gtl·evel** *promotion-level*]]
  [**–com·ponent** *component-selector*] [**–obs·olete**] [**–str·eam** *stream-selector* | **–cvi·ew**]

- List information for one or more specific baselines:

  lsbl [**–s·hort** | **–l·ong** | **–fmt** *format-string*] [**–tre·e**] [**–obs·olete**] [*baseline-selector* ...]

DESCRIPTION

The **lsbl** command lists information for one or more baselines.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE OUTPUT. *Default:* A one-line summary of each baseline.

–s·hort
Displays only the name of each baseline.

–l·ong
For noncomposite baselines, displays detailed information for each baseline, including

ownership, creation, and label information and the stream, component, change sets, and promotion level associated with the baseline.

For composite baselines, displays baselines that the composite directly depends on.

–fmt *format-string*
Displays information in the specified format. See the **fmt_ccase** reference page for details.

–tre·**e**
Displays a list of streams and baselines associated with one or more baselines. The list is indented to show the order of succession for baselines.

FILTERING BY PROMOTION LEVEL. *Default:* All promotion levels.

–lev·**el** *promotion-level*
Displays a list of baselines that are at the specified promotion level. An error results if the specified level is not in the project VOB's current list of valid promotion levels. This option modifies the **–stream** and **–component** options. For general information about promotion levels, see the **setplevel** reference page.

–ltl·**evel** *promotion-level*
Displays a list of baselines whose promotion level is lower than the one specified by the promotion-level argument. For example, your project has four promotion levels in this order: **PROTOTYPE**, **REVIEWED**, **TESTED**, **CERTIFIED**.If you use the argument **–ltlevel TESTED**, the **lsbl** command displays a list of all baselines whose promotion level is **PROTOTYPE** or **REVIEWED**. This option modifies the **–stream** and **–component** options.

–gtl·**evel** *promotion-level*
Displays a list of baselines whose promotion level is greater than the one specified. This option modifies the **–stream** and **–component** options.

SPECIFYING THE BASELINE. *Default:* Baselines in the project VOB of the current directory.

–com·**ponent** *component-selector*
Displays a list of baselines of the specified component.

–str·**eam** *stream-selector*
Displays a list of baselines created in the specified stream.

–cview
Displays a list of baselines created in the stream attached to the current view.

baseline-selector ...
Specifies one or more baselines for which information is displayed.

LISTING OBSOLETE BASELINES. *Default*: Lists only nonobsolete baselines.

–obs·olete

Includes obsolete baselines in the listing. Obsolete baselines are those that have been processed with **lock –obsolete**.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Display a summary (the default) of baselines of the specified component.

  cmd-context **lsbl -component parser@/vobs/core_projects**

  ```
  17-Sep-99.12:06:59  parser_INITIAL.112  bill   "parser_INITIAL"
    component: parser
  ```

- Display a description of baselines created in a stream:

  cmd-context **lsbl -stream java_int@/vobs/core_projects**

  ```
  17-Sep-99.13:56:10  testbl.121  bill   "testbl"
    stream: java_int
    component: parser
  17-Sep-99.14:05:30  new_bl.121  bill   "new_bl"
    stream: java_int
    component: parser
  ```

SEE ALSO

chbl, **deliver**, **describe**, **diffbl**, **lock**, **mkbl**, **rebase**, **rmbl**, **setplevel**

lscheckout

Lists checkouts of an element

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase, ClearCase LT, and Attache on UNIX only—List checkouts:

  lsc·heckout | **lsco** [**–l·ong** | **–s·hort** | **–fmt** *format-string*] [**–cvi·ew**]
  [**–brt·ype** *branch-type-selector*]
  [**–me** | **–use·r** *login-name*]
  [**–r·ecurse** | **–d·irectory** | **–a·ll** | **–avo·bs**] [**–are·plicas**]
  [*pname* ...]

- ClearCase, ClearCase LT, and Attache on Windows only—List checkouts in the command window:

  lsc·heckout | **lsco** [**–l·ong** | **–s·hort** | **–fmt** *format-string*] [**–cvi·ew**]
  [**–brt·ype** *branch-type-selector*]
  [**–me** | **–use·r** *login-name*]
  [**–r·ecurse** | **–d·irectory** | **–a·ll** | **–avo·bs**] [**–are·plicas**]
  [*pname* ...]

- ClearCase and ClearCase LT on Windows only—List checkouts graphically:

  lsc·heckout –g·raphical *pname*

DESCRIPTION

The **lscheckout** command lists the checkout records (the checkouts) for one or more elements. There are many controls for specifying the scope: which elements, directories, or VOBs; which user; which view; and so on.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

(CLEARCASE AND CLEARCASE LT ONLY) LISTING CHECKOUTS GRAPHICALLY. *Default:* Lists checkouts in the command window.

–g·raphical
Starts the **Find Checkouts** tool to display checkouts.

REPORT FORMAT. *Default:* The listing of a checkout event record looks like this:

```
31–Aug.20:19 drp checkout version "ct+lscheckout.1" from /main/4 (reserved)
  "delete extra spaces"
```

–l·ong
Expands the listing to include the view to which the element is checked out.

–s·hort
Restricts the listing to the pathnames of checked-out elements.

–fmt *format-string*
Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this report-writing facility.

SELECTING CHECKOUT RECORDS TO LIST. *Default:* The listing includes all checkouts for the specified elements, including checkouts made in any view by any user.

–me
Restricts the listing to your own checkouts.

–use·r *login-name*
Restricts the listing to checkouts made by the specified user.

–cvi·ew
Restricts the listing to checkouts made in the current view. On UNIX systems, if there is a working directory view, **lscheckout** lists its checkouts; otherwise, it lists the checkouts in the set view.

–brt·ype *branch-type-selector*
Restricts the listing to checkouts on branches of the specified type. Specify *branch-type-selector* in the form [**brtype:**]*type-name*[*@vob-selector*]

type-name	Name of the branch type
vob-selector	VOB specifier
	Specify *vob-selector* in the form [**vob:**]*pname-in-vob*
	pname-in-vob — Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

SPECIFYING THE ELEMENTS. *Default:* The current working directory (equivalent to specifying "**.**" as the *pname* argument). If you don't specify any options, **lscheckout** lists all checkouts of elements in the current directory, to any view. If the current directory is itself checked out, this is also indicated.

pname ...

One or more pathnames, specifying file elements and/or versions of directory elements. (A Windows pathname or a UNIX standard or view-extended pathname to a directory specifies the version in the view.)

- For each *pname* that specifies a file element, the listing includes that element's checkout event records.
- For each *pname* that specifies a version of a directory element, the listing includes checkout event records of elements cataloged in that directory version—but not checkout records for the *pname* directory itself.

The following options modify the processing of the *pname* arguments.

–r·ecurse

Lists the checkouts of elements in the entire subtree below any directory encountered in the current view. VOB symbolic links are not traversed during the recursive descent.

–d·irectory

Lists the checkouts (if any) of a directory itself, rather than the checkouts of elements cataloged in it.

–a·ll

Lists all the checkouts in the VOB containing *pname*. If you don't specify any *pname* arguments, lists all checkouts in the VOB containing the current working directory.

NOTE: A file element can have several names, by virtue of one or more VOB hard links. Checking out such an element under one name causes all the names to be listed as checked out. However, the **–all** option lists the checked-out element only once.

–are·plicas

Lists checkouts of the element specified by *pname* in all replicas of the VOB that contains *pname*. If you don't specify any *pname* arguments, lists all checkouts in all replicas of the VOB containing the current working directory.

The following option is mutually exclusive with *pname* arguments:

–avo·bs

Similar to **–all**, but includes checkouts in all VOBs active (mounted) on the local host. (If environment variable **CLEARCASE_AVOBS** is set to a list of VOB-tags, this set of VOBs is used instead.) This option depends on the MVFS, and so has no meaning for snapshot-view-only hosts.

NOTE: A view context is required when using this option.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List the checkouts in the current working directory.

  cmd-context **lscheckout**

```
08-Dec.12:17 jackson checkout version "hello.c" from /main/4 (reserved)
08-Dec.12:17 jackson checkout version "hello.h" from /main/1 (unreserved)
 "modify local defines"
08-Dec.12:17 jackson checkout version "msg.c" from /main/rel2_bugfix/0
(reserved)
```

- List only the names of elements checked out to the current view.

  cmd-context **lscheckout –short –cview**

```
hello.c
hello.h
hw.c
include
```

- List the checkouts in all directories at or below the current directory.

  cmd-context **lscheckout –recurse**

```
08-Dec.12:17 jackson checkout version "hello.c" from \main\4 (reserved)
08-Dec.12:17 jackson checkout version "hello.h" from \main\1 (unreserved)
    "modify local defines"
08-Dec.12:17 jackson checkout version "msg.c" from \main\rel2_bugfix\0
(reserved)
08-Dec.12:17 jackson checkout directory version "subd" from \main\1
(reserved)
08-Dec.12:17 jackson checkout version ".\subd\util.h" from \main\0
(reserved)
```

- List elements checked out by the user in all mounted VOBs.

  cmd-context **lscheckout –avobs –me**

```
08-Dec.12:17 jackson checkout version "/usr/hw/src/hello.c" from /main/4
(reserved)
08-Dec.12:17 jackson checkout version "/usr/hw/src/hello.h"from /main/1
(unreserved)
    "modify local defines"
08-Dec.12:17 jackson checkout directory version "/usr/hw/release" from
/main/0 (reserved)
08-Dec.12:17 jackson checkout version "/usr/hw/src/msg.c" from
/main/rel2_bugfix/0 (reserved)
08-Dec.12:17 jackson checkout version "/usr/hw/src/util.h" from /main/0
(reserved)
```

- For all checkouts in the current directory, list the checkout date, user name, element name, predecessor version, host, checkout status, and element type. (The command line, including the quoted format string, constitutes a single input line. The input line below is broken to improve readability. Spaces are significant.)

cmd-context **lsco –fmt "%d\t%Lu\t%En\n\tPredecessor: %[version_predecessor]p\n\tHost: %h\n\tStatus: %Rf\n\tElement type: %[type]p\n"**

```
16-Jun-99.15:23:15     lee.user        files.txt
   Predecessor: \main\96
   Host: neon
   Status: unreserved
   Element type: text_file
09-Jun-99.15:39:09     susan.user      mkfile.fm
   Predecessor: \main\27
   Host: pluto
   Status: reserved
   Element type: frame_document
16-Jun-99.12:23:11     cheryl.user     mktype.fm
   Predecessor: \main\115
   Host: troy
   Status: reserved
   Element type: frame_document
10-Jun-99.12:29:30     john.user       files.txt
   Predecessor: \main\26
   Host: marcellus
   Status: reserved
   Element type: text_file
```

SEE ALSO

checkin, checkout, lsprivate, uncheckout

lsclients

Displays the client host list for a ClearCase license or registry server host, or for a ClearCase LT server host

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
Attache	command
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and Attache:

  lsclients –hos·t *hostname* [**–typ·e** { **registry** | **license** | **all** }] [**–s·hort** | **–l·ong**]

- ClearCase LT:

  lsclients [**–s·hort** | **–l·ong**]

DESCRIPTION

On every ClearCase license server host and registry server host, the **albd_server** process maintains a list of current client hosts. On the ClearCase LT server host, **albd_server** maintains a list of hosts that are clients of its registry. If a client host does not access a server host for 30 days, **albd_server** drops it from that server host's client host list. For more information about **albd_server**, see the *Administrator's Guide*.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING A HOST. *Default:* None. You must supply a license or registry server host name.

–hos·t *hostname*

Specifies the registry or license server host. Only license and registry server hosts maintain client lists. If you specify a host that is not a license or registry server host, **lsclients** prints a message indicating that the host has no clients.

SPECIFYING THE TYPE OF SERVER HOST. *Default:* **–all**.

–typ·e { **registry** | **license** | **all** }

Use **–type registry** or **–type license** to restrict the listing to include only clients of registry server hosts or clients of license server hosts, respectively.

LISTING FORMAT. *Default:* Display a one-line description of each client.

–s·hort

Displays client host names only.

–l·ong

ClearCase and Attache—Expands the listing to include each client's registry server host, registry region, license server host, and date and time of the last server access.

ClearCase LT—Displays information about the client host, including the date and time of the last server access.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- Display the sorted client list for license server host **bromacil**.

  cmd-context **lsclients –host bromacil –type license | sort**

```
...
durian: ClearCase 4.1 (OSF1 V5.1 732 alpha)
indeno: ClearCase 4.1 (SunOS 5.7 Generic_106541-08 sun4u)
lead: ClearCase 4.2 (SunOS 5.8 Generic_108528-01 sun4m)
lowell: ClearCase 4.1 (IRIX64 6.5 04151556 IP30)
scorpius: ClearCase 4.2 (HP-UX B.11.00 A 9000/785)
...
```

- Display the long format client list for registry server host **atrium**.

  cmd-context **lsclients –host atrium –long**

```
...
Client: pyramid
  Product: ClearCase 4.1
  Operating system: Windows NT 4.0 (build 1381)Service Pack 6
  Hardware type: Pentium
  Registry host: atrium
  Registry region: atria_r_d_unc
  License host: maple
  Last registry access: 20-Nov-00.04:32:08
  Last license access: never
...
```

UNIX FILES

/var/adm/atria/rgy/rgy_hosts.conf
/var/adm/atria/rgy/rgy_svr.conf
/var/adm/atria/rgy/rgy_region.conf
/var/adm/atria/config/license_host

WINDOWS REGISTRY KEYS

HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\AtriaRegy
HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\Region
HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\ServerType
HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\LicenseHost

SEE ALSO

clearlicense, *Administrator's Guide*

lscomp

Lists information about a component

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

lscomp [**–s·hort** | **–l·ong** | **–fmt** *format-string* | **–tre·e**]
[**–inv·ob** *vob-selector* | *component-selector* ...] [**–obs·olete**]

DESCRIPTION

The **lscomp** command lists information about one or more components.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE OUTPUT. *Default:* A one-line summary.

–s·hort
Displays only the name of each component.

–l·ong
Displays an expanded multiple-line listing for each component, similar to the **describe –long** command.

–fmt *format-string*
Displays information using the specified *format-string*. See the **fmt_ccase** reference page for details.

–tre·e
: Recursively lists baselines and streams in the specified components. The output format is similar to that of the **lsvtree** command.

SPECIFYING THE COMPONENT *Default:* All components in the project VOB of the current directory.

–inv·ob *vob-selector*
: Displays information for all components in the specified project VOB.

component-selector ...
: Specifies one or more components for which information is displayed.

LISTING OBSOLETE COMPONENTS. *Default*: Lists only nonobsolete components.

–obs·olete
: Includes obsolete components in the listing. Obsolete components are those that have been processed with **lock –obsolete**.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Display a description of components in the specified VOB.

  cmd-context **lscomp -invob /vobs/projects**

```
17-Sep-99.12:06:59  parser  bill   "parser"
   root directory: "/vobs/parser"
29-Mar-99.17:23:16  applets  pklenk   "applets"
   root directory: "/vobs/applets"
29-Mar-99.17:23:25  booch  pklenk   "booch"
   root directory: "/vobs/booch"
29-Mar-99.17:23:37  libobj  pklenk   "libobj"
   root directory: "/vobs/libobj"
29-Mar-99.17:23:44  stage  pklenk   "stage"
   root directory: "/vobs/stage"
29-Mar-99.17:23:50  sun5_stage  pklenk   "sun5_stage"
   root directory: "/vobs/sun5_stage"
29-Mar-99.17:24:01  nt_i386_stage  pklenk   "nt_i386_stage"
   root directory: "/vobs/nt_i386_stage"
29-Mar-99.17:24:57  sys  pklenk   "sys"
   root directory: "/vobs/sys"
```

SEE ALSO

describe, **lsbl**, **mkcomp**, **rmcomp**

lsdo

Lists derived objects created by clearmake, omake, or clearaudit (dynamic views only)

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

lsdo [**–r·ecurse**] [**–me**] [**–l·ong** | **–s·hort** | **–fmt** *format-string*] [**–zer·o**]
[**–sti·me** | **–sna·me**] [**–nsh·areable_dos**] [*pname* ...]

DESCRIPTION

The **lsdo** command lists information about one or more derived objects (DOs) in a VOB. Derived objects are created by **clearmake**, **omake**, and **clearaudit** when these tools are invoked from a dynamic view. **lsdo** lists derived objects without respect to which dynamic views (if any) reference them. At any given time, a dynamic view sees at most one derived object at a given pathname.

By default, **lsdo** lists all derived objects built at a given pathname, except for the following kinds of DOs:

- Unshared DOs with a zero reference count (unless you use the **–zero** option).
- DO versions, derived objects that are checked in as versions of elements.
- Nonshareable DOs built in other dynamic views. (The **–nshareable_dos** option lists only nonshareable DOs in the current dynamic view.)
- Derived objects created with one name and subsequently renamed (for example, by **winkin –out** or some operating system command).

You can use *pname* arguments to restrict the listing to derived objects with particular pathnames, or to all the derived objects in particular directories. You can specify a derived object with a standard pathname, or with an extended name that includes a derived object's unique DO-ID.

DOs in Unavailable Dynamic Views

lsdo maintains a cache of tags of inaccessible dynamic views. For each view-tag, **lsdo** records the time of the first unsuccessful contact. Before trying to access a dynamic view, **lsdo** checks the cache. If the view's tag is not listed in the cache, **lsdo** tries to contact the dynamic view. If the view's tag is listed in the cache, **lsdo** compares the time elapsed since the last attempt with the time-out period specified by the CCASE_DNVW_RETRY environment variable. If the elapsed time is greater than the time-out period, **lsdo** removes the view-tag from the cache and tries to contact the dynamic view again.

The default timeout period is 60 minutes. To specify a different time-out period, set CCASE_DNVW_RETRY to another integer value (representing minutes). To disable the cache, set CCASE_DNVW_RETRY to 0.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. *Default:* If any *pname* argument is a directory, the DOs in *pname* are listed, but not the DOs in any subdirectories of *pname*.

–r·ecurse
Includes DOs in the entire subtree below any *pname* that is a directory (or the current working directory if you don't specify any *pname* arguments). VOB symbolic links are not traversed during the recursive descent into a directory.

SELECTION OF DERIVED OBJECTS. *Default:* **lsdo** lists DOs created by any user, but excludes DOs whose data containers no longer exist.

–me
Restricts the listing to derived objects that you created.

–zer·o
Includes in the listing unshared (that is, never-shared) derived objects with zero reference counts. Such objects cannot be candidates for configuration lookup and winkin, because their data containers no longer exist.

–nsh·areable_dos
Lists only nonshareable DOs created in the current dynamic view, by any user.

CONTROLLING REPORT APPEARANCE. *Default:* Each DO's listing includes its extended name (including DO-ID) along with creation-related data: time, user name, and host name. For example:

```
11-Jun.12:00 akp "hello.o@@11-Jun.12:00.554" on neptune
```

In a listing of several DOs, the entries are sorted by derived object name. Within a group of like-named DOs, the entries are sorted chronologically, most recent entry first. The **–long**, **–short**, and **–fmt** options are mutually exclusive; the **–sname** and **–stime** options are mutually exclusive.

–l·ong
: Expands the listing to include a DO's size in bytes, the last access time, the reference count, and the dynamic views that reference the DO.

–s·hort
: Restricts the listing for a DO to its extended name (including DO-ID).

–fmt *format-string*
: Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this report-writing facility.

–sti·me
: Sorts all entries chronologically, most recent entry first.

–sna·me
: (Same as default) Sorts entries alphabetically by name.

SPECIFYING THE DERIVED OBJECTS. *Default:* Lists all derived objects created in the current working directory.

pname ...
: Standard pathnames and/or DO-IDs:

- A directory name causes all derived objects built in that directory to be listed.
- A standard or view-extended pathname of a file causes all derived objects built under that name to be listed.
- A pathname that includes a unique DO-ID (for example, **conv.obj@@19-Nov.21:28.127450**) specifies a particular derived object to be listed.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List, in reverse chronological order, all derived objects that you have created in the current working directory.

  cmd-context **lsdo –stime –me –short**

  ```
  ctl@@14-May.15:18.339307
  ctl_V.o@@14-May.15:18.339305
  libcmd.a@@14-May.15:16.339302
  libcmd_V.o@@14-May.15:16.339300
  cmd_type.o@@14-May.15:15.339297
  cmd_view.o@@14-May.15:15.339294
  cmd_utl.o@@14-May.15:15.339291
  cmd_trig.o@@14-May.15:14.339288
  cmd_lh.o@@14-May.15:14.339285
  ```

- List information on a derived object, identified by its extended pathname.

  cmd-context **lsdo util.obj@@08-Dec.12:06.231**

  ```
  08-Dec.12:06 "util.obj@@08-Dec.12:06.231"
  ```

- List all nonshareable DOs in and under the current working directory.

  cmd-context **lsdo –recurse –nshareable_dos**

  ```
  20-Oct.16:35 "foo4.o@@20-Oct.16:35.2147484252"
  21-Oct.11:39 "foo7.dir/foo.cr_test.o@@21-Oct.11:39.2147484095"
  ```

- List all derived objects created in the current working directory with file name **hello**. Use the long format, to show which dynamic views reference the DOs; include DOs that are not referenced by any dynamic view.

  cmd-context **lsdo –long –zero hello**

  ```
  08-Dec-98.12:06:19 Chuck Jackson (test user) (jackson.dvt@oxygen)
   create derived object "hello@@08-Dec.12:06.234"
   size of derived object is: 18963
   last access: 29-Jan-99.13:56:56
   references: 1 => oxygen:/usr/vobstore/tut/old.vws
  08-Dec-98.12:05:35 Chuck Jackson (test user) (jackson.dvt@oxygen)
   create derived object "hello@@08-Dec.12:05.143"
   size of derived object is: 18963
   last access: 29-Jan-99.13:56:56
   references: 0 (shared)
  ```

- List the name, kind, and reference count of each derived object in the current working directory.

cmd-context **lsdo –fmt "%n\t%[DO_kind]p\t%[DO_ref_count]p\n"**

```
foo.c@@08-May.20:00.354170     shared     3
foo.c@@10-Jun.18:35.236855     shared     2
foo.c@@25-Sep.04:00.456        unshared   1
...
```

SEE ALSO

catcr, **clearaudit**, **clearmake**, **diffcr**, **fmt_ccase**, **omake**, **rmdo**

lsfolder

Lists information about folders

APPLICABILITY

Product	**Command Type**
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

lsfolder [**–s·hort** | **–l·ong** | **–fmt** *format-string* |
  –tre·e [**–fmt** *format-string*] [**–dep·th** *depth*] |
  –anc·estor [**–fmt** *format-string*] [**–dep·th** *depth*]]
  [**–inv·ob** *vob-selector* | **–in** *folder-selector* |
  –vie·w *view-tag* | **–cvi·ew** | *folder-selector* ...] [**–obs·olete**]

DESCRIPTION

The **lsfolder** command displays information that describes one or more folders.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

CHOOSING A DISPLAY FORMAT. *Default*: A one-line summary.

–s·hort
Displays only the the name of each folder.

–l·ong
Displays a detailed listing for a folder.

–fmt *format-string*
Displays information in the specified format. See the **fmt_ccase** reference page for more information.

–tre·e [**–fmt** *format-string*] [**–dep·th** *depth*]
Displays information about a folder and its contents. Default output format is similar to that of the **lsvtree** command.

The **–fmt** option displays information in the format specified by the *format-string* argument. See the **fmt_ccase** reference page for details.

The **–depth** option lists the hierarchy of objects to the level specified by the *depth* argument. The *depth* argument must be a positive integer.

–anc·estor [**–fmt** *format-string*] [**–dep·th** *depth*]
Displays information about a folder and any parent folders.

The **–fmt** option formats information using the specified *format-string.* See the **fmt_ccase** reference page for more information.

The **–depth** option specifies how many levels to display. The *depth* argument must be a positive integer: a value of zero lists the entire hierarchy.

SPECIFYING A FOLDER. *Default:* All folders in the project VOB of the current directory.

–inv·ob *vob-selector*
Displays a list of folders in the specified project VOB.

–in *folder-selector ...*
Displays a list of subfolders of the specified folder or folders.

–vie·w *view-tag*
Displays information about the parent folder of the stream attached to the specified view.

–cvi·ew
Displays information about the parent folder of the stream attached to the current view.

folder-selector ...
Specifies one or more folders to list.

LISTING OBSOLETE FOLDERS. *Default*: Lists only nonobsolete folders.

–obs·olete
Includes obsolete folders in the listing. Obsolete folders are those that have been processed with **lock –obsolete**.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Display a one-line summary of the specified folder.

  cmd-context **lsfolder Core_Parsers@/vobs/core_projects**

  ```
  17-Sep-99.11:21:36  Core_Parsers  bill   "Core_Parsers"
  ```

- Display a long listing for the specified folder.

  cmd-context **lsfolder -long RootFolder@/vobs/core_projects**

  ```
  folder "RootFolder"
   17-Sep-99.10:52:34 by Bill Marrs (bill.user@propane)
    "Predefined Root folder."
    owner: bill
    group: user
    title: Root folder
    contains folders:
      Parsers
      Core_Parsers
    contains projects:
      Java_Parser
  ```

SEE ALSO

chfolder, **lock**, **mkfolder**, **rmfolder**

lshistory

Lists event records for VOB-database objects

APPLICABILITY

Product	**Command Type**
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and Attache on UNIX—Display event records graphically:

  lsh·istory –g·raphical [**–nop·references** [[**–min·or**] [**–nco**]
  [**–use·r** *login-name*] [**–bra·nch** *branch-type-selector*]]]
  [[**–r·ecurse** | **–d·irectory** | **–a·ll** | **–avo·bs**]
  [**–pna·me**] *pname* ...
  | *object-selector* ...
  ]

- ClearCase and Attache on Windows—Display event records graphically:

  lsh·istory –g·raphical [**–nop·references** [[**–min·or**] [**–nco**]
  [**–use·r** *login-name*] [**–bra·nch** *branch-type-selector*]]]
  [[**–r·ecurse** | **–d·irectory** | **–a·ll** | **–avo·bs**]
  [**–pna·me**] *pname* ...
  | *object-selector* ...
  ]

- ClearCase LT on Windows—Display event records graphically:

  lsh·istory –g·raphical [**–nop·references** [[**–min·or**] [**–nco**]
  [**–use·r** *login-name*] [**–bra·nch** *branch-type-selector*]]]
  [[**–r·ecurse** | **–d·irectory** | **–a·ll** | **–avo·bs**]

[**–pna·me**] *pname* ...
| *object-selector* ...
]

- Display event records in the command window:

  lsh·istory [**–l·ong** | **–s·hort** | **–fmt** *format-string*] [**–eve·ntid**]
  [**–min·or**] [**–nco**] [**–las·t** [*num-events*]]
  [**–me** | **–use·r** *login-name*] [**–bra·nch** *branch-type-selector*]
  [[**–r·ecurse** | **–d·irectory** | **–a·ll** | **–avo·bs** | **–local**]
  [**–pna·me**] *pname* ... | *object-selector* ...]

DESCRIPTION

The **lshistory** command lists event records in reverse-chronological order, describing operations that have affected a VOB's data. There are several kinds of listing:

- **File-system data history —** Lists events concerning elements, branches, versions, and VOB links. This includes records for creation and deletion of objects, and records for attaching and removal of annotations: version labels, attributes, and hyperlinks.
- **Hyperlink history —** Lists events concerning hyperlink objects: creation, deletion, attaching/removal of attributes.
- **Type history —** Lists events concerning type objects that have been defined in the VOB.
- **Storage pool history —** Lists events concerning the VOB's storage pools.
- **VOB history —** Lists events concerning the VOB object itself. This includes the deletion of type objects and elements from the VOB.
- **VOB replica history —** Lists events concerning a VOB replica, including synchronization updates.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

Default: If you don't specify any objects to be listed, **lshistory** displays events for the file-system objects in the current working directory and events for the directory element itself. (This is equivalent to specifying "**.**" and "**.@@**" as the *pname* arguments.) The following sections describe how to produce a report on other file system objects, or on other kinds of objects.

LISTING EVENT RECORDS GRAPHICALLY. *Default:* Lists event records in the command window.

–g·raphical
Starts a browser that displays event records.

IGNORING PREFERENCES SETTINGS. *Default:* Displays the history browser with your saved settings for filtering.

–nop·references
: Temporarily overrides filtering settings. When used alone, uses default settings (displays all events except minor events). When used in combination with one or more of **–minor**, **–nco**, **–user**, or **–branch**, overrides your current filtering settings.

  NOTE: You cannot save your History Browser settings during a session that you invoked using the **–nopreferences** option.

REPORT FORMAT. *Default:* Default report formats appear below.

Default report format for an element:

```
02-Feb.10:51 scd create version "msg.c@@/main/rel2_bugfix/1"
 "Version for branch creation test"
02-Feb.10:51 scd create version "msg.c@@/main/rel2_bugfix/0"
02-Feb.10:51 scd create branch "msg.c@@/main/rel2_bugfix"
.
.
.
01-Feb.16:17 scd create file element "msg.c@@"
```

Default report format for a hyperlink:

```
08-Feb.11:25 scd create hyperlink "Merge@535@\tmp\scd_reach_hw"
```

Default report format for a storage pool:

```
01-Feb.16:05 scd create pool "cdft"
 "Predefined pool used to store cleartext versions."
```

–l·ong
: Expands the listing to include other object-specific information.

–s·hort
: Restricts the listing to names only: pathnames of file-system objects, names of type objects, or names of storage pools.

–fmt *format-string*
: Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this report-writing facility.

–eve·ntid
: Displays a numerical event-ID on the line preceding each event record (even if you use **–fmt**). You can change the comment assigned to an arbitrary event record by supplying an event-ID to the **chevent –event** command. Event-IDs remain valid until the VOB is reformatted with **reformatvob**.

SELECTING EVENTS FOR THE SPECIFIED OBJECTS. *Default:* The report includes all the major events in the entire histories of the selected objects.

NOTE: When using one or more of these options with **lshistory –graphical**, you must precede them with the **–nopreferences** option (the **–branch** option also has this requirement).

–min·or
Includes less important events in the listing: attaching of attributes, version labels, and so on. For type objects and storage pools, minor events include rename operations and changes to pool parameters (**mkpool –update**).

–nco
Excludes `checkout version` events (the ones listed by the **lscheckout** command).

–las·t [*num-events*]
Lists the specified number of events, starting with the most recent. If *num-events* is not specified, lists the most recent event.

NOTE: This option is mutually exclusive with **–recurse**.

–me
Lists events recorded for commands entered by the current user.

–use·r *login-name*
Lists events recorded for commands entered by the specified user.

FILE SYSTEM DATA HISTORY. Use the following to specify one or more file-system objects for a history listing.

–bra·nch *branch-type-selector*
Restricts the report to events relating to branches of the specified type. If you use this option with **–graphical**, you must precede **–branch** with the **–nopreferences** option. Specify *branch-type-selector* in the form [**brtype:**]*type-name*

type-name	Name of the branch type

–r·ecurse
Processes the entire subtree below any directory element encountered. VOB symbolic links are not traversed during the recursive descent.

NOTE: This option is mutually exclusive with **–last**.

–d·irectory
Lists information on a directory element itself, rather than on its contents.

–a·ll
Reports on all objects in the VOB containing *pname*: file-system objects, type objects, and

storage pools. If you omit *pname*, this option uses the VOB containing the current working directory. Specifying **–all** implicitly specifies **–local**.

–avo·bs

Similar to **–all**, but includes all VOBs active (mounted) on the local host. (If environment variable **CLEARCASE_AVOBS** is set to a list of VOB-tags, this set of VOBs is used instead.) If a VOB has multiple replicas, events from all the replicas are reported. Specifying **–avobs** implicitly specifies **–local**.

–local

Reports on local copies of types specified with *object-selector*. By default, **lshistory** displays the history of the global type for the object selector you specify. For more information about global types, see the *Administrator's Guide*.

–pna·me

Indicates that *pname* is a file-system object. Use this option when *pname* has the form of an object selector (for example, **lbtype:V3.0**).

pname ...

One or more pathnames, specifying elements and/or VOB symbolic links whose history is to be listed.

NOTE: You cannot use a *pname* argument like **foo.c@@\main** to restrict the report in this way.

object-selector ...

The object whose event records are to be displayed. The object must be in the VOB containing the current working directory, unless you use the *@vob-selector* suffix. Specify *object-selector* in one of the following forms:

vob-selector	**vob:***pname-in-vob* *pname-in-vob* can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system objects within the VOB (if the VOB is mounted). It cannot be the pathname of the VOB storage directory.
attribute-type-selector	**attype:***type-name*[*@vob-selector*]
branch-type-selector	**brtype:***type-name*[*@vob-selector*]
element-type-selector	**eltype:***type-name*[*@vob-selector*]
hyperlink-type-selector	**hltype:***type-name*[*@vob-selector*]
label-type-selector	**lbtype:***type-name*[*@vob-selector*]
trigger-type-selector	**trtype:***type-name*[*@vob-selector*]
pool-selector	**pool:***pool-name*[*@vob-selector*]
hlink-selector	**hlink:***hlink-id*[*@vob-selector*]

oid-obj-selector **oid:***object-oid*[@*vob-selector*]

The following object selector is valid only if you use MultiSite:
replica-selector **replica:***replica-name*[@*vob-selector*]

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List the event history of an element.

  cmd-context **lshistory hello.c**

```
08-Dec.12:05 jackson  import file element "hello.c@@"
20-May.15:41 cory     create version "hello.c@@/main/3" (REL2)
 "include name, home dir, and time in message
 KNOWN BUG: extra NL at end of time message"
07-May.08:34 akp      create version "hello.c@@/main/2" (REL1)
 "ANSI compatibility: declare return value type, make explicit return
value
 also: clean up wording for The Boss"
04-May.13:35 akp      create version "hello.c@@/main/1"
 "first implementation"
04-May.13:35 akp      create version "hello.c@@/main/0"
04-May.13:35 akp      create branch "hello.c@@/main"
04-May.13:35 akp      create file element "hello.c@@"
```

- List the history of a label type, using the long format.

  cmd-context **lshistory –long lbtype:REL1**

```
08-Jan-99.12:05:43 Chuck Jackson (test user) (jackson.dvt@oxygen)
 import label type "REL1"
15-Apr-99.14:45:00 ross.devt@neptune
 create label type "REL1"
 "create label for Release 1 of "hello world" program"
```

- For all elements in the current working directory, list events involving the **rel2_bugfix** branch.

  cmd-context **lshistory –branch rel2_bugfix**

```
24-Mar.12:45 jackson      create version "msg.c@@/main/rel2_bugfix/0"
24-Mar.12:45 jackson      create branch "msg.c@@/main/rel2_bugfix"
 "release 2 bugfixes"
23-Mar.20:40 jackson      create version "util.c@@/main/rel2_bugfix/1"
 "fix bug: extra NL in time string"
23-Mar.20:39 jackson      create version "util.c@@/main/rel2_bugfix/0"
23-Mar.20:39 jackson      create branch "util.c@@/main/el2_bugfix"
```

- List the latest event for every file element in or below the current directory.

  cleartool find . –type f –exec "cleartool lshistory –last %CLEARCASE_XPN%"

```
09-Jun.17:25   lee     create version ".\file.txt@@\main\1"
07-Jun.15:33   cty     create version ".\tests.txt@@\main\33"
17-May.23:44   ben     create version ".\dir1\comp.c@@\main\bugfix\45"
...
```

- List the history of the VOB object itself for the current VOB.

  cmd-context **lsh vob:.**

```
10-Dec.08:01 gomez      unlock versioned object base
"/home/gomez/personal"
09-Dec.15:48 gomez      lock versioned object base "/home/gomez/personal"
 "Locked for all users."
02-Oct.19:46 gomez      create versioned object base
"/home/gomez/personal"
 "gomez's personal vob"
```

- Start a history browser, overriding the saved filtering settings and displaying events for the **hello.c** element that are related to the **v4_test** branch.

  cmd-context **lshistory –graphical –nopreferences –branch v4_test hello.c**

SEE ALSO

chevent, **describe**, **find**, **events_ccase**, **fmt_ccase**, **lscheckout**, **lspool**, **lstype**, **lsvtree**, **xclearcase**

lslocal

Lists the files in the workspace

APPLICABILITY

Product	Command Type
Attache	command

Platform
UNIX
Windows

SYNOPSIS

lslocal [**–r·ecurse**] [**–mod·ified**] [**–nco**] [*pname*...]

DESCRIPTION

The **lslocal** command lists the name of the files in the workspace.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

HANDLING OF DIRECTORY ARGUMENTS. *Default:* For each *pname* that specifies a directory element, **lslocal** lists the contents of that directory, but not the contents of any of its subdirectories.

–r·ecurse
Includes a listing of the entire subtree below any subdirectory included in the top-level listing. VOB symbolic links are not traversed during the recursive descent.

SPECIFYING THE OBJECTS TO BE LISTED. *Default:* If you don't specify any other options, all files in the current working directory are listed; all subdirectory entries are listed, but not the contents of these subdirectories.

–mod·ified
Restricts the listing to writable files only.

–nco
Restricts the listing to writable files corresponding to elements that are not checked out to the workspace view. This will help you to identify files you modified without checking them out, for example, while you were working disconnected from the view.

pname...

Restricts the listing to the specified files and/or directories. Wildcard patterns apply to the workspace contents; / (slash) denotes the root of the workspace. For example, **/*.c** refers to all of the **.c** files in the workspace root. (See the **wildcards** reference page for more information.) You can use either slashes or backslashes.

EXAMPLES

- List the files downloaded to the current workspace working directory.

```
\tmp\jo_agora_hw\src>  lslocal
\tmp\jo_agora_hw
\tmp\jo_agora_hw\src
```

- List the writable files in the current working directory of the workspace that are not checked out.

```
\tmp\jo_agora_hw\src>  lslocal –nco
```

- List the writable files in the specified directory of the current workspace.

```
\tmp\jo_agora_hw\src>  lslocal –modified \tmp\jo_agora_hw\src
\tmp\jo_agora_hw\src\hello.c
```

- List all the files in the current workspace.

```
\tmp\jo_agora_hw\src>  lslocal –recurse \
\tmp
\tmp\jo_agora_hw
\tmp\jo_agora_hw\src
\tmp\jo_agora_hw\src\hello.c
\tmp\jo_agora_hw\src\Makefile
\tmp\jo_agora_hw\src\hello.h
\tmp\jo_agora_hw\src\msg.c
\tmp\jo_agora_hw\src\util.c
```

SEE ALSO

get, **ls**

lslock

Lists locks on objects

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

lslock [**–local**] [**–l·ong** | **–s·hort** | **–fmt** *format-string*] [**–obs·olete**]
[[**–a·ll**] [**–pna·me**] *pname* ...
| *object-selector* ...
]

DESCRIPTION

The **lslock** command lists locks that have been placed on one or more VOB-database objects (with the **lock** command). The listing can include all the locks created within a VOB or a particular set of locks:

- Locks on elements or branches
- Locks on type objects
- Locks on VOB replica objects
- Locks on VOB storage pools
- The lock on the VOB object itself

Obsolete Type Objects

Type objects can be rendered obsolete with the **lock –obsolete xxtype:** command. **lslock** lists an obsolete type object if you specify its name with a *type-name* argument or you use the **–obsolete** option.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

LISTING LOCK STATE OF LOCAL COPIES OF GLOBAL TYPES. *Default:* **lslock** displays the lock state of the global type for the object selector you specify.

–local
Displays the lock state of the local copy of the global type. For more information, see the *Administrator's Guide*.

REPORT FORMAT. *Default:* A lock listing looks like this:

```
01-Sep.08:42 drp lock attribute type "AT2" (locked)
 "Locked for all users."
```

–l·ong
Expands the listing with more time-specific and user-specific information.

–s·hort
Restricts the listing to names of locked objects only.

–fmt *format-string*
Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this report-writing facility.

LISTING OBSOLETE OBJECTS. *Default:* An obsolete object is not listed unless you specify it with a command-line argument.

–obs·olete
Includes obsolete objects in the listing. (Has no effect if you specify one or more objects with arguments.)

SPECIFYING THE LOCKED OBJECTS. *Default:* Lists all the locks created in the VOB containing the current working directory.

[**–pna·me**] *pname* ...
One or more pathnames, each of which specifies an element or branch:

`foo.c`	Element **foo.c**
`foo.c@@`	Element **foo.c**
`foo.c@@/main/bugfix`	Branch of element **foo.c**

(Versions cannot be locked; a pathname to a version references the element object.) Using *pname* arguments restricts the listing to locks on those particular objects (but see the **–all** description below).

If *pname* has the form of an object selector, you must include the **–pname** option to indicate that *pname* is a pathname.

NOTE: Specifying an element lists only the lock on the element itself, not on any of its branches.

–all

For each *pname* argument, lists all locks in the VOB containing *pname*. Has no effect if you don't specify any *pname* argument (because the default is to list all locks in the current VOB).

object-selector ...

One or more non-file-system VOB objects. The objects must exist in the VOB containing the current working directory, unless you specify another VOB with @*vob-specifier*. Specify *object-selector* in one of the following forms:

vob-selector	[**vob:**]*pname-in-vob* *pname-in-vob* can be the pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted). It cannot be the pathname of the VOB storage directory.
attribute-type-selector	**attype:***type-name*[@*vob-selector*]
branch-type-selector	**brtype:***type-name*[@*vob-selector*]
element-type-selector	**eltype:***type-name*[@*vob-selector*]
hyperlink-type-selector	**hltype:***type-name*[@*vob-selector*]
label-type-selector	**lbtype:***type-name*[@*vob-selector*]
trigger-type-selector	**trtype:***type-name*[@*vob-selector*]
pool-selector	**pool:***pool-name*[@*vob-selector*]
oid-obj-selector	**oid:***object-oid*[@*vob-selector*]

The following object selector is valid only if you use MultiSite:

replica-selector	**replica:***replica-name*[@*vob-selector*]

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List the locks on three label types.

  cmd-context **lslock lbtype:REL1 lbtype:REL1.1 lbtype:REL2**

```
08-Dec.12:19 jackson lock label type "REL1" (locked)
 "Locked for all users."
08-Dec.12:19 jackson lock label type "REL1.1" (locked)
 "Locked for all users."
08-Dec.12:19 jackson lock label type "REL2" (locked)
 "Locked for all users."
```

- List the lock on a particular branch of a particular element.

  cmd-context **lslock util.c@@/main/rel2_bugfix**

```
08-Dec.12:19 jackson lock branch "util.c" (locked)
 "Locked for all users."
```

- List the entire-VOB lock on the current VOB, in long format.

  cmd-context **lslock –long vob:.**

```
08-Dec-98.14:57:58 Chuck Jackson (test user) (jackson.dvt@oxygen)
 lock versioned object base "s:\people\chuck\hw.vbs" (locked)
 "Locked for all users."
```

- List all locked objects (including the obsolete ones) in the current VOB.

  cmd-context **lslock –obsolete**

```
08-Dec.12:18 jackson lock file element
   "/usr/hw/src/hello.c@@" (locked)
"Locked for all users."
08-Dec.12:19 jackson lock label type "REL1" (locked)
 "Locked for all users."
08-Dec.12:19 jackson lock label type "REL2" (locked)
 "Locked for all users."
08-Dec.12:18 jackson lock branch type "test" (locked)
 "Locked except for users: gomez jackson"
08-Dec.12:18 jackson lock branch type "patch3" (obsolete)
 "Locked for all users (obsolete)."
08-Dec.12:18 jackson lock file element
   "/usr/hw/src/convolution.c@@" (locked)
"Locked for all users."
08-Dec.12:19 jackson lock branch
   "/usr/hw/src/util.c@@/main/rel2_bugfix@@" (locked)
"Locked for all users."
```

- List the locks on two of the current VOB's storage pools.

  cmd-context **lslock pool:staged pool:cdft**

```
08-Dec.12:19 jackson lock pool "staged" (locked)
 "Locked for all users."
08-Dec.12:19 jackson lock pool "cdft" (locked)
 "Locked for all users."
```

SEE ALSO

lock, **ls**, **lshistory**, **lspool**, **lstype**, **unlock**, **fmt_ccase**

lsmaster

Lists objects mastered by a replica

APPLICABILITY

Product	Command type
ClearCase	cleartool subcommand
MultiSite	multitool subcommand

Platform
UNIX
Windows

SYNOPSIS

lsmaster [**–kind** *object-selector-kind*[**,**...]] [**–fmt** *format-string*] [**–view** *view-tag*]
[**–inr·eplicas** { **–all** | *replica-name*[**,**...] }] *master-replica-selector* ...

DESCRIPTION

This command lists objects mastered by a replica. By default, the command uses only the information known to your current replica. If you list objects mastered by a sibling replica, changes that have not been imported at your current replica are not included in the output. For example, a label type is added at replica **sanfran_hub**, but replica **boston_hub** has not imported the update packet containing the change. If you enter the command **multitool lsmaster sanfran_hub** at the **boston_hub** replica's site, the output does not include the new label type.

To retrieve information from a sibling replica, use **–inreplicas**. This form of the command contacts the sibling replicas and works only between sites that have IP connections. If **lsmaster** cannot contact a replica, it prints an error and tries to contact the next replica you specified.

Object Name Resolution

If you have a view context, **lsmaster** uses the view to resolve object identifiers (OIDs) of file system objects to the names of the objects. If you do not have a view context, **lsmaster** prints OIDs for file system objects. You can specify a view context with the **–view** option.

When you specify **–inreplicas**, **lsmaster** prints OIDs for objects whose creation operations have not yet been imported at your current replica.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE OBJECT KINDS. *Default:* **lsmaster** lists all objects mastered by the replica.

–kind *object-selector-kind*[**,**...]

Limits the listing to the specified object kinds. The list of object kinds must be comma-separated, with no spaces. *object-selector-kind* can be one of the following values:

Values for ClearCase:

attype
branch
brtype
delem (directory element)
eltype
felem (file element)
hlink
hltype
lbtype
slink
vob

Values for ClearCase UCM:

activity
baseline
component
folder
project
stream

Values for MultiSite:

replica

REPORT FORMAT. *Default:* For file-system objects, the master replica, object kind, and OID of each object are listed. For example:

```
master replica: boston_hub@/vobs/dev  file element:oid:40e022a3.241d11ca ...
```

For non-file-system objects, the master replica, object kind, and name of each object are listed. For example:

```
master replica: boston_hub@/vobs/dev  brtype:main
```

–fmt *format-string*
Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this option.

SPECIFYING A VIEW CONTEXT. *Default:* The command uses your current view context.

–view *view-tag*
Specifies a view.

SPECIFYING THE REPLICA FROM WHICH TO RETRIEVE INFORMATION. *Default:* The command uses the information in your current replica.

–inr·eplicas { **–all** | *replica-name*[**,**...] }
With **–all**, retrieves information from all replicas in the VOB family (except deleted replicas). Otherwise, retrieves information from the sibling replicas you specify. The list of replicas must be comma-separated, with no spaces.

SPECIFYING THE REPLICA WHOSE MASTERED OBJECTS ARE DISPLAYED. *Default:* No default; you must specify a replica.

master-replica-selector ...
Lists objects mastered by the specified replica. Specify *master-replica-selector* in the form [**replica:**]*replica-name*[@*vob-selector*]

replica-name	Name of the replica
vob-selector	VOB family of the replica; can be omitted if the current working directory is within the VOB. Specify *vob-selector* in the form [**vob:**]*pname-in-vob* *pname-in-vob* — Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

EXAMPLES

- List all objects mastered by the replica **sanfran_hub**.

  multitool lsmaster –view v4.1 –fmt "%m:%n\n" sanfran_hub@/vobs/dev

```
directory element:/vobs/dev.@@
...
file element:/vobs/dev/lib/file.c@@
...
symbolic link:/vobs/dev/doc
...
hyperlink:Merge@2@/vobs/dev
...
```

- List all label types mastered by the replica **boston_hub**.

  cleartool lsmaster –fmt "%m:%n\n" –kind lbtype boston_hub@\doc
  ```
  label type:LATEST
  label type:CHECKEDOUT
  label type:BACKSTOP
  label type:REL1
  ...
  ```

- List all element types, label types, and branch types mastered by the replica **sanfran_hub**.

  cleartool lsmaster –kind eltype,lbtype,brtype sanfran_hub
  ```
  master replica: sanfran_hub@\dev "element type" file_system_object
  master replica: sanfran_hub@\dev "element type" file
  master replica: sanfran_hub@\dev "element type" directory
  ...
  master replica: sanfran_hub@\dev "branch type" sanfran_main
  master replica: sanfran_hub@\dev "branch type" v1.0_bugfix
  ...
  master replica: sanfran_hub@\dev "label type" LATEST
  master replica: sanfran_hub@\dev "label type" SANFRAN_V2.0
  master replica: sanfran_hub@\dev "label type" V1.0_BUGFIX
  master replica: sanfran_hub@\dev "label type" TOKYO_BASE
  master replica: sanfran_hub@\dev "label type" SYDNEY_BASE
  ...
  ```

- List the name and creation comment of each element type mastered by the replica **boston_hub**. Contact the **boston_hub** replica to retrieve the data.

  **multitool lsmaster –inreplicas boston_hub –fmt "%n\t%c\n" **
  –kind eltype boston_hub@/vobs/dev
  ```
  In replica "boston_hub"
  binary_delta_file       Predefined element type used to represent a file
  in binary delta format.
  ...
  ```

- List information from all replicas in the VOB family about the objects mastered by the replica **sanfran_hub**. Do not use a view context.

  multitool lsmaster –inreplicas –all sanfran_hub@/vobs/dev
  ```
  In replica "boston_hub"
  master replica: sanfran_hub@/vobs/dev "versioned object base" /vobs/dev
  master replica: sanfran_hub@/vobs/dev "directory element"
  (oid:40e0000b.241d23ca.b3df.08:00:69:02:05:33)
  master replica: sanfran_hub@/vobs/dev "directory element"
  (oid:40e0000b.241d23ca.b3df.08:00:69:02:05:33)
  ...
  ```

Use a view context:

multitool lsmaster –view v4.1 –inreplicas –all sanfran_hub@/vobs/dev

```
In replica "boston_hub"
master replica: sanfran_hub@/vobs/dev "versioned object base" /vobs/dev
master replica: sanfran_hub@/vobs/dev "directory element"
/view/v4.1/vobs/dev/.@@
master replica: sanfran_hub@/vobs/dev "directory element"
/view/v4.1/vobs/dev/lib@@
...
```

- List information from the **sanfran_hub** replica about the objects mastered by the replica **boston_hub**.

  multitool lsmaster –view v4.1 –inreplicas sanfran_hub boston_hub@\doc

- List all projects, baselines, and streams mastered by the replica **boston_hub**. Contact the **boston_hub** replica to retrieve the data.

**multitool lsmaster –inreplicas boston_hub –kind project,baseline,stream \
boston_hub@/vobs/projects**

```
In replica "boston_hub"
master replica: boston_hub@/vobs/projects  "project" V4.5.BL3
master replica: boston_hub@/vobs/projects  "project" doc_localize
master replica: boston_hub@/vobs/projects  "stream" 4.5.bl2_int
master replica: boston_hub@/vobs/projects  "project" V4.5.BL2
master replica: boston_hub@/vobs/projects  "stream" 4.5.bl2
master replica: boston_hub@/vobs/projects  "stream" stream000317.160434
master replica: boston_hub@/vobs/projects  "stream" stream000317.173156
master replica: boston_hub@/vobs/projects  "baseline" V4.5.BL2.011005.12820
master replica: boston_hub@/vobs/projects  "baseline" V4.5.BL2.011005.12890
master replica: boston_hub@/vobs/projects  "baseline" V4.5.BL2.011005.17408
master replica: boston_hub@/vobs/projects  "baseline" V4.5.BL2.011005.17695
master replica: boston_hub@/vobs/projects  "baseline" V4.5.BL2.011005.19614
...
```

SEE ALSO

chmaster, **describe**, **reqmaster**
Introduction to MultiSite and *Managing Mastership* in the *Administrator's Guide* for Rational ClearCase MultiSite.

lspool

Lists VOB storage pools

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

lspool [**–l·ong** | **–s·hort** | **–fmt** *format-string*] [**–obs·olete**]
[**–inv·ob** *vob-selector* | *pool-selector* ...]

DESCRIPTION

The **lspool** command lists information about one or more VOB storage pools. This listing does not include the elements assigned to the pool; use the **find** command for this purpose. For example:

cmd-context **find /vobs/include -element 'pool(src_pool_2)' -print**

Obsolete Storage Pools

Storage pools can be rendered obsolete with the **lock –obsolete** command. The obsolete/nonobsolete status of a pool affects some forms of this command.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

LISTING FORMAT. *Default:* A storage pool listing looks like this:

```
20-Nov-1999 drp pool "cdft"
```

–l·ong

Expands the listing to pool parameters and pathnames.

–s·hort

Restricts the listing to pool names only.

–fmt *format-string*

Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this report-writing facility.

LISTING OBSOLETE POOLS. *Default:* If you don't specify any *pool-name* argument, a VOB's obsolete pools are suppressed from the listing.

–obs·olete

Includes obsolete pools in the listing when you don't specify any *pool-name* argument. Has no effect if you specify one or more *pool-name* arguments.

SPECIFYING THE POOLS. *Default:* Lists all storage pools in the VOB containing the current working directory.

–inv·ob *vob-selector*

The VOB whose storage pools are to be listed. Specify *vob-selector* in the form [**vob:**]*pname-in-vob*

pname-in-vob	Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted).

pool-selector ...

One or more names of storage pools to be listed. A pool is listed whether or not it is obsolete. Specify *pool-selector* in the form [**pool:**]*pool-name*[@*vob-selector*]

pool-name	Name of the storage pool
vob-selector	Object-selector for a VOB, in the same format as **–invob**.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, <u>*cmd-context*</u> represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, <u>*cmd-context*</u> represents the interactive **cleartool** prompt. In Attache, <u>*cmd-context*</u> represents the workspace prompt.

- List all storage pools for the VOB containing the current working directory.

cmd-context **lspool**

```
08-Dec.12:1     jackson     pool "c_pool"
 "pool for c source files"
15-Dec.09:34     jenny       pool "cdft"
 "Predefined pool used to store cleartext versions."
08-Dec.12:21     jackson     pool "cltxt2"
15-Dec.09:34     jenny       pool "ddft"
 "Predefined pool used to store derived objects."
08-Dec.12:21     jackson     pool "do1"
08-Dec.12:21     jackson     pool "my_ctpool"
 "alternate cleartext pool"
15-Dec.09:34     jenny       pool "sdft"
 "Predefined pool used to store versions."
08-Dec.12:19     jackson     pool "staged"
```

- List information about a particular storage pool, in long format.

  cmd-context **lspool –long do1**

```
pool "do1"
 08-Dec-99.12:21:13 by jackson.dvt
 owner: jackson
 group: dvt
 kind: derived pool
 pool storage link target pathname "/net/oxygen/usr1/vob_pools/tut/d/do1
 pool storage global pathname "/net/oxygen/usr/vobstore/tut/tut.vbs/d/do1"
 maximum size: 10000 reclaim size: 8000 age: 168
```

- List a particular storage pool, verifying that it is obsolete.

  cmd-context **lspool –short cltxt2**

```
cltxt2 (obsolete)
```

SEE ALSO

chpool, **mkpool**

lsprivate

Lists objects in a dynamic view's private storage area

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

lsp·rivate [**–tag** *view-tag*] [**–inv·ob** *vob-selector*] [**–l·ong** | **–s·hort**]
[**–siz·e**] [**–age**] [**–co**] [**–do**] [**–oth·er**]

DESCRIPTION

The **lsprivate** command lists the file-system objects that belong to a dynamic view:

- View-private files, links, and directories
- Derived objects, including
  - Nonshareable derived objects
  - Unshared derived objects
  - Shared derived objects that are cataloged in (visible through) the dynamic view, even though their data containers are stored in a VOB storage pool
- Checked-out versions of file elements

Except for the shared derived objects, all of these objects are stored in the dynamic view's private storage area.

This command does not list checked-out directory elements, because such a checkout does not produce a view-private object. To list directory checkouts, use the **lscheckout** command.

The objects are listed with full pathnames (thus including the VOB-tag), one per line.

NOTE: **lsprivate** does not work in a snapshot view. In a snapshot view, (**cleartool**) **ls -recurse -view_only** provides output equivalent to that of **lsprivate**.

STRANDED VIEW-PRIVATE FILES

lsprivate sometimes lists a view-private file in a special way, because the file has become *stranded*: it has no name in the VOB namespace, as currently constructed by your dynamic view. There are several possible causes and, hence, several actions you can take.

File Still Accessible Through Some Directory Version

The **lsprivate** listing for a file can include a version-extended pathname to some directory element:

```
/usr/hw/src@@/main/3/subdir1/canUCme
```

In this example, file **canUCme** is stranded because its parent directory, **subdir1**, does not appear in the dynamic view as it is currently configured; but the file can be accessed through **version /main/3**of directory element **src**, which contains an entry for **subdir1**. (Note that you cannot use this pathname to access the view-private object. A version-extended pathname can refer only to an element, branch, or version—not to a view-private file.)

To make a stranded file visible again, you must make its parent directory visible, by reconfiguring the dynamic view (in this case, to select **version /main/3** of directory element **src**).

VOB Is Inactive

If a VOB is not currently active on your host, all view-private files corresponding to that VOB are temporarily stranded. **lsprivate** displays a warning message and prefixes a number sign (#) to pathnames within that VOB:

```
cleartool: Warning: VOB not mounted: "M:\jc_vw\jc_hw"
  VOB UUID is 1127d379.428211cd.b3fa.08:00:69:06:af:65

    .
    .
    .

#M:\jc_vw\jc_hw\src\.cmake.state
#M:\jc_vw\jc_hw\src\findmerge.log.18-Mar-99.13:43:27
#M:\jc_vw\jc_hw\src\hello
#M:\jc_vw\jc_hw\src\hello.o

    .
    .
    .
```

Reactivating the VOB on your host restores **lsprivate** command output to normal for pathnames within that VOB.

VOB Is Inaccessible

If a VOB has been unregistered, all view-private files corresponding to that VOB are temporarily stranded; if the VOB has been deleted, the view-private files are stranded permanently. **lsprivate** cannot distinguish these cases; it may infer the VOB's likely VOB-tag, but it lists the view-private files with an `Unavailable-VOB` prefix:

```
cleartool: Error: Unable to get VOB object registry information for
  replica uuid "1127d379.428211cd.b3fa.08:00:69:06:af:65".
cleartool: Warning: VOB is unavailable -- using name: "<Unavailable-VOB-1>".
  If it has been deleted use 'recoverview -vob <uuid>'
  VOB UUID is 1127d379.428211cd.b3fa.08:00:69:06:af:65
  Last known location of storage is phobos:/usr/people/avid/tut/tut.vbs
#<Unavailable-VOB-1>/<DIR-3587d464.428211cd.b40c.08:00:69:06:af:65>/.cmake.sta
te
#<Unavailable-VOB-1>/<DIR-3587d464.428211cd.b40c.08:00:69:06:af:65>/findmerge.
log.18-Mar-99.13:43:27
```

The procedure for cleaning up stranded view-private files is described in the *Administrator's Guide*.

Directory Element Has Been Deleted

If a directory element (or its entire VOB) has been deleted, all the corresponding view-private files are permanently stranded. They are listed with the VOB's UUID, as above, with no remedy possible, except to use **recoverview** to move the files to the dynamic view's **lost+found** directory (as described in the *Administrator's Guide*).

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE VIEW. *Default:* The current dynamic view is listed; a working directory view takes precedence over a set view.

–tag *view-tag*
: The view-tag of any registered dynamic view to which you have read access.

LISTING STYLE. *Default:* Checked-out versions are annotated with `[checkedout]`.

–l·ong
: Lists objects in the style of **ls –long**.

–s·hort
: Lists pathnames only, without annotations.

–siz·e
Lists each file's size in bytes. At the end of the listing, lists the total size of view private files and of shared DOs in the view.

–age
Lists the last access time of each file.

SELECTING OBJECTS TO LIST. *Default:* All of the dynamic view's objects are listed. You can use **-co**, **-do**, and **-other** in any combination to specify a partial listing.

–inv·ob *vob-selector*
Restricts the listing to objects for the specified VOB. Specify *vob-selector* in the form [**vob:**]*pname-in-vob*

pname-in-vob	Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

NOTE: Specifying a pathname within the VOB does not limit the listing to objects in and below that directory.

–co
Lists checked-out versions of file elements. (Checked-out directory elements are never listed by **lsprivate**.)

–do
Lists derived objects.

–oth·er
Lists view-private files and directories that are neither checked-out versions of file elements nor derived objects.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, <u>*cmd-context*</u> represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, <u>*cmd-context*</u> represents the interactive **cleartool** prompt. In Attache, <u>*cmd-context*</u> represents the workspace prompt.

- List the private objects in the dynamic view with view-tag **jackson_vu**, from the VOB identified by the pathname **/usr/hw/src**.

cmd-context **lsprivate -tag jackson_vu -invob /usr/hw/src**

```
/usr/hw/src/bug.report
/usr/hw/src/convolution.c [checkedout]
/usr/hw/src/edge.sh
/usr/hw/src/hello
/usr/hw/src/hello.c [checkedout]
/usr/hw/src/hello.h [checkedout]
/usr/hw/src/hello.o
/usr/hw/msg.o
/usr/hw/util.c [checkedout]
/usr/hw/util.c.contrib
/usr/hw/util.c.contrib.1
/usr/hw/util.o
```

- List the private objects in the dynamic view with view-tag **jc_vw**, from the VOB identified by the pathname **M:\jc_vw\jc_hw\src**.

  cmd-context **lsprivate -tag jc_vw -invob M:\jc_vw\jc_hw\src**

```
M:\jc_vw\jc_hw\src\bug.report
M:\jc_vw\jc_hw\src\convolut.c [checkedout]
M:\jc_vw\jc_hw\src\edge.sh
M:\jc_vw\jc_hw\src\hello.c [checkedout]
M:\jc_vw\jc_hw\src\hello.h [checkedout]
M:\jc_vw\jc_hw\src\hello.o
M:\jc_vw\jc_hw\msg.o
M:\jc_vw\jc_hw\util.c [checkedout]
M:\jc_vw\jc_hw\util.c.contrib
M:\jc_vw\jc_hw\util.c.contrib.1
M:\jc_vw\jc_hw\util.o
```

- List all checked-out versions of elements in the current dynamic view, from all VOBs.

  cmd-context **lsprivate –co**

```
/usr/hw/src/convolution.c [checkedout]
/usr/hw/src/hello.c [checkedout]
/usr/hw/src/util.c [checkedout]
/vobs/doc/PLAN/DocumentationProposal[checkedout]
/vobs/doc/reference_man/test/attest.dat [checkedout]
/vobs/doc/reference_man/test/testelem.c [checkedout]
```

- List all elements in the current dynamic view, from all VOBs, using a long listing.

  cmd-context **lsprivate –long**

```
view private
object   M:\jc_vw\tmp_vob\scd_reach\src\findmerge.log.04-Feb-99.10:01:01
view private
object   M:\jc_vw\tmp_vob\scd_reach\src\findmerge.log.04-Feb-99.11:00:59
version               M:\jc_vw\doc\reqs@@\main\CHECKEDOUT from \main\33
Rule: element * CHECKEDOUT
version               M:\jc_vw\doc\specs@@\main\CHECKEDOUT from \main\7
Rule: element * CHECKEDOUT
```

- List the size and age of all private objects in the current dynamic view.

  cmd-context **lsprivate –size –age**

```
/tmp/sg_test/.cmake.state
     Size: 2724
     Age: 05-Apr-99.16:01:10
/tmp/sg_test/bar
     Size: 10
     Age: 05-Apr-99.16:00:54
/tmp/sg_test/foo
     Size: 10
     Age: 05-Apr-99.16:00:53
/tmp/sg_test/foobar
     Size: 20
     Age: 05-Apr-99.16:00:55
total size of view private files is 2764
total size of shared derived objects is 0
```

SEE ALSO

checkout, **ls**, **lscheckout**

lsproject

Lists information about a project

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

lsproj·ect [**–s·hort** | **–l·ong** | **–fmt** *format-string* |
–tre·e [**–fmt** *format-string*] [**–dep·th** *depth*] |
–anc·estor [–**fmt** *format-string*] [**–dep·th** *depth*]] [**–obs·olete**]
[**–inv·ob** *vob-selector* | **–in** *folder-selector* |
–vie·w *view-tag* | **–cvi·ew** | *project-selector* ...]

DESCRIPTION

The **lsproject** command lists information for one or more projects.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SELECTING A DISPLAY FORMAT. *Default*: A one-line summary.

–s·hort

Displays project names only.

–l·ong

Displays the following detailed information for a project:

- Name
- The folder it is in
- Integration stream
- Title (if different from the name)

- Modifiable components
- Default rebase promotion level
- Recommended baselines
- Policies

–fmt *format-string*
Displays information in the specified format. See the **fmt_ccase** reference page for details.

–tre·e [**–fmt** *format-string*] [**–dep·th** *depth*]
Displays information for a project, including its hierarchy of streams and activities. You can modify how information is displayed with the **–fmt** and **–depth** options.

The **–fmt** option displays information using the specified format. See the **fmt_ccase** reference page for more information.

The **–depth** option lists the hierarchy of objects to the level specified by the *depth* argument. The *depth* argument must be a positive integer.

–anc·estor [**–fmt** *format-string*] [**–dep·th** *depth*]
Displays information about one or more projects and its parent folders. You can modify how information is displayed with the **–fmt** and **–depth** options.

The **–fmt** option displays information using the specified format. See the **fmt_ccase** reference page for more information.

The **–depth** option lists the hierarchy of objects to the level specified by the *depth* argument. The *depth* argument must be a positive integer.

LISTING OBSOLETE PROJECTS. *Default*: List only nonobsolete projects.

–obs·olete
Includes obsolete projects in the listing. Obsolete projects are those that have been processed with **lock –obsolete**.

SPECIFYING THE PROJECT. *Default:* All projects in the project VOB of the current directory.

–inv·ob *vob-selector*
Displays a list of all projects in the specified project VOB.

–in *folder-selector*
Displays a list of projects in the specified folder.

–vie·w *view-tag*
Displays information for the project containing the stream attached to the specified view.

–cvi·ew
Displays information for the project containing the stream attached to the current view.

project-selector ...

Specifies one or more projects to list.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Display a detailed description of the specified project.

  cmd-context **lsproject -long p7**

```
project "p7"
10-April-01.17:43:08 by Sheila Hunt (sheila.user@echo01)
owner: sheila
group: user
folder: RootFolder@/vobs/sheila_testpvob
integration stream: int7@/vobs/sheila_testpvob
modifiable components:
   c1@/vobs/sheila_testpvob
   c2@/vobs/sheila_testpvob
default rebase promotion level: INITIAL
recommended baselines:
   bld.bl3.161@/vobs/sheila_testpvob (c1@/vobs/sheila_testpvob)
   bld.bl2.181@/vobs/sheila_testpvob (c2@/vobs/sheila_testpvob)
policies:
   POLICY_UNIX_INT_SNAP disabled
   POLICY_UNIX_DEV_SNAP disabled
   POLICY_WIN_INT_SNAP disabled
   POLICY_WIN_DEV_SNAP disabled
   POLICY_INTRAPROJECT_DELIVER_FOUNDATION_CHANGES disabled
   POLICY_INTRAPROJECT_DELIVER_ALLOW_MISSING_TGTCOMPS disabled
   POLICY_INTERPROJECT_DELIVER disabled
   POLICY_INTERPROJECT_DELIVER_FOUNDATION_CHANGES disabled
   POLICY_INTERPROJECT_DELIVER_REQUIRE_TGTCOMP_VISIBILITY disabled
   POLICY_INTERPROJECT_DELIVER_ALLOW_NONMOD_TGTCOMPS disabled
```

- Display a one-line summary of the project containing the stream attached to the specified view.

  cmd-context **lsproject –view java_int**

  ```
  17-Sep-99.11:24:18 Java_Parser_Project_28174 bill "Java Parser Project"
  ```

SEE ALSO

chproject, **mkproject**, **rmproject**

lsregion

Lists ClearCase network regions

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

lsregion [**–s·hort** | **–l·ong**] [**'***region-tag-pattern***'** ...]

DESCRIPTION

The **lsregion** command lists one or more ClearCase network regions.

To be accessible to **cleartool** subcommands, including **lsregion**, a region must have an entry in the **regions** registry file on the network's registry server host. For more information about registry files, see the *Administrator's Guide*.

NOTE: To display the name of the registry server host for which entries are displayed, use the **hostinfo –long** command.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

LISTING FORMAT. *Default:* **lsregion** displays only region names.

–s·hort
Same as the default. Displays only region names.

–l·ong
Displays region names and their comment strings.

SPECIFYING THE REGIONS LISTED. *Default:* Lists all registered network regions.

'*region-tag-pattern*' ...

Confines the listing to regions that match one or more *region-tag-pattern*s. A *region-tag-pattern* can include pattern-matching characters as described in **wildcards_ccase**. Enclose each pattern within single quotes.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List all ClearCase network regions known to the current host's registry server host.

  cmd-context **lsregion**
  ```
  dev_unix
  support
  winnt1
  ...
  ```

- Display all information registered for each network region that matches the wildcard pattern `*winnt*`.

  cmd-context **lsregion –long '*winnt*'**
  ```
  Tag: winnt1  "region defined automatically"
  Tag: winnt2  "region defined implicitly by V2.x client"
  ```

UNIX FILES

/var/adm/atria/rgy/regions

WINDOWS FILES

*ccase-home-dir***var\rgy\regions**

WINDOWS REGISTRY KEYS

HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\Region

SEE ALSO

mkregion, **rmregion**, *Administrator's Guide*

lsreplica

Lists VOB replicas

APPLICABILITY

Product	Command type
ClearCase	cleartool subcommand
Attache	command
MultiSite	multitool subcommand

Platform
UNIX
Windows

SYNOPSIS

lsrep·lica [**–l·ong** | **–s·hort** | **–fmt** *format*]
[**–sib·lings**
| [**–sib·lings**] **–invob** *vob-selector*
| *replica-selector* ...
]

DESCRIPTION

This command lists information about all VOB-replica objects recorded in the VOB database of the current replica (except for deleted replicas). Other replicas may exist, but the packets containing their creation information have not yet been imported at the current replica.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

LISTING FORMAT. *Default:* Includes creation event information for each replica.

–l·ong

Includes each replica's creation information, master replica, mastership request setting, ownership information, and host. If the current replica is in the process of restoration, this option annotates the listings of other replicas from which restoration updates are required. (See the **restorereplica** reference page.)

–s·hort

Lists only replica names.

–fmt *format*

Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this report-writing facility.

–sib·lings

Lists the VOB family members of the current replica, but does not list the current replica itself. This option is useful when you are writing scripts that process only sibling replicas.

SPECIFYING THE VOB FAMILY. *Default:* Lists VOB family members of the replica containing the current working directory.

–invob *vob-selector*

Lists the replicas of the specified VOB family. Specify *vob-selector* in the form [**vob:**]*pname-in-vob*

pname-in-vob	Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

SPECIFYING THE REPLICA. *Default:* Lists all known replicas of the VOB family.

replica-selector ...

Restricts the listing to one or more replicas. Specify *replica-selector* in the form [**replica:**]*replica-name*[@*vob-selector*]

replica-name	Name of the replica
vob-selector	VOB family of the replica; can be omitted if the current working directory is within the VOB. Specify *vob-selector* in the form [**vob:**]*pname-in-vob* *pname-in-vob* — Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

EXAMPLES

- List the names of all replicas of the VOB containing the current working directory.

  multitool lsreplica –short
  ```
  bangalore
  boston_hub
  buenosaires
  sanfran_hub
  ```

- List the names of all siblings of the VOB containing the current working directory.

  multitool lsreplica –short –siblings
  ```
  bangalore
  buenosaires
  sanfran_hub
  ```

- Display a long listing of the current VOB's replicas.

  multitool lsreplica –long
  ```
  replica "bangalore"
   15-Aug-00.15:48:39 by Susan Goechs (susan.user@minuteman)
    replica type: unfiltered
    master replica: boston_hub@/vobs/dev
    request for mastership:enabled
    owner: susan
    group: user
    host: "ramohalli"
  replica "boston_hub"
   19-May-99.15:47:13 by Susan Goechs (susan.user@minuteman)
    replica type: unfiltered
    master replica: boston_hub@/vobs/dev
    request for mastership:enabled
    owner: susan
    group: user
    host: "minuteman"
  replica "buenosaires"
  ```

```
 15-Aug-00.15:48:44 by Susan Goechs (susan.user@minuteman)
  replica type: unfiltered
  master replica: boston_hub@/vobs/dev
  request for mastership:enabled
  owner: susan
  group: user
  host: "mardelplata"
replica "sanfran_hub"
 19-May-99.15:49:46 by Susan Goechs (susan.user@minuteman)
  replica type: unfiltered
  master replica: sanfran_hub@/vobs/dev
  request for mastership:enabled
  owner: susan
  group: user
  host: "goldengate"
```

- List all replicas of the VOB whose VOB-tag is **\doc**.

  multitool lsreplica –invob \doc

  ```
  For VOB replica "\doc":
  11-Mar.13:42        jcole               replica "boston_hub"
  11-Mar.13:45        jcole               replica "sanfran_hub"
  11-Mar.13:48        jcole               replica "tokyo"
  ```

- List the name, master replica, and replica host of all replicas in the VOB family **/vobs/doc**.

  <u>*cmd-context*</u> **lsreplica –fmt **
  **"Name: %n\n\tMaster replica: %[master]p\n\tReplica host: %[replica_host]p\n" **
  -invob /vobs/doc

  ```
  Name: boston_hub
     Master replica: boston@/vobs/doc
     Replica host: minuteman
  Name: sanfran_hub
     Master replica: sanfran_hub@/vobs/doc
     Replica host: goldengate
  Name: tokyo
     Master replica: sanfran_hub@/vobs/doc
     Replica host: shinjuku
  ```

SEE ALSO

mkreplica

lssite

Lists site-wide default properties

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

lssite [**–inq·uire** | *setting-name*]

DESCRIPTION

The **lssite** command lists site-wide properties set in the ClearCase or ClearCase LT site config registry, and properties that are not currently set.

If you have not set any site-wide properties in the registry use **lssite –inquire** to list all available properties and their default values. To change the value of a property (that is, set the property in the registry), use the **setsite** command.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

LISTING PROPERTY NAMES. *Default:* **lssite** lists the site-wide properties that are set in the registry. The properties are displayed in the form *name=value*.

–inq·uire

Lists all available properties and their values. If a property is not set in the registry, **lssite** displays the default value. An asterisk before a property indicates that it is set in the registry.

NOTE: The default view cache size is different for 32-bit and 64-bit computers. Therefore, if the view cache size is not set in the registry, **lssite –inquire** output displays the default size for the computer on which you entered the command.

setting-name
Displays the property and its value. An asterisk before a property indicates that it is set in the registry.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List properties that have been set in the registry.

  cmd-context **lssite**
  ```
  view_cache_size=204800
  ```

- List all available properties.

  cmd-context **lssite –inquire**
  ```
  view_cache_size=204800
  view_shareable_dos=TRUE
  view_interop_text_mode=FALSE
  ```

SEE ALSO

getcache, **mkview**, **setcache**, **setsite**

lsstgloc

Lists view and VOB server storage locations.

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

- ClearCase:

  lsstgloc [**–vie·w** | **–vob**] [**–s·hort** | **–l·ong**] [**–reg·ion** *network-region*] [**–hos·t** *hostname*]
  [*stgloc-name* | **'***stgloc-name-pattern***'** ... | **–sto·rage** *stgloc-pname*]

- ClearCase LT:

  lsstgloc [**–vie·w** | **–vob**] [**–s·hort** | **–l·ong**]
  [**'***stgloc-name-pattern***'** ... | **–sto·rage** *stgloc-pname*]

DESCRIPTION

The **lsstgloc** command lists registry information about server storage locations for views and/or VOBs.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SPECIFYING THE KIND OF SERVER STORAGE LOCATION TO BE LISTED. *Default:* Both view and VOB server storage locations.

–vie·w
Lists server storage locations for views only.

–vob
Lists server storage locations for VOBs only.

SPECIFYING OUTPUT VERBOSITY. *Default:* Displays a one-line summary of the registry information.

–s·hort
Lists server storage location names only.

–l·ong
Lists the server storage location's name, region, UUID, global path (ClearCase), server host, and server host path.

SPECIFYING THE NETWORK REGION. *Default:* The local host's network region. (Use the **hostinfo –long** command to display the network region.) For a discussion of network regions, see the *Administrator's Guide*.

–reg·ion *network-region*
Lists server storage locations in the specified network region.

SPECIFYING THE HOST. *Default:* All server storage locations registered for the local network region.

–hos·t *hostname*
Lists only server storage locations residing at the specified host.

SPECIFYING THE SERVER STORAGE LOCATION. *Default for ClearCase:* All server storage locations in the local network region. *Default for ClearCase LT*: All server storage locations.

'*stgloc-name-pattern***'** ...
Lists server storage locations whose names match the specified patterns (see **wildcards_ccase**). Enclose each name pattern in quotes.

–sto·rage *stgloc-pname*
Lists the specified server storage location.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List the server storage locations.

  cmd-context **lsstgloc**

```
Views /net/saturn/ccstg_d/views
VOBs /net/saturn/ccstg_d/VOBs
test_BRAD /net/pluto/c/temp/test_BRAD
```

- List the server storage location named **test_BRAD**.

  cmd-context **lsstgloc test_BRAD**

```
test_BRAD  \\pluto\c\temp\test_BRAD
```

- List details of the server storage location at **/net/peroxide/export/home/bert/stgloc_view1**.

  cmd-context **lsstgloc -long -storage /net/peroxide/export/home/bert/stgloc_view1**

```
Name: stgloc_view1
Type: View
Region: atria_r_d_unix
Storage Location uuid: 3988ccaa.412d11d4.a313.00:01:80:7c:c6:73
Global path: /net/peroxide/export/home/bert/stgloc_view1
Server host: peroxide
Server host path: /export/home/bert/stgloc_view1
```

SEE ALSO

mkstgloc, **mkview**, **mkvob**, **rmstgloc**, *Administrator's Guide*

lsstream

Lists information about one or more streams

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand

Platform
UNIX
Windows

SYNOPSIS

lsstream [**–s·hort** | **–l·ong** | **–fmt** *format-string*
| **–tre·e** [**–fmt** *format-string*] [**–dep·th** *depth*]
| **–anc·estor** [**–fmt** *format-string*] [**–dep·th** *depth*]] [**–obs·olete**]
[**–inv·ob** *vob-selector* | **–in** *project-selector* | **–vie·w** *view-tag*
| **–cvi·ew** | *stream-selector* ...]

DESCRIPTION

The **lsstream** command displays information about one or more streams.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

SELECTING A DISPLAY FORMAT. *Default*: One-line summary.

–s·hort

Displays only the name of each stream.

–l·ong

Displays the following information for each stream:

- Name
- The associated project
- The default deliver stream
- Title (if different from the name)

- Child development streams
- Activities
- Foundation baselines
- Recommended baselines
- Attached views
- Policies

–fmt *format-string*

Displays information in the specified format. See the **fmt_ccase** reference page for details.

–tre·e [**–fmt** *format-string*] [**–dep·th** *depth*]

Displays information for a stream, including its hierarchy of streams and activities. You can modify how information is displayed with the **–fmt** and **–depth** options.

The **–fmt** option presents information using the specified format string. See the **fmt_ccase** reference page for more information.

The **–depth** option specifies the number of levels to be displayed. The *depth* argument must be a positive integer.

–anc·estor [**–fmt** *format-string*] [**–dep·th** *depth*]

Displays information for a stream, including its containing project and folders. You can modify how information is displayed with the **–fmt** and –**depth** options.

The –**fmt** option presents information using the specified format string. See the **fmt_ccase** reference page for more information.

The **–depth** option sets the number of levels displayed. The *depth* argument must be a positive integer.

LISTING OBSOLETE STREAMS. *Default*: Lists only nonobsolete streams.

–obs·olete

Includes obsolete streams in the listing. Obsolete streams are those that have been processed with **lock –obsolete**.

SPECIFYING THE STREAM. *Default:* **–cview**.

–inv·ob *vob-selector*

Displays a list of all streams in the specified project VOB.

–in *project-selector*

Displays a list of all streams for the specified project and highlights the integration stream.

–**vie·w** *view-tag*

Displays information for the stream connected to the specified view.

–cvi·ew

Displays information for the stream connected to the current view.

stream-selector ...

Displays information for specified stream or streams.

You can specify the stream as a simple name or as an object selector of the form [**stream**]**:***name@vob-selector*, where *vob-selector* specifies a project VOB (see the **cleartool** reference page). If you specify a simple name and the current directory is not a project VOB, then this command assumes the stream resides in the project VOB associated with the current view. If the current directory is a project VOB, then that project VOB is the context for identifying the stream.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, cmd-context represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, cmd-context represents the interactive **cleartool** prompt. In Attache, cmd-context represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- Display a one-line summary of the stream attached to the specified view.

  cmd-context **lsstream –view java_int**

  ```
  17-Sep-99.11:54:50  java_int  bill   "Deliver your changes here"
  ```

- For all streams in the project VOB, display a detailed listing for the current stream using a tree format. The asterisk (*) indicates **java_int** is the stream attached to the current view.

  `#` **cd /vobs/core_projects**

cmd-context **lsstream -tree**

```
*java_int                           stream     "Deliver your changes here"
   rebase.java_int.19990917.132524       activity   "rebase Deliver your
    changes here on 09/17/99 13:25:24."
   activity990917.133218        activity   "activity990917.133218"
   activity990917.133255        activity   "my new activity"
   new_activity                 activity   "new_activity"
   toms_edit                    activity   "toms_edit"
   activity990917.134751        activity   "activity990917.134751"
   deliver.java_dev.19990917.140443      activity   "deliver java_dev
    on 09/17/99 14:04:43."
   deliver.java_dev.19990917.141046      activity   "deliver java_dev
    on 09/17/99 14:10:46."

java_dev                            stream     "java_dev"
   activity990917.140331        activity   "activity990917.140331"
```

- Display a detailed listing for development stream **p7_sheila_devstr1**.

cmd-context **lsstream -long p7_sheila_devstr1**

```
stream “p7_sheila_devstr1”
10-Apr-01.17:43:33 by Sheila Hunt (sheila.usr@echo01)
owner: sheila
group: user
project: p7@/vobs/sheila_testvob
default deliver stream: int7@/vobs/sheila_testpvob
development streams:
   p7_kevin_devstr11@/vobs/sheila_testpvob
contains activities:
   dact14@/vobs/sheila_testpvob
foundation baselines:
   bld.bl3.161@/vobs/sheila_testpvob (c1@/vobs/sheila_testpvob)
    (modifiable)
   bld.bl2.181@/vobs/sheila_testpvob (c2@/vobs/sheila_testpvob)
    (modifiable)
recommended baselines:
   devstr.bl1.212
   bld.bl2.181
views:
   sheila_p7_devstr1
policies:
   POLICY_DELIVER_REQUIRE_REBASE disabled
   POLICY_DELIVER_NCO_DEVSTR disabled
   POLICY_INTRAPROJECT_DELIVER_FOUNDATION_CHANGES disabled
   POLICY_INTRAPROJECT_DELIVER_ALLOW_MISSING_TGTCOMPS disabled
   POLICY_INTERPROJECT_DELIVER disabled
   POLICY_INTERPROJECT_DELIVER_FOUNDATION_CHANGES disabled
```

```
POLICY_INTERPROJECT_DELIVER_REQUIRE_TGTCOMP_VISIBILITY disabled
POLICY_INTERPROJECT_DELIVER_ALLOW_NONMOD_TGTCOMPS disabled
```

SEE ALSO

chstream, **lock**, **mkstream**, **rmstream**

lstype

Lists a VOB's type objects

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase, ClearCase LT and Attache on UNIX—List type objects:

  lstype [**–local**] [**–l·ong** | **–s·hort** | **–fmt** *format-string*] [**–obs·olete**]
  { **–kin·d** *type-kind* [**–inv·ob** *vob-selector*]
  | *type-selector* ...
  }

- ClearCase and ClearCase LT on Windows only—List type objects graphically:

  lstype –g·raphical [**–kin·d** *type-kind*] [**–inv·ob** *vob-selector*]

- ClearCase, ClearCase LT and Attache on Windows only—List type objects in the command window:

  lstype [**–local**] [**–l·ong** | **–s·hort** | **–fmt** *format-string*] [**–obs·olete**]
  { **–kin·d** *type-kind* [**–inv·ob** *vob-selector*]
  | *type-selector* ...
  }

DESCRIPTION

The **lstype** command lists information about one or more of a VOB's type objects.

Obsolete Type Objects

Type objects can be rendered obsolete with the **lock –xxtype –obsolete** command. **lstype** lists an obsolete type object only if you either specify its name with a *type-name* argument or use the **–obsolete** option.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

LISTING TYPE OBJECTS GRAPHICALLY. *Default:* Lists type objects in the command window.

–g·raphical
Starts a type object browser to display type objects.

LISTING LOCAL COPIES OF GLOBAL TYPES. *Default:* In addition to types in the specified VOB, **lstype** lists all global types in associated administrative VOBs.

–local
Lists ordinary types and local copies of global types. For more information about global types, see *Administrator's Guide*.

LISTING FORMAT. *Default:* A type object listing looks like this:

```
07-Nov-1998     sakai     element type "text_file"
```

–l·ong
Expands the listing to include any type-specific parameters (for example, that a label type is one-per-element or that an element type inherited its type manager from the **text_file** supertype and so on.)

–s·hort
Restricts the listing to type names only.

–fmt *format-string*
Lists information using the specified format string. See the **fmt_ccase** reference page for details on using this report-writing facility.

LISTING OBSOLETE TYPES. *Default:* If you don't specify any *type-name* argument, only the nonobsolete types of the specified kind are listed.

–obs·olete
Includes obsolete type objects in the listing when you don't specify any individual type objects with *type-name* arguments. Has no effect if you specify one or more *type-name* arguments.

SPECIFYING THE KIND OF TYPE OBJECT. *Default:* None.

–kin·d *type-kind*

A kind of type object. All objects of this kind are listed. *type-kind* can be one of:

attype, **brtype**, **eltype**, **hltype**, **lbtype**, **trtype**

SPECIFYING THE VOB. *Default:* Lists type objects in the VOB that contains the current working directory.

–inv·ob *vob-selector*

The VOB whose type objects are to be listed. Specify *vob-selector* in the form [**vob:**]*pname-in-vob*

pname-in-vob	Pathname of the VOB-tag (whether or not the VOB is mounted) or of any file-system object within the VOB (if the VOB is mounted)

SPECIFYING INDIVIDUAL TYPE OBJECTS. *Default:* None.

type-selector ...

One or more names of type objects. The listing includes only the named objects. Specify type-selector in the form [*type-kind*:]*type-name*[@*vob-selector*]

type-kind	One of	
	attype	Attribute type
	brtype	Branch type
	eltype	Element type
	hltype	Hyperlink type
	lbtype	Label type
	trtype	Trigger type
type-name	Name of the type object.	
vob-selector	Object-selector for a VOB, in the same format as with **–invob**, above.	

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form **/vobs/***vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- List all branch types defined in the VOB containing the current working directory.

  cmd-context **lstype –kind brtype**

```
15-Dec.09:34   jenny       branch type "main"
 "Predefined branch type used to represent the main branch of elements."
08-Dec.12:12   jackson     branch type "test"
 "test development branch"
08-Dec.12:12   jackson     branch type "patch2"
08-Dec.12:12   jackson     branch type "patch3"
08-Dec.12:12   jackson     branch type "rel2_bugfix"
```

- List all label types defined in the current VOB. Use the short format, and include obsolete label types.

  cmd-context **lstype –kind lbtype –obsolete –short**

```
BACKSTOP
CHECKEDOUT
LATEST
REL1 (obsolete)
REL2
REL3
V2.7.1 (obsolete)
```

  Note that the listing includes the three predefined label types, **BACKSTOP**, **LATEST**, and **CHECKEDOUT**.

- List information about a particular user-defined element type, in long format.

  cmd-context **lstype –long eltype:c_source**

```
element type "c_source"
 08-Dec-98.12:12:38 by Chuck Jackson (test user) (jackson.dvt@oxygen)
 owner: jackson
 group: dvt
 scope: this VOB (ordinary type)
 type manager: text_file_delta (inherited from type "text_file")
 supertype: text_file
 meta-type of element: file element
```

- List information about a particular trigger type, in long format.

  cmd-context **lstype –long trtype:trig1**

```
trigger type "trig1"
 08-Dec-98.12:14:08 by jackson.dvt@oxygen
 owner: jackson
 group: dvt
 element trigger
 pre-operation MODIFY_ELEM
 action: -exec checkcmt
```

- List information about a particular hyperlink type.

  cmd-context **lstype –long hltype:design_spec**

```
hyperlink type "design_spec"
 08-Dec-98.12:13:31 by Chuck Jackson (test user) (jackson.dvt@oxygen)
 "source to design document"
 owner: jackson
 group: dvt
 scope: this VOB (ordinary type)
```

- List the name, lock status, master replica, and scope of all label types in the VOB **/vobs/stage**. (The command line, including the quoted format string, constitutes a single input line. The input line below is broken to improve readability. Spaces are significant.)

  cmd-context **lstype –fmt "%n\n\tLock status: %[locked]p\n\tMaster replica: %[master]p\n\tScope: %[type_scope]p\n" -kind lbtype**

```
V3.BL3
    Lock status: unlocked
    Master replica: lex
    Scope: ordinary
V3.BL4
    Lock status: locked
    Master replica: lex
    Scope: global
V4.0.DOC
    Lock status: unlocked
    Master replica: doc_clone
    Scope: ordinary
V4.0.HELP
    Lock status: unlocked
    Master replica: doc_clone
    Scope: ordinary
```

- List the name, kind, and creation comment of a particular trigger type.

  cmd-context **lstype -fmt "%n\t%[trigger_kind]p\n\t%c" trtype:cmnt**
  ```
  cmnt     element trigger
       prompt user for comment
  ```

SEE ALSO

describe, **rmtype**, **rename**, *Administrator's Guide*

lsview

Lists view registry entries

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and Attache:

lsview [**–s·hort** | **–l·ong**] [**–hos·t** *hostname*]
[**–pro·perties** [**–ful·l**] | **–age**] [**–reg·ion** *network-region*]
[**–cview** | *view-tag* ... | **–sto·rage** *view-storage-dir-pname* ... | **–uui·d** *view-uuid*]

- ClearCase LT:

lsview [**–s·hort** | **–l·ong**] [**–pro·perties** [**–ful·l**] | **–age**]
[**–cview** | *view-tag* ... | **–sto·rage** *view-storage-dir-pname* ... | **–uui·d** *view-uuid*]

DESCRIPTION

In ClearCase and Attache, the **lsview** command lists one or more views, including nonactive dynamic views. In ClearCase LT, **lsview** lists views registered at the ClearCase LT server host. To be accessible to **cleartool** subcommands and Attache commands, including **lsview**, a view requires these registry entries:

- One entry in the **view_object** registry file
- One or more entries in the **view_tag** registry file

These files, **view_object** and **view_tag**, constitute the view registry on the network's registry server host. For more information about registry files, see the *Administrator's Guide*.

ClearCase and Attache—Default Output

In ClearCase and Attache, the **lsview** command lists all views registered for the current network region by default, whether or not they are active. The default output line for each listed view shows:

- Whether the view is active on the host (* indicates the active dynamic view; active snapshot views are listed, but not annotated with a *)

  NOTE: **lsview** does not report unmounted local views as active even if they have running servers. Also, **lsview** does not report removed views as active.

- The view-tag
- The view-storage directory pathname

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

LISTING FORMAT. *Default for ClearCase and Attache:* See the *ClearCase and Attache—Default Output* section. *Default for ClearCase LT:* Similar to that of ClearCase and Attache, but no information about dynamic views is listed.

–s·hort
Restricts the listing to view-tags only.

–l·ong
Expands the listing to include all information stored in the view registry regarding the listed views.

ClearCase and Attache—The network accessibility information includes the global path to the view if a value is set for this property; otherwise, lists the host name and host-local path only.

ClearCase LT—The network accessibility information lists the host name and the host-local path.

–pro·perties [–ful·l]
Reports the following properties:

- When and by whom the view was created, last modified, and last accessed
- Permissions for the view owner, view group members, and others

With the **–full** option, reports the following additional properties:

- When and by whom view-private data was last accessed
- When and by whom a view-private object was last updated

- When and by whom the config spec was last updated
- For a dynamic view, when and by whom a derived object was last created, promoted, and winked in
- For a dynamic view, whether it creates shareable derived objects or nonshareable derived objects
- The view's text mode
- Whether the view is a dynamic view or a snapshot view
- Whether the view is read-only or writable

–age
Reports when and by whom the view was last accessed.

SPECIFYING THE VIEWS. *Default for ClearCase and Attache:* Views registered for the local network region. *Default for ClearCase LT:* All views registered at the ClearCase LT server host.

–hos·t *hostname*
Confines the listing to views whose storage directories reside on host *hostname*.

–reg·ion *network-region*
Confines the listing to the views registered for a particular network region. (The **mkview** and **mktag** commands have a **–region** option, which can be used to assign view-tags to specific network regions.) The *network-region* argument can include pattern-matching characters as described in **wildcards_ccase** (ClearCase) or **wildcards** (Attache). If the *network-region* argument includes pattern-matching characters, enclose it in single quotes.

–cvi·ew
Lists the current view. On UNIX systems, **lsview** lists the working directory view, if there is one; otherwise, it lists the set view.

view-tag ...
Specifies a single view to be listed. The view must be registered, but it need not be active to be listed with **lsview**. The *view-tag* argument can include pattern-matching characters as described in **wildcards_ccase** or **wildcards**. Enclose in single-quotes any *view-tag* argument that includes pattern-matching characters.

–sto·rage *view-storage-dir-pname* ...
One or more views, identified by full pathnames to their storage directories.

–uui·d *view-uuid*
A single view, specified by its UUID (universal unique identifier).

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List the views registered for the local network region.

  cmd-context **lsview**

```
*   mainRel5     /net/host5/usr/viewstore/ainRel5.vws
    anneRel5     /net/host5/usr/viewstore/anneRel5.vws
*   anneTest     /net/host2/usr/anne/viewstore/anneTest.vws
    nordTest     /net/host2/usr/nord/nordTest.vws
*   nordRel5     /net/host4/usr/viewstore/nordRel5.vws
    nordRel4      /net/host8/usr/viewstore/nordRel4.vws
```

- List, using the long display format, the registry information for the view with view-tag **mainRel5**.

  cmd-context **lsview –long mainRel5**

```
Tag: mainRel5
 Global path: \\pluto\viewshare\mainRel5.vws
 Server host: pluto
 Region: main_headqtrs
 Active: YES
 View tag uuid:a9c1ba4d.853e11cc.a96b.08:00:69:06:05:d8
View on host: pluto
View server access path: C:\views\mainRel5.vws
View uuid: a9c1ba4d.853e11cc.a96b.08:00:69:06:05:d8
```

- For a particular host, list the views whose view-tags match a wildcard pattern.

  cmd-context **lsview –host saturn '*anne*'**

```
*   anne_main     /net/saturn/usr/anne/views/anne_main.vws
    anne_rel2     /net/saturn/usr/anne/views/anne_rel2.vws
```

- List full view properties.

  cmd-context **lsview –properties –full anne_main**

```
  anne_main         \\saturn\disk1\views\anne_main.vws
Created 18-Jun-99.17:41:34 by anne.user@saturn
Last modified 22-Jul-99.15:42:16 by sue.user@pluto
Last accessed 22-Jul-99.15:42:16 by sue.user@pluto
Last read of private data 21-Jul-99.17:06:20 by anne.user@saturn
Last derived object promotion 11-Mar-99.12.31.20 by anne.user@saturn
Last config spec update 18-Jun-99.17:55:27 by anne.user@saturn
Last derived object winkin 21-Jul-99.17:05:49 by anne.user@saturn
Last derived object creation 21-Jul-99.17:06:18 by anne.user@saturn
Last view private object update 22-Jul-99.15:42:16 by sue.user@pluto
Text mode: msdos
Properties: dynamic readwrite shareable_dos
Owner: ACME\anne   : rwx (all)
Group: ACME\user   : rwx (all)
Other:             : r-x (read)
```

SEE ALSO

mktag, **mkview**, **register**, **unregister**, *Administrator's Guide*

lsvob

Lists VOB registry entries

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and Attache—List VOBs:

  lsvob [**–s·hort** | **–l·ong**] [**–hos·t** *hostname*] [**–reg·ion** *network-region*]
  [*vob-tag* ... | **–sto·rage** *vob-storage-dir-pname* ...
  | **–uui·d** *vob-uuid* | **–fam·ily** *vob-family-uuid*]

- ClearCase LT—List VOBs:

  lsvob [**–s·hort** | **–l·ong**] [*vob-tag* ... | **–sto·rage** *vob-storage-dir-pname* ... | **–uui·d** *vob-uuid*]

- ClearCase—List VOBs using the graphical VOB browser:

  lsvob –g·raphical [**–reg·ion** *network-region*]

- ClearCase LT on Windows—List VOBs using the graphical VOB browser:

  lsvob –g·raphical

DESCRIPTION

The **lsvob** command lists one or more VOBs. To be accessible to **cleartool** subcommands and Attache commands, including **lsvob**, a VOB must be registered. That is, it must have an entry in the **vob_object** file on the registry server host (ClearCase and Attache), or on the ClearCase LT server host (ClearCase LT). In addition, each VOB typically has one or more entries in the **vob_tag** registry file; you cannot mount, or even create, a VOB without assigning a tag to it. See the **mkvob** reference page. For more information about registry files, see the *Administrator's Guide*.

ClearCase and Attache—Default Output

In ClearCase and Attache, **lsvob** lists all VOBs registered for the current network region by default, whether or not they are mounted (active). The default output line for each listed VOB looks like this:

```
* /vobs/src /net/host2/usr/vobstore/src_vob public
```

The five output fields report:

- Whether the VOB is mounted (*)
- The VOB-tag
- The VOB storage directory pathname
- Whether the VOB is public or private (see the **mkvob** reference page)
- Whether the VOB is a UCM VOB or a project VOB

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

LISTING FORMAT. *Default for ClearCase and Attache:* See the *ClearCase and Attache—Default Output* section. *Default for ClearCase LT:* One-line summary.

–l·ong

Expands the listing to include all information stored in the VOB registry regarding the listed VOBs.

ClearCase and Attache—The network accessibility information includes the global path to the view if a value is set for this property; otherwise, lists the host name and host-local path only.

ClearCase LT—The network accessibility information lists the host name and the host-local path.

–s·hort

Restricts the listing to VOB-tags only.

LISTING THE VOBS GRAPHICALLY. *Default:* Lists the VOBs in the command window.

–g·raphical

Starts the VOB Browser to list the VOBs.

SPECIFYING THE VOBS. *Default for ClearCase and Attache:* Lists all VOBs registered for the local network region, both mounted and unmounted, public and private. *Default for ClearCase LT:* Lists all VOBs on the ClearCase LT server.

–hos·t *hostname*
Confines the listing to VOBs whose storage directories reside on host *hostname*.

–reg·ion *network-region*
Confines the VOB listing to include only the VOBs registered for one or more network regions. (The **mkvob** and **mktag** commands have a **–region** option, which can be used to assign VOB-tags to specific network regions.) Unless you use the **–graphical** option, the *network-region* argument can include pattern-matching characters as described in the **wildcards_ccase** reference page and the Attache **wildcards** reference page. Single-quote the *network-region* argument, if it includes pattern-matching characters.

vob-tag ...
Specifies one or more VOBs to be listed. A VOB must be registered, but it need not be mounted, to be listed with **lsvob**. The *vob-tag* argument can include pattern-matching characters as described in **wildcards_ccase** or **wildcards**. Enclose in single quotes any *vob-tag* argument that includes pattern-matching characters.

–sto·rage *vob-storage-dir-pname* ...
One or more VOBs, identified by full pathnames to their storage directories.

–uui·d *vob-uuid*
Lists the VOB with the specified universal unique identifier (UUID).

–fam·ily *vob-family-uuid*
Lists the VOB with the specified VOB family UUID.

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

NOTE: In the UNIX examples that follow, arguments and output that show multicomponent VOB tags are not applicable to ClearCase LT, which recognizes only single-component VOB tags. In this manual, a multicomponent VOB tag is by convention a two-component VOB tag of the form

/vobs/*vob-tag-leaf*—for example, **/vobs/src**. A single-component VOB tag consists of a leaf only—for example, **/src**. In all other respects, the examples are valid for ClearCase LT.

- List the VOBs registered for the local network region.

  cmd-context **lsvob**

```
* /vobs/demo    /net/host5/usr/vobstore/demo_vob public
* /vobs/src     /net/host2/usr/vobstore/src_vob public
* /vobs/design  /net/host2/usr/vobstore/publicdesign_vob public
  /vobs/pvob    /net/host7/usr/vobstore/p_vob private (pvob)
  /vobs/doc     /net/host2/usr/vobstore/doc_vob public
* /vobs/stage   /net/host4/usr/vobstore/stage_vob public
* /vobs/ucmvob  /net/host6/usr/vobstore/ucm_vob private (ucmvob)
  /vobs/bugvob   /net/host8/usr/anne/vobstore/bug_vob private
```

```
*   \demo     \\neptune\vbstore\demo_vob public
*   \src1     \\sunfield\users\vbstore\src1_vob public
    \ucmvob   \\pluto\users\vbstore\ucm_vob private (ucmvob)
*   \design   \\luna\vbstore\design_vob public
    \doc      \\sunfield\users\vbstore\doc_vob public
*   \stage    \\pluto\users\vbstore\stage_vob public
*   \pvob     \\sunfield\users\vbstore\p_vob private (pvob)
    \bugvob    \\luna\users\anne\vbstore\bug_vob private
```

- List, using the long display format, the registry information for the VOB with VOB tag **\vob12**. The output line `Active: YES` indicates that the VOB is currently mounted.

  cmd-context **lsvob –long \vob12**

```
Tag: \vob12
 Global path: \\sol\vbstore\vob12.vbs
 Server host: sol
 Access: public
 Region: us_west
 Active: YES
 Vob tag replica uuid:cb4caf2f.f48d11cc.abfc.00:01:53:00:e8:c3
Vob on host: sol
Vob server access path: c:\vbstore\vob12.vbs
Vob family uuid: aed00001.9d3e11ca.bc4c.00:01:53:00:e8:c3
Vob replica uuid: cb4caf2f.f48d11cc.abfc.00:01:53:00:e8:c3
```

- List the VOB with a particular VOB family UUID.

  cmd-context **lsvob –family 12a3456b.78c901d2.e3ab.45:67:89:c0:1d:e2**

```
* /vobs/dev /net/minuteman/vobstg/dev.vbs public
```

- For a particular host, list the VOBs whose VOB-tags match a wildcard pattern.

  cmd-context **lsvob –host host4 '*anne*'**
  ```
  * /usr/anne/vobs/test2 /net/host4/usr/anne/vobs/test2.vbs public
  * /usr/anne/vobs/work /net/host4/usr/anne/vobs/work.vbs private
  ```

- Use a VOB browser to list all VOBs in the rd_east region.

  cmd-context **lsvob –graphical –region rd_east**

SEE ALSO

mktag, **mkvob**, **mount**, **register**, **umount**, **unregister**, *Administrator's Guide*

lsvtree

Lists version tree of an element

APPLICABILITY

Product	Command Type
ClearCase	cleartool subcommand
ClearCase LT	cleartool subcommand
Attache	command

Platform
UNIX
Windows

SYNOPSIS

- ClearCase and ClearCase LT on UNIX only—Display the version tree in graphical form:

  lsvtree –g·raphical [**–a·ll**] [**–nme·rge**] [**–nco**]
  [**–opt·ions** *pass-through-opts*] *pname* ...

- Windows only—Display the version tree in graphical form:

  lsvtree –g·raphical [**–a·ll**] [**–nme·rge**] [**–nco**] *pname* ...

- List the version tree in the command window:

  lsvtree [**–nr·ecurse**] [**–s·hort**] [**–a·ll**] [**–mer·ge**] [**–nco**] [**–obs·olete**]
  [**–bra·nch** *branch-pname*] *pname* ...

DESCRIPTION

The **lsvtree** command lists part or all of the version tree of one or more elements. By default, the listing includes all branches of an element's version tree except for obsolete branches. The listing excludes certain versions on the included branches. Command options control which branches, how many branches, and which versions are listed. You can also control the way versions are annotated with version labels and merge arrows.

RESTRICTIONS

None.

OPTIONS AND ARGUMENTS

DISPLAYING THE VERSION TREE GRAPHICALLY. *Default:* Lists the version tree in nongraphical form.

–g·raphical

Starts a Version Tree Browser (in Attache, a Version Tree Browser window) for each element you specify as an argument.

LISTING SUBBRANCHES. *Default:* Lists the entire subtree of the branch selected as the starting point.

–nr·ecurse

Omits all subbranches from the listing, showing only versions on a single branch.

SELECTING AND ANNOTATING VERSIONS ON A BRANCH. *Default:* For each branch included in the listing, these selected versions are listed:

- Checked-out versions (annotated with the view name) and their predecessors
- Versions that are the **LATEST** on their branches
- Versions with labels
- Versions at which a subbranch was created
- Versions that are hyperlink endpoints.

A version is annotated with up to five of its version labels; an ellipsis (...) indicates that the version has additional labels.

–s·hort

Restricts the listing to version-extended pathnames. Version labels, merge annotations, and checkout annotations are omitted.

–a·ll

Lists all versions on a branch, not the selected versions only; annotates each version with all of its version labels.

–mer·ge

Includes all versions that are at the from-end of one or more merge arrows (hyperlinks of type **Merge**). Annotations on each such version indicate the corresponding to-objects.

–nme·rge

Excludes versions that have merge arrows.

–nco

Excludes checked-out versions from the listing or display. The predecessor of a checked-out version is also excluded, unless there is another reason to include it (for example, it has a version label).

LISTING OBSOLETE BRANCHES. *Default:* Obsolete branches (locked with the obsolete option) and instances of obsolete branch types are not listed.

–obs·olete
Lists obsolete branches and instances of obsolete branch types.

GRAPHICAL OPTIONS. *Default:* None.

–opt·ions *pass-through-options*
Specifies one or more **xclearcase** command options that are not directly supported on the **lsvtree** command line. In particular, **xclearcase** accepts all the standard X Toolkit command-line options (for example, **–display**), as described in the **X(1)** reference page. If the option string includes white space, enclose it with quotes.

SELECTING THE STARTING POINT. *Default:* Starts the version tree listing at an element's **main** branch.

–bra·nch *branch-pname*
Starts the version tree listing at the specified branch. You can also use an extended name as the *pname* argument (for example, **foo.c@@\main\bug405**) to start the listing at a particular branch.

SPECIFYING THE ELEMENTS OR BRANCHES. *Default:* None. You must specify at least one element.

pname ...
One or more pathnames, specifying elements or branches of elements. (Alternatively, use the **–branch** option to specify a branch of an element.)

EXAMPLES

The UNIX examples in this section are written for use in **csh**. If you use another shell, you may need to use different quoting and escaping conventions.

The Windows examples that include wildcards or quoting are written for use in **cleartool** interactive mode. If you use **cleartool** single-command mode, you may need to change the wildcards and quoting to make your command interpreter process the command appropriately.

In **cleartool** single-command mode, *cmd-context* represents the UNIX shell or Windows command interpreter prompt, followed by the **cleartool** command. In **cleartool** interactive mode, *cmd-context* represents the interactive **cleartool** prompt. In Attache, *cmd-context* represents the workspace prompt.

- List selected versions from an element's version tree.

  cmd-context **lsvtree util.c**
  ```
  util.c@@/main
  util.c@@/main/1 (REL2)
  util.c@@/main/rel2_bugfix
  util.c@@/main/rel2_bugfix/1
  util.c@@/main/3 (REL3)
  util.c@@/main/4
  ```

- List all versions and all obsolete branches in an element's version tree.

  cmd-context **lsvtree –all –obsolete util.c**
  ```
  util.c@@\main
  util.c@@\main\0
  util.c@@\main\1 (REL2)
  util.c@@\main\rel2_bugfix
  util.c@@\main\rel2_bugfix\0
  util.c@@\main\rel2_bugfix\1
  util.c@@\main\2
  util.c@@\main\3 (REL3)
  util.c@@\main\rel3_patch
  util.c@@\main\rel3_patch\0
  util.c@@\main\rel3_patch\1
  util.c@@\main\4
  ```

- List all versions on the **rel2_bugfix** branch of an element's version tree.

  cmd-context **lsvtree -branch /main/rel2_bugfix -all util.c**
  ```
  util.c@@/main/rel2_bugfix
  util.c@@/main/rel2_bugfix/0
  util.c@@/main/rel2_bugfix/1
  ```

- Start a version tree browser to display all versions in an element's version tree.

  cmd-context **lsvtree –graphical –all util.h**

SEE ALSO

describe, **ls**, **lshistory**

lsws

Lists local workspace registry entries

APPLICABILITY

Product	Command Type
Attache	command

Platform
UNIX
Windows

SYNOPSIS

lsws

DESCRIPTION

The **lsws** command lists all workspaces registered on the local machine, showing their workspace storage directories and workspace helper hosts.

The output line for each listed workspace looks like this:

```
jo_main          c:\users\jo\jo_main          agora
```

The three output fields report:

- The workspace name (view tag)
- The workspace-storage directory pathname
- The ClearCase host serving as the workspace helper host (running **ws_helper**)

RESTRICTIONS

None.

EXAMPLES

- List all of your workspaces.

  cmd-context **lsws**

```
Workspace name      Local storage directory    Server host
  jed_ws            C:\users\jo\jed_ws         agora
  jo_main           C:\users\jo\jo_main        agora
```

SEE ALSO

attache_command_line_interface, mkws, rmws, setws