Rational Software Corporation®

RATIONAL® CLEARCASE® LT

ADMINISTRATOR'S GUIDE

UNIX/WINDOWS EDITION

VERSION: 2002.05.00 AND LATER

PART NUMBER: 800-025075-000

Rational®
the software development company

Administrator's Guide
Document Number 800-025075-000 October 2001
Rational Software Corporation 20 Maguire Road Lexington, Massachusetts 02421

IMPORTANT NOTICE

Copyright

Copyright © 1992, 2001 Rational Software Corporation. All rights reserved.
Copyright 1989, 1991 The Regents of the University of California
Copyright 1984–1991 by Raima Corporation

Permitted Usage

THIS DOCUMENT IS PROTECTED BY COPYRIGHT AND CONTAINS INFORMATION PROPRIETARY TO RATIONAL. ANY COPYING, ADAPTATION, DISTRIBUTION, OR PUBLIC DISPLAY OF THIS DOCUMENT WITHOUT THE EXPRESS WRITTEN CONSENT OF RATIONAL IS STRICTLY PROHIBITED. THE RECEIPT OR POSSESSION OF THIS DOCUMENT DOES NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISTRIBUTE ITS CONTENTS, OR TO MANUFACTURE, USE, OR SELL ANYTHING THAT IT MAY DESCRIBE, IN WHOLE OR IN PART, WITHOUT THE SPECIFIC WRITTEN CONSENT OF RATIONAL.

Trademarks

Rational, Rational Software Corporation, the Rational logo, Rational the e-development company, Rational Suite ContentStudio, ClearCase, ClearCase MultiSite ClearQuest, Object Testing, Object-Oriented Recording, Objectory, PerformanceStudio, PureCoverage, PureDDTS, PureLink, Purify, Purify'd, Quantify, Rational Apex, Rational CRC, Rational PerformanceArchitect, Rational Rose, Rational Suite, Rational Summit, Rational Unified Process, Rational Visual Test, Requisite, RequisitePro, RUP, SiteCheck, SoDA, TestFactory, TestMate, TestStudio, The Rational Watch, among others are trademarks or registered trademarks of Rational Software Corporation in the United States and in other countries. All other names are used for identification purposes only, and are trademarks or registered trademarks of their respective companies.

Sun, Solaris, and Java are trademarks or registered trademarks of Sun Microsystems, Inc.

Microsoft, the Microsoft logo, the Microsoft Internet Explorer logo, Windows, the Windows logo, Windows NT, the Windows Start logo are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.

FLEXlm and GLOBEtrotter are trademarks or registered trademarks of GLOBEtrotter Software, Inc. Licensee shall not incorporate any Globetrotter software (FLEXlm libraries and utilities) into any product or application the primary purpose of which is software license management.

Patent

U.S. Patent Nos. 5,193,180 and 5,335,344 and 5,535,329 and 5,574,898 and 5,649,200 and 5,675,802 and 5,835,701. Additional patents pending.

Government Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Rational License Agreement and in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) 1995, FAR 52.227-19, or FAR 52.227-14, as applicable.

Warranty Disclaimer

This document and its associated software may be used as stated in the underlying license agreement. Rational Software Corporation expressly disclaims all other warranties, express or implied, with respect to the media and software product and its documentation, including without limitation, the warranties of merchantability or fitness for a particular purpose or arising from a course of dealing, usage, or trade practice.

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

Administering the ClearCase Network

Administering VOBs

Administering Views

Administering Scheduled Jobs

Administering Web Servers

Tuning for Performance

Troubleshooting

Figures

Tables

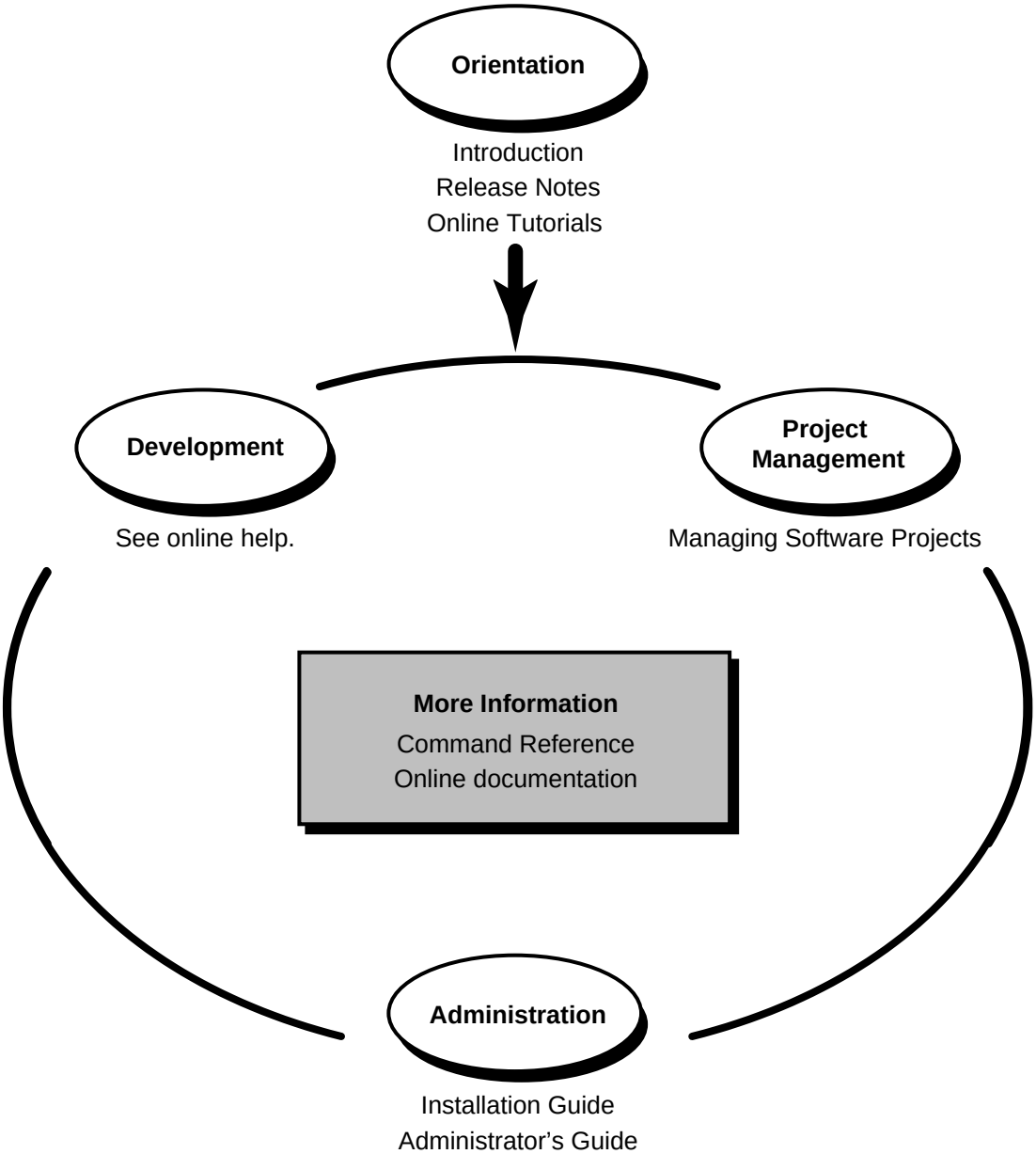
Preface

About This Manual

This manual is for ClearCase LT users or administrators who are responsible for tasks such as creating and maintaining data repositories, managing servers, and administering user and group accounts. *Administrator's Guide LT* discusses these subjects in depth:

- Managing the ClearCase LT server, clients, and user and group accounts
- Administering a mixed network of UNIX and Windows computers using ClearCase LT
- Creating ClearCase VOBs, or data repositories, and managing their storage requirements
- Creating ClearCase views, or user workspaces, and managing their storage requirements
- Administering ClearCase LT licenses
- Managing the ClearCase scheduler, which runs jobs periodically
- Setting up Web servers for the ClearCase Web interface and integrations with Web authoring tools
- Tuning the ClearCase LT server for better performance
- Troubleshooting problems with ClearCase LT

ClearCase LT Documentation Roadmap



Online Documentation

The ClearCase graphical interface includes an online help system.

There are three basic 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 ClearCase 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.

ClearCase also provides access to full "reference pages" (detailed descriptions of ClearCase commands, utilities, and data structures) with the **cleartool man** subcommand. Without any argument, **cleartool man** displays the **cleartool** overview reference page. Specifying a command name as an argument gives information about using the specified command. For example:

cleartool man *(display the cleartool overview page)*

cleartool man man *(display the cleartool man reference page)*

cleartool man checkout *(display the cleartool checkout reference page)*

ClearCase's **–help** command option or **help** command displays individual subcommand syntax. Without any argument, **cleartool help** displays the syntax for all **cleartool** commands. **help checkout** and **checkout –help** are equivalent.

cleartool uncheckout –help

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

Additionally, the online *ClearCase Tutorial* provides a step-by-step tour through ClearCase's most important features. To start the tutorial:

- On Windows, choose **Tutorial** in the appropriate ClearCase folder off the **Start** menu.
- On UNIX, type **hyperhelp cc_tut.hlp**.

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**

Administering the ClearCase Network

1 Understanding the ClearCase LT Network

This chapter presents a system administrator's overview of a local area network of computers running Rational ClearCase LT. It also serves as a roadmap to other chapters in this manual and to detailed reference information in the *Command Reference*.

This manual is intended for use by experienced UNIX or Windows administrators who also have administrative responsibility for a ClearCase LT network. Many of the procedures we describe in this manual assume that the reader is familiar with the scripting and command-line conventions, file system access controls, networking protocols, and system administration tools for UNIX and Windows computers.

In this manual, the term *Windows* refers to any of these operating systems:

- Windows XP Professional
- Windows 2000
- Windows NT

References to Windows Me or Windows 98 refer only to the specific operating system. References to UNIX refer to all UNIX and Linux platforms supported by Rational ClearCase LT.

1.1 Network Overview

A ClearCase administrator has three principal concerns:

- **ClearCase hosts.** ClearCase LT can be installed and used on any number of hosts in a network. ClearCase LT includes two types of hosts: ClearCase LT *clients* that provide

desktop platforms for software developers, and a ClearCase LT *server* that manages client access to critical ClearCase data. ClearCase LT supports any combination of UNIX and Windows clients accessing a UNIX or Windows server. Any UNIX or Windows computer can be a ClearCase LT server or client. Windows Me and Windows 98 computers can only be ClearCase LT clients.

- **ClearCase data storage.** ClearCase data is stored in *versioned object bases* (*VOBs*) and *views*, which are managed by the ClearCase LT server.

  For many organizations, the set of all VOBs constitutes a central data repository, which you may need to administer as a unit. Most views are created and used by individuals; however, it is likely that one or more shared views will be created, requiring some central administration.

- **ClearCase users.** ClearCase LT users are identified by the user name and group memberships established when they log on to either UNIX or Windows. Any number of people can use ClearCase LT on any number of hosts; the floating license scheme limits the number of concurrent users, but not the number of hosts. ClearCase LT node-locked licenses (Windows only) limit use of ClearCase features to hosts that are licensed.

1.2 ClearCase Hosts

ClearCase LT is a distributed application with a client/server architecture. Any development task (for example, execution of a single ClearCase command) may involve programs and data on both the ClearCase LT client and the server.

- **Client host.** Each ClearCase LT user works at a ClearCase LT client host, running programs such as **cleartool**, **clearmake**, and various ClearCase LT graphical user interfaces (GUIs), as well as other software (for example, development tools, a Rational Suite, and operating system utilities). ClearCase LT must be installed on each client host.

- **ClearCase LT server**. Every ClearCase LT client requires the services of a ClearCase LT *server host* to manage all of the client's access to VOB and view data. The ClearCase LT server runs server processes and supplies storage areas for VOB and view storage. A ClearCase LT client host can access only one ClearCase LT server. A ClearCase LT server host can support multiple VOBs and views, and can service the needs of many ClearCase LT clients.

- **Rational license server host**. One or more hosts in the network must be configured as *license server hosts*. ClearCase LT client hosts require a license to access data on the ClearCase LT server, and to perform most operations. The license server host may also be

the license server for other Rational products. Because all Rational products use the same licensing mechanism, we recommend that you simplify license administration by using a single host to serve licenses for all Rational products in use at your site. See Chapter 5, *Administering Licenses*, for more information on this topic.

- **Networkwide release host.** One host in the network acts as the networkwide *release host*. A directory on this host stores an entire ClearCase LT release that has been extracted from the distribution CD. When necessary, ClearCase patches can be applied to a release area to update the entire release with the latest enhancements and defect fixes. Certain installation options allow ClearCase hosts running UNIX to access ClearCase programs and data through symbolic links to the release area instead of having the programs and data installed on local storage. ClearCase hosts running Windows do not access the networkwide release host after installation is complete. The networkwide release host does not need to run ClearCase.

  NOTE: On Windows computers, ClearCase LT does not use a networkwide release host.

- **ClearCase Web server host.** If you want to use the ClearCase Web interface, you'll need at least one ClearCase Web server host. See Chapter 17, *Configuring a Web Server for the ClearCase Web Interface*, for more detail.

1.3 ClearCase Data Storage

All ClearCase data is stored in VOBs and views on the Clearcase LT server. VOB and view maintenance is a critical aspect of ClearCase administration.

Versioned Object Bases (VOBs)

The ClearCase network's permanent data repository consists of one or (usually) more VOBs located on the Clearcase LT server. Each VOB occupies a *VOB storage directory*, which holds file system objects and an embedded database.

VOB administration responsibilities include the following:

- **Registration and tagging.** Access information for all VOBs is kept in a networkwide registry on the ClearCase LT server. In a typical network, registry maintenance is minimal; ClearCase commands that create VOBs and VOB-tags update the registry automatically, though you may need to change a tag if you move a VOB.

- **Backup.** VOBs have special backup and recovery requirements, and must be backed up frequently and reliably. ClearCase does not include data-backup tools. You will have to select appropriate operating system backup and archive utilities or third-party backup tools for this critical task. VOB backup and restore procedures are described in Chapter 9, *Backing Up and Restoring VOBs*.

- **Periodic maintenance.** VOB administration requires that you balance the need to preserve important data with the need to conserve disk space. ClearCase LT includes tools for collecting data on disk space used and for occasional *scrubbing* of unneeded data. You can specify what data is unneeded on a per-VOB basis.

  ClearCase LT includes a job scheduler that manages periodic execution of various administrative tasks, including disk-space data collection and VOB scrubbing. The job scheduler is installed with a predefined schedule of these maintenance operations, which you can change as needed. For more information on creating and managing scheduled jobs, see Chapter 16, *Managing Scheduled Jobs*.

- **Access control.** Each VOB has an *owner*, a *primary group*, a supplemental *group list*, and a *protection mode*. Together, they control access to VOB data. Understanding and managing VOB access controls is an important task for the ClearCase administrator. For more information, see Chapter 3, *Understanding ClearCase Access Controls*.

- **Growth.** As new projects begin or existing projects are placed under ClearCase control, you may need to create new VOBs and incorporate them into your data backup and periodic maintenance schedule.

- **Reformatting**. Occasionally, a major new ClearCase LT release may include a feature that requires reformatting of your existing VOBs. This process updates the *schema* of the embedded VOB database.

Views

ClearCase *views* provide

- Access to VOB data

- Short-Term storage for other data created during the development process

A view stores versions of file elements that have been loaded from one or more VOBs, as well as *view-private files* which have no counterpart in the VOB (for example, text editor backup files).

ClearCase supports two types of views:

- Snapshot views, which contain copies of versions of specified elements, along with view-private objects. The view never updates itself with new versions created from other views. Instead, the **update** command reevaluates the view's config spec and loads the newly selected versions into the view.

- Dynamic views, which provide transparent access to versions of elements in the VOB and to view-private objects. Each time you access an element through a dynamic view, the view's **view_server** process evaluates the view's config spec and selects a particular version of the element. Thus, such a view updates itself with new versions created in other views.

  NOTE: ClearCase LT does not support dynamic views.

Unlike VOBs, which are long-lived artifacts created by an administrator, views tend to be shorter lived and are usually created by individual developers. View administration is simpler than VOB administration, involving little more than occasional backups, periodic maintenance, and attention to access control issues.

1.4 ClearCase Users

ClearCase LT does not maintain a database of its users. Any user who is logged in to a ClearCase LT host and can acquire a license is able to use the software. Because ClearCase LT relies on a host's operating system to establish a user's identity (user ID, principal group ID, and optional supplementary group IDs), you must make certain that user identities are consistent on every ClearCase LT host. This consistency is usually achieved by using networkwide databases maintained by the operating system such as the NIS **passwd** and **group** maps on UNIX or, on Windows, Windows NT or Active Directory domains.

NOTE: In environments where users access a common set of VOBs and views from UNIX and Windows hosts, we strongly recommend that each user's user name, password, and group memberships be the same whether the user is logged in to UNIX or Windows.

Most **cleartool** commands check the user's identity before granting access to particular objects—element, version, and so on. See Chapter 3, *Understanding ClearCase Access Controls*, for detailed information on this topic.

1.5 The ClearCase Registry

ClearCase maintains a networkwide registry of VOBs and views so that users and programs can access these objects without having to know the details of where they are stored.

The ClearCase registry database is maintained on the ClearCase LT server.

Objects and Tags

The ClearCase registry contains two types of information for every VOB and every view:

- A single object entry that describes where the VOB or view is stored on the network (a host name and storage directory, for example) and includes other information needed by ClearCase.
- One or more tag entries that define the name by which the VOB or view is referenced by other programs and provide a global path, expressed using network file naming conventions, to the VOB or view storage directory.

Server Storage Locations

In addition to VOB and view tag and object entries, the ClearCase registry records the names and network paths for server storage locations that can be created to provide a standard or default storage location for newly created VOBs and views.

1.6 ClearCase Server Processes

This section provides brief descriptions of server processes that run on the Clearcase LT server. No ClearCase server processes run on a ClearCase LT client.

albd_server

The Atria location broker daemon or **albd_server** handles a variety of tasks:

- Manages the operation of all ClearCase services on the ClearCase LT server.
- Helps set up network communications between ClearCase LT clients and the ClearCase LT server.
- Manages execution of tasks run by the ClearCase **schedule** service.

When you start ClearCase (using the ClearCase control panel on Windows or the **atria_start** script on UNIX), the **albd_server** starts first. It then starts other servers as needed.

When you stop ClearCase, the **albd_server** stops all running ClearCase servers and then exits.

In addition to starting and stopping services, the **albd_server** helps client programs like **cleartool** and ClearCase GUIs use remote procedure calls (RPCs) to connect with the ClearCase LT server. When a client program wants to access the ClearCase LT server, it uses an RPC to send a request to the **albd_server** process on that host. The **albd_server** starts the requested service if it is not already started, then issues a response telling the client the service's port number (socket address). Thereafter, the client communicates directly with the specific service, without involving the **albd_server**.

The **albd_server** reads configuration file **albd.conf** during startup to determine which services to provide. Do not modify this file.

admin_server

The ClearCase administration server **admin_server** is invoked as needed by the **albd_server** process. This short-lived server performs miscellaneous administrative support functions:

- Retrieving server log files for display by the **getlog** command and the ClearCase Administration Console.
- Retrieving and changing the ClearCase LT server's ClearCase properties when requested by the ClearCase Administration Console.

view_server

A **view_server** is a long-lived process that manages activity in a particular view.

For each view, a long-lived **view_server** process runs on the ClearCase LT server. The **view_server** is started by the host's **albd_server** process when necessary. On UNIX systems, it

runs with the identity of the owner of the view storage directory (usually, the user who created the view). On Windows systems, it runs with the identity of the **albd_server** (NT AUTHORITY\SYSTEM) A **view_server** remains active until it is terminated by a **cleartool endview –server** command, a system shutdown, or an operating system command that terminates the **view_server** process.

When it begins execution, a **view_server** reads configuration information from the **.view** file in the view-storage directory. Values in this file are established by **mkview**, **chview**, and similar commands. Do not edit this file yourself.

vob_server

For each VOB, a long-lived **vob_server** process runs on the ClearCase LT server. The **vob_server** runs with the identity of the VOB owner and manipulates data in the VOB's storage pools in response to requests from client processes.

The **vob_server** is the only process that ever creates or deletes data containers; the VOB owner is the only user who can modify data containers and storage pools. These severe restrictions protect VOB data against careless or malicious users.

A **vob_server** process is started as needed by **albd_server**. It remains active until any of the following events occurs:

- The operating system is restarted.
- The VOB is deleted with the **rmvob** command.
- ClearCase is stopped (UNIX only, see the **init_ccase** reference page).

When it begins execution, the **vob_server** reads configuration information from the file **vob_server.conf** in the VOB storage directory. Values in this file are established by the **vob_snapshot_setup** utility and similar commands. Do not edit this file yourself.

db_server

A host's **db_server** processes handle VOB database transactions on that host in response to requests from client programs. Because client programs cannot access VOB databases directly, they must send database transaction requests to a **db_server** process. Typical requests include:

- Creating and modifying metadata (such as attaching a label to a version)
- Reading metadata (such as finding the labels attached to a version)

- Writing event records (such as the one that records a **checkout** command)
- Writing configuration records
- Reading event records and configuration records

Each **db_server** process services a single client at a time, but can operate on any number of VOBs. A client establishes a connection to a **db_server** with the help of the **albd_server** on the VOB host. If necessary, the **albd_server** starts a new **db_server** process to handle a request. The connection is broken when the client exits or becomes idle (stops requesting database transactions for an extended period). At that point, the **db_server** becomes available for use by another client; eventually, an unconnected **db_server** is terminated by **albd_server**.

vobrpc_server

The ClearCase LT server runs up to five **vobrpc_server** processes for each of its VOBs. Each process handles requests from **view_server** processes throughout the network. The request can generate both metadata (VOB database) and file system data (storage pool) activity. The **vobrpc_server** accesses the VOB database in exactly the same way as a **db_server**. It forwards storage pool access requests to the **vob_server**.

Multiple server processes are started by **albd_server**, which also routes new requests to the least-busy servers and terminates unneeded **vobrpc_server** processes when the system is lightly loaded.

lockmgr

The ClearCase LT server runs one database lock manager process, **lockmgr**. This process arbitrates transaction requests to all VOB databases on that host from ClearCase LT client programs throughout the network. The calling program polls **lockmgr**, which either grants or prohibits access to the requested data. If the data is available, the transaction proceeds immediately: the data is read or written, and output is returned to the calling program. If the data is unavailable (locked because another caller has been granted write access to the data), the caller waits until **lockmgr** grants it access to the data.

Unlike most other ClearCase services, the **lockmgr** is not started by the **albd_server**. Instead, it is started when the ClearCase LT server starts. On UNIX systems, the **lockmgr** is started by the **atria_start** script. On Windows, the **lockmgr** is started by the Service Control Manager.

NOTE: On UNIX systems, the **lockmgr** communicates with other processes through **var/adm/atria/almd**, which is a shared-memory file on some platforms and a socket on others. To reduce the likelihood of accidental deletion, **/var/adm/atria/almd** is owned by **root**.

Lock manager startup options can be changed if necessary to improve VOB server performance for certain configurations. See *Lock Manager Startup Options* on page 286 for more information on this topic.

Server Error Logs

Each ClearCase server process maintains an error log on the ClearCase LT server. For details on the error logs and how to display them, see the **getlog** reference page in the *Command Reference*. In addition, Windows hosts can access error logs for all ClearCase hosts (Windows and UNIX) using the ClearCase Administration Console.

Error Logging on UNIX

On ClearCase LT servers running UNIX, log files are located in the directory **/var/adm/atria/log**. Log files record error and status information from various server programs and user programs. These files include the following:

albd_log	Used by the **albd_server**
db_server_log	Used by the **db_server**
error_log	General-purpose error log. Used by user programs such as **cleartool**
event_scrubber_log	Used by the **event_scrubber** program
install_log	Used by **install_release** (installation script)
lockmgr_log	Used by the **lockmgr** program
scrubber_log	Used by the **scrubber** program
view_log	Used by the **view_server**
vob_log	Used by the **vob_server**
vob_scrubber_log	Used by **vob_scrubber** program
vobrpc_server_log	Used by **vobrpc_server**

Error log files are ordinary text files. A typical entry includes the date and time of the error, the software module in which the error occurred, the current user, and an error-specific message.

As errors accumulate, the error log files grow. By default, the scheduler periodically runs a job that renames error log files to *logfile_name*`.old`, and creates empty template files in their place. See the **schedule** reference page for information on describing and changing scheduled jobs.

Error Logging on Windows

On ClearCase LT servers running Windows, ClearCase server processes and other ClearCase programs write informational, warning, and error messages to the Windows application event

log. The source of these messages is displayed as **ClearCase LT**. A typical event log entry includes the date and time of the error, the software module in which the error occurred, the current user, and an error-specific message.

As errors accumulate, the error log files grow. Use the Event Viewer to save or delete logs.

1.7 ClearCase Client/Server Processing

Because ClearCase is a distributed client/server application, multiple processes, often running on multiple hosts, can play a role in the execution of ClearCase commands. For example, this is what happens when a user checks out a file element:

1. The user's *client* process—**cleartool**, for example, or a GUI—issues a checkout request, in the form of a remote procedure call.
2. The checkout procedure requested in the RPC is handled by several *server processes*, which run on the ClearCase LT server.
3. The view database is updated and the file is made writable in the file system (view directory). These operations involve the ClearCase LT server and client, and may involve other computers if either or both of these hosts use network-accessible storage.

ClearCase server processes handle all this automatically and reliably. Users need not be concerned with server-level processing. As an administrator, your responsibilities require a good understanding of the client and server process distribution across your local network.

1.8 ClearCase Startup and Shutdown

ClearCase is normally started and stopped when the ClearCase LT server starts up or shuts down, using the startup and shutdown conventions for the platform type. Starting and stopping ClearCase starts and stops ClearCase server processes. No ClearCase server processes run on a ClearCase LT client.

On UNIX

When UNIX is bootstrapped, the ClearCase startup script is executed by **init(1M)**. The startup script does the following:

- Starts the host's **albd_server** process.
- Starts a **lockmgr** process.

You can also run the **atria_start** script manually, as *root*. For example:

*ccase-home-dir***/etc/atria_start stop**

Stops ClearCase.

*ccase-home-dir***/etc/atria_start start**

Starts (or restarts) ClearCase.

See the **init_ccase** reference page for pointers to system-specific variants of these commands.

On Windows

Unless explicitly configured to be started manually, the **lockmgr** and **albd_server** services are started when a ClearCase LT server starts. You can also start and stop ClearCase services manually from the **ClearCase** program in Control Panel.

2 Administering ClearCase LT Hosts

This chapter discusses a variety of general issues related to ClearCase host administration.

2.1 ClearCase Administration Console

Rational ClearCase LT on Windows has an administration console that centralizes administration of views, VOBs, scheduled jobs, server logs, and VOB tags for UNIX and Windows hosts. Because the ClearCase Administration Console is based on Microsoft Management Console technology, the user interface runs only on Windows. But because it uses standard ClearCase communication protocols to access data on other ClearCase hosts, it is an effective tool for administering your ClearCase LT server whether it is running on Windows or UNIX.

The ClearCase Administration Console helps you manage VOB and view storage, scheduled jobs, and server logs on the ClearCase LT server from any ClearCase LT server or client host running Windows. It also provides easy access to the ClearCase customer Web site. To start the ClearCase Administration Console, click **Start** > **Programs** > **Rational ClearCase LT** > **ClearCase Administration Console**.

In this document, we usually suggest using the ClearCase Administration Console to perform any administrative operation of which it is capable. We also provide information on using the **cleartool** command line to perform administrative tasks.

NOTE: Because the Administration Console is implemented as an MMC snap-in, ClearCase users can create customized administration consoles by adding or removing snap-ins using the MMC model. At ClearCase installation, two **.msc** files (ClearCase snap-in configuration files) are

installed in *ccase-home-dir***bin**. Users gain access to the new administration tools by using MMC and these **.msc** files.

Because these files are modified by MMC to store user preferences and can change over time, the installation also places copies of the installed versions of both files in *ccase-home-dir***config\ui\preferences**. Users who need to restore the default version of one or both files can copy the versions in *ccase-home-dir***config\ui\preferences** to *ccase-home-dir***bin.**

2.2 ClearCase Data and Non-ClearCase Hosts

A host on which ClearCase has not been installed can still access VOB data. There are several ways to provide this access, some more restrictive than others:

- **Use a snapshot view on a ClearCase host.** You can use this method when the non-ClearCase host can use a remote file access facility, such as NFS or Microsoft Windows networking, to access files in the snapshot view directory.

  This method offers good performance because it uses native software for remote file access. However, you are restricted to those element versions that the snapshot view selects.

  To use this method:

  a. On a ClearCase host, create a view and load into it the files you want to access from a non-ClearCase host.

  b. On this host, make the files accessible to other hosts on your network. For a UNIX host, export the file system on which the view directory resides. For a Windows host, share the drive or directory on which the view directory resides.

  c. On a non-ClearCase host, use remote file access to read and write files in the view directory.

  d. To modify VOB data, check out and check in versions in the view on the ClearCase host.

- **Use the ClearCase Web interface.** You can use this method when the non-ClearCase host has a browser that the ClearCase Web interface supports and when a ClearCase host is configured as a Web server for the ClearCase interface. You can use the Web interface to modify VOB data without running ClearCase tools on the non-ClearCase host. For information about setting up the Web interface, see Chapter 17, *Configuring a Web Server for the ClearCase Web Interface*.

2.3 Administering Networkwide Release Areas on UNIX

Networkwide release areas for ClearCase LT on UNIX support use of symbolic links to the release area, so some release area administration may be necessary.

Changing the Location of the Release Area

To change the location of the release area:

1. **Reload the distribution medium.** Create a new release area, using the procedure in the *Installation Guide* for Rational ClearCase LT.
2. **Reinstall hosts, as appropriate.** Reinstall any client whose previous installation involved one or more symbolic links to the networkwide release area. (The standard installation model copies some files and links others.) Use the procedure in the *Installation Guide* for Rational ClearCase LT.
3. **Remove the old release area.** When all hosts have been reinstalled, you can remove the old release area.

Renaming a Release Area Host

Because UNIX installations sometimes include symbolic links to the release area—links that include the name of the release host—renaming the release host may make such installations inoperable. If you rename the networkwide release host, reinstall any host whose previous installation involved one or more symbolic links to the networkwide release area.

3 Understanding ClearCase Access Controls

This chapter describes how Rational ClearCase LT controls access to the data it maintains.

3.1 Fundamentals of ClearCase Access Control

ClearCase implements access controls that determine which users can create, read, write, execute, and delete data in ClearCase. Access control depends on the interaction of users and the groups they belong to, ClearCase objects, and user processes or application programs that access ClearCase data on behalf of the users.

Users and Groups

ClearCase does not have its own implementation of user and group accounts. Instead, it relies on the operating system to authenticate users at login time and to provide the details of user identity and group membership that determine a user's rights to execute various ClearCase operations. Both UNIX and Windows provide networkwide databases of user and group names that are well suited to the needs of a distributed application like ClearCase. On UNIX, this database is part of the Network Information System (NIS, NIS+). On Windows, it is part of the Active Directory or Windows NT Domain system.

On both operating systems, a user logs on with a unique user name, which ClearCase uses as the user's identification or *user ID*. A user ID can be a member of one or more groups, one of which is distinguished as the user's *primary group*. On UNIX, the primary group is part of the user's

entry in the NIS *passwd* database. On Windows, a primary group can be specified when the user's domain account is created, although each user should also set the user environment variable **CLEARCASE_PRIMARY_GROUP** to the name of that group (see *Setting the ClearCase Primary Group* on page 33). ClearCase considers both the user ID and primary group when determining a user's access rights to ClearCase objects.

In this document, the term *community* refers to a set of users and computers that access a common ClearCase LT server. A typical ClearCase LT community includes two classes of user:

- **Privileged users** have rights to create, modify, and delete all artifacts under ClearCase control. Access to privileged user accounts should be restricted to a few ClearCase administrators.
- **Ordinary users** have rights to modify VOBs and views that they create, or that have been created by a member of their primary group. We recommend that all ClearCase users in a community be members of the same primary group. This can be a group that already exists, or one that you create expressly for use by the ClearCase community.

NOTE: ClearCase on Windows has a few special requirements for creation of users and groups in Windows domains. See *Domain User and Group Accounts* on page 33.

Privileged Users and Groups

Some users and groups have special importance for ClearCase access control. For example, each ClearCase object has an *owner*, usually the creator of the object, who often has special access rights to the object. If the object is a container object, such as a VOB that contains elements, the ownership of the container object may in part determine who has access to the objects it contains.

ClearCase has a general notion of the *privileged user*, a term used throughout this book to describe a user who is authorized to perform certain critical operations on the ClearCase LT server. (There is no privileged user on the ClearCase LT client.)

- On a ClearCase LT server running UNIX, the privileged user is the **root** user logged in to the local host. Several limitations apply to the privileges of a **root** user logged in to a remote host. These limitations are described in detail in *Restricted Privileges for Remote root* on page 19.
- On a ClearCase LT server running Windows, the privileged user is any member of the local Administrators group on the ClearCase LT server host.

On either UNIX or Windows, the privileged user has the right to create, delete, and modify VOBs and the objects that VOBs contain. Access to the privileged user account should be restricted to ClearCase administrators.

Restricted Privileges for Remote root

Although many ClearCase operations on UNIX hosts treat a remote **root** user (one who is not logged in to the local host) as a privileged user, there are several exceptions to this rule.

- When a user logged in as **root** on a UNIX host attempts to access a view on another UNIX host, the user's identity is interpreted as **nobody.nobody** (unidentified user, unidentified group). This interpretation is typical of NFS implementations on UNIX, and provides a level of security comparable to that provided by NFS. ClearCase does not support any mount option that controls how requests for access by a remote **root** user are treated, so it will not allow view access by a remote **root** user unless the view has been created to be usable by **nobody.nobody**.
- Operations that change the user or group ownership of an object in a VOB (**cleartool protect -chown** or **-chgrp**) will not succeed if they are requested by a remote **root** user.

User Processes

Any user request to read or write ClearCase data causes a user process (such as a ClearCase GUI or command-line program, or a non-ClearCase program such as a text editor that accesses ClearCase data in a dynamic view) running with that user's identity to request access to the data.

A user process has several properties that ClearCase evaluates to determine whether to grant the process access to the requested data:

- **User.** This is the user who starts the process.
- **Primary group.** This is the primary group of the user who starts the process.
- **Supplemental group list.** These are any other groups of which the user who starts the process is a member.

This chapter refers to a process's primary group and other groups together as the process's groups.

ClearCase Objects

The following ClearCase objects are subject to access control:

- VOBs
- Elements and versions
- Types and instances of types, such as labels, branches, and attributes
- Unified Change Management objects, such as projects, folders, activities, and streams
- VOB storage pools
- Views

NOTE: The ClearCase scheduler maintains its own access control mechanism. See *Managing the Scheduler Access Control List* on page 250.

Each of these objects has one or more of these properties, which are important for access control:

- **Owner.** The owner is a user. The initial owner is the user identity of the process that creates the object. For some objects, the initial owner can be changed.

- **Group.** The initial group is the primary group of the process that creates the object. For some objects, the initial group can be changed.

- **Protection mode.** Some objects also have a protection mode. The mode consists of three sets of permissions, one for each of these user categories:

  - The object's owner
  - Any member of the object's group
  - All other users

  Each set of permissions consists of three Boolean values for a user in its category. Each value determines whether the user has one of these permissions to act on the object:

  - Read permission, or permission to view the object's data.

  - Write permission, or permission to modify the object's data. For an object that contains other objects, such as a VOB or a directory, write permission generally means permission to create or delete objects within the containing object.

  - Execute permission. For a file object, execute permission is permission to run the file as an executable program. For a directory object, execute permission is permission to search the directory.

The protection mode for a ClearCase object is summarized in Table 1. Information about an object's protection mode usually takes the form of a single-character abbreviation of each Boolean value in the protection mode; these abbreviations appear in Table 1.

Table 1 Protection Mode for a ClearCase Object

User Category	Read Permission?	Write Permission?	Execute Permission?
Object's Owner	Yes (`r`) or No (`-`)	Yes (`w`) or No (`-`)	Yes (`x`) or No (`-`)
Member of Object's Group	Yes (`r`) or No (`-`)	Yes (`w`) or No (`-`)	Yes (`x`) or No (`-`)
Other	Yes (`r`) or No (`-`)	Yes (`w`) or No (`-`)	Yes (`x`) or No (`-`)

For example, suppose you are working in a view and want to see the protection mode for the directory element **myproj** in the **projects** VOB. Suppose the owner and group of **myproj** have read, write, and execute permission, but other users have only read and execute permissions. The **cleartool describe** command displays the mode, along with the elements's owner (**sue**) and group (**clearusers**), as follows:

cleartool describe myproj

```
...
  Element Protection:
    User : sue         rwx
    Group: clearusers  rwx
    Other:             r-x
...
```

In addition to these single-character abbreviations, ClearCase sometimes uses integers to display object permissions in a compact form. For each user category (owner, group, and others), a single digit from 0 through 7 represents the permissions for that category, as described in Table 2.

Table 2 Protection Mode Digits for a ClearCase Object (Part 1 of 2)

Digit	Read Permission?	Write Permission?	Execute Permission?
0	No	No	No
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes

Table 2 Protection Mode Digits for a ClearCase Object (Part 2 of 2)

Digit	Read Permission?	Write Permission?	Execute Permission?
4	Yes	No	No
5	Yes	No	Yes
6	Yes	Yes	No
7	Yes	Yes	Yes

A sequence of three digits in a row expresses the protection mode for an object, in this order: owner's permissions, group's permissions, others' permissions. For example, the protection mode 750 for an object means that it has these permissions:

Digit	User Category	Permissions
7	Owner	Read, Write, Execute
5	Group	Read, Execute
0	Others	None

Access to ClearCase Data

Whether a process has access to a ClearCase object depends on these factors:

- The user and groups of the process
- The owner and group of the object
- The protection mode of the object, if any

When a process seeks access to a protected object, the following algorithm usually determines whether access is granted:

1. Does the process have the user ID of the owner of the object?
   - Yes: grant or deny access according to the object's protection mode for the **Owner** category.
   - No: go to Step #2.
2. Does the process have the group ID of the group of the object

- Yes: grant or deny access according to the object's protection mode for the **Group** category.
- No: go to Step #3.

3. Grant or deny access according to the object's protection mode for the **Other** category.

If an object has no protection mode, ClearCase determines whether to grant access by using rules that depend on the type of object. See the descriptions in *Access Control for VOBs and VOB Objects* on page 23 and *Access Control for Views and View Objects* on page 29.

3.2 Access Control for VOBs and VOB Objects

VOBs are the principal repositories for ClearCase data. Both VOBs themselves and objects within VOBs participate in access control. These objects include the following:

- Elements and versions
- Types and instances of types, such as labels, branches, and attributes
- Unified Change Management objects, such as projects, folders, activities, and streams
- VOB storage pools

Access Control for VOBs

These VOB properties are important for access control:

- **Owner.** The initial owner is the user of the process that creates the VOB.
- **Group.** The initial group is the primary group of the process that creates the VOB.
- **Supplemental group list**. The initial supplemental group list is empty for a Windows VOB. For a UNIX VOB, it contains the group list of the VOB owner.

A VOB has no protection mode. This chapter refers to a VOB's primary group and other groups as the VOB's groups.

You can use the **cleartool describe** command to display the owner, group, and supplemental group list for a VOB.

After a VOB is created, a privileged user can use the **cleartool protectvob** command to change the VOB's owner, group, or supplemental group list.

NOTE: You cannot use **protectvob** to add the ClearCase administrators group to a VOB's supplemental group list. Members of this group already have full access rights to all VOB objects.

Permission to Create VOBs

Any user can create a VOB.

Permission to Delete VOBs

Only the VOB owner or a privileged user can delete a VOB.

Permission to Read VOBs

You cannot read a VOB directly. Read operations on a VOB are read operations on objects within the VOB. See *Access Control for Elements* and *Access Control for Other VOB Objects*.

Permission to Write VOBs

You cannot write a VOB directly. Write operations on a VOB include creating and deleting objects within the VOB. See *Access Control for Elements* and *Access Control for Other VOB Objects*.

Permission to Execute VOBs

You cannot execute a VOB directly. Execute operations on a VOB are execute operations on objects within the VOB. See *Access Control for Elements* and *Access Control for Other VOB Objects*.

Access Control for Elements

An element has these properties that are important for access control:

- **Owner.** The initial owner is the user ID of the process that creates the element.
- **Group.** The initial group is determined differently on UNIX and Windows hosts.
  - On UNIX, it is the primary group ID of the process that creates the element.

- On Windows, it is the primary group ID of the process that creates the element if that group is on the VOB's group list. Otherwise, it can be any group that appears on the group list of process that creates the element and also on the group list of the VOB.

The group of an element must be one of the VOB's groups.

- **Protection mode.** The initial protection mode for a file element is determined differently on UNIX and Windows hosts.

  - On UNIX, if you create the element from an existing file, the element has the same protection mode as the file, except that no user category has write permission. If you create a file element any other way, the element has only read permission for all user categories. If your **umask** is 0, a directory element initially has read, write, and execute permission for all user categories. Otherwise, the initial read, write and execute permissions are determined by the value of your **umask**.

  - On Windows, a file element initially has only read permission for all user categories. You must explicitly add execute permission for an element by using the **cleartool chmod** command. A directory element initially has read, write, and execute permission for all user categories.

An element's owner, group, and protection mode are the same for all versions of the element.

You can use the **cleartool describe** command or, on Windows, the **Properties of Element** dialog box to display the owner, group, and protection mode for an element.

After an element is created, the element owner, the VOB owner, or the privileged user can use the **cleartool protect** command to change the element's owner, group, or protection mode.

Permission to Create Elements

When you create a VOB, ClearCase creates an initial element, the VOB root directory. This element is the top-level container for other elements in the VOB. Its initial owner is the owner of the VOB, and its initial group is the group of the VOB.

Only a process whose primary group is one of the VOB's groups can create any other element. That process must also have permission to check out a version of the directory element that contains the new element. See *Permission to Write Elements*.

Permission to Delete Elements

Only the element owner, the VOB owner, or the privileged user can delete an element. Deleting an element, using the **cleartool rmelem** command, is not the same as removing the element from a directory version. See *Permission to Write Elements*.

The creator of a version, the element owner, the VOB owner, or the privileged user can delete the version.

Permission to Read Elements

An algorithm that considers the process's user and group and the element's owner, group, and protection mode determines whether to grant read permission for an element. See *Access to ClearCase Data* on page 22.

Permission to Write Elements

A process cannot write elements directly. You modify an element by checking out a version of it and checking in a new version.

The element's protection mode is not considered when determining whether a process can check out or check in a version. A process can check out a version if any of these conditions exist:

- The process has the user identity of the element's owner.
- Any of the process's group identities is the same as the element's group.
- The process has the user identity of the VOB owner.
- The process has the user identity of the privileged user.

A process can check in a version if any of these conditions exist:

- The process has the user identity of the user who checked out the element.
- The process has the user identity of the element's owner.
- The process has the user identity of the VOB owner.
- The process has the user identity of the privileged user.

When a directory element is checked out, you can modify the directory by creating elements or by removing elements from it. Removing an element's name from a directory, using the **cleartool rmname** command, is not the same as deleting the element itself. See *Permission to Delete Elements*.

Permission to Execute Elements

An algorithm that considers the process's user and group and the element's owner, group, and protection mode determines whether to grant execute permission for an element. See *Access to ClearCase Data* on page 22. In addition, two special cases can restrict permission to execute an element:

- On Windows, a file element does not have execute permission until you add it by using the **cleartool chmod** command. For example:

  cleartool chmod +x command.exe

  If you add an executable program to source control, you will not be able to execute it until you take this step.

- On UNIX, a VOB mounted with the **nosuid** mount option will not allow setuid executables to run.

Access Control for Other VOB Objects

In addition to elements and versions, a VOB contains other kinds of objects that are subject to access control:

- *Metadata* types, such as label types, branch types, and attribute types
- Unified Change Management objects, such as projects, activities, folders, and streams
- Storage pools

In general, each of these objects has two properties that are important for access control:

- **Owner.** The initial owner is the user of the process that creates the object.

- **Group.** The initial group is the primary group of the process that creates the object.

You can use the **cleartool describe** command to display the owner and group of an object. After the object is created, the object's owner, the VOB owner, or the privileged user can use the **cleartool protect** command to change the object's owner or group. The group of the object must be one of the VOB's groups.

Permission to Create Other VOB Objects

Any user can create a type or a UCM object. Only the VOB owner or the privileged user can create a storage pool.

Instances of types, such as labels, branches, and attributes, are usually associated with element versions. To create an instance of one of these types, one of the following conditions must exist:

- The process has the user identity of the element's owner.
- Any of the process's group identities is the same as the element's group.
- The process has the user identity of the VOB owner.
- The process has the user identity of the privileged user.

Permission to Delete Other VOB Objects

The owner of the object, the owner of the VOB, or the privileged user can delete a type or UCM object.

Instances of types, such as labels, branches, and attributes, are usually associated with element versions. In general, if you can create an instance of a type, you can also delete the instance. See *Permission to Create Other VOB Objects*. In addition, the creator of a branch can delete the branch.

Permission to Read Other VOB Objects

Any user can display information about a type, a UCM object, or a storage pool.

Permission to Write Other VOB Objects

Any user can change a UCM object. The owner of the object, the owner of the VOB, or the privileged user can change a type.

Locks on VOB Objects

The ClearCase permissions scheme is intended for use as a long-lived access-control mechanism. ClearCase also provides for temporary access control, through explicit *locks* on individual VOB objects. You can use the **lock** command to restrict or prohibit changes at various levels. At the lowest level, you can lock an individual element, or even an individual branch of an element. At the highest level, you can lock an entire VOB, preventing all modifications to it.

When an object is locked, it cannot be modified by anyone, even the privileged user or the user who created the lock. (But these users have permission to unlock the object.) The **lock** command accepts an exception list that specifies which users can modify the object despite the lock.

Locking Type Objects

You can lock type objects; this prevents changes to the instances of those types. For example:

- You can lock the branch type **main** to all but a select group of users. This group can then perform integration or release-related cleanup work on the **main** branches of all elements. All other users can continue to work, but must do so on subbranches, not on the **main** branch.
- Locking a label type prevents anyone from creating or moving that label. Labeling all the element versions used in a particular release, and then locking that label, provides an easy way to re-create the release later.

3.3 Access Control for Views and View Objects

Views mediate user access to ClearCase elements and versions. Like VOBs and objects within VOBs, views participate in access control. In a ClearCase LT snapshot view, native file-system permissions on the snapshot view directory tree determine access to files and directories in the snapshot view, including copies of element versions. Creating, deleting, and modifying elements in a snapshot view require the process to have the appropriate permissions for those elements.

NOTE: On UNIX hosts, ClearCase treats view access requests from a remote **root** user as requests from the user **nobody.nobody**. See *Restricted Privileges for Remote root* on page 19 for details.

3.4 ClearCase and Native File-System Permissions

ClearCase maintains data for VOBs and views in special directory trees in the native file system on the ClearCase LT server. These special directory trees are the *VOB storage directory* and the *view storage directory*. ClearCase creates the VOB storage directory when you create a VOB and the view storage directory when you create a view.

ClearCase manages all access to the VOB and view storage directories; users never read from or write to these directories directly. The VOB and view storage directories have native file-system

permissions, but ClearCase itself creates and maintains these permissions. Although ClearCase stores objects subject to access control in these directories, you must manage ClearCase access control using the mechanisms described in this chapter.

WARNING: **Never change native file-system permissions on the directories and files in any VOB storage directory or view storage directory tree.**

If you have inadvertently changed permissions on a VOB or view storage directory, you may be able to repair the damage using ClearCase tools. See Chapter 20, *Repairing VOB and View Storage Directory ACLs on Windows*.

A snapshot view directory is a native file-system directory tree that contains copies of ClearCase versions and file-system objects that are not under ClearCase control. The owner of a snapshot view can manage native file-system permissions on these files. For example, the view owner can add or remove group write permission for files in a snapshot view directory that are not under ClearCase control.

A view also has a view storage directory, which may or may not be located within the snapshot view directory. ClearCase creates and maintains the view storage directory. Never change native file-system permissions on the view storage directory.

4 ClearCase and Windows Domains

Rational ClearCase LT relies on the operating system to establish a user's identity. On Windows computers, user and group identity are established when the user logs in to a Windows NT or Active Directory domain. This chapter describes what a domain administrator needs to know about the user, group, and resource accounts that ClearCase LT requires, and about the effect of domain trust relationships on ClearCase LT.

This chapter is intended for domain administrators who have experience with Windows NT domains, Active Directory domains, or both, and are familiar with Microsoft's domain administration and account maintenance tools.

4.1 Domain Configurations Compatible with ClearCase LT

ClearCase LT is compatible with a wide variety of domain configurations. The simplest configuration—a single domain that includes all users, groups, and computers—entails the least administrative overhead, but other common configurations work equally well:

- Master and multi-master Windows NT domain configurations in which user and group accounts are created in a master domain and host (computer) accounts are created in one or more resource domains that trust the master domain.
- Active Directory domains that are part of a single forest.
- Domain upgrade and domain migration environments that support a combination of Windows NT and Active Directory domains.

What ClearCase LT Requires from Any Domain

ClearCase LT imposes a few simple requirements on any domain environment in which it operates:

- All members of a ClearCase LT community (all users who access the same VOBs and views) must have domain accounts and have at least one group membership in common.
- All ClearCase LT hosts must be members of a domain. If the hosts are not members of the same domain in which the user and group accounts are created, they must be members of a domain that trusts that domain.
- Additional steps must be taken to enable users from multiple domains to access a common set of VOBs and views.

ClearCase LT on Nondomain Hosts

Windows computers that are not in a domain can be ClearCase LT hosts, though with severely limited functionality:

- They cannot be ClearCase LT servers.
- They cannot access VOBs and views hosted on a ClearCase LT server running Windows.

Nondomain systems can function as ClearCase LT clients in networks whose ClearCase LT server is on a UNIX computer.

Nondomain systems can also function as stand-alone systems on which all VOBs and views are used locally. This situation typically arises when you are evaluating ClearCase LT software. For more information, see *Installation Guide* for ClearCase LT.

Nondomain systems can also function as stand-alone systems that act as both a ClearCase LT server and client. This situation typically arises when you are evaluating ClearCase LT software.

4.2 Domain User and Group Accounts

Any ClearCase LT community must define a group to which all users who perform routine ClearCase LT operations using a common set of VOBs and views belong. We refer to this group as the ClearCase users group. It can be an existing domain global group or one created specifically for this purpose. In examples throughout this document, the ClearCase users group is named **clearusers**.

The ClearCase users group must have the following characteristics:

- It must be a domain global group or an Active Directory universal group. We recommend using a domain global group unless the group needs to include other groups.
- Its name must be the same as the name of the ClearCase users group on UNIX if members of the group must access VOBs or views on a UNIX host. UNIX and Windows place different restrictions on the length of group names, as well as on the characters that are allowed in them. Be sure that the ClearCase users group name is acceptable in both environments.

NOTE: If your ClearCase LT community includes multiple groups that share VOBs and views among their members but do not share these VOBs and views with members of other groups, you must designate a different ClearCase users group for each of these groups.

Setting the ClearCase Primary Group

Although you can designate a user's primary group using various Windows domain account maintenance tools, this group name is not always returned when an application requests the name of a user's primary group. We strongly recommend that you ask each user to set the user environment variable **CLEARCASE_PRIMARY_GROUP** to the domain-qualified name of the ClearCase users group. For example:

MYDOMAIN\clearusers

This setting guarantees an unambiguous definition of the group that ClearCase considers the user's primary group.

The **CLEARCASE_PRIMARY_GROUP** assignment has no security or access-control implications outside the context of VOB access. Users who have not set **CLEARCASE_PRIMARY_GROUP** correctly are likely to have problems creating elements or otherwise accessing VOBs, especially in complex domain configurations.

Users must set the value of **CLEARCASE_PRIMARY_GROUP** as a user environment variable on each Windows platform from which they will access any VOB or view.

NOTE: Members of the ClearCase users group who are also members of the ClearCase administrators group must set **CLEARCASE_PRIMARY_GROUP** to the name of the ClearCase users group, not the name of the ClearCase administrators group.

On a computer running Windows Me or Windows 98, you must set the **CLEARCASE_PRIMARY_GROUP** variable in the **AUTOEXEC.BAT** file.

To verify that **CLEARCASE_PRIMARY_GROUP** has been properly set.

1. Click **Start** > **Programs** > **Rational ClearCase LT** > **ClearCase Doctor.**
2. Click **Start Analysis**.
3. When the analysis is finished, click the **Topics** tab and open the **User Login Account** folder.
4. Double-click **Primary Group**, read the user's primary group, and verify that it is correct.

4.3 Multiple User Account Domain Support

If your existing domain configuration requires it, you can enable ClearCase access for users and computers in multiple domains. Because this configuration can be more complicated to set up and administer, we recommend that you avoid using it unless organizational or security concerns require you to do so.

Using Active Directory Universal Groups

When a ClearCase LT community includes users from multiple Active Directory domains that are part of the same forest, you can use an Active Directory universal group to provide users logged on to different domains with access to a common set of VOBs and views.

To create an Active Directory universal group that can be used as the ClearCase primary group by users from multiple Active Directory domains in a single forest:

1. Verify that the Active Directory environment is operating in native mode. (Universal groups cannot be created in an Active Directory domain that is operating in mixed mode.)

2. Create the ClearCase users group as an Active Directory universal group.

3. Make each domain global group whose members are part of the ClearCase community a member of the ClearCase users group. We do not recommend adding individual user accounts to a universal group. Instead, group the users from each Active Directory domain into a domain global group defined in that domain, and make each of those domain global group a member of the universal group.

4. Require each user in each of the domain global groups that are members of the ClearCase users group to set **CLEARCASE_PRIMARY_GROUP** to the domain-qualified name of the (universal) ClearCase users group. (You cannot use Active Directory account management tools to specify a universal group as a user's primary group.)

NOTE: If you are upgrading a multi-master Windows NT domain environment to Active Directory, use the procedure described in *Converting Proxy Groups* on page 42 to convert the proxy groups to members of an Active Directory universal group.

Using Proxy Groups and Domain Mapping in Windows NT Domains

When a ClearCase LT community includes users from multiple Windows NT domains, you must enable the ClearCase domain mapping feature as described in this section to provide all users with access to a common set of VOBs and views.

Suppose that ClearCase LT users have accounts in domains named **ATLANTA**, **BOSTON**, and **CUPERTINO**, and that the primary group of each VOB they need to share is **ATLANTA\clearusers**. To use ClearCase LT in this environment, create *proxy groups* and enable the domain mapping feature as follows:

1. Ensure that each ClearCase LT host is a member of a resource domain that trusts the **ATLANTA**, **BOSTON**, and **CUPERTINO** domains.

2. Create the ClearCase users group in one of the user account domains. In this example, the domain is Atlanta and the group is **ATLANTA\clearusers**. VOBs to be shared by users taking advantage of domain mapping must be owned by the **ATLANTA\clearusers** group.

3. Create two more domain global groups, one in each of the other domains:

   - In the Boston domain, create the group **BOSTON\clearusers_Boston**.
   - In the Cupertino domain, create the group **CUPERTINO\clearusers_Cupertino**.

   When creating these groups, make sure their description strings contain the following substring:

```
ClearCaseGroup(Atlanta\clearusers)
```

This string must be case-correct and contain no spaces. When this text string is present in a group description, ClearCase LT recognizes the group as a proxy group for the group whose name is delimited by the parentheses (in this case, the group **ATLANTA\clearusers**). When evaluating VOB access rights, ClearCase LT treats members of a proxy group as though they were members of the group named in the **ClearCaseGroup** substring. In this example, a member of **BOSTON\clearusers_Boston** will have the same VOB access rights as a member of **ATLANTA\clearusers** if the description of **BOSTON\clearusers_Boston** includes the string **ClearCaseGroup(ATLANTA\clearusers).**

4. Make ClearCase LT users members of the appropriate groups:

   - Make users whose accounts are in domain Atlanta members of **ATLANTA\clearusers**.
   - Make users whose accounts are in domain Boston members of **BOSTON\clearusers_Boston**.
   - Make users whose accounts are in domain Cupertino members of **CUPERTINO\clearusers_Cupertino**.

5. Enable domain mapping on each ClearCase LT host. To do so, edit the Windows registry on that host using the following procedure:

   a. Using a Windows registry editor, navigate to **HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion**.

   b. Click **Edit** > **Add Value**.

   c. In the **Add Value** dialog box, enter **DomainMappingEnabled** in the Value Name and select **REG_DWORD** as the value type.

   d. Click **OK** to start the DWORD editor

   e. In the DWORD editor, enter 1 in the **Data** box. Make sure that the Radix is set to Hex (the default).

   f. Click **OK** to add the value.

   NOTE: To enable domain mapping for a Windows 98 or Windows Me computer, enable it on the Windows host that is designated as the credentials mapping server; then shut down and restart the Windows Me or Windows 98 computer.

6. Require each ClearCase LT user to set the user environment variable **CLEARCASE_PRIMARY_GROUP** to the value **ATLANTA\clearusers**. See *Setting the ClearCase Primary Group* on page 33.

Setting VOB Element Permissions

All elements in any VOB that will be accessed by users who are members of proxy groups must allow **Read** rights for **Other**. Newly created elements grant this right by default. You can examine an element's protection from Windows Explorer:

1. Right-click the element to display the shortcut menu.
2. Click **ClearCase** > **Properties of Element**.
3. In the **Properties of Element** dialog box, click the **Protection** tab. The **Read** check box in the **Other** group must be selected.

To reprotect a large number of elements, use the **cleartool protect** command.

Setting VOB Storage ACLs

If necessary, you can restrict access to world-readable elements to a smaller set of users by setting the access control list (ACL) on the share that contains the VOB storage directory. For example, if a VOB is registered with the global path **\\myserver\vobstorage\src_vob**, you can set the ACL on the **vobstorage** share to restrict access to **ATLANTA\clearusers**, **BOSTON\clearusers_Boston**, and **CUPERTINO\clearusers_Cupertino**.

4.4 Conversion to Active Directory

Like any enterprise-scale application in a Windows network, ClearCase LT is affected when the network is converted from Windows NT domains to Active Directory domains. This section and the following sections describe how this conversion affects ClearCase LT, and how to manage ClearCase LT users, groups, hosts, and data during and after the conversion.

NOTE: If you are using ClearCase LT in an environment that is already running Active Directory, these sections do not apply to you.

Understanding Active Directory

Microsoft provides tools and documentation to facilitate conversion of a Windows network from Windows NT domains to Active Directory. In this section, we assume you have read the applicable documents from Microsoft and are familiar with the terminology they use and the

procedures they describe. In particular, we assume you have read the Microsoft white paper entitled *Planning Migration from Microsoft Windows NT to Microsoft Windows 2000.* (It is distributed as part of the Windows 2000 Support Tools and is also available on Microsoft's Web site.) That document and related documents introduce several key concepts— including *native mode, mixed mode, domain upgrade, domain migration, SID history,* and *cloning* of principals—that we use throughout this chapter.

How Active Directory Affects ClearCase LT

In an Active Directory environment, some details of user and group identity are handled differently than they are in a Windows NT domain environment. Depending on how your Windows NT domain environment is configured, where your ClearCase user and group accounts exist in this domain structure, and how your organization plans to convert Windows NT domains to Active Directory domains, you may need to take steps during and after the conversion process to maintain user access to artifacts under ClearCase control.

Conversion to Active Directory affects ClearCase LT in several ways:

- In Active Directory, trust relationships between domains are created and maintained differently than they are in Windows NT domains. During and after the conversion to Active Directory, these differences will affect ClearCase LT communities in which users from multiple domains access a common set of VOBs and views.
- Windows Security Identifiers (SIDs) for users and groups can change in some conversion scenarios. Because ClearCase stores SIDs in VOB databases (to represent owners of objects), VOBs must be updated with new SIDs in these scenarios.

In general, sites that have the simplest domain structure (all ClearCase LT users and hosts in a single domain) will encounter very few problems during the conversion process. Sites with more complex domain structures (users from multiple domains accessing a common set of VOBs and views) can benefit from Active Directory's improved interdomain security features after they modify some existing user and group account information.

Planning Your Active Directory Upgrade or Migration Strategy

Microsoft provides tools and documentation to facilitate conversion of domains from Windows NT to Active Directory. The conversion can take one of two forms:

- An upgrade (often referred to as an in-place upgrade), in which a Windows NT domain controller is converted to an Active Directory domain controller operating in mixed or native mode. After an upgrade, all users, groups, and resources have the same SIDs as they had in their original Windows NT domain.

- A migration, in which user, group, and resource accounts migrate (using a process referred to as *cloning*) from a Windows NT domain to an Active Directory domain. After a migration is complete, all users, groups, and resources have new SIDs. Because a native mode Active Directory maintains information about each principal's current and former SIDs (referred to by Microsoft as the principal's SID history), both types of domains can be used together for as long as needed.

We recommend that a knowledgeable ClearCase administrator who has reviewed this chapter and applicable documents from Microsoft, and who understands the impact of various conversion or migration strategies on ClearCase, be familiar with (and if possible help plan) your organization's conversion from Windows NT domains to Active Directory.

CAUTION: Microsoft supplies tools for converting the SIDs stored in NTFS ACLs. Never use these tools (or any tools that change native file system protection information) on a VOB or view storage directory. Only ClearCase utilities should be used to convert SIDs in VOB or view storage directories. See *Migrating Multiple Domains* on page 43 for details.

Preparing ClearCase LT Hosts

Before you begin the conversion process, your ClearCase LT hosts must be configured for use in an Active Directory environment.

- All ClearCase LT hosts must be running ClearCase LT version 5.0 or later.

  NOTE: Hosts running earlier releases of ClearCase LT can be converted to Active Directory, although various restrictions apply. For more information, see the ClearCase customer site at **www.rational.com**. This chapter does not apply to hosts running earler releases.

- All VOBs on Windows hosts must be at schema version 54. Schema version 54 stores Windows user and group identity information in SID form to better support Active Directory's improved handling of user and group authentication.

- All views must be reformatted. A view is reformatted automatically the first time it is started after ClearCase LT version 5.0 has been installed on the view host. You can also reformat a view manually using the **reformatview** command.

- The user environment variable **CLEARCASE_PRIMARY_GROUP** must be defined for all users. We recommend that the value of this variable be a domain-qualified group name of the form *DOMAIN_NAME\group_name*.

Verify that all ClearCase LT hosts have been configured as described in this section and that ClearCase LT is operating normally for all users and hosts before you proceed with the conversion to Active Directory.

4.5 Domain Upgrade Scenarios

This section describes several scenarios in which one or more Windows NT domains are upgraded to Active Directory domains using the in-place upgrade procedure defined by Microsoft. If your site is not using this procedure, see *Domain Migration Scenarios* on page 43.

NOTE: In this section, references to upgrading a domain refer to upgrading the primary domain controller of that domain.

Upgrading a Single Domain

Use this procedure if all ClearCase users, groups, and hosts are members of a single Windows NT domain:

1. Prepare ClearCase LT hosts as described in *Preparing ClearCase LT Hosts* on page 39.
2. If you do not have a backup domain controller online during the upgrade, stop ClearCase on the ClearCase LT server to prevent ClearCase operations during the upgrade process.
3. Use the procedures defined by Microsoft to perform an in-place upgrade of the Windows NT domain to an active directory domain.
4. After the upgrade of the primary domain controller is complete, shut down and restart all ClearCase LT hosts.

Upgrading a Master Domain and Its Resource Domains

Use this procedure if all ClearCase users and groups are members of a single Windows NT domain (the master domain) that is trusted by one or more Windows NT resource domains to which ClearCase hosts belong.

To upgrade a master domain and its resource domains:

1. Upgrade the master domain as described in *Upgrading a Single Domain* on page 40.
2. Upgrade each resource domain as described in *Upgrading a Single Domain* on page 40. Configure the upgraded resource domain as a child of the upgraded master domain.
3. After the upgrade of a resource domain is complete, shut down and restart all ClearCase LT hosts in that resource domain.

Upgrading Multiple Master and Resource Domains

Use this procedure if ClearCase users and groups are members of more than one Windows NT domain and you are using proxy groups to enable users from these domains to access a common set of VOBs and views. The domains can include resources as well as users and groups or can be master domains trusted by resource domains.

After the upgrade is complete, we recommend converting the Active Directory domains to native mode so that you can use an Active Directory universal group to eliminate the need for proxy groups and domain mapping.

To upgrade multiple master and resource domains:

1. Prepare ClearCase LT hosts as described in *Preparing ClearCase LT Hosts* on page 39.
2. If you do not have a backup domain controller online for each of the master and resource domains during the upgrade, stop ClearCase on the ClearCase LT server to prevent ClearCase operations during the upgrade process.
3. Use the procedures defined by Microsoft to perform an in-place upgrade of the first Windows NT master domain to an Active Directory domain. Upgrade the domain in which the ClearCase users group is defined before upgrading the domains in which the proxy groups are defined.
4. Upgrade the remaining master domains.

5. After the master domains have been upgraded, you may begin to upgrade the resource domains. As long as you do not alter existing trust relationships between domains that have been upgraded and those that have not, you may upgrade the resource domains in any order and on any schedule that is appropriate for your organization. We recommend making all of the upgraded domains members of the same forest, which will allow you to take advantage of Active Directory universal groups, as described in *Converting Proxy Groups*.

6. After the upgrade of a resource domain is complete, shut down and restart all ClearCase LT hosts in that resource domain.

7. After all domains have been upgraded and the Active Directory domain has been converted to native mode, follow the procedure described in *Converting Proxy Groups* on page 42 to eliminate proxy groups and domain mapping.

Converting Proxy Groups

Use the following procedure to replace the proxy groups required in a Windows NT domain environment with a ClearCase users group that is an Active Directory universal group. The universal group includes the groups formerly used as proxy groups.

NOTE: You can only use this procedure in an Active Directory domain that is operating in native mode. The ClearCase users group and the proxy groups must all exist in the same forest.

1. Use Active Directory management tools to rename the ClearCase users group in the primary ClearCase domain. For example, the **ATLANTA\clearusers** group created in the procedure described in *Using Proxy Groups and Domain Mapping in Windows NT Domains* on page 35 could be renamed to **ATLANTA\clearusers_Atlanta**.

2. Use Active Directory management tools to create a new (universal) ClearCase users group. We suggest keeping the name the same (**ATLANTA\clearusers** in our example), so that users do not have to change the value of their **CLEARCASE_PRIMARY_GROUP** environment variable.

3. Use Active Directory management tools to add the former proxy groups (**BOSTON\clearusers_Boston** and **CUPERTINO\clearusers_Cupertino** in our example) and the group you renamed in Step #1 of this procedure as members of the new (universal) ClearCase users group.

4. Disable domain mapping on all ClearCase LT client and server hosts. To disable domain mapping on a ClearCase LT host, use a registry editor to navigate to the registry key **HKEY_CURRENT_USER\SOFTWARE\Atria\ClearCase\CurrentVersion**, and then remove the DWORD value **DomainMappingEnabled**. (Some ClearCase LT hosts may store this value in **HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion**.)

5. Shut down and restart all ClearCase LT client and server hosts.

4.6 Domain Migration Scenarios

Sites for which a domain upgrade is impossible can use a domain migration process. This process uses Microsoft tools to populate a native-mode Active Directory domain with user, group, and resource accounts that have been cloned from existing accounts in a Windows NT domain. Both types of domain can operate in parallel and, if the appropriate trust relationships exist between the Windows NT and Active Directory domains, users and groups in either type of domain have equivalent ClearCase access rights.

Migration can take place over an extended period, if necessary. When all user and group accounts have been migrated to the Active Directory domain, the migration process can be completed and the Windows NT domains decommissioned. After a migration, all users, groups, and hosts have new SIDs. This has several implications for ClearCase LT:

- VOB storage directories must be re-protected so that they include the new SIDs of the VOB owner in the directory ACL.
- VOB databases must be updated with the new SIDs. The ClearCase utility program **vob_sidwalk** replaces old SIDs with new ones. See *Using vob_sidwalk to Change or Update VOB Users and Groups* on page 47 and the **vob_sidwalk** reference page for more information on this program.

Detailed instructions for performing these procedures are included in the remainder of this section.

Migrating Multiple Domains

All migration scenarios are essentially the same, differing only in their level of complexity as measured by the number of domains being migrated and the trust relationships among them. The procedure defined in this section can be used whether you are migrating a single domain or multiple master and resource domains.

NOTE: Before you begin any migration process, prepare all ClearCase LT hosts as described in *Preparing ClearCase LT Hosts* on page 39. Even though the hosts may not be migrating right away, they will not be accessible by migrated users and groups until you have taken this step.

Migrating Users and Groups

We recommend that you begin any domain migration by migrating users and groups from the Windows NT domains in which they were created to new Active Directory domains. Follow these steps to migrate ClearCase users and groups:

1. Use the procedures defined by Microsoft to migrate users and groups from the Windows NT domains to the Active Directory domains.

2. In many migration scenarios, there will be a period when users logged in to the Active Directory domain will share a VOB with users logged in to a Windows NT domain. To ensure access to VOBs by users in either type of domain, do one of the following:

   - Add the domain-qualified name of the ClearCase users group that has been migrated to the Active Directory domain to the VOB's supplemental group list. For example, you can use the **cleartool protectvob** command as shown here to add the **clearusers** group in the Active Directory domain **AD-DOMAIN** to the group list of the VOB with storage on the VOB server host at **C:\vobsvr\vobstg\srcs.vbs**:

     cleartool protectvob -add_group AD-DOMAIN\clearusers C:\vobstg\srcs.vbs

   - Ask users logged in to an Active Directory domain to set their **CLEARCASE_PRIMARY_GROUP** environment variable to the string representation of the SID of the ClearCase users group in the Windows NT domain. To find the SID string, run the **creds** command—on a computer that is a member of the Windows NT domain or a domain that trusts the Windows NT domain—as shown in this example:

     *ccase-home-dir***etc\utils\creds –g NT-DOMAIN\clearusers**

     .
     .
     .

     ```
     ClearCase group info:
     Name: NT-DOMAIN\clearusers
     GID:  0x100423
     SID credentials NT:S-1-5-21-103034363-981818062-1465874335-1064
     ```

     In this case, the user should set **CLEARCASE_PRIMARY_GROUP** to the value **NT:S-1-5-21-103034363-981818062-1465874335-1064**

3. Because migrated accounts include SID history, user accounts in the Active Directory domain include twice as many group memberships as they had in the Windows NT domain. (Each user's group list includes groups from both domains.) Users who are members of multiple groups in a Windows NT domain may find that their group list includes more than 32 groups after migration. Because ClearCase only recognizes 32 groups, these users should

set their **CLEARCASE_GROUPS** environment variable to include the SID string that represents the ClearCase users group in the Windows NT domain, as well as the name of the ClearCase users group in the Active Directory domain. See Step #2 of this procedure for information on how to use **creds** to obtain the SID string.

If the user environment variable **CLEARCASE_GROUPS** exists for any user, ClearCase will consider the semicolon-separated list of groups specified in the value of this variable first when determining (or displaying) which groups a user belongs to. This example includes the name of the ClearCase users group from the Active Directory domain and the SID of the ClearCase users group from the Windows NT domain.

CLEARCASE_GROUPS=AD-DOMAIN\clearusers;NT:S-1-5-21-103034363-981818062-1465874335-1064

If You Must Add a New User While Migration Is In Progress

If a new ClearCase user must be created while a domain migration is in progress, use either of the following methods:

- Create the user's account in the Active Directory domain and ask the user to set **CLEARCASE_PRIMARY_GROUP** environment variable to the domain-qualified name of the ClearCase users group that has already been cloned and exists in the Active Directory domain.
- Create the user in the Windows NT domain, and then migrate the user account.

These users may also need to change their **CLEARCASE_PRIMARY_GROUP** and **CLEARCASE_GROUPS** environment variables as described in *Migrating Users and Groups*.

Migrating Individual ClearCase LT Hosts

When all users and groups have been migrated and no users are required to log in to the Windows NT domain, take the following steps to migrate ClearCase LT hosts to the Active Directory domain. If you cannot migrate all your ClearCase LT hosts at the same time, we recommend migrating VOB servers first. (Registry and license servers can migrate at any time, because they do not store SIDs in their databases.)

NOTE: The procedures in this section require you to use the **vob_sidwalk** utility. Before executing any of these procedures, review the **vob_sidwalk** reference page to get a better understanding of the capabilities of **vob_sidwalk**.

To migrate an individual ClearCase LT host:

1. Stop ClearCase on the ClearCase LT server.
2. Use the procedures defined by Microsoft to migrate the host to the Active Directory domain.
3. Restart ClearCase on the ClearCase LT server.
4. Reprotect any VOB and view storage on the host by running the ClearCase **fix_prot** utility on each VOB or view storage directory as shown in this example:

   *ccase-home-dir***etc\utils\fix_prot –replace** *storage-dir-pname*

   where *storage-dir-pname* is the pathname to the VOB or view storage directory.
5. Use the following procedure to update VOB databases with the new SIDs representing the cloned user and group accounts.
   a. Log on to the VOB server host as the VOB owner or privileged user
   b. Lock the VOB for all users but yourself (**–nusers** *your-user-name*).
   c. Replace the old SIDs with the new ones. Use the **vob_sidwalk** utility as shown in the following example, where *vob-tag* is the VOB-tag of the VOB you are updating and *SIDfile-path* is the name of a file where **vob_sidwalk** will log the changes it makes:

      *ccase-home-dir***etc\utils\vob_sidwalk -s -execute** *vob-tag SIDfile-path*

      For additional details on how **vob_sidwalk** remaps SIDs, see *Remapping Historical SIDs After Domain Migration* on page 48.
   d. Unlock the VOB.

NOTE: If you had been using proxy groups to enable users from multiple domains to access a common set of ClearCase artifacts, we recommend using an Active Directory universal group to eliminate the need for proxy groups and domain mapping. Follow the procedure described in *Converting Proxy Groups* on page 42

If VOB Servers Cannot Migrate When Clients Do

If any VOB server cannot migrate when its clients do and you need to preserve the clients' ability to access VOBs on that server, you must use **vob_siddump** to establish the mapping between new SIDs and old ones. After the mapping has been established, you can use **vob_sidwalk** to update the VOB database with the new SIDs.

1. Log on to a client that has been migrated to the Active Directory domain.

2. Lock the VOB for all users but yourself (**–nusers** *your-user-name*).

3. Run **vob_siddump** to generate a map file. (You must use **vob_siddump** because you cannot run **vob_sidwalk** from a remote host.) In the following example, *vob-tag* is the VOB-tag of a VOB on a server that is still in the Windows NT resource domain, and *SIDfile-path* is the pathname to the map file that **vob_siddump** will generate. (If *SIDfile_path* cannot be created on a drive that is accessible to the VOB server host, you must copy it to the VOB server host before you perform Step #4 of this procedure.)

   vob_siddump -sidhistory *vob-tag SIDfile-path*

4. Log on to the VOB server that hosts the VOB whose tag you used in Step #3 of this procedure. With the VOB still locked for all users but yourself, run **vob_sidwalk** to update the SID information stored in the VOB

   vob_sidwalk -execute -map *mapfile-path vob-tag SIDfile-path*

   In this example, *mapfile-path* is the map file you generated in Step #3 of this procedure and *SIDfile-path* is the name of a file in which **vob_sidwalk** will log the changes it makes. For more information, see *Using vob_sidwalk to Change or Update VOB Users and Groups* on page 47.

5. Unlock the VOB.

NOTE: Unless the VOB remains locked from the time you begin Step #3 until the time you complete Step #4, users may create new objects in the VOB between the steps. If they do, you will have to perform both steps again.

4.7 Using vob_sidwalk to Change or Update VOB Users and Groups

Any time you move a VOB to a host in a domain that does not trust the domain in which the VOB's original host exists, all SIDs stored in the VOB database become invalid, because they do not resolve to an account in the new domain. This scenario occurs during domain migration (the host moves to a different domain, but the VOB stays on the host). It also occurs when a VOB is moved from a host in one domain to a host in a different domain.

The **vob_sidwalk** command provides a flexible means of reassigning ownership of objects in a VOB, updating the SIDS that represent the groups in a VOB's group list, and correcting VOB storage directory protections. Common uses for **vob_sidwalk** include:

- Migrating a VOB from a Windows NT domain to an Active Directory domain

- Moving a VOB to a host in a domain that does not trust the original domain
- Moving a VOB from a Windows host to a UNIX host or vice versa
- Moving a VOB server host to a domain that does not trust the original domain

This section provides several examples of procedures that use **vob_sidwalk** and **vob_siddump**. There are additional examples of procedures that use **vob_sidwalk** in Chapter 11, *Moving VOBs*. The **vob_sidwalk** reference page provides complete information on all **vob_sidwalk** and **vob_siddump** options.

Regardless of the procedure you are using, we recommend that you run **vob_siddump** (or **vob_sidwalk** without the **-execute** option) and then examine the output to determine which objects in the VOB would have new owners. After you are sure that the changes in ownership are the ones you want, run **vob_sidwalk** with the **-execute** option to actually remap the SIDs. The output SID file is written in comma-separated-value (.csv) form, so it can be viewed and changed with any text editor or any spreadsheet program that can read a file of this format.

NOTE: On Windows, **vob_sidwalk** can only be used on VOBs formatted with schema version 54. It is only installed on hosts that are configured to support local views and VOBs and support VOB schema version 54.

Remapping Historical SIDs After Domain Migration

In a domain migration scenario, a VOB database includes additional SIDs that represent the SID histories of the security principals (users and groups) who own objects in the VOB. These historical SIDs were associated with the security principals before migration and are stored in the principal's **sIDHistory** attribute in an Active Directory domain.

To replace the historical SIDs stored in the VOB database with new ones that resolve to the appropriate security principals in the Active Directory domain, use a command like this one:

vob_sidwalk -sidhistory -execute *vob-tag SIDfile-path*

When invoked with the **-sidhistory** option, **vob_sidwalk** uses the following algorithm to determine SID history:

1. Look up a SID to find the account name.
2. Look up the account name found in Step #1 to find its SID.

3. If the SID returned in Step #2 is different from the SID used in Step #1, the SID used in Step #1 is assumed to be an historical SID, and the SID returned in Step #2 is written to the new-SID field of the current line of *SIDfile-path*.

If **vob_sidwalk** was invoked with the **-execute** option, the SID used in Step #1 is replaced with the SID returned in Step #2.

Remapping Current SIDs When Moving a VOB to a New Domain

When you move a VOB to another domain, you must use **vob_sidwalk** to retrieve and store (in the VOB database) the new SIDs for all security principals who own objects in the VOB. The procedure is essentially the same whether you are moving the VOB to a host in another domain or simply moving a VOB server host to another domain and leaving the VOB on the host. *Moving a VOB to a Different Domain* on page 156 provides a detailed description of the procedure for remapping SIDs as part of a VOB move.

Reassigning Ownership to the VOB Owner

To reassign ownership of objects in the VOB by mapping all existing SIDs to the new SIDs of the VOB owner and group, use a command line like this one:

vob_sidwalk -unknown -execute *vob-tag SIDfile-path*

When invoked with the **-unknown** and **-execute** options, **vob_sidwalk** maps unresolvable user SIDs to the SID of the VOB owner and maps unresolvable group SIDs to the SID of the VOB's group.

NOTE: If you are using UCM, you may not want to reassign ownership with **-unknown**. Reassigning an open activity to the VOB owner makes it unusable by its creator (unless it was created by the VOB owner).

Resetting VOB Storage Directory Protections

If VOB storage directory ACLs have been damaged during a migration (or by any other event), you can use **vob_sidwalk** as shown here to correct the ACLs on the VOB storage directory and container files:

vob_sidwalk -recover_filesystem *vob-tag SIDfile-path*

When used with the **-recover_filesystem** option, **vob_sidwalk** also corrects the SIDs for the VOB's supplementary group list.

Administering Licenses

5

This chapter provides an introduction to license administration for Rational ClearCase LT.

5.1 Overview of ClearCase LT Licensing

ClearCase LT, like most Rational Software products, requires a license to run. ClearCase LT licensing is based on the FLEXlm license management tool from GLOBEtrotter Software, Inc. The ClearCase LT installation procedure installs FLEXlm client software on ClearCase LT client hosts.

Most end users configure their own systems for licensing using software that is included in ClearCase LT. If a license server is needed, an administrator typically configures a host as the FLEXlm license server, using software provided by Rational and GLOBEtrotter, then acquires and manages the licenses served by that host. If you have a FLEXlm license server at your site, especially if it is configured to serve licenses for other Rational Software products, you may be able to use it as the ClearCase LT license server host too.

Administering Licenses for Rational Software provides detailed descriptions of all aspects of Rational Common Licensing on Windows platforms; it includes all the information a user or administrator needs to install, configure, and troubleshoot licensing on Windows clients, as well as instructions for setting up a FLEXlm license server host on Windows.

The *Installation Guide* for Rational ClearCase LT includes detailed information about setting up a FLEXlm license server on UNIX platforms and adding licenses to a new or existing FLEXlm server on UNIX.

Floating and Node-Locked Licenses

FLEXlm supports two types of licenses: *floating* and *node-locked*.

- A floating license is installed on a license server host and permits a specified number of users to use all ClearCase LT features on any ClearCase LT client host served by that server.
- A node-locked license is installed on a single ClearCase LT client host and permits any user logged in to that host to use all ClearCase LT features on that host.

ClearCase LT on Windows platforms can use any node-locked Rational Suite license or a floating license specific to Clearcase LT. ClearCase LT on UNIX platforms can use only the floating license type.

To use ClearCase LT, a user must obtain a license, which grants the privilege to use ClearCase commands and data on any ClearCase LT host in the local area network. When a user runs a ClearCase client utility, such as **cleartool** or a GUI program, that utility attempts to obtain a license. On a client using a node-locked license, this attempt always succeeds as long as a valid license key is installed on the host. If the client is using a floating license, the attempt succeeds as long as the license server host can be reached over the network and enough floating licenses are available for all concurrent ClearCase LT users. After a user obtains a floating license, entering any ClearCase command renews the license. If the user doesn't enter a ClearCase command for a substantial period—by default, 30 minutes—another user can take the license.

Temporary, Permanent, and Term Licenses

Both floating and node-locked licenses can be either temporary, permanent, or subject to a term license agreement (TLA). Temporary licenses are shipped with ClearCase LT and allow users to begin using the product immediately while requests for permanent license keys are being processed. TLA licenses keys are a variation on permanent license keys. Contact your local Rational sales team for more information about TLA licenses.

ClearCase LT and Rational Suites Licenses

On Windows platforms, ClearCase LT can use a Rational Suite floating license or a dedicated ClearCase LT floating license. It can also use a node-locked license.

5.2 Installing and Configuring a License Server

If you use other Rational Software products, you may already have a FLEXlm license server host at your site. In this case, all you need to do is install ClearCase LT licenses on this host. You do not need to set up a dedicated license server host for ClearCase LT.

NOTE: If all of your ClearCase LT clients are running on Windows computers and using node-locked licenses, you do not need to install or configure a license server

A FLEXlm license server hosted on either a Windows NT or UNIX computer can serve licenses for ClearCase LT on any platform.

- If you need to set up a license server host on a computer running Windows NT, refer to Chapter 4 of *Administering Licenses for Rational Software*.
- If you need to set up a license server host on a computer running UNIX, refer to the *Installation Guide* for Rational ClearCase LT.

5.3 Installing and Configuring Windows Client Licenses

After a license server host has been set up, you must install ClearCase LT licenses on it before users can run ClearCase LT. Users and administrators have several options when installing and configuring licenses on ClearCase LT client hosts.

The License Key Administrator

On Windows platforms, ClearCase LT includes the Rational License Key Administrator program. This program manages configuration of a ClearCase LT client host to use either a floating or node-locked license and automates the procedure for obtaining a node-locked license from Rational.

5.4 Installing and Configuring UNIX Client Licenses

On UNIX platforms, ClearCase LT only supports the floating license type. Floating licenses require a license server host, and must be installed on that host by an administrator before they can be used. Refer to the *Installation Guide* for Rational ClearCase LT for information about how to set up a FLEXlm license server host and obtain permanent ClearCase LT licenses.

Administering VOBs

6 Understanding VOBs and VOB Storage

This chapter introduces the operational details, storage layout, and data model of versioned object bases, or VOBs. The **mkvob** reference page has additional information about VOBs.

6.1 Introduction to VOBs and VOB Administration

Any ClearCase development environment requires one or more VOBs. A VOB is the principal repository for ClearCase data and metadata. VOBs are a global resource; any VOB on the ClearCase LT server can be made accessible to any ClearCase user. Data can be centralized in a few VOBs, organized into multiple components at the VOB level, or distributed across a number of VOBs in other ways.

A typical VOB is created and populated with an initial collection of data by an administrator or project leader and accessed by many users. Significant administrative responsibilities associated with VOBs include the following:

- VOB creation and access control
- Importing data into a VOB from another source control system
- Backing up and recovering VOBs
- Relocating data from one VOB to another VOB
- Monitoring VOB integrity

- Moving, removing, and managing the storage used by VOBs

CAUTION: Moving a VOB is a complex procedure that requires several steps. You must follow these steps carefully or you risk losing or corrupting VOB data. See Chapter 11, *Moving VOBs*, for more information.

Rational ClearCase LT provides various VOB storage management tools as well as a job scheduler that can automate many routine VOB maintenance tasks. Chapter 16, *Managing Scheduled Jobs*, provides details on the job scheduler and related tools.

Types of VOBs

There are four types of VOBs. All have the same on-disk directory structure, and all have similar administrative requirements. You may not need all types at your site.

- Ordinary VOBs serve as the permanent repositories for data under ClearCase control. Most of the VOBs in a ClearCase network are ordinary VOBs. These VOBs are likely to consume more disk space and other server resources (memory, CPU cycles, and network bandwidth) than the other types of VOBs.
- Administrative VOBs allow centralized management of certain types of metadata that can be shared by other VOBs. Administrative VOBs are optional. See Chapter 8, *Using Administrative VOBs and Global Types* for more on this topic.
- UCM component VOBs, which function like ordinary VOBs but are treated by UCM as components used by a project.
- UCM project VOBs that store information about UCM artifacts such as projects, baselines, folders, and components. If your site does not use UCM, you will not need any UCM project VOBs. A UCM project VOB is the default administrative VOB for all of the UCM component VOBs that are included in the UCM projects stored in that VOB.

Access to VOB Data and Metadata

Users cannot access VOB data directly. They must access it using a view and refer to the VOB by its VOB-tag.

Views

Users access VOB data through views, which select specific versions of file and directory elements stored in the VOB and present them to a ClearCase LT user as part of the file system—a standard directory tree, whose top-level directory is called the *VOB root directory*. Users can also access VOB data and metadata using **cleartool** commands and ClearCase GUIs that do not need to use a view. Users cannot access VOB data directly.

Tags

ClearCase commands and utilities access a VOB by referring to its *VOB-tag*, which is a name with an associated global path that all ClearCase hosts can use to access the VOB. A VOB-tag is created whenever a VOB is created. All VOB tags are stored in the ClearCase registry, where they are accessible to all ClearCase LT hosts. Certain network configurations may require you to create additional tags or modify existing ones to ensure access to VOB data by all hosts.

VOB Server Processes

Access to VOB data is managed by several server processes that run on the ClearCase LT server:

- A **db_server** process handles requests from a single client program for access to any VOB on a ClearCase LT server.
- One or more **vobrpc_server** processes are started for each VOB on the ClearCase LT server. Each **vobrpc_server** process handles requests from one or more **view_server** processes.
- A **vob_server** process provides access to VOB storage pools. This process handles data-access requests from clients, forwarded to it by the **vobrpc_server**.
- A single **lockmgr** (lock manager) process manages transactions in the VOB databases of all VOBs on a ClearCase LT server

ClearCase Server Processes on page 6 provides additional information on these and other ClearCase server processes.

6.2 The VOB Storage Directory

Every VOB has a *VOB storage directory*. This directory resides on the ClearCase LT server.

CAUTION: A VOB storage directory and all of its contents are created and managed by ClearCase commands and services. Never use non-ClearCase commands or utilities to alter either the contents or the protection of the VOB storage directory or any of its files or subdirectories.

A VOB's directory structure is visible when you use native operating system utilities to list the VOB storage directory and its subdirectories and files, which include the following:

.pid	A one-line text file that lists the process ID of the VOB's associated **vob_server** process.
admin	A directory that contains administrative data related to the amount of disk space a VOB and its derived objects are using. Use **cleartool space –vob** to list this data.
vob_oid	A one-line text file that lists the VOB's object identified (OID) expressed as a universal unique identifier (UUID). This UUID is the same for all the replicas in a VOB family (ClearCase MultiSite). This value is also stored in the VOB's database; do not change it.
replica_uuid	A one-line text file that lists the replica UUID of this particular replica of the VOB. Different replicas created with ClearCase MultiSite have different identifiers.
.**identity**	On UNIX hosts, a subdirectory whose files establish the VOB's owner and group memberships. See *The .identity Directory* on page 63.
identity.sd	On Windows hosts, a binary data file that contains the security descriptor for the VOB storage directory. This security descriptor includes a Windows ACL that assigns all permissions (**full control**) to the VOB owner.
groups.sd	On Windows, security descriptors of the VOB's supplementary groups.
s	A subdirectory in which the VOB's source storage pools reside.
d	A subdirectory in which the VOB's derived object storage pools reside.
c	A subdirectory in which the VOB's cleartext storage pools reside.
db	A subdirectory containing the files that implement the VOB's embedded database. See *VOB Database* on page 62.
vob_server.conf	A text file read by the **vob_server** at startup. It contains the setting for deferred deletion of source containers; deferred deletion is activated if you have turned on semi-live backup. See the **vob_snapshot_setup** reference page for more information.
.**hostname**	A text file that records the name of the VOB's server host.
.msadm_acls	Stores MultiSite administration server's ACL

The sections that follow discuss the subdirectories of the VOB storage directory.

VOB Storage Pools

The **c**, **d**, and **s** subdirectories of the VOB storage directory contain the VOB's *storage pools*. The storage pools in these directories hold *data container* files, which store versions of elements and shared binaries. Depending on the *element type*, the versions of an element may be stored in separate data container files or may be combined into a single structured file that contains interleaved *deltas* (version-to-version differences).

Each VOB storage pool directory is created with a single subdirectory known as the default storage pool. There are three default pools.

s\sdft	Default source storage pool, for permanent storage of the contents of files under ClearCase control.
c\cdft	Default cleartext storage pool, for temporary storage of the **cleartext** versions currently in use (for example, reconstructed versions of **text_file** elements).
d\ddft	Default derived object storage pool, for storage of promoted/shared derived objects.

Source Storage Pools

Each source pool holds all the source data containers for a set of file elements. A source data container holds the contents of one or more versions of a file element. For example, a single source data container holds all the versions of an element of type **text_file**. The *type manager* program for this element type handles the task of generating individual cleartext versions from *deltas* in the data container. It also updates the source container with a new delta when a new version is checked in.

Source pools are accessed by **cleartool** commands such as **checkout** and **checkin**, as well as by any program that opens a file or directory in a dynamic view (whether or not the file or directory is checked out). If a cleartext version of the element is available, it is used. If it is not available, the request to access the file element causes the cleartext to be constructed and stored in the cleartext pool.

Cleartext Storage Pools

Each *cleartext pool* holds all the cleartext *data containers* for a set of file elements. A cleartext data container holds the contents of one version of an element. These pools are caches that accelerate access to element types (**text_file** and **compressed_text_file**) for which all versions are stored in a single data container.

For example, the first time a version of a **text_file** element is required, the **text_file_delta** type manager reconstructs the version from the element's source data container. The version is cached as a cleartext data container—an ordinary text file—located in a cleartext storage pool. On subsequent accesses, ClearCase looks first in the cleartext pool. A cache hit eliminates the need to access a source pool, thus reducing the load on that pool; it also eliminates the need for the type manager to reconstruct the requested version.

Cache hits are not guaranteed, because cleartext storage pools are periodically *scrubbed*. (See Chapter 10, *Administering VOB Storage*.) A cache miss forces the type manager to reconstruct the version.

Derived Object Storage Pools

Derived objects are part of the implementation of a ClearCase feature not used by ClearCase LT. ClearCase LT VOBs contain storage pools for these objects, but the objects are never created and the pools are never used.

VOB Database

Each VOB has its own database, which is managed by an embedded database management system (DBMS) and implemented as a set of files in the **db** subdirectory of the VOB storage directory. The database stores several kinds of data:

- Version-control information: elements, their branch structures, and their versions
- *Metadata* associated with the file system objects: version labels, attributes, and so on
- *Event records* and *configuration records*, which document ClearCase development activities
- *Type* objects, which are involved in the implementation of both the version-control structures and the metadata
- (UCM project VOBs only) UCM objects: folders, projects, streams, activities, components, baselines, and so on

The permanent contents of files that are under ClearCase control are stored in the source pools (**.s** and its subdirectories), not in the VOB database.

The **db** directory contains these files:

vob_db.dbd	A compiled database schema, used by embedded DBMS routines for database access. The schema describes the structure of the VOB database.
vob_db_schema_version	A schema version file, used by embedded DBMS routines to verify that the compiled schema file is at the expected revision level.
vob_db.d0*n* **vob_db.k0***n*	Files in which the database's contents are stored.
vista.*	Database control files and transaction logs.
db_dumper (UNIX)	Backup copy of *ccase-home-dir***/etc/db_dumper**. **reformatvob** invokes this copy of **db_dumper** only if it cannot invoke **/usr/atria/etc/dumpers/db_dumper.***num*, where *num* is the revision level of the VOB.
db_dumper (Windows)	A copy of *ccase-home-dir***\bin\db_dumper.exe**. This is an executable program, invoked during the **reformatvob** command's dump phase. Each VOB gets its own copy of **db_dumper** so that it can always dump itself to ASCII files. (Typically, it needs to be dumped after a newer release of ClearCase has already been installed on the host; with this strategy, the *ccase-home-dir***\bin\db_dumper** program in the newer release need not know about the older VOB database format.)
vob_db.str_file	Database string file that stores long strings.

Preserved Database Subdirectories

The root directory of any VOB that has been reformatted using the **reformatvob** command may include a preserved **db** directory that contains the VOB database as it existed before the reformat. **reformatvob** does its work by creating a new VOB database. By default, it preserves the old database by renaming it. Thus, a VOB storage directory may contain old (and usually unneeded) VOB database subdirectories, with names like **db.0318**. If **reformatvob** is interrupted, it may leave a partially reformatted database with the name **db.reformat**.

The .identity Directory

On UNIX platforms, a directory named **.identity** on UNIX records the VOB's ownership and group membership information. Access to this directory is restricted to the VOB owner and **root**.

The **.identity** directory contains these files:

uid	The owner of this file is the VOB owner.

gid	The group to which this file belongs is the VOB's principal group.
group.*nn*	Each additional file (if any) indicates by its group membership an additional group on the VOB's group list. In addition, the file's name identifies the group by numeric ID (**group.30**, **group.2**, and so on).

You can use the **cleartool** subcommands **describe** and **protectvob** to, respectively, display and if necessary change the VOB ownership information recorded in **.identity** or **identity.sd**.

NOTE: On Windows platforms, similar identity information is maintained in two files, **identity.sd** and **groups.sd**, in the VOB root directory. These files contain Windows security descriptors that describe the VOB's owner, principal group, and supplementary groups.

6.3 The lost+found Directory

Each VOB storage directory includes a special directory element named **lost+found**. ClearCase uses **lost+found** to hold elements that are no longer cataloged in any directory version in the VOB. This occurs when you do any of the following:

- Create new elements, and then cancel the checkout of directory in which they were created
- Delete the last reference to an element with the **rmname** command
- Delete the last reference to an element by deleting a directory version with the **rmver**, **rmbranch**, or **rmelem** command

When an element is moved to **lost+found**, it gets a name of the form

element_leaf_name.id-number

For example:

foo.41a00000bcaa11caacd0080069021c7

The **lost+found** directory has several unique properties:

- It cannot be checked out.
- Its contents can be modified even though it is not checked out.
- No branches can be created within it.

To move an element from the **lost+found** directory to another directory within the VOB, use the **cleartool mv** command. To move an element from the **lost+found** directory to another VOB, use the **cleartool relocate** command.

To conserve disk space, periodically clean up the **lost+found** directory:

- If you need an element in **lost+found**, catalog it in a versioned directory using **cleartool mv**.
- Use **cleartool rmelem** to remove unneeded elements.

NOTE: To access the **lost+found** directory from a snapshot view, you must include the directory in the view's load rules. Once **lost+found** and its elements have been loaded into the view, you can move or remove elements as needed.

6.4 VOB Datatypes

From the user's standpoint, a VOB contains file system objects and metadata. Some metadata is stored in the form of objects; other metadata is stored as records or annotations attached to objects. This section describes VOB datatypes.

Some of these datatypes are created automatically by ClearCase commands. Others must be created explicitly by users or (more often) administrators. Users never access metadata directly. Instead, they use **cleartool describe** and various ClearCase GUIs, most of which can retrieve metadata whether or not they are run in a view context.

The VOB Object

Each VOB database contains a VOB object that represents the VOB itself. The VOB object provides a handle for certain operations. For example:

- Listing event records of operations that affect the entire VOB (see the **lshistory** command). This includes creation and deletion of type objects, removal of elements, and so on.
- Placing a lock on the entire VOB (see the **lock** command).

File System Objects

A VOB database keeps track of users' file system objects using the following database objects:

File element	An object with a version tree, consisting of branches and versions. Each version of a file element has file system data: a sequence of bytes. Certain element types constrain the nature of the versions' file system data; for example, versions of **text_file** elements must contain text lines, not binary data.
Directory element	An object with a version tree, consisting of branches and versions. Each version of a directory element catalogs a set of file elements, directory elements (subdirectories), and VOB symbolic links. An extra name for an element that is already entered in some other directory version is termed a VOB hard link.
VOB symbolic link	An object that contains a text string. On UNIX systems, this string is interpreted by standard commands in the same way as an operating system symbolic link.

Link Counts for UNIX File System Objects

Link counts for UNIX file system objects, which are required when accessing VOB objects in a view on a UNIX computer, are stored in the VOB database and are reported by the UNIX **ls** command as follows:

Symbolic link	1
File element	1
File version	1
Directory element	2
Directory version	2 plus number of directory elements cataloged in that version
Branch	2 plus number of subbranches (each branch appears as a subdirectory in the extended-namespace representation of an element's version tree)

This scheme satisfies two UNIX rules:

- The link count is at least 1 for an object that has a name.
- The link count of a directory is 2 + *number-of-subdirectories*.

The scheme does not satisfy the rule that the link count should be the number of names the object has in the current namespace.

Type Objects

A type object is a prototype for one or more instances of type objects stored in a VOB database. If a type object exists, a user can create an instance of it by entering the appropriate command (for example, **cleartool mklabel** to create an instance of a label type object).

A VOB can store several kinds of type objects:

Type	Mnemonic	Description
Element type	**eltype**	Instances are elements.
Branch type	**brtype**	Instances are branches.
Hyperlink type	**hltype**	Instances are VOB hyperlinks (used to connect pairs of objects).
Trigger type	**trtype**	Instances are triggers.

In addition, VOBs can store type objects that can only be used to modify instances of other type objects:

Type	Mnemonic	Description
Label type	**lbtype**	Instances are labels that can be attached to any version object.
Attribute type	**attype**	Instances are attributes (name/value pairs) that can be attached to any instance of a type object.

The mnemonic associated with each type object can be used in an object-selector prefix to **cleartool** commands like **describe**. For example, to describe a branch type named **v4_patch**, use the **brtype** mnemonic as shown here:

cleartool describe brtype:v4_patch

Instances of Type Objects

After a type object is created, users can create any number of instances of the type. Creating an instance of a type object creates a reference to the type object. For example, attaching version label **BASELEVEL_4.2** to a particular version does not make a copy of the **BASELEVEL_4.2** type object. Instead, it establishes a connection between the version object and the label type object.

This scheme makes it easy to administer type objects and their instances. For example, renaming the label type object from **BASELEVEL_4.2** to **BL4.2** renames all its existing instances.

NOTE: Creating an instance does not make a copy of the type object, but in certain cases it does create a new object. For example, the **mkbranch** command creates a new branch object and creates a reference connecting the new branch object to an existing branch type object.

Element	Each file or directory element in a VOB is created by **mkelem** or **mkdir** as an instance of an existing element type in that VOB.
Branch	Each branch in an element is created by **mkbranch** as an instance of an existing branch type in that element's VOB.
Version label	The **mklabel** command annotates a version with a version label, by creating an instance of an existing label type.
Attribute	The **mkattr** command annotates a version, branch, element, VOB symbolic link, or hyperlink with an attribute, by creating an instance of an existing attribute type. Each instance of an attribute has a particular value—a string, an integer, and so on.
Hyperlink	The **mkhlink** command creates a hyperlink object, which is an instance of an existing hyperlink type. A typical hyperlink connects two objects, in the same VOB or in different VOBs.
Trigger	The **mktrigger** command creates a trigger object, which is an instance of an existing trigger type. The trigger may be attached to one or more elements.

Predefined and User-Defined Type Objects

Each VOB is created with a set of predefined type objects. Users can create additional type objects as needed using the **cleartool mkeltype** command. The online help for **mkeltype** lists the predefined element types for the current release of ClearCase. You can also use the ClearCase Administrator Console or the **cleartool lstype** command to list the type objects defined in a given VOB.

Scope of Type Objects

Each VOB has its own set of type objects, for use in creating instances of the types in that particular VOB. ClearCase also provides a global type facility that you can use to increase the scope of a type object from a single VOB to a group of VOBs. For more information about using global types, see Chapter 8, *Using Administrative VOBs and Global Types*.

Changing an Element's Type

You can use **chtype** to convert an element from one element type to another (for example, from **file** to **text_file**). Typically, you change an element's type to change the way its versions are stored. For example, versions of a *file* element are stored in separate data containers in a VOB

source pool. Converting the element to type **text_file** causes all its versions to be stored in a single data container, as a set of deltas (version-to-version differences); this saves disk space.

NOTE: All versions of an element must fit the new element type. For example, converting an element to type **text_file** fails if any of its versions contains binary data, rather than text. You cannot convert files to directories, and vice versa.

Event Records

Nearly every operation that modifies the VOB creates an event record in the VOB database. See the **events_ccase** reference page for more information.

The **vob_scrubber** utility deletes unneeded event records. By default, the scheduler runs **vob_scrubber** periodically. See the **schedule** reference page for information on describing and changing scheduled jobs.

7 Setting Up VOBs

Rational ClearCase LT automates the creation of several VOBs during the installation and setup of the ClearCase LT server. However, you may need to create additional VOBs later on. If so, follow the guidelines in this chapter.

7.1 ClearCase LT Server Configuration Guidelines

The ClearCase LT server must be appropriately configured for the number and size of VOBs it supports. Server configuration is critical to obtaining satisfactory performance.

NOTE: ClearCase has special requirements for the Windows domain membership of the ClearCase LT server and ClearCase LT users. See Chapter 4, *ClearCase and Windows Domains*.

The computer chosen as the ClearCase LT server must satisfy these requirements:

- **Main memory (RAM).** The minimum recommended main memory size is 128 MB or half the size of all the VOB databases the server will host, whichever is greater. To this amount, add 7 MB of memory per VOB, regardless of VOB database size, as well as 750 KB of memory for each view that the ClearCase LT server will support. Adequate physical memory is the most important factor in VOB performance; increasing the size of a VOB host's main memory is the easiest (and most cost-efficient) way to make VOB access faster and to increase the number of concurrent users without degrading performance. For the best performance, configure the ClearCase LT server with enough main memory to hold the entire VOB database of all the VOBs you plan to create.
- **Disk capacity**. A VOB database must fit in a single disk partition, and VOB databases tend to grow significantly as development proceeds and projects mature. We recommend a high-

performance disk subsystem, one with high rotational speed, low seek times, and a capacity of at least 10 GB.

If possible, use a RAID or similar disk subsystem that takes advantage of disk striping and mirroring. Mirrors are useful for backups, although there is a slight performance degradation associated with their use. However, striping helps overall performance and more than makes up for any degradation caused by mirroring. For more information, see *Maximize Disk Performance* on page 284.

- **Processing power.** A ClearCase LT server must have adequate CPU capacity. The definition of *adequate* in this context varies from one hardware architecture to another. With ClearCase and similar enterprise applications, server CPU capacity is a critical factor governing performance of client operations. Make the most of the available server CPU cycles by keeping nonessential processes—including ClearCase client tools and views—off the ClearCase LT server.

See also Chapter 19, *Improving ClearCase LT Server Performance*.

VOB Feature Levels

A feature level is an integer that defines the set of features that a VOB supports. Whenever a ClearCase LT release introduces features that require support in the VOB database, you must raise the feature level of a VOB before clients can take advantage of the new features when accessing data in that VOB.Every ClearCase release is associated with a feature level. See the *Release Notes* for ClearCase LT for information on which feature levels a release supports.

Displaying the Feature Level

To display the feature level of a VOB, use the Windows explorer or ClearCase Administration Console to display the properties of the VOB. The feature level is listed on the **Custom** tab. You can also use the **cleartool** command line.

cleartool describe vob:\dev

```
versioned object base "\dev"
  created 15-Aug-00.14:19:03 by Susan Goechs (susan.user@minuteman)
  VOB family feature level: 2
...
```

Changing the Feature Level

The **chflevel** command changes the feature level of a VOB. To raise the feature level of a VOB:

1. Log on to the ClearCase LT server.
2. Issue the **chflevel** command with the **–auto** option. The command lists each VOB served by the host. It then offers to raise the feature level of each VOB that is not already at the feature level corresponding to the release of ClearCase LT that is installed on the server.

VOB Schema Versions

Every VOB has a database schema version that denotes the format of the VOB database, and determines the types of data the VOB can accommodate. There are two schema versions:

- Schema version 53, which has been supported by every ClearCase release since 3.0, but is not supported by ClearCase LT
- Schema version 54, which provides better support for Windows security identifiers and is required when using ClearCase in an Active Directory environment. Schema version 54 also provides support for larger VOB database files (greater than 2 GB) on certain platforms.

All VOBs on a VOB server host must be formatted with the same schema version. When you install ClearCase server software on a host, you are prompted to select a schema version to be used by VOBs created on that host.

To change the schema version of an existing VOB to a higher-numbered schema version, use the **reformatvob** command. You cannot reformat a VOB to a lower-numbered schema version.

7.2 Planning for One or More Additional VOBs

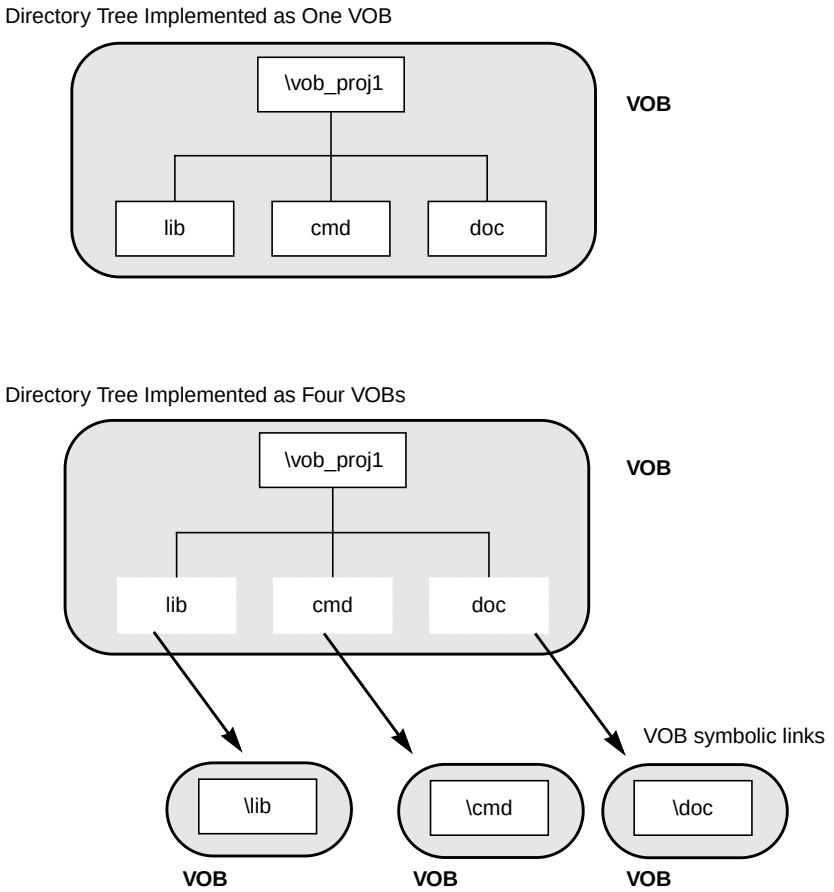
The ClearCase LT server is initially set up with three VOBs: a UCM Project VOB, a VOB for use by the ClearCase LT Tutorial, and a VOB to hold source elements. Your organization must decide how to allocate data to additional VOBs on the ClearCase LT server. These are the principal trade-offs to consider:

- Splitting data into several small VOBs increases your flexibility. It is easy to move an entire VOB to another disk. It is more difficult to split a large VOB into several new VOBs and move one or more of them to another disk.
- Typically, you have fewer performance bottlenecks when you use several smaller VOBs rather than one very large VOB.

To users, a VOB appears to be a single directory tree. Thus, it makes sense to consider whether to organize your development artifacts into logically separate trees and create a VOB for each one. If two projects do not share source files, you may want to place the sources in different VOBs. Typically, several or all projects share some header (**.h**) files. You may want to assign these shared sources to their own VOB.

In your VOB planning, keep in mind that you can make several VOBs appear to be a single directory tree, using *VOB symbolic links* (Figure 1).

Figure 1 Linking Multiple VOBs into a Single Directory Tree



NOTE: Be sure that the text of a VOB symbolic link is a relative pathname, not a full or absolute pathname. For example, the following command creates a VOB symbolic link that makes the VOB **\vob_lib** appear as a subdirectory named **lib** in the VOB **\vob_proj1**:

cleartool ln –s ..\vob_lib lib *(..\vob_lib, not \vob_lib)*
Link created: "lib".

cleartool ls lib
```
lib --> ../lib
```

dir
...
```
12/18/94 10:23a        <DIR>        lib
...
```

Relative pathnames ensure that the link is traversed correctly in all view contexts. See the **pathnames_ccase** reference page for more on this topic.

Planning for Release VOBs

VOBs are not for source files only; you can also use them to store product releases (binaries, configuration files, bitmaps, and so on). Such VOBs tend to grow quickly; we recommend that in a multiple-architecture environment, you store releases for different platforms in separate VOBs.

7.3 Modifying a ClearCase LT Server on UNIX

Each VOB is managed by several server processes, which run on the ClearCase LT server. Because these servers make significant demands on system resources, server configuration is an important contributor to application performance.

UNIX Kernel Resources

You may need to adjust the following kernel resources on a ClearCase LT server running UNIX:

- **Overall process table.** The operating system's process table must support 96 or more concurrent user processes. If more than three or four VOBs are to reside on the host, increase the size of the process table to at least 128.
- **Overall file descriptor table.** The size of the operating system's file descriptor table must be at least 700. If more than three or four VOBs are to reside on the host, increase the size of the file descriptor table.

You may also find it beneficial to adjust kernel resources for the ClearCase LT server after ClearCase LT has been up and running for some time. For more information, see *Tune Block Buffer Caches on UNIX* on page 284.

7.4 Creating VOB Storage Locations

ClearCase LT server installation creates a default VOB storage location on the ClearCase LT server. You can use this storage location to hold additional VOBs you create after the server setup process is finished. You may also create additional storage locations if you need them.

VOB storage locations should be created on a disk partition on the ClearCase LT server that has plenty of room for VOB database growth and is accessible to all ClearCase LT clients.

For Windows servers, the directory (folder) must be shared. You can control whether a directory is shared by using the Windows **net share** command or by setting the directory's sharing properties using Windows Explorer.

NOTE: By default, newly created shares have few access restrictions. If you modify the ACL of a share that corresponds to one or more VOB or view storage directories, you must preserve full access rights for all users who need access to the VOB or view. In addition, you must grant full access to the ClearCase administrators group.

See the **mkstgloc** reference page for more on this topic.

7.5 Creating a VOB

Follow these steps to plan and create each new VOB:

1. **Log on to the ClearCase LT server**. If possible, log on as a user who is a member of the same primary group as all other users who will need access to the VOB.

   NOTE: The identity of the user who creates a VOB (user ID, primary group, and umask on UNIX; user ID and primary group on Windows) is used to initialize VOB access permissions. If a VOB is created by a user whose primary group is not the same as the primary group of other users who must access the VOB, you will need to edit the VOB's supplementary group list before those users can access VOB data. See also the **protect** and **protectvob** reference pages.

2. **Choose a location for the VOB storage directory**. You can use an existing server storage location, create a new server storage location, or use any other directory that has the proper characteristics for good VOB storage as described in *Creating VOB Storage Locations* on page 76.

   - **Choose a VOB-tag.** A VOB-tag is a globally unique name by which all clients can refer to a VOB. A VOB-tag should indicate what the VOB contains or is used for. The tag must begin with a backslash or slash character and cannot include any spaces. Regardless of how a tag is created, ClearCase LT clients on UNIX will refer to the tag as /<*tag*>, while ClearCase LT clients on Windows will refer to it as \<*tag*>.

3. **Create the VOB.** This example explains how to create a VOB using the **cleartool** command line. You can also create VOBs using various GUI tools on UNIX or Windows.

   The following command creates a VOB on Windows with the VOB-tag **flex** whose storage is in the directory shared as **vobstore** on a ClearCase LT server named **pluto**, and then returns location and ownership information about the newly created VOB.

   cleartool mkvob –tag \flex \\pluto\vobstore\flex.vbs

   ```
   Host-local path: c:\vobstore\flex.vbs
   Global path: \\pluto\vobstore\flex.vbs

   VOB ownership:
     owner vobadm
     group dvt
   ```

   Whenever you create a VOB, you are prompted to enter a comment, which is stored in an event record (as the event "create versioned object base") in the new VOB's database.

   To create a Unified Change Management *Process VOB*, which can store UCM objects such as baselines, projects, and streams, add the **–ucmproject** option when you run **mkvob**.

In many cases, the VOB-creation process is now complete. The following sections describe special cases and optional adjustments you may want to make to the new VOB.

Linking a VOB to an Administrative VOB

If you want a VOB to use global types defined in an *administrative VOB*, you must link the client VOB to the administrative VOB. On Windows, if you use the VOB Creation Wizard to create the VOB, you can specify the administrative VOB at the time of VOB creation. To link a VOB to an

administrative VOB after you have created the client VOB, see *Linking a Client VOB to an Administrative VOB* on page 95.

Adjusting the VOB's Ownership Information

This section discusses changes that you may need to make to a new VOB's ownership information.

Access-control issues may arise when all prospective users of the VOB do not belong to the same group. (For detailed information on the topic of user identity and ClearCase access rights, see Chapter 3, *Understanding ClearCase Access Controls*.)

Case 1: One Group for All VOBs, Views, and Users

In organizations where all ClearCase users are members of the same group, all VOBs and views must also belong to the common group. A VOB or view belongs to the principal group of its creator and is fully accessible only to those users who are in the VOB creator's group.

The commands in Step #1–Step #3 on page 76 are sufficient to create a VOB in such a situation.

Case 2: Accommodating Multiple User Groups

If your organization has multiple user groups, there are special considerations when members of different groups share a VOB:

- Users can create an element only if their primary group is in the VOB's group list. If members of more than one group need to create elements, you must add the primary groups of all these users to the VOB's group list. Use the **cleartool protectvob** command.
- If members of more than one group need read access to an element, you must grant read access (and, for a directory, execute access) to **others** for that element. You must also grant read and execute access to **others** for all directories in the element's path, up to and including the VOB root directory. Use the **cleartool protect** command to change permissions for an element.
- Users cannot change an element (by checking out a version and checking in a new version), unless they belong to the element's group. The element's group does not have to be the user's primary group; it can be any group the user belongs to.

  Note that to create an element, you must be able to check out the containing directory. Thus, a user can create an element only if both of the following are true:

- The user's primary group is in the VOB's group list.
- Any of the user's groups is the group of the containing directory.

7.6 Coordinating the New VOB with Existing VOBs

A typical project involves multiple VOBs. To ensure that they all work together, you must coordinate their *type* objects: branch types, label types, and so on. For example, the following config spec can be used to create a view that selects the source versions for a previous release:

```
element * REL_3
```

For this strategy to succeed, all relevant VOBs must define version label **REL3**; that is, label type **REL3** must exist in each VOB:

- If you are using *global types*, create the label type with the **–global** option to **mklbtype**. The VOB that stores this global type becomes, by definition, an administrative VOB. Users in other VOBs then link to the administrative VOB by creating instances of the predefined hyperlink type **AdminVOB** that point to the administrative VOB. Thereafter, users in any VOB can create instances of label **REL3**.
- If you are not using global types, use **cptype** to copy type objects from one VOB to another.

7.7 Populating a VOB with Data

The new VOB is now ready to be populated with data. At this point, the VOB contains one directory element, the *VOB root directory*. (It also contains a **lost+found** directory. See the **mkvob** reference page for a discussion of this directory.) ClearCase LT includes several utilities for exporting data from other configuration management systems, as well as an Import Wizard that automates the process of moving legacy data into a VOB when the ClearCase LT server is set up. The Import Wizard is designed to be used just once as part of server setup. This section presents examples of how the other data import utilities can be used.

Importing Data into a UCM Project

You cannot run any of the **clearimport** procedures described in this section in a UCM view. To import data into a UCM project, you must first import the data in a non-UCM view, and then follow the procedures for setting up a project described in *Managing Software Projects*.

Example: Importing RCS Data

A simple conversion scenario illustrates the migration of RCS sources to ClearCase. In this example, we use an entire UNIX RCS source tree to populate a newly created VOB. The root of the RCS tree is **/usr/libpub** on a host where the empty VOB has already been activated, at **/proj/libpub**.

Creating the Data File

1. **Go to the source data.** Change to the root directory of the existing RCS source tree:

   cd /usr/libpub

2. **Run the export utility.** Use **clearexport_rcs** to create the data file and place descriptions of RCS files (**,v** files) in it:

   clearexport_rcs

   ```
   Exporting element "./Makefile,v" ...
   Extracting element history ...
   .
   Completed.
   Exporting element ...
   Creating element ...
   Element "./Makefile" completed.
     .
     .
   Element "./lineseq.c" completed.
   Creating datafile ./cvt_data ...
   ```

   The data file, **cvt_data**, is created in the current directory.

Running clearimport

3. **Use any view. clearimport** ignores the config spec.

4. **Go to the target VOB.** Change to the root directory of the newly created **libpub** VOB—that is, the directory specified by its *VOB-tag*:

   cd /libpub

5. **Run clearimport.** Run **clearimport** on the data file, **cvt_data**, to populate the **libpub** VOB:

   clearimport /usr/libpub/cvt_data

```
Converting files from /usr/libpub to .
Created element "././Makefile" (type "text_file").
Changed protection on "././Makefile".

Making version of ././Makefile

Checked out "././Makefile" from version "/main/0".
Comment for all listed objects:
Checked in "././Makefile" version "/main/1".
  .
  .
```

   There is no need to check out or check in the VOB's root directory element; this is handled automatically. If problems cause **clearimport** to terminate prematurely, you can fix the problems and run **clearimport** again.

Example: Importing PVCS Data

A simple conversion scenario illustrates the migration of PVCS sources to ClearCase. In this example, entire Windows PVCS source tree is used to populate a newly created VOB. The PVCS tree is located at **c:\libpub**, on a host where the empty VOB has been activated, at **\vob_libpub**.

Creating the Data File

1. **Go to the source data.** Change to the directory of the existing PVCS source tree:

   `c:\>` **cd libpub**

2. **Run the export utility.** Use **clearexport_pvcs** to create the data file and place descriptions of PVCS files in it:

```
c:\libpub> clearexport_pvcs
Exporting element ".\makefile" ...
Extracting element history ...
.
Completed.
Exporting element ...
Creating element ...
Element ".\makefile" completed.
  .
  .
Element ".\lineseq.c" completed.
Creating datafile .\cvt_data ...
```

The data file, **cvt_data**, is created in the current directory.

Running the Conversion Scripts

3. **Use any view. clearimport** ignores the config spec.

   NOTE: You cannot run **clearimport** in a UCM view.

4. **Go to the target VOB.** Change to the root directory of the newly created **\vob_libpub** VOB—that is, to the directory specified by its *VOB-tag*:

5. **Run clearimport.** Run **clearimport** on the datafile, **cvt_data**, to populate the **\vob_libpub** VOB:

   ```
   z:\vob_libpub> clearimport c:\libpub\cvt_data
   Converting files from c:\libpub to .
   Created element ".\.\makefile" (type "text_file").
   Changed protection on ".\.\makefile".

   Making version of .\.\makefile

   Checked out ".\.\makefile" from version "\main\0".
   Comment for all listed objects:
   Checked in ".\.\makefile" version "\main\1".
     .
     .
   ```

   There is no need to check out or check in the VOB's root directory element; this is handled automatically. If problems cause **clearimport** to terminate prematurely, you can fix the problems and run **clearimport** again.

For more information on the ClearCase exporters and importer, see the **clearexport_ccase**, **clearexport_pvcs**, **clearexport_rcs**, **clearexport_sccs**, **clearexport_ssafe**, **clearexport_cvs**, **clearimport** and **clearfsimport** reference pages.

7.8 Converting a SourceSafe Configuration

This section uses a sample configuration to show how to use the **clearexport_ssafe** and **clearimport** utilities to migrate a set of source files from the SourceSafe configuration management tool to ClearCase control. This section describes the sample SourceSafe configuration, shows how to convert it to a ClearCase configuration, and examines how the conversion process treats various SourceSafe features.

This example assumes that you have created a VOB named **\payroll** to hold the imported data.

Overview of Payroll Configuration

The sample payroll configuration contains source files used to build a payroll application.

Figure 2 Sample SourceSafe Payroll Configuration

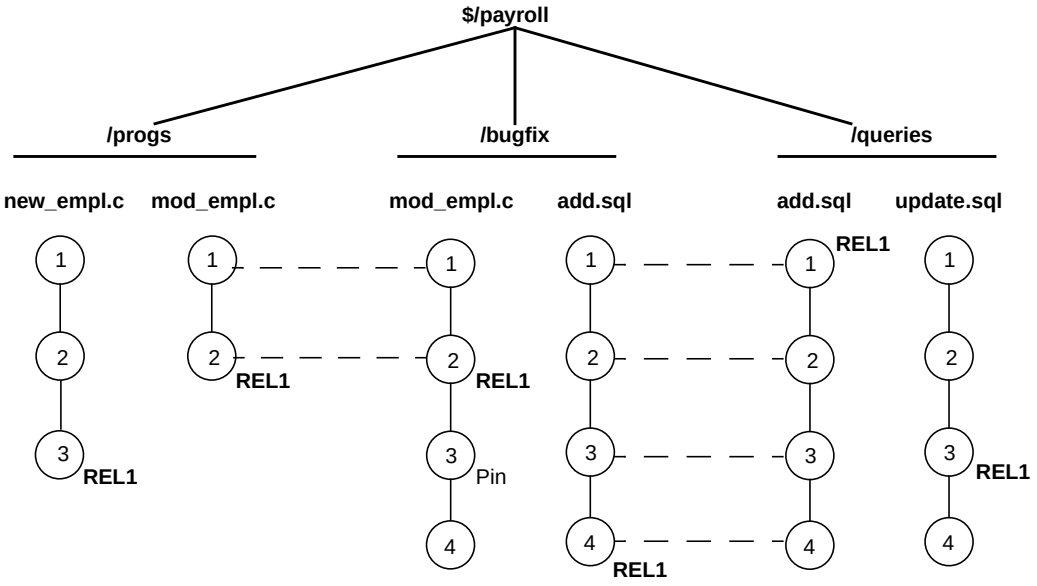


Figure 2 shows the configuration hierarchy in SourceSafe, which consists of these projects:

- **$/payroll** project
- **/progs** subproject for storing C programs
- **/bugfix** subproject where developers store source files while fixing bugs
- **/queries** subproject for storing SQL database access files

The payroll configuration contains the following objects: shares, branches, labels, and pins. **clearexport_ssafe** recognizes and handles some of them during the export phase of the conversion process.

Shares

In SourceSafe, shares are similar to hard links. In the example, the **$/payroll/bugfix/add.sql** and **$/payroll/queries/add.sql** files are SourceSafe shares. In SourceSafe, any changes that you make to one of these shares appears in the other file.

Branches

In SourceSafe, you must create a share before you can create a branch. The **$payroll/progs/mod_empl.c** and **$payroll/bugfix/mod_empl.c** files are shared for versions 1 and 2. At version 3, the **$payroll/bugfix/mod_empl.c** file forms its own branch.

Labels

Labels identify a particular version or a project. The Payroll configuration uses the **REL1** label to identify the versions of source files used to build the first release of the Payroll application.

Pins

By default, SourceSafe uses the latest version of files. Pins allow you to direct SourceSafe to use a version other than the latest. In the Payroll configuration, version three of **$payroll/bugfix/mod_empl.c** is pinned.

Setting Your Environment

Before you start the conversion process, make sure that certain environment variables and your SourceSafe current project are set correctly.

Setting Environment Variables

Verify that the PATH environment variable includes the location of the SourceSafe executable file, **ss.exe** in SourceSafe Version 5.0, and the **ssafeget.bat** batch file. For example, if **ss.exe** is in **C:\ss5.0\win32**, and **ssafeget.bat** is in **C:\atria\bin**, set the PATH environment variable, as follows:

```
set path=c:\ss5.0\win32;c:\atria\bin;%PATH%
```

Set the TMP environment variable to a location where **clearimport** can safely store temporary files. For example:

```
set TMP=c:\temp
```

Setting Your SourceSafe Current Project

The **clearexport_ssafe** utility exports data for files and directories in your SourceSafe current project. Before running **clearexport_ssafe**, verify that your SourceSafe current project is set so that **clearexport_ssafe** exports the desired files and directories. For example:

ss cp
```
Current Project is $/payroll/bugfix
```

ss cp ..
```
Current Project is $/payroll
```

ss dir
```
$/payroll
$bugfix
$progs
$queries
3 item(s)
```

Running clearexport_ssafe

The **clearexport_ssafe** utility reads the files and subprojects in your SourceSafe current project and generates a data file, which **clearimport** uses to create equivalent ClearCase elements. By default, **clearexport_ssafe** names the data file **cvt_data** and stores it in your current working directory. You can specify a different name and storage location for the data file by using the **–o** option.

Using the Recursive Option

By default **clearexport_ssafe** exports the files and subprojects in the SourceSafe current project, but it does not export any files contained in subprojects. For example, with the SourceSafe current project set to **$/payroll**, **clearexport_ssafe** exports subprojects **/progs**, **/bugfix**, and **/queries** but does not export any of the files in those subprojects. To export all files in all subprojects, specify the **–r** option.

Example

In the following example, the SourceSafe current project is **$/payroll**. Because the command line specifies the **–r** option, **clearexport_ssafe** exports all files in the **/progs**, **/bugfix**, and **/queries** subprojects. The **–o** option directs **clearexport_ssafe** to store the output in a data file named **paycvt** in the **c:\datafiles** directory.

ss cd

```
Current project is $/payroll
```

clearexport_ssafe –r –o c:\datafiles\paycvt

```
VOB directory element “.”.
VOB directory element “bugfix”.
Converting element “bugfix\add.sql” ...
Extracting element history ...
....
Completed.
Converting element ...
Creating element ...
Element “bugfix/add.sql” completed.
Converting element “bugfix\mod_empl.c;3” ...
Extracting element history ...
...
Completed.
Converting element ...
Creating element ...
Element “bugfix/mod_empl.c” completed.
VOB directory element “progs”.
Converting element “progs\mod_empl.c” ...
Extracting element history ...
...
Completed.
Converting element ...
Creating element ...
Element “progs/mod_empl.c” completed.
Converting element “progs\new_empl.c” ...
Extracting element history ...
...
Completed.
Converting element ...
Creating element ...
Element “progs/new_empl.c” completed.
```

```
VOB directory element “queries”.
Converting element “queries\add.sql” ...
Extracting element history ...
...
Completed.
Converting element ...
Element “queries/add.sql” completed.
Converting element “queries\update.sql” ...
Extracting element history ...
...
Completed.
Converting element ...
Creating element ...
Element “queries/update.sql” completed.
Creating script file c:\datafiles\paycvt ...
```

Running clearimport

Use any view. **clearimport** ignores the config spec.**NOTE:** You cannot run **clearimport** in a UCM view.

Set your working directory to the VOB directory in which you plan to import the configuration. You can run **clearimport** from a location other than the VOB directory by specifying the VOB directory with the **–d** option. You must specify the pathname of the data file created by **clearexport_ssafe**.

clearimport c:\datafiles\paycvt

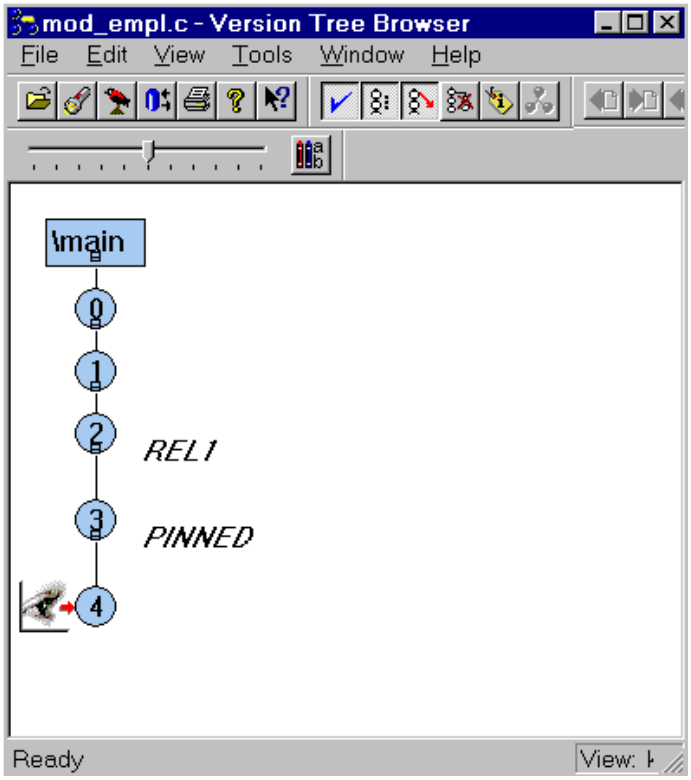
```
Validating label types.
Validating directories and symbolic links.
Validating elements.
Creating element “.\bugfix/add.sql”.
     version “1”
     version “2”
     version “3”
     version “4”
Creating element “.\bugfix/mod_empl.c”.
     version “1”
     version “2”
     version “3”
     version “4”
Creating element “.\progs/mod_empl.c”.
     version “1”
     version “2”
```

```
Creating element ".\progs/new_empl.c".
     version "1"
     version "2"
     version "3"
Creating element ".\queries/add.sql".
     version "1"
     version "2"
     version "3"
     version "4"
Creating element ".\queries/update.sql".
     version "1"
     version "2"
     version "3"
     version "4"
Closing directories.
```

Examining the Results

After **clearimport** finishes populating the VOB, examine the version trees of the new ClearCase elements to verify that **clearexport_ssafe** and **clearimport** converted the SourceSafe configuration as you expected. In Windows Explorer, open the VOB folder and select an element. Then, click **File** > **ClearCase** > **Version Tree**. Figure 3 shows the version tree for the **\bugfix\mod_empl.c** element.

Figure 3 ClearCase Version Tree of \bugfix\mod_empl.c Element



Version Numbers

As it does with all elements, ClearCase adds a version 0 as a placeholder at the start of the version tree.

Labels

The conversion process maps SourceSafe labels directly to ClearCase labels, so version 2 of **mod_empl.c** has the **REL1** label as it does in the SourceSafe configuration.

Branches

In the SourceSafe Payroll configuration, at version 3, the **$payroll/bugfix/mod_empl.c** file forms its own branch. **clearexport_ssafe** does not convert SourceSafe branches to ClearCase branches. Instead, **clearexport_ssafe** creates separate elements. In this case, it creates versions 1 and 2 of the **mod_empl.c** element in the **\progs** directory, and versions 1 through 4 of **mod_empl.c** in the **\bugfix** directory.

Pins

ClearCase does not have a feature equivalent to a SourceSafe pin. Sometimes, pins perform the same function as labels; therefore, the conversion process maps pins to labels. In the SourceSafe Payroll configuration, version 3 of **mod_empl.c** is pinned. The conversion process applies a label with the name **PINNED**.

Shares

ClearCase does not have a feature equivalent to a SourceSafe share. **clearexport_ssafe** does not preserve shares as hard links during conversion. Instead, shares become separate elements.

8 Using Administrative VOBs and Global Types

This chapter describes how to use administrative VOBs and global types.

8.1 Overview of Global Types

An administrative VOB stores definitions of global types and makes them available to all client VOBs that link to the administrative VOB. You can use global types to increase the scope of a type object from a single VOB to a group of VOBs. For example, you can have all VOBs in your local area network use the same set of type objects by linking them to the same administrative VOB. You can create any number of global type objects in one or more administrative VOBs.

A client VOB uses global types from an administrative VOB as follows:

1. A developer attempts to create an instance of a type, but there is no type object in the client VOB. For example, a developer wants to create a **v3_bugfix** branch in a particular element, but branch type **v3_bugfix** does not exist in the client VOB.
2. Rational ClearCase LT determines which administrative VOB is associated with the client VOB by scanning for an **AdminVOB** hyperlink in the client VOB.
3. If it finds a global type (branch type **v3_bugfix** in this example) in the administrative VOB, ClearCase LT creates a copy of the type object in the client VOB. This capability is called *auto-make-type*.

4. ClearCase LT uses the *local copy* of the type object to create the instance that the developer wants. (In this case, it creates the **v3_bugfix** branch for the desired element in the client VOB.)

A local copy is linked to the global type object by a **GlobalDefinition** hyperlink. Operations performed on a global type affect all its local copies. Similarly, operations performed on a local copy affect the global type, and by extension, all other local copies.

8.2 Why Use Global Types?

Using global types offers the following benefits:

- **Automatic creation of types.** Local metadata types are created from global types as needed at client VOBs when you create instances of the types (with the **mkattr**, **mkbranch**, **mkelem**, **mkhlink**, and **mklabel** commands).
- **Centralized administration.** Groups of VOBs can share metadata type definitions. You can control global type objects and their local copies from the administrative VOB.

8.3 Working with Administrative VOBs

The following sections describe how to work with administrative VOBs.

Creating an Administrative VOB

To create an administrative VOB, follow the procedures described in *Creating a VOB* on page 76.

NOTE: An administrative VOB can also store elements.

To put a VOB into use as an administrative VOB:

1. Link client VOBs to it. See *Linking a Client VOB to an Administrative VOB* on page 95.
2. Create global types in it. See *Creating a Global Type* on page 100.

Linking a Client VOB to an Administrative VOB

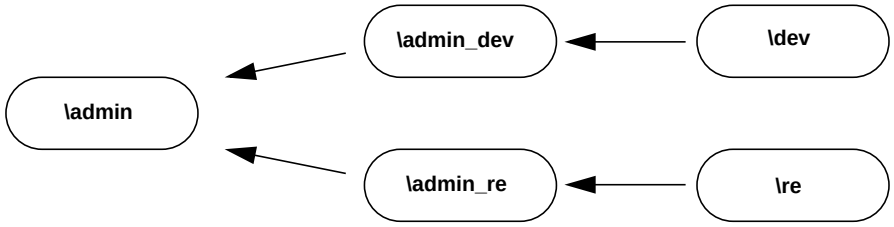
To associate a client VOB with an administrative VOB, create a hyperlink of type **AdminVOB** from the client VOB to the administrative VOB.

cleartool mkhlink –c "link to admin VOB" AdminVOB vob:\dev vob:\admin_dev
```
Created hyperlink "AdminVOB@40@\dev".
```

Administrative VOB Hierarchies

A client VOB can have only one administrative VOB. However, you can create administrative VOB hierarchies, in which a client VOB is linked to an administrative VOB, which is linked to another administrative VOB. (Circular relationships are prohibited.) For example, you can create an administrative VOB that contains global types used by all VOBs at your site, plus two other administrative VOBs, each containing global types specific to the needs of particular teams. Figure 4 illustrates this example.

Figure 4 Administrative VOB Hierarchy



You can add an administrative VOB to the middle of a hierarchy by removing an existing **AdminVOB** hyperlink and adding two new ones. This operation does not disrupt existing type definitions, because the hyperlink between a local copy and its associated global type remains intact.

To add an administrative VOB to a hierarchy:

1. Remove the **AdminVOB** hyperlink at the point where you want to add the new administrative VOB. For example, if you want to add an administrative VOB between **\admin** and **\admin_re**:

cleartool describe –l vob:\admin

```
...
  Hyperlinks:
    AdminVOB@40@\admin_re <- vob:\admin_re
```

cleartool rmhlink –c "insert admin VOB" AdminVOB@40@\admin_re

```
Removed hyperlink "AdminVOB@40@\admin_re"
```

2. Create the new administrative VOB.

3. Associate the new administrative VOB with its higher level administrative VOB. For example, if the new administrative VOB is **\admin_lb**:

   cleartool mkhlink –c "link admin_lb to admin" AdminVOB vob:\admin_lb vob:\admin

   ```
   Created hyperlink "AdminVOB@40@\admin_lb".
   ```

4. Associate the client VOB with the new administrative VOB.

   cleartool mkhlink –c "link re to admin_lb" AdminVOB vob:\re vob:\admin_lb

   ```
   Created hyperlink "AdminVOB@40@\re".
   ```

Listing an AdminVOB Hyperlink

Use the ClearCase Administration Console or the **cleartool describe** command. The **describe** command shows the hyperlink that associates a client VOB with an administrative VOB. The hyperlink always points from the client VOB to the administrative VOB. The following examples show **AdminVOB** hyperlinks.

- Client VOB **\dev**, whose administrative VOB is **\admin_dev**

  cleartool describe vob:\dev

  ```
  versioned object base "\dev"
  ...
    Hyperlinks:
      AdminVOB -> vob:\admin_dev
  ```

- Administrative VOB **\admin** with two client VOBs

cleartool describe vob:\admin

```
versioned object base "\admin"
...
  Hyperlinks:
    AdminVOB <- vob:\admin_dev
    AdminVOB <- vob:\admin_re
```

- A VOB that is both a client VOB and an administrative VOB

cleartool describe vob:\admin_dev

```
versioned object base "\admin_dev"
...
  Hyperlinks:
    AdminVOB -> vob:\admin
    AdminVOB <- vob:\dev
```

To display the hyperlink ID, use **describe –long**. The hyperlink ID includes the VOB-tag of the VOB in which the hyperlink was created. For example:

cleartool describe –long vob:\admin_dev

```
...
  Hyperlinks:
    AdminVOB@40@\admin_dev -> vob:\admin
    AdminVOB@40@\dev <- vob:\dev
```

Restrictions on Administrative and Client VOBs

The following restrictions apply to administrative and client VOBs:

- If you try to link a client VOB to an administrative VOB and any global type definitions in the administrative VOB would be eclipsed by ordinary types in the client VOB, the operation fails unless the **-acquire** option has been used. See the **mkhlink** reference page for more information.
- A VOB can have only one **AdminVOB** hyperlink pointing from a client VOB to an administrative VOB. The **mkhlink** command prevents the creation of a second **AdminVOB** hyperlink to any administrative VOB.

If an Administrative VOB Becomes Unavailable

If an administrative VOB becomes unavailable to a client VOB for any reason, attempts at the client VOB to create instances based on a now-inaccessible global type definition produce the following error:

`cleartool: Error: Unable to access administrative VOB "`*adminVOB*`" of` *clientVOB*

In addition, the output of **cleartool describe** for the client VOB may not show the administrative VOB.

Breaking a Link Between a Client VOB and an Administrative VOB

You can convert a client VOB to a regular VOB by removing the **AdminVOB** hyperlink and all **GlobalDefinition** hyperlinks between the client VOB and its administrative VOB. You must remove all such hyperlinks to sever the connection between a VOB and its administrative VOB. The following sections describe how to remove the hyperlinks using the command line. You can also use the ClearCase Administration Console.

Removing the AdminVOB Hyperlink

To remove the **AdminVOB** hyperlink between the client VOB and the administrative VOB:

1. Determine the name and ID of the **AdminVOB** hyperlink:

   cleartool describe vob:\dev
   ```
   versioned object base "\dev"
   ...
     Hyperlinks:
       AdminVOB@40@\dev -> vob:\admin_dev
   ```

2. Remove the hyperlink with the **rmhlink** command:

   cleartool rmhlink AdminVOB@40@\dev
   ```
   Removed hyperlink "AdminVOB@40@\dev".
   ```

Removing All GlobalDefinition Hyperlinks

To remove all **GlobalDefinition** hyperlinks that connect local copies in the client VOB to global types in the administrative VOB:

1. Determine the names of all local copies:

   cleartool lstype –local –fmt "%n\t%[type_scope]p\n" –kind attype –invob \dev
   ```
   Tested         local copy
   Feature Level  ordinary
   ...
   ```

   cleartool lstype –local –fmt "%n\t%[type_scope]p\n" –kind brtype –invob \dev
   ```
   ...
   ```

   cleartool lstype –local –fmt "%n\t%[type_scope]p\n" –kind eltype –invob \dev
   ```
   ...
   ```

   cleartool lstype –local –fmt "%n\t%[type_scope]p\n" –kind hltype –invob \dev
   ```
   ...
   ```

   cleartool lstype –local –fmt "%n\t%[type_scope]p\n" –kind lbtype –invob \dev
   ```
   ...
   ```

2. For each local copy, determine the name and ID of the hyperlink linking the local copy to its global type. For example:

   cleartool describe –local –long –ahlink GlobalDefinition attype:Tested
   ```
   Tested
     Hyperlinks:
       GlobalDefinition@58@\dev -> attype:Tested@\admin_dev
   ```

3. Remove each hyperlink with the **rmhlink** command:

   cleartool rmhlink GlobalDefinition@58@\dev
   ```
   Removed hyperlink "GlobalDefinition@58@\dev".
   ```

Removing an Administrative VOB

Remove an administrative VOB with the **rmvob** command or by using the ClearCase Administration Console. When you remove an administrative VOB:

- All **AdminVOB** hyperlinks connecting to the deleted VOB are removed.
- All **GlobalDefinition** hyperlinks connecting to global types in the deleted VOB are removed.
- All local copies in the deleted VOB's client VOBs are converted to ordinary types.

Fixing Global Type Problems After Restoring a VOB from Backup

Any time you restore an administrative VOB or any of its clients from backup, there is the possibility of a mismatch in global type information (**AdminVOB** hyperlinks) if the VOBs were backed up on different schedules. For example, if an administrative VOB is backed up before a new global type is created, its client VOBs will not be able to use instances of the new global type after that backup is restored. To prevent problems caused by this type of mismatch:

- After you restore an administrative VOB from backup, you should check for and fix any broken hyperlinks in that VOB and all of its client VOBs.
- After you restore a client VOB from backup, you must check for and clean up any broken hyperlinks from that VOB to its administrative VOB.

To remove broken hyperlinks, use **checkvob –hlink**. For more information, see *Checking Hyperlinks* on page 176 and the **checkvob** reference page.

8.4 Working with Global Types

In general, all operations on a global type or a local copy of a global type apply to the global type and all its local copies. ClearCase LT prevents attempts to eclipse global types (that is, it prevents creating an ordinary type with the same name as a global type).

Examples in this section use the **cleartool** command line. You can also use the Metadata subnode of a VOB node in the ClearCase Administration Console.

Creating a Global Type

The **cleartool** commands **mkattype**, **mkbrtype**, **mkeltype**, **mkhltype**, and **mklbtype** include the **–global** option, which creates a global type object in the current VOB. Client VOBs linked to this VOB can use the global types.

Local copies of global types are created in client VOBs only when a user creates an instance of the type in the client VOB.

The following command creates a global label type in VOB **\admin**:

cleartool mklbtype –c "final label for REL6" –global REL6@\admin
```
Created label type "REL6".
```

You cannot create a global type if any client VOB contains types with the same name. When you create a new global type, you can check for types with the same name in client VOBs. If the types are identical (except for comments and locks, which can be different), the creation operation converts the existing types to local copies of the global type and changes their comments to match. Use the **–acquire** option with the **mk**type** command to check for and acquire identical ordinary types.

For example:

cleartool describe –fmt "%n\t%[type_scope]p\n" lbtype:V3.2@\dev
```
V3.2    ordinary
```

cleartool mklbtype –c "Release 3.2" –global –acquire V3.2@\admin
```
Created label type "V3.2".
```

cleartool describe –local –fmt "%n\t%[type_scope]p\n" lbtype:V3.2@\dev
```
V3.2    local copy
```

If the types are not identical, the operation prints a warning and fails. If a type is locked, it is reported as not acquirable, and the operation continues with other types. To correct this problem, remove the lock and enter a new **mk**type** command with the **–replace –global –acquire** options, or use the **checkvob –global** command.

Use the **–replace –global –acquire** options either to acquire and replace eclipsing ordinary types that were created after the global type was first created or to convert an ordinary type in an administrative VOB to a global type.

Auto-Make-Type Operations

In general, when you create an instance of a global type in a client VOB, ClearCase LT creates a local copy of the global type in the client VOB. Specifically:

- When you create attributes, branches, elements, hyperlinks, or labels, ClearCase LT creates a local copy of the global type.
- When a checkout operation causes the creation of a branch (auto-make-branch), ClearCase LT creates a local copy of the global branch type.
- When you attach an attribute or a hyperlink to a local copy of a type, ClearCase LT creates the local copy if it does not already exist.

➤ If the global type has supertypes, ClearCase LT creates local copies of the supertypes and fires any **mk**type** triggers associated with them.

In addition, ClearCase LT sets the permissions and ownership of the local copy to be the same as those of the global type.

EXCEPTION: When you create a trigger type and specify a global type as an argument to a built-in action (the arguments **–mklabel**, **–mkattr**, and so on), ClearCase LT does not create a local copy of the global type. This behavior preserves the rule that built-in actions cannot cause cascading triggers. Therefore, if you create the trigger in a client VOB that does not contain a local copy of the global type, the **mktrtype** command fails.

The following example shows the creation of an instance of a global label type. The output of the command includes the VOB-tag of the administrative VOB.

cleartool mklabel –c "Release 6" REL6 \dev\file.c

```
Automatically created label type "REL6" from global definition in VOB
"\admin".
Created label "REL6" on "\dev\file.c" version "/main/rel6_main/31".
```

Describing Global Types

By default, the **describe** command shows the description of the global type for the object selector you specify. You can enter the command in the context of a client VOB even if the client VOB does not contain a local copy of the type. To describe the local copy, use the **–local** option.

The following command describes a global type:

cleartool describe –long lbtype:REL6@\dev

```
label type "REL6"
  created 28-Jul-99.14:00:26 by Suzanne Gets (smg.user@neon)
  "final label for REL6"
  owner: smg
  group: user
  scope: global
  constraint: one version per element
  Hyperlinks:
    GlobalDefinition@47@\dev <- lbtype:REL6@\dev
```

The following command describes the local copy of a global type:

cleartool describe –local –long lbtype:REL6@\dev

```
label type "REL6"
  created 28-Jul-99.14:23:45 by Suzanne Gets (smg.user@neon)
  "Automatically created label type from global definition in VOB "\admin"."
  owner: smg
  group: user
  scope: this VOB (local copy of global type)
  constraint: one version per element
  Hyperlinks:
    GlobalDefinition@47@\dev -> lbtype:REL6@\admin
```

If you specify **–local** and no local copy exists, **describe** prints an error:

cleartool describe –local lbtype:NOLOCAL@\dev

```
cleartool: Error: Not a vob object: "lbtype:NOLOCAL@\dev".
```

Listing Global Types

By default, the **lstype** command lists global types associated with local copies, even if you specify the client VOB in the **–invob** option. The output also includes global types from all administrative VOBs above this VOB in the administrative VOB hierarchy, even if the client VOB does not contain local copies of the type. To show client information only, use the **–local** option.

The following command lists all label types in the client VOB, including all global types from administrative VOBs in the hierarchy:

cleartool lstype –fmt "%n\t%[type_scope]p\n" –kind lbtype –invob \dev

```
BACKSTOP        ordinary
CHECKEDOUT      ordinary
LABEL1  global
LATEST  ordinary
REL6    global
```

The following command lists ordinary types and local copies of global types (if the specified VOB is an administrative VOB, global types are also listed):

cleartool lstype –local –fmt "%n\t%[type_scope]p\n" –kind lbtype –invob \dev

```
BACKSTOP        ordinary
CHECKEDOUT      ordinary
LATEST  ordinary
REL6    local copy
```

Listing History of a Global Type

By default, the **lshistory** command lists the history of the global type for the object selector you specify, even if there is no local copy of the type in the client VOB. To list the history of a local copy, use the **–local** option. Specifying **–all** or **–avobs** implicitly specifies **–local**.

The following command lists the history of a global label type:

cleartool lshistory –minor lbtype:REL6@\dev

```
28-Jul.14:00  smg        make hyperlink "GlobalDefinition" on label type
"REL6"
"Attached hyperlink "GlobalDefinition@47@\dev".
Automatically created label type from global definition in VOB "\admin"."
28-Jul.13:57  smg       create label type "REL6"
```

The following command lists the history of a local copy of a global label type:

cleartool lshistory –local –minor lbtype:REL6@\dev

```
28-Jul.14:00  smg        make hyperlink "GlobalDefinition" on label type
"REL6"
"Attached hyperlink "GlobalDefinition@47@\dev".
Automatically created label type from global definition in VOB "\admin"."
28-Jul.14:00  smg       create label type "REL6"
"Automatically created label type from global definition in VOB "\admin"."
```

Changing Protection of a Global Type

Changing the protection of a global type or of a local copy of a global type changes the protection of the global type and all its local copies. You must have permission to change the protection of the global type. You can enter the command in the context of a client VOB even if the client VOB does not contain a local copy of the type.

In this example, the owner of the label type **LABEL1** is changed to **jtg**. The **describe** command shows that the protection change is made to all local copies of the global type.

cleartool protect –chown jtg lbtype:LABEL1@\dev

```
Changed protection on "LABEL1".
```

cleartool describe –local lbtype:LABEL1@\re

```
label type "LABEL1"
...
  owner: jtg
  group: user
  scope: this VOB (local copy of global type)
...
```

If the protection cannot be changed on one or more of the local copies, the operation fails and the global type's protection is not changed. This failure leaves the global type and its local copies in inconsistent states. You must fix the problem and run the **protect** command again.

Locking or Unlocking a Global Type

Locking or unlocking a global type or one of its local copies locks or unlocks all local copies. The **describe** command does not list local copies as locked, but access checking on local copies checks for a lock on the global type.

For example, the following command locks the global label type **REL6** and its local copies:

cleartool lock –c "freeze" lbtype:REL6@\dev

```
Locked label type "REL6".
```

Attempts to create instances of the label type fail:

cleartool mklabel –c "last version" REL6 \re\tests.txt

```
cleartool: Error: Lock on label type "REL6" prevents operation "make
hyperlink".
cleartool: Error: Unable to create label "REL6" on "\re\tests.txt" version
"/main/5".
```

If you enter a **lock** command in a client VOB that does not contain a local copy of the specified type, ClearCase LT searches for the global type in the administrative VOB hierarchy.

By default, **lslock** lists the lock state of the global type. To list the lock state of the local copy, use the **–local** option.

Changing the Type of an Element or Branch

You can use the **chtype** command to change the type of an element (convert the element from one type to another) or a branch (rename the branch). If the new type is a global type and a local copy does not exist in the client VOB, the **chtype** command creates the local copy.

For more information on changing a type, see the **chtype** reference page.

Copying a Global Type

When you copy a global type to the same name (in a different VOB), the **cptype** command preserves the global type associations of the copied global type when either of these conditions is true:

- The source VOB of the original type and destination VOB of the copy are both members of the same administrative VOB hierarchy. (The copy then points to that administrative VOB hierarchy.)
- The original global type resides in a VOB that is the administrative VOB of the copy's destination VOB (where **cptype** creates a local copy).

In all other cases, the type is created as an ordinary (that is, nonglobal) type.

Renaming a Global Type

Renaming a global type renames all its local copies. Also, renaming a local copy of the global type renames the specified local copy, all other local copies, and the global type itself. If you enter a **rename** command in a client VOB that does not contain a local copy of the specified type, ClearCase LT searches for the global type in the administrative VOB hierarchy.

All local copies are renamed first; then the global type is renamed. If any of the local copies cannot be renamed, the command fails and the global type is not renamed. This failure leaves the global type and its local copies in inconsistent states. You must correct the problem and enter the **rename** command again.

For more information on renaming types, see the **rename** reference page.

Changing the Scope of a Type

To convert an existing ordinary type to a global type, enter a **mkattype**, **mkbrtype**, **mkeltype**, **mkhltype**, or **mklbtype** command with the options **–replace –global –acquire**. These commands convert the type and convert any identical types in client VOBs to local copies of the type. For example:

1. An administrative VOB and one of its client VOBs contain identical ordinary label types named **IDENT**:

   cleartool describe lbtype:IDENT@\admin
   ```
   label type "IDENT"
     created 02-Aug-99.15:32:52 by Suzanne Gets (smg.user@neon)
     owner: smg
     group: user
     scope: this VOB (ordinary type)
     constraint: one version per element
   ```

   cleartool describe lbtype:IDENT@\dev
   ```
   label type "IDENT"
   created 01-Aug-99.15:33:00 by Suzanne Gets (smg.user@neon)
   owner: smg
   group: user
   scope: this VOB (ordinary type)
   constraint: one version per element
   ```

2. Convert the label type in the administrative VOB to be a global type:

   cleartool mklbtype –replace –global –acquire IDENT@\admin
   ```
   Replaced definition of label type "IDENT".
   ```

3. The output of the **describe** command shows that the label type in the administrative VOB is now global, and the label type in the client VOB is now a local copy of the global type:

cleartool describe –local lbtype:IDENT@\admin lbtype:IDENT@\dev

```
label type "IDENT"
  created 02-Aug-99.15:32:52 by Suzanne Gets (smg.user@neon)
  owner: smg
  group: user
  scope: global
  constraint: one version per element
  Hyperlinks:
    GlobalDefinition <- lbtype:IDENT@\dev
label type "IDENT"
  created 02-Aug-99.15:32:52 by Suzanne Gets (smg.user@neon)
  owner: smg
  group: user
  scope: this VOB (local copy of global type)
  constraint: one version per element
  Hyperlinks:
    GlobalDefinition <- lbtype:IDENT@\dev
```

To convert an existing global type to an ordinary type, enter a **mkattype**, **mkbrtype**, **mkeltype**, **mkhltype**, or **mklbtype** command with the options **–replace –ordinary**. These commands convert the type and all its local copies to ordinary types. You must specify the global type in the command; you cannot specify a local copy of the type. For example, to convert the global element type **doc_file** to an ordinary type, and the administrative VOB is **\admin**, enter the following command:

cleartool mkeltype –replace –ordinary –nc eltype:doc_file@\admin

You can also use the **–replace** option to change the constraints of a type if the normal ClearCase LT restrictions allow the change.

Removing a Global Type

Removing a global type removes all the local copies. Also, removing a local copy removes the specified copy, all other local copies, and the global type itself. The **rmtype** command lists the client VOBs that have local copies of the global type, then prompts for confirmation of the removal. You must use the **–rmall** option with the **rmtype** command.

For example:

cleartool rmtype –nc lbtype:LABEL1@\dev

```
cleartool: Error: There are labels of type "LABEL1".
cleartool: Error: Unable to remove label type "LABEL1".
```

cleartool rmtype –nc –rmall lbtype:LABEL1@\dev

```
There are 1 instance(s) of label type "LABEL1" in \re.
There are 1 instance(s) of label type "LABEL1" in \dev.
Remove all instances of label type "LABEL1"?  [no] yes
Removed label type "LABEL1".
```

Notes on removing global types:

- If you enter a **rmtype** command in a client VOB that does not contain a local copy of the global type, ClearCase LT tries to find a matching global type in the administrative VOB hierarchy.
- All local copies are deleted first; then the global type is removed. If any of the local copies cannot be removed, the command fails and the global type is not removed. You must correct the problem and enter the **rmtype** command again.

For more information on removing types, see the **rmtype** reference page.

Cleaning Up Global Types

Use **checkvob –global** to check and fix global types that are in an inconsistent state. For more information, see Chapter 13, *Using checkvob*.

9 Backing Up and Restoring VOBs

The most important maintenance task for a ClearCase LT administrator is to ensure frequent, reliable backups of essential ClearCase LT data. To ensure the integrity of your VOB backups, follow closely the instructions and guidelines in this chapter.

This chapter describes these operations:

- Backing up VOBs
- Restoring VOBs
- Synchronizing VOBs and views after restoring a VOB.

For information on backing up views, see *Backing Up a View* on page 229.

9.1 Choosing Backup Tools

Rational ClearCase LT does not include any backup tools. All ClearCase LT data is stored in standard files, within standard directory trees; thus, you can use any backup tools available to you that can handle the special needs of VOB backup and recovery. Because file-system conventions and backup tools for UNIX and Windows differ, keep platform-specific considerations in mind when choosing a VOB backup tool.

UNIX Backup Issues

On some systems (HP-UX and Solaris, for example), **tar** resets file access times, which can disrupt cleartext storage pool scrubbing patterns. For example, the **scrubber** utility, by default, scrubs

cleartext files that have not been accessed in more than 96 hours. A nightly **tar** operation that backs up cleartext pools and resets cleartext file access times will prevent the pools from ever being scrubbed.

The standard UNIX utility **cpio(1)** is well suited to backing up ClearCase LT data.

Windows Backup Issues

On Windows, VOB backup programs must be able to handle the file system protection and file locking issues unique to that platform. In particular:

- It must preserve the protections of the VOB storage directory so that they can be restored correctly when the backup is recovered.
- If possible, it should be able to copy files even when they open for writing.

Common Windows backup utilities do not back up files that are open for writing. Unless you have stopped ClearCase on the VOB server host, several VOB database files remain open for writing even when the VOB is locked. If a backup operation skips these files, the backup will be useless. To avoid this problem, do one of the following:

- Purchase and configure backup software that can back up files that are open for writing.
- Stop ClearCase before performing the backup. The VOB will be inaccessible while ClearCase is stopped. You may be able to decrease the length of time the VOB is inaccessible by copying the VOB storage directory to another on-disk location, re-starting ClearCase, and then backing up the copy

NOTE: Utilities such as **ccopy** (a ClearCase LT utility) and **scopy** (from the Windows NT Resource Kit) do not back up files that are open for writing.

9.2 Backing Up a VOB

Because of operating system differences, VOB backup operations are handled differently on UNIX and Windows. All VOB backup operations begin with locking the VOB. This step is critical to the integrity of any VOB backup.

Backing Up a VOB on UNIX

A VOB backup on UNIX can be summarized as follows:

1. Lock the VOB.
2. Back up the VOB storage directory.
3. Unlock the VOB.

By default, a VOB storage directory is wholly contained in a single directory tree, which resides in a single disk partition. If you use a file-oriented backup tool, you specify only the VOB storage directory to ensure a complete backup. If you use a disk-partition-oriented backup tool, you specify only the partition name.

Backing Up a VOB on Windows

VOB storage on Windows is wholly contained in a single directory tree, which simplifies the backup process. A VOB backup on Windows can be summarized as follows:

1. If your backup software can be configured to capture files that are open for write access:
   a. Lock the VOB
   b. Back up the VOB storage directory.
   c. Unlock the VOB.
2. If your backup software cannot process files that are open for write access, you must stop ClearCase on the ClearCase LT server, back up the VOB storage directory, restart ClearCase, and unlock the VOB.

WARNING: Many Windows backup utilities do not back up files that are open for writing. Because the VOB database files are typically in this state while ClearCase is running, *even when the VOB is locked*, your backup operation skips these files unless you stop ClearCase before performing the backup. Unless your backup software can capture files open for write access, *you must stop ClearCase on the VOB host before performing a backup.* To stop ClearCase on the ClearCase LT server, use the ClearCase program in Control Panel.

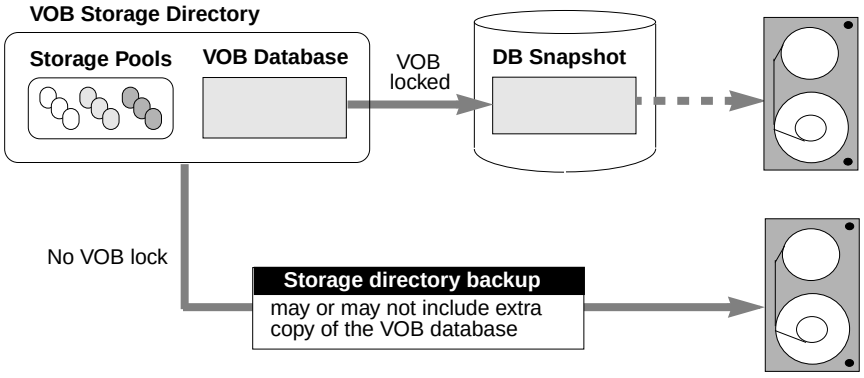
Choosing Between Standard and Semi-Live Backup

The standard backup procedure is to lock the VOB and back up the entire VOB—VOB database, plus VOB storage pools and various other files in the VOB storage directory. The VOB must remain locked for the duration of the backup. Keeping VOB database and storage pools synchronized on backup media is desirable; the standard backup procedure is recommended for all sites that can accept the duration of required locks. Locking a VOB prevents checkins, checkouts, and other operations that affect VOB data and metadata. These include UCM deliver operations and any UCM rebase operations that require merging. Other development tasks, such as editing, compiling, and debugging, can take place while a VOB is locked.

As illustrated in Figure 5, semi-live backup involves backing up the two major pieces of a VOB separately, as follows:

- **VOB database.** Configure the VOB to have the **vob_snapshot** utility periodically lock the VOB and copy the VOB database to another location on disk (from which it can be backed up as part of the site's normal backup process).
- **VOB storage pools.** Back up the VOB storage directory routinely, without locking the VOB. The backed up VOB storage directory may include a copy of the VOB database. However, because it is backed up while the VOB is unlocked, this database is useless and is discarded when the VOB is restored.

Figure 5 Semi-Live Backup



Benefits of Semi-Live Backup

- **VOB lock time reduced.** The VOB storage directory can be backed up unlocked. The **vob_snapshot** utility locks the VOB, copies the VOB database to another disk location, and unlocks the VOB.

Costs of Semi-Live Backup

- **More disk space is required.** A second copy of the VOB database must be captured to disk. In addition, each of the VOB's source pool data containers, when replaced by a container with new version data, is retained for an additional 30 minutes to guarantee that source containers can be reconstructed when **checkvob** resynchronizes the VOB database and the storage pools at VOB restore time.

- **The VOB restore procedure is more complex.** The VOB storage pools and the VOB database have different reference times. VOB database and storage pools must be resynchronized when you restore the VOB. In particular, version data added or removed in the interval between the database snapshot and storage pool backup cause database or pool skew that must be resolved. The **vob_restore** utility runs the **checkvob** utility to do this work.

- **Some data may be lost at VOB restore time.** If the restored pools are older than the restored VOB database, data missing from the pools is lost (as expected). If the restored pool backup is newer than the database, pool version data newer than the snapshot is not added to the restored database. See also *vob_restore: Restoring with a Database Snapshot* on page 135.

Enabling Semi-Live Backup

To enable database snapshots, run **vob_snapshot_setup** on a VOB. This command causes the ClearCase LT scheduler to run **vob_snapshot** periodically on the VOB (daily, by default). The **vob_snapshot** and **vob_snapshot_setup** reference pages explain these operations. Consult them if you choose to use semi-live backup.

The backup procedures that follow accommodate (but do not require) snapshot-enabled VOBs. However, their focus is the standard backup approach, which requires the VOB to be locked for long enough to back up the database and all the pools. The **vob_restore** procedure applies equally to VOBs with and without database snapshots.

Deferred Source Container Deletion

Deferred deletion is enabled by the **vob_snapshot_setup** program. It ensures that the source pool will contain all needed containers when a backup program archives an active VOB. When a

container is replaced by new version data (for example, during a checkin), the new container is created and the client requests the **vob_server** to remove the old container. If deferred deletion is deactivated, the container is immediately removed. If deferred deletion is enabled, the old container is added to a list of pending deletions, and it is removed in 30 minutes. Keeping source data containers for 30 minutes increases disk space requirements. The increase can be substantial during any 30-minute interval of heavy VOB checkin activity.

On UNIX platforms, you can execute the **kill –HUP** command on the **vob_server** process to send deferred deletion statistics to the **vob_server** log (see also **getlog** and **getlog –gr**).

When **checkvob** examines source pools, it reports containers on the deferred deletion list.

The deferred deletion list is written every five minutes to the file **delete_list.db** in the VOB storage directory.

Determining a VOB's Location

To determine the location of a VOB storage directory, use the ClearCase Administration Console or the cleartool **lsvob** command. If your backup program runs locally, it probably uses the `VOB server access path`. If your backup program runs over the network, it probably uses the `Global path`.

cleartool lsvob –long \vob_flex

```
Tag: \flex
  Global path: \\ccsvr01\vobstore\vob_flex.vbs
   .
   .
 .
VOB on host: ccsvr01
VOB server access path: c:\vobstore\vob_flex.vbs
```

Specify the appropriate pathname to your backup program.

NOTE: Some UNIX utilities that back up entire disk partitions require you to specify the disk partition where the VOB storage directory resides.

Ensuring a Consistent Backup

A backup represents a self-consistent snapshot of a VOB storage directory's contents only if the VOB is not modified while the backup program is working. That's why it is essential to lock the VOB before backing it up and unlock it after the backup completes.

WARNING: Regardless of your chosen backup strategy (see *Choosing Between Standard and Semi-Live Backup* on page 114), you must lock the VOB against all users. Use the ClearCase Administration Console or the Windows Explorer shortcut menu to lock the VOB. If you use the **cleartool lock** command, do not use the **–nusers** option. The lock is mandatory. It is not sufficient to capture a VOB that happens to be idle. The lock does more than ensure that no one modifies the VOB. It also causes the VOB to flush a database checkpoint to disk before backup. A VOB lock applied with the **–nusers** option does not perform the required database checkpoint.

Locking and Unlocking a VOB

You must be a privileged user to lock or unlock a VOB. There are several GUIs that support VOB locking, as well as two **cleartool** commands.

You can lock or unlock a VOB on any UNIX or Windows host from the ClearCase Administration Console:

1. Navigate to the VOB storage node for the VOB. This is a subnode of the host node for the host where the VOB storage directory resides.
2. Click **Action** > **Properties**. In the **Properties of VOB** dialog box, click the **Lock** tab.

From a Windows host, you can lock or unlock a VOB using the Windows Explorer shortcut menu for the VOB. Click **ClearCase** > **Properties of VOB** and click the **Lock** tab. On the command line, use **cleartool lock** and **unlock**:

cleartool lock vob:/flex
```
Locked versioned object base "/net/pluto/vobstore/flex.vbs".
```

<perform backup>

cleartool unlock vob:/flex
```
Unlocked versioned object base "/net/pluto/vobstore/flex.vbs".
```

Partial Backups

If you use a file-oriented backup program, you may want to exclude some subdirectories within the VOB storage directory to save time. Use the guidelines in Table 3 to determine the relative importance of the various directories.

Table 3 Importance of VOB Directories in Partial Backups

VOB Directory	Importance for Backup
Top-level VOB storage directory	Required
VOB database subdirectory	Required
Source storage pools	Required
Cleartext storage pools	Optional
Administrative directory	Important, but not required

Cleartext Pool Backup

Backing up cleartext storage pools is not important, because they are caches that enhance performance. Type managers re-create cleartext data containers as necessary.

If you do not back up cleartext pools, include each pool's roots in the backup—the pool's root directory (**c\cdft**, for example) and **pool_id** file, but not its subdirectories. Doing so prevents pool root check failure at restore time (see also *Database or Storage Pool Inconsistencies* on page 183) and possible cleartext construction errors in the restored VOB.

Administrative Directory Backup

The **admin** directory contains data on how much disk space has been used by the VOB and its derived objects. The ClearCase LT scheduler runs periodic jobs that collect data on disk space use and store it in the **admin** directory. By default, the scheduler stores data for the previous 30 days. This historical data cannot be re-created. If the data is important to you, back up the **admin** directory.

Incremental Backups of a VOB Storage Directory

Using a base-plus-incremental scheme to restore a VOB storage directory typically restores too much data. Depending on your particular VOB and disk, this data may fill up the disk.

ClearCase LT stores version data in *delta* format (for example, the container of a **text_file** element). Instead of modifying an existing data container when a new version of an element is checked in, ClearCase LT creates a new container at a different pathname within the source storage pool. (It then deletes the old container.)

An example demonstrates how this storage strategy works with an incremental backup scheme. Suppose that one or more new versions of a particular element are created each day for a week. Each day's incremental backup picks up a different source data container for that element. If a crash occurs on the ClearCase LT server at the end of the week, restoring the VOB places all those source data containers in the source storage pool, even though only one of them (the most recent) corresponds to the current state of the VOB database. Also, **checkvob** reports large numbers of unreferenced data containers when it runs at VOB restore time. (See the **checkvob** reference page for details.)

Given this situation, we recommend that you perform only a few incremental backups on a VOB storage directory before the next full backup. This practice minimizes the extra data involved in a base-plus-incremental restoration of the VOB. Run **checkvob** to detect and clean up the extra *debris* containers.

9.3 Restoring a VOB from Backup with vob_restore

Given a complete and consistent VOB storage backup, **vob_restore** can accommodate a variety of restore scenarios. You can use **vob_restore** with or without a VOB snapshot. Any valid VOB backup (as defined in section 9.2) will work. **vob_restore** handles all of the following subtasks:

- Stops and restarts ClearCase
- Updates the ClearCase LT VOB registry
- Merges the VOB database snapshot and VOB storage directory
- Copies the temporary storage directory to the target location
- Runs **checkvob** to resynchronize the VOB database and storage pools

NOTE: You cannot use **vob_restore** to restore a VOB that is located on a Network Attached Storage device.

Before examining alternative restore scenarios, it is useful to summarize the restoration procedure. We recommend that you do not retrieve VOB storage from backup media until **vob_restore** prompts you to do so in Step #3.

1. **Log on to the ClearCase LT server.** Log on as a user with permission to stop and start ClearCase—typically the **root** user on UNIX or a local administrator on Windows.

2. **Check available disk space**. If you are not restoring the VOB to the same location, make sure that there is enough free space in the VOB's disk partition to load the backup copy into a temporary storage location. To do so, use the VOB storage node in the ClearCase Administration Console or the **cleartool space** command.

   ```
   # cleartool space /vobstore/flex.vbs
     Use(Mb)  %Use  Directory
        27.0    2%  VOB database /net/pluto/vobstore/flex.vbs
        33.0    3%  cleartext pool /net/pluto/vobstore/flex.vbs/c/cdft
      .
      .
   ------------------------------------------------------------------------
      312.9   28%  Subtotal
      828.4   74%  Filesystem /net/pluto/vobstore (capacity 1115.1 Mb)
   ```

   If the available space is insufficient, delete the VOB storage directory or use other means to make enough space available.

3. **Run vob_restore.** Exit any current working view and run a command like this one:

   vob_restore /flex *(the VOB-tag is the only argument)*

   When prompted, supply the information required by **vob_restore**—target directory for VOB restoration, location of database snapshot (if any), and so on.

   NOTE: On Windows, you must use UNC names (*host**share**rest-of-path*) for all path information you supply to **vob_restore**.

   For more information on **vob_restore** operations, refer to these sections:

   - *vob_restore: Sample Session* on page 121
   - *vob_restore: Restoration Scenarios* on page 129
   - *vob_restore: Restoring with a Database Snapshot* on page 135

4. **While the VOB is still locked, run some consistency checks.** For example, create a new view with the default config spec, load it from the restored VOB, and verify that all versions of recently changed elements are present.

5. **Unlock the restored VOB**. **vob_restore** always leaves the VOB locked when it exits.

   ```
   # cleartool unlock vob:/flex
   Unlocked versioned object base "/vobstore/flex.vbs".
   ```

6. **If necessary, resynchronize the VOB and views.** See *VOB and View Resynchronization* on page 142.

vob_restore: Sample Session

The sample session presented here restores VOB **\vob_src**. To complete the recovery, **checkvob** is run to find and fix inconsistencies between the restored VOB database and the restored VOB storage pools.

Additional details about this scenario:

- The VOB is being restored to its current location (**\\io\vobstore\vob_src.vbs)**. This is the in-place scenario, which is the most common.
- The data currently in the VOB is invalid, so there is no need to maintain read-only access to the VOB during the restore operation.
- A database snapshot for this VOB, **\\io\e\vob_snaps\src**, is used.
- The administrator is logged on as *Administrator* on the VOB host, **io**.
- As is recommended, the restored data is not retrieved from backup media before running **vob_restore**.

To start the process, run the **vob_restore** command.

*ccase-home-dir***\etc\vob_restore \vob_src**

```
This utility helps recover a damaged VOB or replica from backup. It will first
prompt you for the information it needs to perform its job. Then, it will
describe the 'restore scenario' it believes you have in mind, based on the
information you have supplied. You will then be asked various questions as the
script proceeds through the restore process. Some prompts ask for more
information, others simply request verification for the next step. Each prompt
includes a list of valid responses and the default, where appropriate.

At many prompts, a possible response is the string 'quit'. If you choose to
quit at one of these prompts, a '–restart <pathname>' string will be displayed
as the script exits. Save this output and supply it as a command line option
to vob_restore (before the VOB-tag argument) when you want to resume the
interrupted restore operation on the same VOB or replica. If you choose to
quit at any point, DO NOT in any way alter the state of the VOB or replica
before restarting the restore operation.

Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
```

Target Prompt

vob_restore prompts for a target destination for the reassembled VOB storage directory.

```
Specify the full path for target storage directory to contain the VOB or
VOB replica. The default is the currently registered local path:
    \\io\vobstore\vob_src.vbs
Type a full pathname, "help", or "quit": <RETURN>
```

Pressing RETURN restores the VOB to its current, registered location. Supplying a different pathname moves the VOB. The pathname supplied here must be on the local host.

Storage Directory Prompt

Next, **vob_restore** prompts for the location of the storage directory backup, which may or may not be retrieved at this point.

```
Please specify the full path for the VOB or replica storage that was
either restored from backup media or will be during this restoration
process.
Valid responses are (Full path, quit or help)
There is no default response: \\io\vobstore\vob_src.vbs
```

This response indicates that the retrieved backup storage directory is already in, or will be loaded into (recommended), the currently registered storage location, overwriting the existing storage directory.

Snapshot Prompt

vob_restore prompts for the location of the VOB database snapshot, if any. If you are not using a semi-live backup strategy for this VOB, press RETURN.

```
If a merge of a snapped database with this retrieved backup data is
desired, please specify the full pathname of the snapped database.
(Full path, help, quit): \\io\e\vob_snaps\src
```

Backup-Loaded Prompt

vob_restore prompts you to specify whether the backup has been loaded.

```
Is the data currently contained in the directory
  \\io\vobstore\vob_src.vbs
the data restored from backup media?
Valid responses are (yes,no,help)
The default is no: no<RETURN>
```

This response confirms that the VOB storage directory is not yet retrieved from backup media.

Sample VOB Restoration Scenario

At this point, **vob_restore** has the information it needs to determine the restore scenario. See also *vob_restore: Restoration Scenarios* on page 129.

```
The information you supplied, and the state of the registry at that time,
resulted in the following initial restoration parameters. If this
invocation is a restart, the current registry state may be different.

 o The vob is currently registered.
 o The restored vob will be placed in its previously registered location
       \\io\vobstore\vob_src.vbs
   on its previously registered host
       io
 o The restoration will be done in place.
 o The database will be copied from the snapshot directory
       \\io\e\vob_snaps\src
```

```
If you wish, you may quit for now  and restart the script at a later time.
Do you want to proceed?
Valid responses are (yes,quit)
The default is yes: yes<RETURN>
```

If the VOB is registered in a registry that is served by a different registry host, you must go to a host served by that registry server and unregister the VOB with **cleartool unregister**. You can also use the ClearCase Administration Console.

NOTE: ClearCase LT does not support multiple registries. Press RETURN when prompted for a response.

```
The vob must now be made unavailable. This script will unregister the vob
from this hosts's registry. If it is registered in any other registries
you must unregister it from those registries before continuing. Do not
unregister it from this host's registry. You may quit for now to perform
this operation. Press <RETURN> to continue only when this host's registry
is the only registry the replica remains registered in.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>

The full current registry settings will be saved in the file
    \\io\vobstore\register_save_io
Press return to continue.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
Removing vob tag \vob_src...rmtag complete
Unregistering \\io\vobstore\vob_src.vbs...unregister complete
```

At this point, **vob_restore** is ready to shut down ClearCase.

```
Is it all right to shutdown Clearcase on this host?
Valid responses are (yes,quit,help)
The default is yes: yes<RETURN>
Shutting down Clearcase...Clearcase shutdown complete
```

You must now move or remove the old, damaged storage directory before **vob_restore** restarts ClearCase:

```
You must either move or remove the previous storage directory
    \\io\vobstore\vob_src.vbs
now. Press <RETURN> to continue only when it has been completed.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
Starting Clearcase...Clearcase start complete
```

Now, load the backup into the directory supplied at *Storage Directory Prompt*.

```
The restored data must now be retrieved from backup media and placed in
    \\io\vobstore\vob_src.vbs
You may quit to restore the data now or press <RETURN> to continue when
the restoration has been completed.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
```

If the VOB is configured for database snapshots, but not for **db_check** operations at snapshot time (see **vob_snapshot_setup**), run **db_check** now. Note that **db_check** is forced in either of these situations:

- There is no snapshot, and the VOB was unlocked at backup time.
- The snapshot was taken when the VOB was unlocked.

```
A dbcheck was not done during the snapshot. Do you wish to do one now?
Valid responses are (yes,no,quit)
The default is no: yes<RETURN>

Performing data base check. This may take some time...checked clean
```

ClearCase LT does not support remote storage pools. Press RETURN at the next prompt.

```
If there are any remote pools that should be restored now is the time to
restore them. You may quit for now to perform this operation or press
<RETURN> to continue.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
```

Confirm that you have supplied a VOB database snapshot. **vob_restore** merges it with the retrieved VOB storage backup, overwriting any VOB database retrieved with the storage directory:

```
The restored storage area contains a copy of a directory being restored
from the snapshot area.
    \\io\vobstore\vob_src.vbs\db
Is it ok to remove this copy?
Valid responses are (yes,quit,help)
The default is yes: yes<RETURN>
Making tag for storage \\io\vobstore\vob_src.vbs\db...mktag complete
Registering \\io\vobstore\vob_src.vbs\db...register complete
```

If you are testing backup or restore procedures and not restoring a broken VOB, let **vob_restore** temporarily disable VOB database snapshot activity on the VOB:

```
The VOB database snapshot utility is enabled for this VOB/replica. If you
are restoring this VOB/replica because it was broken, this is probably ok.
However, if you are merely testing your backups, and this replica is
actually active in some other region, this may cause a problem. If the
snap path
    \\io\e\vob_snaps\src
is visible on this host, the test backup VOB's database will be snapped,
overwriting the real snap for the VOB/replica . This can be prevented by
removing the snap parameter attribute for this VOB/replica at this
time.You should answer yes to this prompt if this recovery is a test of
backups otherwise answer no.
Do you want this script to remove the snap parameter attribute ?
Valid responses are (yes,no)
There is no default response: no<RETURN>
```

If you supplied a database snapshot when prompted for one, you must run **checkvob** to resynchronize the VOB database and storage pools. We recommend that you run **checkvob** whenever you restore a VOB, because it may expose problems in the restored VOB.

NOTE: **vob_restore** replaces the VOB lock with one that permits access by the **checkvob** process, and relocks the VOB against all users when **checkvob** exits. Make sure that the user ID under which **vob_restore** or **checkvob** runs does not modify the VOB in any way while **checkvob** is running.

```
Do you want to have checkvob examine the vob for possible problems?
Valid responses are (yes,quit,help)
The default is yes: yes<RETURN>

Would you like to have checkvob run in 'check only' mode?
Valid responses are (yes,no,quit,help)
The default is yes: yes<RETURN>

Source pools?
Valid responses are (yes,no)
The default is yes: yes<RETURN>

Derived object pools?
Valid responses are (yes,no)
The default is yes: yes<RETURN>
```

```
Cleartext pools?
Valid responses are (yes,no)
The default is yes: yes<RETURN>

Checkvob expects to be run in a view context. If you are not currently in
a view, checkvob will accept a view tag argument. Supply one now if you
are not in a view.
view_tag: adm_view<RETURN>

Checkvob will now be run in 'check only' mode. This may take a while on
large vobs. Checkvob will generate logs in the directory
    checkvob_sum.03-Oct-96.14.21.06
It will emit rather verbose information to standard output as it runs. All
of this information is also saved in the log directory for later viewing.
You may be prompted for information as well. You may quit for now or press
<RETURN> to continue.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
```

At this point, **checkvob** takes control temporarily from **vob_restore**:

```
The session's log directory is 'checkvob_sum.03-Oct-96.14.21.06'.
================================================================
Starting "source pool" processing at 03-Sep-96.14:21:59

... lots of checkvob output ...

The VOB's derived pools are healthy.

Poolkind transcript log:
checkvob_sum.03-Oct-96.14.21.06\poolkind_derived\transcript
================================================================
```

checkvob's check-only pass is complete. Control has returned to **vob_restore**, which summarizes the results and prompts you run **checkvob** in fix mode to repair any problems:

```
1 source containers are either missing or corrupt
0 derived object containers are either missing or corrupt
0 source containers are misprotected
0 derived object containers are misprotected
Cleartext containers were not checked.
More details may be found in the log files in the above directory

Do you want to have checkvob run in repair mode?
Valid responses are (yes,no,quit)
The default is yes: yes<RETURN>
```

```
Source pools?
Valid responses are (yes,no)
The default is yes: yes<RETURN>

Checkvob will now be run in fix mode. This may take a while on
large vobs. Checkvob will generate logs in the directory
    checkvob_fix.03-Oct-96.14.22.02
It will also emit rather verbose information to standard output as it
runs. All of this information is also saved in the log directory for
later viewing. You may be prompted for information as well.
Do you want to proceed?
Valid responses are (yes,quit)
The default is yes: yes<RETURN>

While running checkvob the the vob will be locked for all but user
Administrator (or other account used to invoke vob_restore)
The vob will be open to possible modifications by anyone running as
this user. This MUST NOT be allowed to happen. If you need to make
arrangements to ensure that no one with this identity will attempt to
modify this vob before proceeding you may quit for now. Press <RETURN>
if you are satisfied that no such modifications will occur.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
```

Once again, **checkvob** is back in control, running in fix mode.

```
The session's log directory is 'checkvob_fix.03-Oct-96.14.22.02'.

If any version data is missing from source pool data containers,
fix processing involves irreversibly updating the VOB database
with the equivalent of 'cleartool rmver -data' operations.
By default, checkvob does not allow this class of fix processing.

Do you want to override the default and fix elements with
missing version data? [no] yes<RETURN>

You must specify a time limit for acceptable missing data.
Refer to the reference manual for more information.

Allow missing version data created since: [date-time, <CR>] <RETURN>

Allowing missing version data created since 02-Oct-96.00:00:00.
```

```
WARNING: You are allowing fix processing for missing version data.
If the allowable missing version data limit encompasses
more versions than you expected, you will have to restore
this VOB from backup media to undo the effects of this fix
processing.

Do you want to continue with force mode processing? [no] yes<RETURN>

================================================================
Starting "source pool" processing at 03-Oct-96.14:26:21

... lots of checkvob output ...

The VOB's source pools are healthy.

Poolkind transcript log:
checkvob_fix.03-Oct-96.14.22.02\poolkind_source\transcript
================================================================

0 source containers remain either missing or corrupt
0 source containers remain misprotected
0 source elements experienced loss of version data
0 source versions were lost
0 derived object containers remain either missing or corrupt
0 derived object containers remain misprotected
0 rmdo operations were done
0 derived objects were lost
Cleartext containers were not checked.

Check has either detected no problems or has repaired what it did detect.
This is a non replicated vob so no further action should be required.
```

VOB restoration is now complete. Go to Step #4 on page 120.

vob_restore: Restoration Scenarios

vob_restore can accommodate a number of restoration scenarios:

- In place: restore a VOB without changing its location
- VOB is active: restore a VOB that has read-only access
- Move VOB on same host
- Move VOB to new host
- VOB is unregistered

All have two variants: with and without a VOB snapshot.

The previous example restored a VOB in place, loading the backed up VOB storage directly into the currently registered location. The examples also merged in a VOB database snapshot from its on-disk location. This scenario can be called “in place, with snapshot.” There are other possibilities.

How vob_restore Determines the Scenario

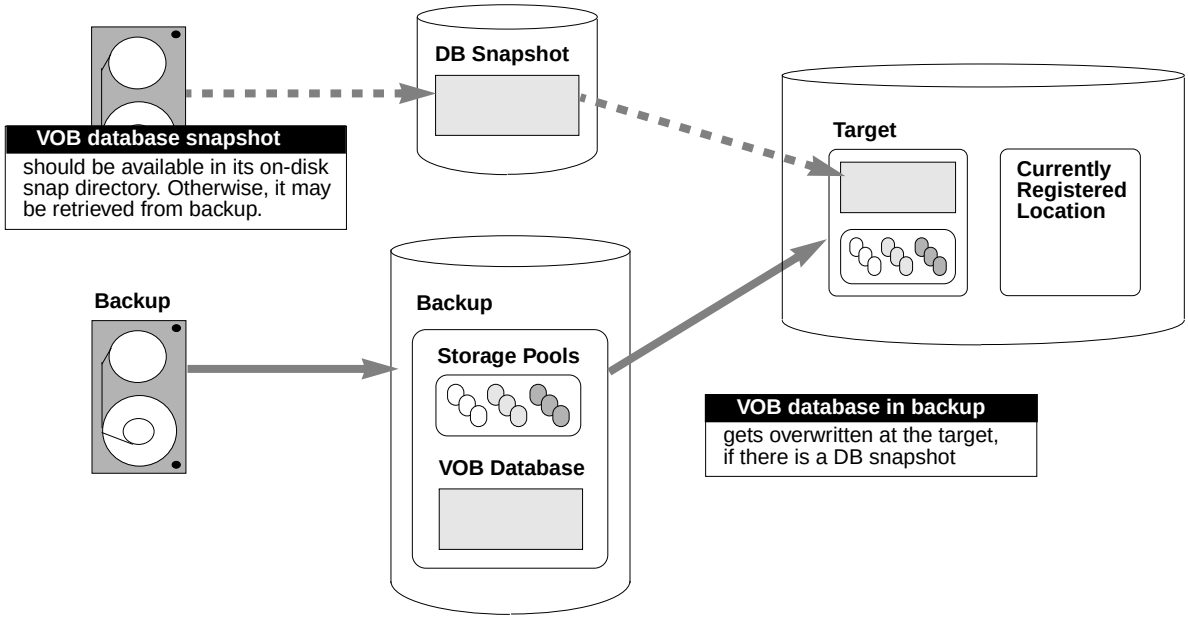
When you run **vob_restore**, it prompts for three critical pieces of information:

- What is the target? That is, where will the restored VOB storage ultimately reside?
- Where are you going to put the VOB storage directory backup (or, where did you put it)?
- If you want to merge in a VOB database snapshot, where is it?

vob_restore also derives the VOB’s *currently registered storage location* from the VOB-tag argument supplied on the command line.

As shown in Figure 6, **vob_restore**’s task is to merge the *backup* and the *snapshot* (if there is one) at the *target*, and to correctly re-register the target, which may be at a location other than the VOB’s *currently registered location*.

Figure 6 VOB Restoration



vob_restore uses your input to describe the restore scenario. For example, Figure 7 shows the scenario from the sample restore session in the previous section.

Figure 7 Restore Scenario Summarized by Output from vob_restore

```
o The vob is currently registered.
o The restored vob will be placed in its previously registered location
      \\io\vobstore\vob_src.vbs
  on its previously registered host
      io
o The restoration will be done in place.
o The database will be copied from the snapshot directory
      \\io\e\vob_snaps\src
```

This output can vary, depending on the information you supply at **vob_restore** prompts.

Restoration Rules and Guidelines

At restore time, remember these rules and guidelines:

- *Target* must be on the ClearCase LT server.

- *Snapshot* can reside on a remote host.
- Whenever possible, the *backup* and *target* pathnames ought to be the same. That is, always try to load the backup into its final destination. This practice avoids a time-consuming copy operation. (In fact, **vob_restore** copies the backup to the target only if a move operation is impossible.) This avoidance is possible for all scenarios but *vob_restore: VOB Is Active* on page 133.
- If the *backup* and *target* pathnames are different, try to keep them on the same host to avoid possible problems with permissions, network load, and so on.
- If you retrieve the *backup* before running an in-place restoration, you must unregister the VOB, stop ClearCase, load the backup, and restart ClearCase.

Now let's look at the main scenarios, one at a time.

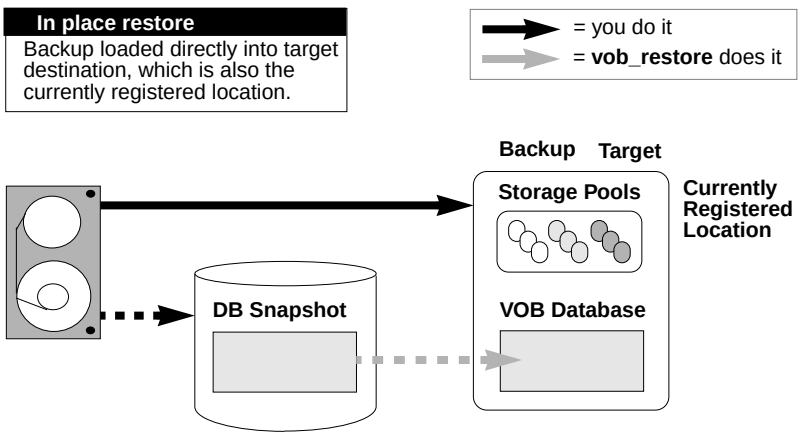
vob_restore: In Place

This scenario is illustrated in Figure 8

Formula. *backup = target = currently-registered-location*

Advice. Use this scenario whenever practical.

Figure 8 vob_restore: In Place



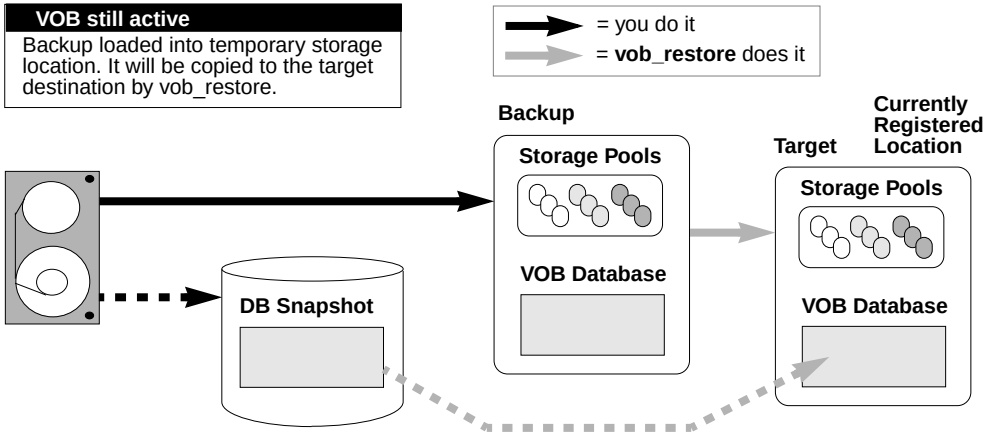
vob_restore: VOB Is Active

In this scenario, the VOB is still active for read-only access, so the backup must be loaded into a temporary storage location. (See Figure 9.)

Formula. (*backup* different from *target*) and (*target = currently-registered-location*)

Advice. Do not combine this scenario with a move VOB scenario. Keep the VOB at its currently registered location.

Figure 9 vob_restore: VOB Is Active



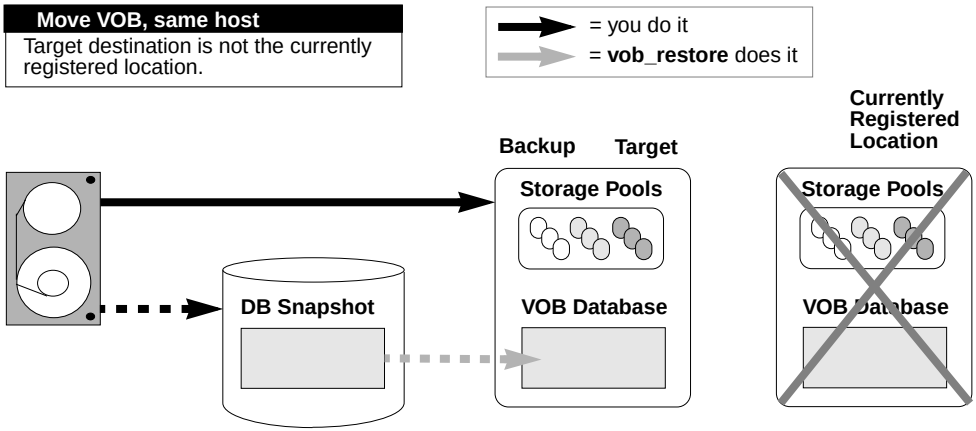
vob_restore: Move VOB on Same Host

In this scenario, the backup is restored to a different location on the same host. (See Figure 10.)

Formula. *target* different from *currently-registered-location*

Advice. Load the backup directly into the target destination.

Figure 10 vob_restore: Move VOB on Same Host



vob_restore: Move VOB to New Host

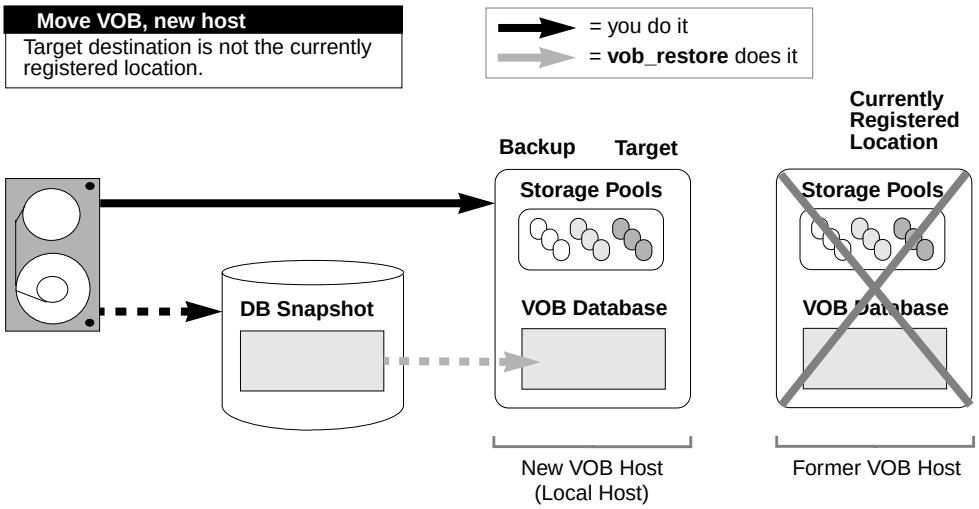
When moving a VOB as part of a **vob_restore** operation, the target host must have the same architecture (hardware and operating system). (See Figure 11.)

NOTE: Because a ClearCase LT client can only access VOBs on a single ClearCase LT server, moving a VOB off a ClearCase LT server makes it inaccessible to clients of that server. In a ClearCase LT environment, server-to-server VOB moves are generally undertaken only in conjunction with a planned change in ClearCase LT server host hardware. In this scenario, you must run **vob_restore** on the target host.

Formula. *target* different from *currently-registered-location*

Advice. Load the backup directly into the target destination.

Figure 11 vob_restore: Move VOB to New Host



vob_restore: Unregistered

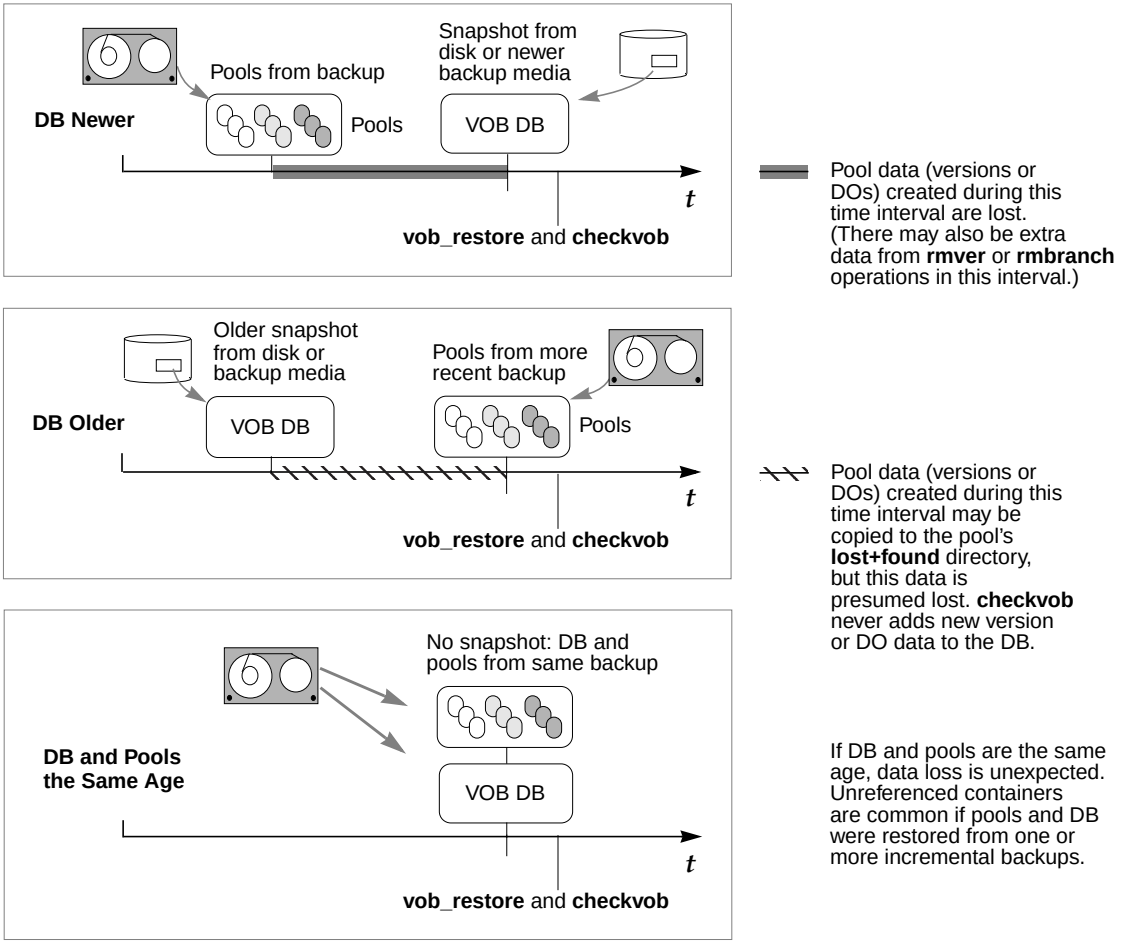
Formula. Same as in place, but no *currently-registered-location*

Advice. Load the backup directly into the target destination.

vob_restore: Restoring with a Database Snapshot

Any restore scenario that includes a VOB database snapshot introduces an additional complication: the VOB database and VOB storage pools have different reference times; that is, one is newer than the other. This skew must be resolved in the restored VOB. Run **checkvob** to resolve it. Figure 12 illustrates the problem and summarizes what **checkvob** does when the a VOB database is newer or older than the VOB's storage pools (the *backup*) at restore time.

Figure 12 VOB Database or Storage Pool Skew Associated with VOB Snapshots



NOTE: Derived objects (DOs) are not supported by ClearCase LT. ClearCase LT VOBs include DO pools, but the pools never contain DOs. Both **vob_restore** and **checkvob** include code to deal with DOs and DO pools. This code may produce log output that refers to DOs, but you can ignore such output in a ClearCase LT environment.

VOB snapshots have minimal impact on the your work at restore time. When prompted by **vob_restore**, you need only supply the correct snapshot directory. However, the **checkvob** portion of a **vob_restore** operation is critical. You must be prepared to respond intelligently to **checkvob** prompts and to understand the implications of any errors reported in its output.

Typically, **checkvob** finds and repairs numerous small problems, but it reports version data loss for the interval between storage pool and VOB database reference times.

For more information, see the **checkvob** reference page and Chapter 13, *Using checkvob*.

9.4 Restoring a VOB from Backup Without vob_restore

In most cases, the easiest way to restore your VOB is to run the **vob_restore** utility and follow its instructions. The procedure presented here is an alternative approach for circumstances in which you cannot use **vob_restore** and for administrators who prefer to do it themselves.

The following procedure restores a VOB backup without disrupting ongoing work. The VOB is the same one discussed in *Backing Up a VOB* on page 112.

1. **Log on to the ClearCase LT server.** Log on as a user with permission to stop and start ClearCase; typically this is the **root** user on UNIX or a local administrator on Windows.

2. **Check available disk space**. If you are restoring the VOB to a new location, make sure that there is enough free space in the VOB's disk partition to load the backup copy into temporary storage. Use the VOB storage node in the ClearCase Administration Console or the **cleartool space** command to check available disk space.

   ```
   # cleartool space /vobstore/flex.vbs
     Use(Mb)  %Use  Directory
        27.0    2%  VOB database /net/pluto/vobstore/flex.vbs
        33.0    3%  cleartext pool /net/pluto/vobstore/flex.vbs/c/cdft
      .
      .
   ----------------------------------------------------------------------
      312.9   28%  Subtotal
      828.4   74%  Filesystem /net/pluto/vobstore (capacity 1115.1 Mb)
   ```

   If the available space is insufficient, delete the VOB storage directory, or use other means to make enough space available.

WARNING: Step #3 and Step #4 are critical to the integrity of your restored VOB.

3. **Shut down ClearCase on the ClearCase LT server.** This ensures that ClearCase processes associated with the VOB are terminated. On Windows, use the ClearCase program in Control Panel. On UNIX, use the following command:

```
# ccase-home-dir/etc/atria_start stop
```

4. **Rename the VOB storage directory**. If it still exists, rename the VOB storage directory. A new one with the same name will be created during restore.

   ```
   # mv /vobstore/vob_flex.vbs /vobstore/vob_flex.OLD
   ```

5. **Restart ClearCase on the ClearCase LT server.** Starting ClearCase makes other VOBs on the ClearCase LT server available. Do this on Windows by using the **ClearCase** program in Control Panel. On UNIX, use the following command:

   ```
   # ccase-home-dir/etc/atria_start start
   ```

6. **Load the backup**. Re-create the VOB storage directory if necessary; then restore the VOB storage directory's contents from the backup medium.

   ```
   # mkdir /vobstore/vob_flex.vbs
   # cd /vobstore/vob_flex.vbs
   <enter restore command>
   ```

   If your Windows backup tool does not backup and restore ACLs correctly, you may need to fix them now. See Chapter 20, *Repairing VOB and View Storage Directory ACLs on Windows*, for details.

   NOTE: Each UNIX VOB storage area includes a directory named **.identity**, which stores files with special permissions: the *setUID* bit is set on file **uid**; the *setGID* bit is set on file **gid**. You must preserve these special permissions when you restore a VOB backup:

   - If you used **tar**(1) to back up the VOB, use the **–p** option when restoring the VOB. In addition, make sure to enter the **tar** command as the **root** user. If you do not, the **–p** flag is ignored.
   - If you used **cpio**(1) to back up the VOB, no special options are required in the **cpio** command that restores the backup data.

7. **Lock the VOB.** As a precaution, lock the VOB as soon as the restore is complete. This will prevent any users from changing VOB data until the restore has been checked for consistency.

8. **If you restored the VOB to a new location, re-register the VOB**. For example, if you restored the VOB to new location **/vobst_aux/flex.vbs**:

   ```
   # cleartool unregister –vob /vobstore/flex.vbs   (run unregister first)
   # cleartool register –vob /vobst_aux/flex.vbs
   # cleartool mktag –vob –replace –tag /vobs/flex /vobst_aux/flex.vbs
   ```

9. While the VOB is still locked, run some consistency checks. Load VOB data into a view and confirm that it is intact by checking event history on various components, examining recently changed elements, running **checkvob** as described in *Using checkvob* on page 175, and so on.

10. **Unlock the restored VOB**.

    ```
    # cleartool unlock vob:/flex
    Unlocked versioned object base "/vobstore/flex.vbs".
    ```

11. **If necessary, resynchronize the VOB and views.** See *VOB and View Resynchronization* on page 142.

9.5 Restoring an Individual Element from Backup

If you mistakenly delete an element with **rmelem**, you can restore it from a backup copy, using the procedure presented in this section.

NOTE: This procedure cannot be used to recover an element in a UCM (component) VOB.

Removing a directory element does not remove the file elements cataloged within it; file elements exist independently of directory elements. In many cases, deleting a directory element causes the files within it to be transferred to the VOB's **lost+found** directory. Look there first if you've accidentally removed a directory element.

This is the procedure for restoring a single element from a backup of a non-UCM VOB:

1. Unregister the VOB whose element has been deleted.
2. Restore the most recent backup of the VOB storage directory to a temporary location on the ClearCase LT server.
3. Register the restored backup.
4. Create a temporary VOB to hold the element you need to restore. Because the VOB and the restored backup cannot both be active at the same time, this temporary VOB serves as a staging point to which the element you're recovering can be copied.
5. Use **cleartool relocate** to relocate the element from the backup VOB to the temporary VOB.
6. Unregister the backup VOB.

7. Re-register the original VOB.

8. Use **cleartool relocate** to relocate the element from the temporary VOB to the original VOB.

9. Delete the backup VOB and the temporary VOB.

As an example, suppose that file element **util.c** is deleted from directory **\proj\src**. The VOB-tag is **\proj**, and the VOB storage directory is **c:\vobstore\proj.vbs** on the local host. Here's how the VOB owner can restore the element from a backup copy.

NOTE: The procedure described here restores the element to the version of its directory that is selected by the view in which the procedure takes place. It does not restore the element to earlier versions of the directory.

1. **Remove the VOB-tag and registry entries**. These entries prevent use of an old version of the same VOB.

   cleartool rmtag –vob \proj
   cleartool unregister –vob \\sol\vobstore\proj.vbs

2. **(UNIX server only) Terminate the VOB's server processes**. Search the process table for the **vob_server** and **vobrpc_server** processes that manage that VOB. Use **ps –ax** or **ps –ef**, and search for `/vobstore/proj.vbs`; use **kill**(1) to terminate any such processes. (Only the **root** user can kill a **vobrpc_server** process.)

3. **Restore the VOB storage directory from the backup to a temporary location**. Because this is a temporary copy that only one client will need to access for a brief period, you may want to follow the procedures for a simple manual VOB restore as described in *Restoring a VOB from Backup Without vob_restore* on page 137.

4. **Register the backup VOB**. Use **cleartool** commands like the following:

   cleartool register –vob \\sol\users\tmp\proj.vbs
   cleartool mktag –vob –tag \oldproj \\sol\users\tmp\proj.vbs

5. **Create a new, temporary VOB.** Because the original VOB and the backup VOB cannot be active at the same time, you need this temporary VOB to hold a copy of the element you're recovering. Later, you can unregister the backup, register the original VOB, and copy the element from the temporary VOB to the original VOB. Unless the element is very large, this temporary VOB does not need much disk space.

```
cleartool mkvob –nc –tag \tmpvob \\sol\users\tmp\tmpvob.vbs
Created versioned object base.
  .
  .
```

6. **Relocate the element from the backup VOB to the temporary VOB.** Use the **cleartool relocate** command to relocate the element into the temporary VOB. Create the data file in the old VOB and run **clearimport** in the temporary VOB. In this example, **Z:** is the root of a view that uses the default config spec.

   ```
   z:\> cd oldproj\src
   z:\oldproj\src> cleartool relocate util.c \tmpvob
   ```

7. **Unregister the backup VOB**. You can use these **cleartool** commands to unregister the VOB.

   ```
   cd \
   cleartool rmtag –vob \oldproj
   cleartool unregister –vob \\sol\users\tmp\proj.vbs
   ```

8. **(UNIX server only) Terminate the backup VOB's server processes**. This is similar to Step #2 of this procedure. This time, search the process table for a **vob_server** and/or **vobrpc_server** invoked with **/usr/tmp/proj.vbs**.

9. **Re-register the original VOB**. You can use these **cleartool** commands to re-register the VOB.

   ```
   cleartool register –vob \\sol\vobstore\proj.vbs
   cleartool mktag –vob –tag \proj \\sol\vobstore\proj.vbs
   ```

   NOTE: If you are registering a UCM project VOB, you must use the **–ucmproject** option with the **register** command.

10. **Relocate the element from the backup VOB to the temporary VOB.** Use the **cleartool relocate** command program as in Step #6 to relocate the element from the temporary VOB to the original VOB.

    ```
    z:\> cd \tmpvob
    z:\tmpvob> cleartool relocate util.c \proj\src
    ```

11. **Clean Up.** Unregister the temporary VOB. Remove the temporary VOB and the backup VOB. Use the VOB storage node for the VOB in the ClearCase Administration Console or the **rmvob** command, which removes the VOB's registry entries and terminates all of its server processes.

cleartool rmvob c:\users\tmp\tmpvob.vbs

```
Remove versioned object base "c:\users\tmp\tmpvob.vbs"[no] yes
Removed versioned object base "c:\users\tmp\tmpvob.vbs".
```

cleartool rmvob c:\users\tmp\oldproj.vbs

```
Remove versioned object base "c:\users\tmp\oldproj.vbs"[no] yes
Removed versioned object base "c:\users\tmp\oldproj.vbs".
```

9.6 VOB and View Resynchronization

A VOB database maintains references to one or more view databases, and vice versa. When a VOB or view is restored from backup, the view and VOB are potentially out of sync; some of the references are no longer valid. This skew can cause several problems.

- Elements can become "checked out but removed" when the VOB has recorded that the view has the element checked out, but the view doesn't have a view-private file for the element.
- Elements can become *eclipsed* by a view-private file when the checkout that was once valid in the view is no longer recognized in the VOB. (The VOB says the element isn't checked out in the view.)

There are a several things you can do to correct the skew and identify and/or eliminate the problems. You can perform the resynchronization described in *Resynchronizing Views and VOBs* on the restored view immediately following VOB or view restore, or you can wait until problems surface.

Resynchronizing Views and VOBs

To resynchronize views and VOBs, run **cleartool ls –recurse –view_only**. Look at the output for any checkouts that are marked as checked out but removed.

If the VOB was restored, the user had probably once checked out the version but then either checked it in or canceled the checkout after the VOB was backed up but before it was restored.

- If the version was checked in, the data has been lost.
- If the view was restored, any view-private modifications to the checked-out version have been lost.

In either case, the user must now cancel the checkout (**uncheckout**) or re-create the version data and check it in.

10 Administering VOB Storage

VOB storage administration has two primary goals:

- Preserving critical data and metadata
- Managing the disk space required for VOB storage

Implementation of a daily routine for backing up VOB data is the more critical task of the two. See Chapter 9, *Backing Up and Restoring VOBs* for more information on this topic. After you have implemented a backup and recovery strategy for VOB storage, consider the ways in which you can minimize the amount of storage your VOBs require and manage distribution of that storage across an appropriate set of network resources. These topics are covered in this chapter.

10.1 VOB Storage Management

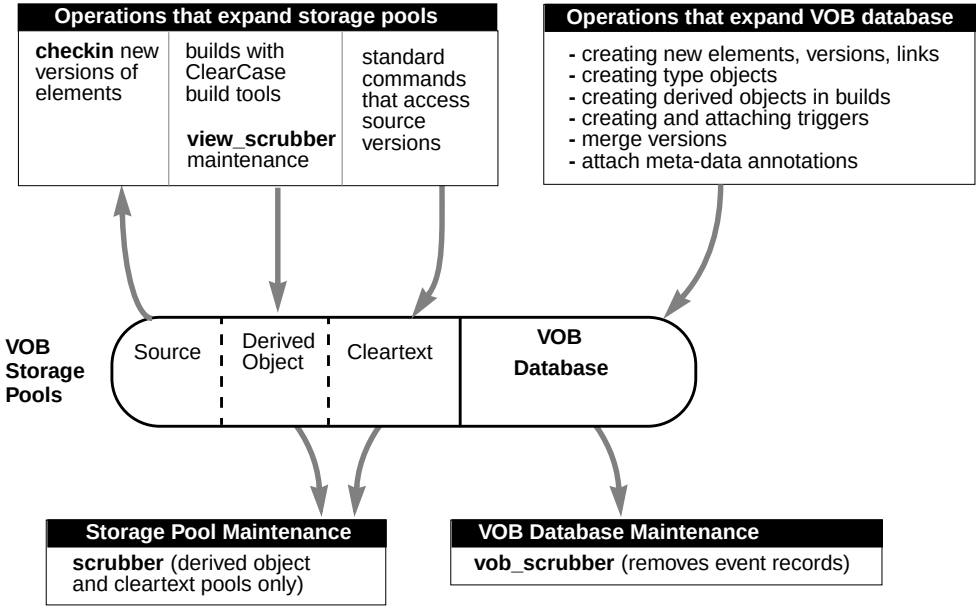
VOB storage grows in proportion to the number of developers using the VOB and the rate at which they create and change the data under ClearCase LT control. Much of this data needs to be preserved, often for an extended period. But some of it loses value quickly and can be safely removed from the VOB. Rational ClearCase LT provides tools that perform the following tasks:

- Monitor the disk space used by VOBs
- Scrub VOBs to remove unneeded data and metadata on a schedule you define, using criteria you specify
- Relocate data from one VOB to another
- Remove versions and, if necessary, elements when they are no longer needed

You can use any or all of these tools to keep VOB storage requirements to a practical minimum.

Figure 13 shows how VOB storage pools and VOB databases grow in regular use; it also lists the maintenance commands (scrubbers) that control growth of these storage areas.

Figure 13 Controlling VOB Growth



Monitoring VOB Storage

ClearCase LT provides command line and GUI tools that display information about disk space used by VOBs.

- In the ClearCase Administration Console, the VOB storage node for a VOB subnode of the ClearCase LT Server node shows current and historical disk space use for the VOB.
- The **cleartool space –vob** command shows current and historical disk space use for a VOB.

Using the Scheduler

The ClearCase LT scheduler runs several jobs that gather data about disk space used by VOBs and that can reclaim excess disk space used by local VOBs:

- Daily data gathering on VOB disk space used
- Daily scrubbing of VOB storage pools using the **scrubber** utility
- Weekly scrubbing of VOB databases using the **vob_scrubber** utility
- Daily and weekly execution of jobs that you can customize to run your own programs

In their default configuration, many of these jobs do little or nothing. (For example, the daily VOB space monitoring job does not gather statistics on any VOB until it has been configured to do so.) We recommend reviewing the list of scheduled jobs and enabling all of the ones your VOB storage management routine requires. For more information on the ClearCase LT scheduler, see Chapter 16, *Managing Scheduled Jobs*.

10.2 Scrubbing to Control VOB Storage Growth

The first step in managing the growth of VOB storage is to understand the **scrubber** and configure each VOB's pools to be scrubbed in a way that is appropriate to the growth and access patterns of the VOB. Several basic rules the govern use of the **scrubber**:

- Cleartext pools are scrubbed to control their size. These pools are essentially caches; scrubbing unneeded data containers in a cleartext pool has little or no effect on ClearCase LT performance.

- Source pools are never scrubbed.

In the default configuration, the **scrubber** runs nightly as part of a scheduled task managed by the ClearCase LT schedule service. You can also run it manually.

ClearCase LT also includes a **vob_scrubber** utility that can remove unneeded metadata from the VOB database itself.

Scrubbing VOB Storage Pools

Storage pools are scrubbed by the **scrubber** utility. Each cleartext storage pool has its own scrubbing parameters, which are set when ClearCase LT is installed.

Scrubbing VOB Databases

Almost every change to a VOB is recorded in the VOB database as an *event record*. Some event records have permanent value, such as those for the creation of elements and versions. Others may not be useful to your organization or may lose their value as time passes. (For example, you probably don't care about the removal of an unneeded or obsolete version label.)

Each ClearCase LT server has a scrubber configuration file, *ccase-home-dir***config\vob\vob_scrubber_params**, which controls **vob_scrubber** operation for all VOBs. If you need more control over scrubbing schedules, you can create a scrubber parameters file for each VOB. This file is also named **vob_scrubber_params**, but it is located in the VOB storage directory. See the **vob_scrubber** reference page for more information.

NOTE: Deleting event records and other metadata from the VOB database, using the **vob_scrubber** or any other mechanism (**rmver**, **rmelem**, **relocate**, and so on) increases the amount of free space within these files so that they can accommodate newly created event records, but does not reduce the disk space used by the VOB database. A regularly scrubbed VOB database grows slowly and should not require further intervention to keep growth under control. However, if you must occasionally force a reduction in the size of a VOB database, scrub it, and then run the **reformatvob** command.

Adjusting Default Scrubbing Parameters

Typical motivations for adjusting a VOB host's default procedures for storage pool scrubbing include:

- **Not enough space.** The disk partitions in which VOB storage pools reside may fill up frequently. A more aggressive scrubbing strategy may be necessary.
- **Not enough time.** The **scrubber** utility may be taking too much time to complete, interfering with other overnight activities, such as nightly software builds. A less aggressive scrubbing strategy may be necessary.

You may need to experiment. For example, adjusting scrubbing to take place less frequently may cause disk-space problems that you had not previously experienced. Before making any scrubbing adjustments on a VOB host, be sure to analyze its **scrubber_log** file. The **scrubber** reference page explains how to read this file.

NOTE: If you use a Windows backup tool that changes the time stamps in VOB storage directories, the DO and cleartext pools in those directories may never be scrubbed. The **scrubber** command, by default, scrubs objects that have not been referenced for the last 96 hours. If such a backup tool runs every night, the access time on objects in the pools is reset every night and the objects are never scrubbed.

The following sections present some simple examples of adjusting the way VOB storage pools are scrubbed.

Scrubbing Derived Objects More Often

By default, **scrubber** allows data containers of unreferenced derived objects to remain in their storage pools for 4 days (96 hours). If DOs are filling up a VOB's disk partition, you can shorten this grace period. This command empties a VOB's default DO storage pool (**ddft**) of unneeded data containers every 24 hours:

cleartool mkpool –update –age 24 ddft@*vob-tag*
```
Updated pool "ddft".
```

Fine-Tuning Derived Object Scrubbing

Suppose that the adjustment in the grace period is not enough to keep the disk partition from filling up. You may decide to run the **scrubber** utility more often: during the work day as well as overnight. To minimize the impact on users during the work day, you can pinpoint the scrubbing—perhaps to the DOs created in a particular directory. This example shows the UNIX command line syntax for moving a DO pool and scrubbing it more often:

1. **Determine the directory's current DO storage pool assignment.** You will need to clean up this storage pool.

   cd /vobs/proj
   cleartool describe –long reorg@@
   ```
   directory element "reorg@@":
     .
     .
     ...  derived pool: ddft
   ```

   This directory uses the default DO pool.

2. **Assign the directory to a separate storage pool.** This assignment enables finer control of scrubbing, which can be invoked on a per-pool basis:

   cd /vobs/proj
   cleartool mkpool –derived new_do_pool
   `Comments for "new_do_pool":`
   pool for DOs created in /vobs/proj/reorg
   .
   `Created pool "new_do_pool".`
   cleartool chpool new_do_pool reorg
   `Changed pool for "reorg" to "new_do_pool".`

3. **Determine the location of the VOB storage directory**. You'll need the pathname of the VOB's storage directory for **scrubber**. Use **lsvob** to determine the pathname:

   cleartool lsvob /vobs/proj
   `* /vobs/proj /net/ccsvr03/vobstore/proj.vbs`

4. **Scrub the new storage pool thoroughly and often**. There are many ways to accomplish this. You can create a new task for the ClearCase LT scheduler that invokes the **scrubber** utility for your new pool:

   ```
   "$ATRIAHOME/etc/scrubber –e –p new_do_pool \
      /net/ccsvr03/vobstore/proj.vbs"
   ```

   This script invokes the **scrubber** utility on the derived object storage pool **new_do_pool**. The **–e** option to **scrubber** empties the pool of all zero-referenced DOs.

   You can then register your task in the scheduler's task database and create a new scheduled job to execute the task several times per day. For more information on tasks and jobs, see Chapter 16, *Managing Scheduled Jobs*.

5. **Clean up the old DO storage pool**. The **chpool** command in Step #2 does not move existing DO data containers; it only affects where a new DO's data container is stored. Accordingly, you should clean up the old storage pool:

   *ccase-home-dir***/etc/scrubber –e –p ddft /net/ccsvr03/vobstore/proj.vbs**

Scrubbing Less Aggressively

If the ClearCase LT scheduler's scrubbing regimen takes too long (perhaps spilling over into the work day), you can make the starting time for the ClearCase LT default Daily VOB Pool Scrubbing job earlier. Alternatively, you can disable the Daily VOB Pool Scrubbing job and define your own job that changes the way that scrubber is invoked, so that it takes less time to run.

Here's how you can revise scrubbing to process DO pools only, leaving cleartext pools alone:

1. Define a task whose executable program invokes the scrubber as follows:

   ```
   ccase_home_dir/etc/scrubber –f –a –k do
   ```

2. Register the task in the scheduler's task database.

3. Define a new job in the scheduler that runs your task daily. Choose an appropriate starting time.

4. Disable the ClearCase LT default Daily VOB Pool Scrubbing Job in the scheduler. You can disable a job by setting its end date to a time in the past or by deleting the job.

5. Check all other jobs with sequential schedules. Change the schedule for any job that is defined to follow the ClearCase LT default Daily VOB Pool Scrubbing job to follow your new job instead. The ClearCase LT default Daily VOB Snapshots job follows Daily VOB Pool Scrubbing, so you must change this job to follow your new job.

10.3 Removing Unneeded Versions from a VOB

Scrubbing only removes artifacts that can be regenerated. Scrubbing never removes versions of elements. Because elements and versions are historical data, you should approach their removal with extreme caution. Removing entire elements, using **rmelem**, is particularly dangerous:

- Even if an element is no longer needed for ongoing work, you may need it to reproduce and maintain earlier work.

- **rmelem** removes the element's name from all directory versions in which it was ever cataloged. This erasing of history means that the element does not appear in listings or comparisons of old directory versions. Removing an element's name from a VOB directory, using the **rmname** command, preserves its history but makes its name invisible in subsequent versions of the directory.

- Making a mistake can be costly; there is a procedure for recovering from backup an element that was deleted mistakenly, but it's cumbersome. (See *Restoring an Individual Element from Backup* on page 139.)

If you need to reclaim disk space, it is more prudent to remove individual versions of elements, rather than entire elements. The **rmver** command makes it easy to remove versions that you believe you will probably never need again.

By default, **rmver** removes only versions of little use:

- Versions that are unrelated to branching: not located at a branch point and not the first or last version on a branch
- Versions that have no metadata annotations: version labels, attributes, or hyperlinks

You can also use the **cleartool relocate** command to move entire VOB directories and all the element versions they contain from one VOB to another. Chapter 14, *Splitting VOBs with relocate*, has more information on this procedure.

11 Moving VOBs

VOBs cannot be copied from one location to another using an ordinary file copy utility. Special procedures must be followed to maintain the integrity of VOB data and to preserve permissions and ownership on VOB storage directories. This chapter presents procedures for moving VOBs. The following scenarios are covered:

- Moving a VOB from one disk partition to another on Windows or between two Windows hosts in the same domain.
- Moving a VOB from a Windows host in one domain to a Windows host in another domain.
- Moving a VOB from one disk partition to another on UNIX or between one UNIX host and another UNIX host with the same binary format.
- Moving a VOB from one UNIX host to another UNIX host with a different binary data format.
- Moving a VOB from a Windows host to a UNIX host.
- Moving a VOB from a UNIX host to a Windows host.

NOTE: Only a ClearCase LT server can be a VOB host.

11.1 Important Steps to Take When Moving Any VOB

Although the procedures described in this section differ in various ways, the following steps should be considered when moving any VOB.

- Unless you are moving a VOB between UNIX and Windows, or from one Windows domain to another, make sure that the ownership and access control information for the VOB storage directory is preserved when the directory is copied. Not all file system copy utilities—especially on Windows—preserve this information. If this information is changed during the copy step of the VOB move procedure, the VOB will not be usable in its new location until the protections are corrected. Rational ClearCase provides several programs that can correct damaged file system protections if necessary. See Chapter 20, *Repairing VOB and View Storage Directory ACLs on Windows* for more information on this topic.

- All of the VOB move procedures described in this chapter preserve the original VOB storage directory. After you are sure that the VOB can be accessed at its new location, delete the old VOB storage directory to free the storage it would otherwise consume.

- If VOB database snapshot backups are enabled for the VOB, use the **vob_snapshot_setup** program to disable them before you begin the move, and enable them again after the move is complete.

- Optionally, scrub the VOB's cleartext pools before moving the VOB. This reduces the size of the VOB storage directory.

CAUTION: Never register more than one copy of a VOB in a ClearCase registry. Although the procedures described in this chapter copy the VOB storage directory, none of them registers the copy before the original has been unregistered. If more than one copy of a VOB is registered in the same registry, even if they have tags in separate regions, severe ClearCase errors, including potential data loss, will occur.

11.2 Moving a VOB on Windows

This section describes procedures for the following types of VOB moves involving Windows platforms:

- moving a VOB to another host in the same domain. This procedure also applies when moving a VOB from one partition to another on a host.

- moving a VOB to a host in another domain

Moving a VOB Within a Domain

The following procedure describes how to move the VOB **\libpub** from storage directory **C:\ClearCaseStorage\VOBs\libpub.vbs** on VOB server host **\\sol** to a storage directory shared as **vobstg** on VOB server host **\\ccsvr01**. The procedure to move the VOB to a new partition on **\\sol** would be similar.

1. **Log on to the VOB's current server as the VOB owner or privileged user.** In this example, the VOB's current server is **\\sol**.

2. **Lock the VOB**.

3. **Stop ClearCase** on the VOB server host.

4. **Copy the VOB storage directory, preserving all ownership information**. You must use a copy utility that preserves ownership information contained in the VOB storage directory ACLs. Typical Windows file system copy utilities like **copy** and **xcopy** do not preserve this information. We recommend using the ClearCase utility *ccase-home-dir***\etc\utils\ccopy** for this purpose.

   C:\ClearCaseStorage\VOBs> net use E: \\ccsvr01\vobstg
   C:\ClearCaseStorage\VOBs> ccopy libpub.vbs E:\libpub.vbs

   NOTE: Although **ccopy** copies all of the ownership information required by ClearCase, it does not copy the full security descriptor of an object, and therefore effectively grants the user who executes this step full access to the copy of the VOB storage directory. If this step is not executed by the VOB owner, you may want to use a different copy program, such as **scopy /s /o** from the Windows NT Resource Kit or **xcopy /o** on Windows 2000 or Windows XP, that copies the VOB storage directory but does not grant the user additional rights to it.

5. **Restart ClearCase** if you have copied the VOB storage directory to a new location on the same VOB server host.

6. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console, or the following commands (this example applies to the destination on server **ccsvr01**):

   cleartool register -vob -replace \\ccsvr01\vobstg\libpub.vbs
   cleartool mktag -vob -replace –tag \libpub \\ccsvr01\vobstg\libpub.vbs

   NOTE: If you are registering a Unified Change Management project VOB, you must supply the **–ucmproject** option to the **register** command.

7. **Unlock the VOB.**

8. **Verify that all clients can access the VOB at the new location.**

Moving a VOB to a Different Domain

On Windows, VOBs formatted with schema version 54 store Windows security identifiers (SIDs) to represent users, groups, and resources (hosts). When you move a VOB to a different domain, these SIDs become invalid and must be changed (mapped) to ones that are valid in the new domain. ClearCase LT includes a utility program, **vob_sidwalk**, that provides a flexible means of mapping SIDs when you move a VOB to a different domain. We strongly recommend that you review the reference page for **vob_sidwalk** before continuing with this procedure.

The following procedure describes how to move the VOB **\libpub** from storage directory **C:\ClearCaseStorage\VOBs\libpub.vbs** on VOB server host **\\sol**, which is in the **OLD** domain, to a storage directory shared as **vobstg** on VOB server host **\\ccsvr-new**, which is in the **NEW** domain. To execute this procedure, you must be able to log in to both the **OLD** and **NEW** domains as the VOB owner of **\libpub** or as the privileged user.

1. **Be sure the VOB has been formatted with schema version 54**. Earlier schema versions do not support moving a VOB across domains. You can use the ClearCase Administration Console or the **cleartool describe** command to determine a VOB's schema version. If the VOB is not formatted with schema version 54, reformat it using **reformatvob**. (All VOBs on a server must be formatted with the same schema version.)

2. **Log on to the VOB server host as the VOB owner or privileged user.** In this example, the VOB server host for this step is **\\sol**.

3. **Lock the VOB.** This ensures that no new VOB objects will be accidentally created while you are performing Step #4 of this procedure.

4. **Generate a SID file** that lists the names of users and groups associated with objects in **\libpub**. Run **vob_siddump** as shown in the following example to generate a SID file in comma-separated-value (csv) format:

   *ccase-home-dir***\etc\utils\vob_siddump \libpub ^**
   C:\ClearCaseStorage\VOBs\libpub.vbs\libpub.csv

   We suggest creating the SID file in the VOB storage directory so that it will be available on the new VOB host after the storage directory has been moved. You will need it Step #10 of this procedure.

5. **Stop ClearCase** on the VOB server host **\\sol**.

6. **Copy the VOB storage directory** to the new location.

   C:\ClearCaseStorage\VOBs> net use E: \\ccsvr-new\vobstg
   C:\ClearCaseStorage\VOBs> ccopy libpub.vbs E:\libpub.vbs

   NOTE: Because the existing VOB storage directory ACLs will be meaningless in the new domain, you can use **xcopy** or a similar utility instead of the ClearCase utility *ccase-home-dir***\etc\utils\ccopy** in this procedure.

7. **Fix the VOB storage directory protections.** Log on to the VOB server host in the new domain (**\\ccsvr-new** in our example) as the VOB owner for **\libpub** or as a privileged user. Run the **fix_prot** utility as shown in the following example, where **vobadm** is the name of the new VOB owner, **ccusers** is the name of the VOB's new principal group, and **V:\vobstg\libpub.vbs** is the host-local pathname of the VOB storage directory on **\\ccsvr-new**.

   *ccase-home-dir***\etc\utils\fix_prot -root -r -chown vobadm -chgrp ccusers ^**
   V:\vobstg\libpub.vbs

8. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console, or the following commands:

   cleartool register -vob -replace \\ccsvr-new\vobstg\libpub.vbs
   cleartool mktag -vob -replace –tag \libpub \\ccsvr-new\vobstg\libpub.vbs

   If **\\ccsvr-new** is not in the same registry region as **\\sol**, you do not need to use the **-replace** option to **cleartool register** and **cleartool mktag**, but you should remove the old registration and tag for **\libpub**, which will be invalid after the move.

   NOTE: If you are registering a Unified Change Management *Process VOB*, you must supply the **–ucmproject** option to the **register** command.

9. **Lock the VOB.** Although the VOB is now registered and has a tag, it will not be usable until this procedure is complete. If you are concerned that users may try to access the VOB before it is ready, lock it now.

10. **Create a map file.** Open the SID file you generated in Step #4 of this procedure (**\\ccsvr-new\vobstg\libpub.vbs\libpub.csv**). Editing this file may be simplified if you use a spreadsheet program that can read the comma-separated-value format. This example

shows one line of such a file. We've added a header row for clarity and truncated the SID string to save space.

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
OLD\akp	USER	NT:S-1-2-21-532...	IGNORE	USER		137

For each line in the file, replace the string **IGNORE** in the **New-name** field with a string made up of the new domain name and the user name from the **Old-name** field; then delete the last three fields (**Type**, **New-SID**, and **Count**). In this example, old domain's name is OLD and the new domain's name is NEW, so the line would change as shown here:

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
OLD\akp	USER	NT:S-1-2-21-532...	NEW\akp			

Although this example shows a user name that is the same in the old and new domains, the procedure can also be used to map a user or group name from the old domain to a different user or group name in the new domain.

After you have edited all the rows of the SID file, save it as a comma-separated-value file and use it as the mapping file required when you run **vob_sidwalk -map**. Each line of the mapping file must have exactly four fields, separated by commas. The example row created in this step would look like this in .csv form:

OLD\akp,USER,NT:S-1-2-21-532...,NEW\akp

NOTE: You may reassign ownership of any object in a VOB to the VOB owner by placing the string **DELETE** in the **New-name** field. You may also reassign ownership of all objects in a VOB to the VOB owner without creating a mapping file. See *Reassigning Ownership to the VOB Owner* on page 49.

11. **Test the map file.** Run **vob_sidwalk** without the **-execute** option. The list of mappings prescribed by the map file **libpub-map.csv** is written to the SID file (**libpub-test.csv** in this example), but no changes are made to the VOB.

    *ccase-home-dir***etc\utils\vob_sidwalk -map ^**
    \\ccsvr-new\vobstg\libpub.vbs\libpub-map.csv \libpub libpub-test.csv

12. **Unlock the VOB.** If you are concerned that users may try to access the VOB before this procedure is complete, lock the VOB again for all users except yourself.

13. **Update user and group identities stored in the VOB.** When you are satisfied that the map file is correct, run **vob_sidwalk** as shown in the following example, where **libpub-map.csv** is the map file you created in Step #10 of this procedure:

    *ccase-home-dir***etc\utils\vob_sidwalk -execute -map ^**
    \\ccsvr-new\vobstg\libpub.vbs\libpub-map.csv \libpub libpub-exec.csv

    vob_sidwalk remaps ownership as specified in the map file and records the changes that were made in **libpub-exec.csv**.

14. **Recover file system ACLs.** Finally, while you are still logged in to **\\ccsvr-new** as the VOB owner or privileged user, use **vob_sidwalk** with the **-recover_filesystem** option to apply the correct ACLs to the VOB storage directory.

    *ccase-home-dir***etc\utils\vob_sidwalk -recover_filesystem \vobstore2\libpub**

15. **Verify that all clients in the new domain can access the VOB.** Unlock the VOB if it is still locked.

16. **Verify that all ClearCase users in the new domain can access objects in the VOB.** Users should be able to create new objects as well as change or remove objects they own.

    NOTE: If the user's name in the new domain is not the same as in the old domain, the user ill loses rights associated with the creator of a version or a branch. For example, the user would not be able to remove a version even though the user had created that version. These operations can still be executed by a more privileged user (VOB owner, member of the ClearCase administrators group).

11.3 Moving a VOB on UNIX

This section describes procedures for the following types of VOB moves involving UNIX platforms:

- Moving a VOB to another UNIX VOB server host that has the same architecture (binary data format). This procedure also applies when moving a VOB from one partition to another on a UNIX VOB server host.
- Moving a VOB to another UNIX VOB server host that has a different architecture.

For clarity, the procedures in this section use an example:

- The current location of the VOB storage directory to be moved is **/vobstore/libpub.vbs**, on a host named **sol**.
- The VOB tag is **/vobs/libpub**.
- The new location for the VOB storage directory is **/vobstore2/libpub.vbs**. The example includes these cases:
  - The new location is also on **sol**.
  - The new location is on another host, named **ccsvr04**.

Moving a VOB Between UNIX Hosts (Same Architecture)

Use the following procedure to move a VOB from one disk partition to another on a UNIX VOB server host, or between one UNIX VOB server host and another one with the same architecture.

1. **Log on to the VOB server host as root.**
2. **Lock the VOB**.
3. **Stop ClearCase** on the VOB server host:
4. **Copy the VOB storage directory**. The procedure you use depends on whether you're moving the VOB within the same disk partition or to another disk partition.

   a. If you are moving the VOB to another disk partition, use **tar** or a similar command to copy the entire VOB storage directory tree, but not any remote storage pools, to the new location.

   - To the same host on a different disk partition:

     ```
     # cd /vobstore
     # tar –cf – libpub.vbs | ( cd /vobstore2 ; tar –xBpf – )
     ```

   - To a different host:

     ```
     # cd /vobstore
     # tar –cf – libpub.vbs | rsh ccsvr04 'cd /vobstore2 ; tar –xBpf –'
     ```

   NOTE: The **-B** option to the **tar** command may not be supported on all architectures. Also, the **rsh** command may have a different name, such as **remsh**, on some platforms. Refer to the reference pages for your operating system.

b. If you are moving the VOB storage directory within the same disk partition, use a simple **mv** command to relocate the VOB storage directory to the new location.

5. **Restart ClearCase** if you have copied the VOB storage directory to a new location on the same VOB server host.

6. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console or the following commands (this example applies to the destination on server **sol**):

   cleartool register -vob -replace /net/sol/vobstore2/libpub.vbs
   cleartool mktag -vob -replace –tag /vobs/libpub /net/sol/vobstore2/libpub.vbs

   NOTE: If you are registering a UCM project VOB, you must supply the **–ucmproject** option to the **register** command.

7. **Unlock the VOB.**

8. **Verify that all clients can access the VOB at the new location.**

Moving a VOB Between UNIX Hosts (Different Architectures)

Use the following procedure to move a VOB from one UNIX VOB server host to another UNIX VOB server host that has a different architecture. The procedure is similar to the one described in *Moving a VOB Between UNIX Hosts (Same Architecture)*, but includes the additional steps required to dump the VOB database before it is moved and reformat it on the target host.

1. **Log on to the VOB server host as the VOB owner or root.**

2. **Dump the VOB's database** to ASCII dump files using the **cleartool reformatvob** command. You do not need to lock the VOB beforehand; the **reformatvob** command does this automatically.

   # cleartool reformatvob –dump /vobstore/libpub.vbs

   reformatvob -dump marks the VOB database as invalid. It will be unusable by clients until it is processed by a **reformatvob –load** command.

3. **Copy the VOB storage directory**. First, make sure that the parent directory of the target location exists and is writable. Then, copy the VOB storage directory to the new host.

```
# cd /vobstore
# tar –cf – libpub.vbs | rsh ccsvr04 'cd /src/vobstore ; tar –xBpf –'
```

NOTE: The **-B** option to the **tar** commands may not be supported on all architectures. Also, the **rsh** command may have a different name, such as **remsh**, on some platforms. Refer to the *Platform-Specific Guide* in online help for more information or check the reference pages for your operating system.

4. **Terminate the old VOB's server processes**. You may either stop and re-start ClearCase on the VOB server host, or search the process table for the **vob_server** and **vobrpc_server** processes that service the old VOB. Use **ps –ax** or **ps –ef**, and search for the VOB storage directory name (**libpub.vbs** in our example); then use **kill** to terminate any such processes.

5. **Log on to the new VOB server host as the VOB owner or root.**

6. **Re-create the VOB database from the dump files**:

   ```
   # cleartool reformatvob –load /src/vobstore/libpub.vbs
   ```

7. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console, or the following commands (this example applies to the destination on server **sol**):

   ```
   cleartool register -vob -replace /net/sol/vobstore2/libpub.vbs
   cleartool mktag -vob -replace –tag /vobs/libpub /net/sol/vobstore2/libpub.vbs
   ```

   NOTE: If you are registering a UCM project VOB, you must supply the **–ucmproject** option to the **register** command.

8. **Unlock the VOB.**

9. **Verify that all clients can access the VOB at the new location.**

11.4 Moving a VOB Between Windows and UNIX

When you move a VOB from a Windows host to a UNIX host or vice versa, all user identity information stored in the VOB storage directory and VOB database must change. The binary data format of the VOB database must change as well. To accommodate these requirements, you must take a number of additional steps beyond those required in most other VOB move scenarios, including the following:

- Run **vob_sidwalk** before the move to capture information about ownership of objects in the VOB.
- Use **reformatvob** to dump the VOB database into a portable ASCII form.
- Copy the VOB storage directory, which includes the dumped database, to the new host.
- Use **reformatvob** to load the VOB database in the proper binary format.
- Reset the file system protections on the VOB storage directory after the move.
- Remap the SIDs (or, on UNIX, UIDs and GIDs) of owners of objects in the VOB.

These steps, along with other ones that are typical of all VOB moves, are detailed in the procedures described in this section.

NOTE: You can move a VOB between Windows and UNIX only if both hosts are configured to support schema version 54.

Schema Version Compatibility

vob_sidwalk and **vob_siddump** are not compatible with VOB schema version 53. **vob_sidwalk** is installed only on hosts that are configured to support local VOBs and views and to support VOB schema version 54. **vob_siddump**, which is not restricted to operating on local VOBs, is installed on all hosts.

Before a VOB formatted with schema version 54 can be moved from Windows to a UNIX host, the UNIX host must be configured to support VOB schema version 54. Not all UNIX hosts have this capability. You cannot reformat a VOB from schema version 54 to schema version 53. You cannot move a VOB formatted with schema version 53 from a UNIX host to a Windows host that supports schema version 54.

Moving a VOB from Windows to UNIX

For clarity, the procedures in this section use an example:

- The current location of the VOB storage directory to be moved is **C:\ClearCaseStorage\libpub.vbs** on Windows host **ccsvr-nt**. The VOB-tag for this VOB is **\libpub**.

➤ The new location for the VOB storage directory is **/vobstg/libpub.vbs** on UNIX host **ccsvr-ux**. VOB-tag **/vobs/libpub** will be created for this VOB.

To move a VOB from Windows to UNIX:

1. **Log on to the Windows VOB server host as the VOB owner or privileged user.** In this example, the VOB server host for this step is **\\ccsvr-nt**.

2. **Lock the VOB.** This ensures that new VOB objects are not created while you are performing Step #3 of this procedure.

3. **Generate a SID file** that lists the names of users and groups associated with objects in **\libpub**. Run the **vob_siddump** utility as shown in this example:

   *ccase-home-dir***\etc\utils\vob_siddump -raw_sid \libpub ^**
   C:\ClearCaseStorage\libpub.vbs\libpub.csv

   We suggest creating the SID file in the VOB storage directory so that it will be available on the new VOB host after the storage directory has been moved. You will need the SID file in Step #15 of this procedure.

4. **Dump the VOB database** using the **cleartool reformatvob** command:

   cleartool reformatvob –dump C:\ClearCaseStorage\libpub.vbs

   reformatvob -dump marks the VOB database as invalid. It cannot be used by clients until it is processed by a **reformatvob –load** command.

5. **Copy the VOB storage directory**. Use any file system copy utility to copy the entire VOB storage directory to the UNIX host. This example assumes that the target UNIX host **ccsvr-ux** is running an SMB server and has shared its **\vobstg** partition.

   C:\ClearCaseStorage\VOBs> net use E: \\ccsvr-ux\vobstg
   C:\ClearCaseStorage\VOBs> *ccase-home-dir***\etc\utils\ccopy libpub.vbs E:\libpub.vbs**

   NOTE: Because the existing VOB storage directory ACLs will be meaningless on the UNIX host, you can use **xcopy** or another copy program instead of the ClearCase **ccopy** utility in this case.

6. **Terminate the VOB's server processes on Windows**. Stop and restart ClearCase on the Windows VOB server host (**\\ccsvr-nt** in our example).

7. **Log on to the UNIX VOB server host as root.**

8. **Update VOB owner identity information.** Use the **fix_prot** utility as shown here to create a new **.identity** directory for the VOB. This example sets the VOB's owner to user **vobadm** and the VOB's primary group to **ccusers**:

   *ccase-home-dir***/etc/utils/fix_prot -root -recurse -chown vobadm -chgrp ccusers ^ /vobstg/libpub.vbs**

9. **Re-create the VOB database from the dump files**. Use the cleartool **reformatvob** command:

   # **cleartool reformatvob –load /vobstg/libpub.vbs**

10. **Create a tag for the VOB.** Use the ClearCase Administration Console, or the following command:

    cleartool mktag -vob -tag /vobs/libpub /net/ccsvr-ux/vobstg/libpub.vbs

    NOTE: You do not need to register the VOB, because **reformatvob -load** registers the VOB after the reformat is complete.

11. **Create a map file.** Open the SID file generated in Step #3 of this procedure (**/vobstg/libpub.vbs/libpub.csv**). Editing this file may be easier if you use a spreadsheet program that can read the comma-separated-value format. This example shows one line of such a file. We've added a header row for clarity and truncated the SID string to save space.

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
OLD\akp	USER	SID:3.0105037...	IGNORE	USER		137

    For each line in the file, replace the string **IGNORE** in the **New-name** field with a user or group name that is valid on the UNIX VOB server host; then delete the last three fields (**Type**, **New-SID**, and **Count**).

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
OLD\akp	USER	SID:3.0105037...	akp			

    Although this example shows a user name that is the same on UNIX as it was on Windows, the procedure can also be used to map a Windows user or group name to a different user or group name on UNIX.

    After you have edited all the rows of the SID file, save it as a comma-separated-value file and use it as the mapping file required when you run **vob_sidwalk -map**. Each line of the

mapping file must have exactly four fields, separated by commas. The example row created in this step looks like this in .csv format:

OLD\akp,USER,SID:3.0105037...,akp

NOTE: You may reassign ownership of any object in a VOB to the VOB owner by placing the string **DELETE** in the **New-name** field. You may also reassign ownership of all objects in a VOB to the VOB owner without creating a mapping file. See *Reassigning Ownership to the VOB Owner* on page 49

12. **Test the map file.** Run **vob_sidwalk** without the **-execute** option. The list of mappings prescribed by the file **libpub-map.csv** is written to the SID file (**libpub-test.csv** in this example), but no changes are made to the VOB.

    *ccase-home-dir***/etc/utils/vob_sidwalk -map /vobstg/libpub.vbs/libpub-map.csv \ /vobs/libpub /libpub-test.csv**

13. **Unlock the VOB.** If you are concerned that users may try to access the VOB before this procedure is complete, lock the VOB again for all users except yourself.

14. **Update user and group identities stored in the VOB.** When you are satisfied that the map file is correct, run **vob_sidwalk** as shown in the following example, where **libpub-map.csv** is the map file created in Step #11 of this procedure:

    *ccase-home-dir***/etc/utils/vob_sidwalk -execute -map /vobstg/libpub.vbs/libpub-map.csv \ /vobs/libpub /libpub-exec.csv**

    vob_sidwalk makes the changes specified in the map file and records the changes that were made in a new SID file, **libpub-exec.csv**.

15. **Update the VOB's group list and container protections**. Use the **vob_sidwalk** program as shown in the following example:

    *ccase-home-dir***/etc/utils/vob_sidwalk -recover_filesystem /vobs/libpub libpub.csv**

16. **Verify that all clients can access the VOB at the new location.** Unlock the VOB if it is still locked.

17. **Verify that all ClearCase users in the new domain can access objects in the VOB.** Users should be able to create new objects as well as change or remove objects they own.

Moving a VOB from UNIX to Windows

For clarity, the procedures in this section use an example:

- The current location of the VOB storage directory to be moved is **/vobstg/libpub.vbs** on UNIX host **ccsvr-ux**. The VOB-tag for this VOB is **/vobs/libpub**.
- The new location of the VOB storage directory is **C:\ClearCaseStorage\VOBs\libpub.vbs** on Windows host **ccsvr-nt**. VOB-tag **\libpub** will be created for this VOB.

To move a VOB from UNIX to Windows:

1. **Log on to the VOB server host as the VOB owner or root.** In this example, the VOB server host for this step is **ccsvr-ux**.
2. **Lock the VOB.** This ensures that new VOB objects are not created while you are performing Step #3 of this procedure.
3. **Generate a SID file** that lists the names and UIDs/GIDs of users and groups associated with objects in **/vobs/libpub**. Run the **vob_siddump** utility as shown in this example:

   *ccase-home-dir***/etc/utils/vob_siddump /vobs/libpub /vobstg/libpub.vbs/libpub.csv**

   We suggest creating the SID file in the VOB storage directory so that it will be available on the new VOB host after the storage directory has been moved. You will need it in Step #15 of this procedure.
4. **Dump the VOB database** using the **cleartool reformatvob** command:

   cleartool reformatvob –dump /vobstg/libpub.vbs

   reformatvob -dump marks the VOB database as invalid. It will be unusable by clients until it is processed by a **reformatvob –load** command.
5. **Copy the VOB storage directory**. Use any file system copy utility to copy the entire VOB storage directory to the Windows host. This example assumes that the UNIX host **ccsvr-ux** is running an SMB server and has shared its **\vobstg** partition. From the Windows host, run these commands:

   C:\ClearCaseStorage\VOBs> net use E: \\ccsvr-ux\vobstg
   C:\ClearCaseStorage\VOBs> *ccase-home-dir***\etc\utils\ccopy E:\libpub.vbs libpub.vbs**

NOTE: Because the existing VOB storage directory ACLs are not supported on the UNIX host, you can use **xcopy** or a similar utility instead of the ClearCase utility *ccase-home-dir***etc\utils\ccopy** in this case.

6. **Terminate the VOB's server processes on UNIX**. You may either stop and restart ClearCase on the VOB server host or search the process table for the **vob_server** and **vobrpc_server** processes that service the old VOB. Use **ps –ax** or **ps –ef**, and search for the VOB storage directory name (**libpub.vbs** in our example), then use **kill**(1) to terminate any such processes.

7. **Fix the VOB storage directory protections.** Log in to the Windows VOB server host (**ccsvr-nt** in our example) as the VOB owner of **\libpub** or as a privileged user. Run the **fix_prot** utility as shown in the following example, where **vobadm** is the name of the new VOB owner, **ccusers** is the name of the VOB's new principal group, and **C:\ClearCaseStorage\VOBs\libpub.vbs** is the host-local pathname of the VOB storage directory.

   *ccase-home-dir***etc\utils\fix_prot -root -r -chown vobadm -chgrp ccusers ^ C:\ClearCaseStorage\VOBs\libpub.vbs**

8. **Re-create the VOB database from the dump files**. Use the cleartool **reformatvob** command:

   cleartool reformatvob -load C:\ClearCaseStorage\VOBs\libpub.vbs

9. **Create a tag for the VOB.** Use the ClearCase Administration Console, or the following command (assuming that **C:\ClearCaseStorage** is shared as **\\ccsvr-nt\ClearCaseStorage**):

   cleartool mktag -vob -tag \libpub \\ccsvr-nt\ClearCaseStorage\VOBs\libpub.vbs

   NOTE: You do not need to register the VOB, because **reformatvob -load** registers the VOB after the reformat is complete.

10. **Lock the VOB.** Although the VOB is now registered and has a tag, it will not be usable until this procedure is complete. If you are concerned that users may try to access the VOB before it is ready, lock it now.

11. **Create a map file.** Open the SID file generated in Step #4 of this procedure (**\\ccsvr-nt\ClearCaseStorage\VOBs\libpub.vbs\libpub.csv**). Editing this file may be easier if you use a spreadsheet program that can read the comma-separated-value format. This example shows one line of such a file. We've added a header row for clarity.

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
akp	USER	UNIX:UID-1247	IGNORE	USER		137

For each line in the file, replace the string **IGNORE** in the **New-name** field with a domain-qualified name of the user or group to which the old name should be mapped; then delete the last three fields (**Type**, **New-SID**, and **Count**).

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
akp	USER	UNIX:UID-1247	NEW\akp			

Although this example shows a user name that is the same on Windows as it was on UNIX, this procedure can also be used to map a UNIX user or group name to a different user or group name on Windows.

After you have edited all the rows of the SID file, save it as a comma-separated-value file and use it as the mapping file required when you run **vob_sidwalk -map**. Each line of the mapping file must have exactly four fields, separated by commas. The example row created in this step looks like this in .csv format:

akp,USER,UNIX:UID-1247,NEW\akp

NOTE: You may reassign ownership of any object in a VOB to the VOB owner by placing the string **DELETE** in the **New-name** field. You may also reassign ownership of all objects in a VOB to the VOB owner without creating a mapping file. See *Reassigning Ownership to the VOB Owner* on page 49.

12. **Test the map file.** Run **vob_sidwalk** without the **-execute** option. The list of mappings prescribed by the file **libpub-map.csv** is written to the SID file (**libpub-test.csv** in this example), but no changes are made to the VOB.

    *ccase-home-dir***etc\utils\vob_sidwalk -map^**
    \\ccsvr-nt\ClearCaseStorage\VOBs\libpub.vbs\libpub-map.csv ^
    \libpub libpub-test.csv

13. **Unlock the VOB.** If you are concerned that users may try to access the VOB before this procedure is complete, lock the VOB again for all users except yourself.

14. **Update user and group identities stored in the VOB.** When you are satisfied that the map file is correct, run **vob_sidwalk** as shown in the following example, where **libpub-map.csv** is the map file you created in Step #10 of this procedure:

    *ccase-home-dir***etc\utils\vob_sidwalk -execute -map**
    \\ccsvr-nt\ClearCaseStorage\VOBs\libpub.vbs\libpub-map.csv \libpub
    libpub-exec.csv

vob_sidwalk remaps ownership as specified in the map file and records the changes that were made in a new SID file, **libpub-exec.csv**.

15. **Recover file system ACLs.** Finally, while you are still logged in to **\\ccsvr-nt** as the VOB owner or privileges user, use **vob_sidwalk** with the **-recover_filesystem** option to apply the correct ACLs to the VOB storage directory.

    *ccase-home-dir***\etc\utils\vob_sidwalk -recover_filesystem \libpub libpub.csv**

16. **Verify that all clients in the region can access the VOB.** Unlock the VOB if it is still locked.

17. **Verify that all ClearCase users on Windows can access objects in the VOB.** Users should be able to create new objects as well as change or remove objects they own.

    NOTE: If the user's name on Windows is not the same as on UNIX, the user loses rights associated with the creator of a version or a branch. For example, the user would not be able to remove a version even though the user had created that version. These operations can still be executed by a more privileged user (VOB owner, member of the ClearCase administrators group).

12 Removing VOBs

This chapter presents procedures for making a VOB inaccessible to clients temporarily and for removing a VOB altogether.

12.1 Locking as an Alternative to VOB Deactivation

In some situations, you may not need to make the VOB inaccessible. For example, to prevent a VOB from being modified, you can lock it using the ClearCase Administration Console, or you can use the following command:

cleartool lock vob:*vob-specifier*

To lock a VOB, you must be the VOB owner or the privileged user. The pathname you specify as the *vob-specifier* can be either the VOB storage directory or the VOB-tag.

12.2 Taking a VOB Out of Service

Suppose that the VOB to be taken out of service has storage directory **/net/sol/vobstore/libpub.vbs** and has VOB-tag **/libpub**. To make the VOB inaccessible:

1. **Remove the VOB from the object registry.** Removal prevents anyone from reactivating the VOB. Use the ClearCase Administration Console or the **cleartool unregister** command:

   cleartool unregister –vob /net/sol/vobstore/libpub.vbs

2. **(Optional) Remove the VOB from the tag registry.** Use the VOB Tags node of the ClearCase Administration Console to remove the VOB's VOB-tag.

3. **Terminate the VOB's server processes.** If the ClearCase LT server is a UNIX computer, search the server's process table for the **vob_server** and **vobrpc_server** processes that manage that VOB. Use **ps –ax** or **ps –ef**, and search for **/vobstore/libpub.vbs**; use **kill**(1) to terminate any such processes. (On UNIX, only the **root** user can kill a **vobrpc_server** process.)

Restoring the VOB to Service

To restore a VOB to service:

1. **Restore the VOB to the object registry.** Use this command:

   cleartool register –vob /net/sol/vobstore/libpub.vbs

   NOTE: If you are registering a UCM project VOB, you must use the **–ucmproject** option with the **register** command.

2. **(If necessary) Restore the VOB to the tag registry**. This is necessary only if you removed the VOB-tag in Step #2 of *Taking a VOB Out of Service*. Use the VOB-Tags node of the ClearCase Administration Console. Or use this command:

   cleartool mktag –vob –tag /libpub /net/sol/vobstore/libpub.vbs

12.3 Removing a VOB

VOBs are usually repositories for critical data. Removing a VOB destroys all of its data. Do not remove a VOB unless the data it contains is no longer valuable.

To remove a VOB:

1. Make sure that no ClearCase LT clients are using the VOB.

2. Use the ClearCase Administration Console to remove the VOB.

   ➢ Navigate to the VOB storage node for the VOB. This is a subnode of the host node for the host where the VOB storage directory resides.

➢ Click **Action** > **All Tasks** > **Remove VOB**.

You can also use the **cleartool rmvob** command

3. Stop and restart ClearCase on the ClearCase LT server.

13 Using checkvob

This chapter explains how to use the **checkvob** utility to find and fix inconsistencies between a VOB database and its storage pools, to clean up broken cross-VOB hyperlinks, and to find and fix global type problems.

See also the **checkvob** reference page.

13.1 When to Use checkvob

Use **checkvob** for these purposes:

- Checking VOB database or storage pool consistency after a VOB restore operation
- (Optional) Cleaning up broken cross-VOB hyperlinks
- (Optional) Finding and fixing problems in an administrative VOB hierarchy
- (Optional) Monitoring VOBs
- (Optional) Diagnosing and repairing damaged VOBs

Running **checkvob** to verify a VOB restore operation is always recommended, but it is required (and happens automatically) when you restore a VOB that uses the semi-live backup strategy. (See also **vob_snapshot**, **vob_restore**, and *Choosing Between Standard and Semi-Live Backup* on page 114.) We recommend that you run **checkvob** as part of your normal maintenance routine.

13.2 Checking Hyperlinks

In hyperlink mode (**–hlinks**), **checkvob** cleans up hyperlinks that **rmhlink** does not delete because they appear to be broken. It examines the hyperlinks on each object you specify and tries to determine whether they are intact. Hyperlinks typically break because an object in a cross-VOB hyperlink is unavailable. This usually happens when a VOB at one end of a cross-VOB hyperlink is offline. By default, **checkvob** prompts you before deleting each partially unavailable hyperlink that it detects. Use **–force** to suppress prompts.

NOTE: When you specify a VOB, **checkvob –hlinks** does not look at all hyperlinks in that VOB; it looks only at hyperlinks attached to the VOB object itself.

13.3 Checking Global Types

In global types mode (**–global**), **checkvob** verifies that no VOB has more than one administrative VOB. If any VOB has more than one administrative VOB, **checkvob** lists it and stops. If the administrative VOB hierarchy is valid, **checkvob** lists this information:

- Global types with local copies whose names do not match
- Eclipsing types
- Eclipsing locks on local copies of global types
- Mismatched protections between global types and their local copies

If you specify a VOB selector, **checkvob** starts its search for global types in that VOB and checks all global types it finds in the administrative VOB hierarchy associated with that VOB. If you specify a type selector, **checkvob** checks only the specified type.

You can restrict the checks by specifying one or more of **–acquire**, **–protections**, **–lock**, or **–unlock**. See the **checkvob** reference page for descriptions of these options.

Fix Processing

With the **–fix** option, **checkvob** does the following:

- Changes the name of each local copy to match the name of its global type. **checkvob** queries for confirmation unless you specify **–force**.

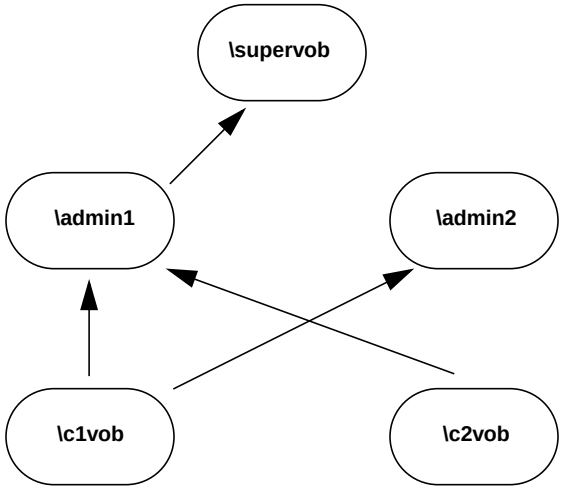
- Attempts to acquire eclipsing local copies and eclipsing ordinary types. **checkvob** queries for confirmation unless you specify **–force**. If a type being acquired is locked, the locks are discarded if there is a lock on the global type. If there is no lock on the global type, the lock information from the first acquired type is applied to the global type.
- Corrects eclipsing locks on local copies. **checkvob** queries for confirmation and whether to unlock or lock the copy unless you specify **–force** and either **–lock** or **–unlock**.
- Changes ownership of each local copy to match the ownership of its global type. **checkvob** queries for confirmation unless you specify **–force**.

Output Log for Global Type Checking

The output from **checkvob** is captured in a log file. By default, the file is named **checkvob.***date***.***time* and written to the current directory.

Example Check or Fix Scenario

In this scenario, the VOB hierarchy looks like this:



To check and fix the global types problems in the hierarchy:

1. Run **checkvob –global** in any VOB in the hierarchy.

   cleartool checkvob –global vob:\supervob

   ```
   The session's log file is "checkvob.03-Aug-99.17:15:15".
   Starting analysis of Admin VOB hierarchy.

   cleartool: Error: There are too many Admin VOBs for vob "\c1vob".

   Analysis of Admin VOB hierarchy complete.
   cleartool: Error: 4 VOBs analyzed, 1 VOBs had problems.
   cleartool: Error: Please fix the listed problems, then rerun this command.
   ```

2. List the **AdminVOB** hyperlinks for **\c1vob**.

   cleartool describe –long vob:\c1vob

   ```
   versioned object base "\c1vob"
   ...
     Hyperlinks:
       AdminVOB@2@\c1vob -> vob:\admin2
       AdminVOB@3@\c1vob -> vob:\admin1
   ```

3. Delete one of the **AdminVOB** hyperlinks. In this example, the hyperlink to **\admin2** is deleted.

   cleartool rmhlink AdminVOB@2@\c1vob

   ```
   Removed hyperlink "AdminVOB@2@\c1vob".
   ```

4. Run **checkvob –global** again to check for problems with the global types in the hierarchy.

cleartool checkvob –global vob:\supervob

```
The session's log file is "checkvob.03-Aug-99.17:23:25".
Starting analysis of Admin VOB hierarchy.

Analysis of Admin VOB hierarchy complete.
4 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
cleartool: Error: Definition of element type "global_el" in \supervob is
eclipsed by definition(s):
        element type "global_el" in \admin1
        element type "global_el" in \c2vob
cleartool: Error: Definition of branch type "global_br" in \supervob is
eclipsed by definition(s):
        branch type "global_br" in \admin1
        branch type "global_br" in \c2vob
cleartool: Error: Global branch type "global_br" in "\supervob" has local
copies with non-matching names:
        "global_br_mismatch" in "\c1vob"
cleartool: Error: Definition of attribute type "global_at" in \supervob is
eclipsed by definition(s):
        attribute type "global_at" in \admin1
        attribute type "global_at" in \c2vob
cleartool: Error: Definition of hyperlink type "global_hl" in \supervob is
eclipsed by definition(s):
        hyperlink type "global_hl" in \c2vob
cleartool: Error: Global hyperlink type "global_hl" in "\supervob" has
local copies with non-matching names:
        "global_hl_wrong" in "\admin1"
```

```
cleartool: Error: Definition of label type "global_lb" in \supervob is
eclipsed by definition(s):
        label type "global_lb" in \admin1
cleartool: Error: Definition of element type "global_el" in \admin1 is
eclipsed by definition(s):
        element type "global_el" in \c2vob
cleartool: Error: Definition of branch type "global_br" in \admin1 is
eclipsed by definition(s):
        branch type "global_br" in \c2vob
cleartool: Error: Definition of attribute type "global_at" in \admin1 is
eclipsed by definition(s):
        attribute type "global_at" in \c2vob
cleartool: Error: Global label type "global_lb" in "\admin1" has local
copies with non-matching names:
        "nt_global_lb" in "\c2vob"


Completed "global type" processing.
Processed 9 global types in 4 VOBs.
The following 9 global types may still have problems:
        element type "global_el" in "\supervob"
        branch type "global_br" in "\supervob"
        attribute type "global_at" in "\supervob"
        hyperlink type "global_hl" in "\supervob"
        label type "global_lb" in "\supervob"
        element type "global_el" in "\admin1"
        branch type "global_br" in "\admin1"
        attribute type "global_at" in "\admin1"
        label type "global_lb" in "\admin1"
```

5. Review the log file, and then run **checkvob –global –fix** to correct the problems.

   cleartool checkvob –global –fix vob:\supervob

```
The session's log file is "checkvob.03-Aug-99.17:31:06".
Starting analysis of Admin VOB hierarchy.

Analysis of Admin VOB hierarchy complete.
4 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: ENABLED
Global element type "global_el" in "\admin1" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global element type "global_el" in "\admin1" has local
copies with inconsistent protections in VOBs:
        \c2vob
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global branch type "global_br" in "\admin1" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global branch type "global_br" in "\admin1" has local
copies with inconsistent protections in VOBs:
        \c2vob
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global attribute type "global_at" in "\admin1" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global attribute type "global_at" in "\admin1" has local
copies with inconsistent protections in VOBs:
        \c2vob
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
cleartool: Error: Global label type "global_lb" in "\admin1" has local
copies with non-matching names:
        "nt_global_lb" in "\c2vob"
```

```
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global element type "global_el" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
Global branch type "global_br" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global branch type "global_br" in "\supervob" has local
copies with non-matching names:
        "global_br_mismatch" in "\c1vob"
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global attribute type "global_at" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
Global hyperlink type "global_hl" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global hyperlink type "global_hl" in "\supervob" has
local copies with non-matching names:
        "global_hl_wrong" in "\admin1"
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
cleartool: Error: Global hyperlink type "global_hl" in "\supervob" has
local copies with inconsistent protections in VOBs:
        \c2vob
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global label type "global_lb" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.

Completed "global type" processing.
Processed 9 global types in 4 VOBs.
```

6. Run **checkvob –global** again to confirm that there are no problems.

   cleartool checkvob -global vob:\supervob

```
The session's log file is "checkvob.03-Aug-99.17:36:49".
Starting analysis of Admin VOB hierarchy.


Analysis of Admin VOB hierarchy complete.
4 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 5 global types in 4 VOBs.
```

13.4 Database or Storage Pool Inconsistencies

Under normal circumstances, the VOB database maintains a complete and uninterrupted record of all activity in the VOB's *source pools*. (Cleartext pools are caches and are expendable.) The VOB database reflects all additions (checkins, branch creation, element creation, and so on) and deletions (removal of versions, branches, and elements) in the pools. Each data container is referenced from the database.

NOTE: Derived objects (DOs) are a ClearCase feature that ClearCase LT does not use. ClearCase LT VOBs contain storage pools for these objects, but the pools are never used. **checkvob** examines these pools as part of its normal procedure, but they never contain any data on a ClearCase LT server.

In general, inconsistencies between the database and storage pools occur because of damage to the VOB database, to the storage pools, or to both. If the damage requires VOB restoration from backup, database or pool inconsistency occurs at restore time if the VOB has been backed up using the semi-live backup approach (database and pools were backed up at different times). That is why a **vob_restore** operation includes **checkvob**. In general, **checkvob** tries to make the pools match the database. If the pools are missing data, **checkvob** cannot re-create the data, and must adjust the database to compensate for the missing information.

Here are three scenarios:

- **VOB database damage.** The database must be retrieved from snapshot or backup and is older than existing storage pools. It does not record recent pool changes (**checkin**, **rmver**, and so on).
- **VOB storage pool damage.** The database is newer than the storage pools, which must be retrieved from backup. The database records development activity that is not reflected in the older pools.
- **Database and pool damage, with semi-live VOB backup procedures in use.** The database snapshot and storage pool backups occur at different times (see **vob_snapshot**). The database retrieved from backup may be older or newer than storage pools retrieved from backup.

In addition, system crashes or network failures can prevent **cleartool** operations from completing cleanup tasks. This often results in unreferenced data containers. Any aborted operation that fails to clean up properly can have the same effect, as can OS bugs and disk problems.

Given a time lapse between current instances of VOB database and storage pools, any operation that modifies storage pools can create database or source pool inconsistencies. **checkvob** can identify—and in most cases, fix—these kinds of data container problems:

- **Misprotected data containers—**containers with incorrect access control information
- **Missing data containers—**VOB database references to nonexistent data containers
- **Unreferenced data containers—**data containers for which there is no corresponding VOB database entry

Of these problems, missing containers is the most serious, as it may represent loss of data.

Problem	**Found by checkvob**
All pool kinds (source, DO, cleartext)	
Bad pool roots (pool names, identity info)	Yes (**–pool**)
Source and DO pools only	
Misprotected containers	Yes (**–protections**)
Missing containers	Yes (**–data**)

Problem	Found by checkvob
Unreferenced containers (debris)	Yes (**–debris**)
Corrupted containers	No

By default, **checkvob** prints detailed reports on one or more storage pools (**–pool**) or on specific file elements passed in the command line. With the **–fix** option, **checkvob** fixes as many problems as it can, adjusting *data containers* to match what is expected by the VOB database.

WARNING: Fixing problems detected with **–data** can update the VOB irreversibly. If version data is recorded in the database but missing from the storage pools, **checkvob** updates the VOB database by removing the version (**rmver -data**).

Updating the VOB Database

checkvob never updates the VOB database to reference newer pool data. If versions are stored in the pools but not recorded in the database, **checkvob** moves the unreferenced data containers to the pool's **lost+found** directory. That is, if pool storage is newer than the database, **checkvob** does not salvage the latest versions from the unreferenced pool containers and update the database. It uses the unreferenced containers to return the pool to the state expected by the database, and it moves the unreferenced containers (with the latest version data) to *pool-dir***lost+found**. These versions should be presumed lost. Contact Rational Technical Support for more information. (By contrast, barring unexpected events, any missing data that **checkvob** reports in this scenario can generally be attributed to **rmver** operations that were not recorded in the older database; and the fix processing that accepts this data loss is inconsequential.)

Although it does not add new version information to the VOB database, **checkvob** updates the VOB database in the following ways:

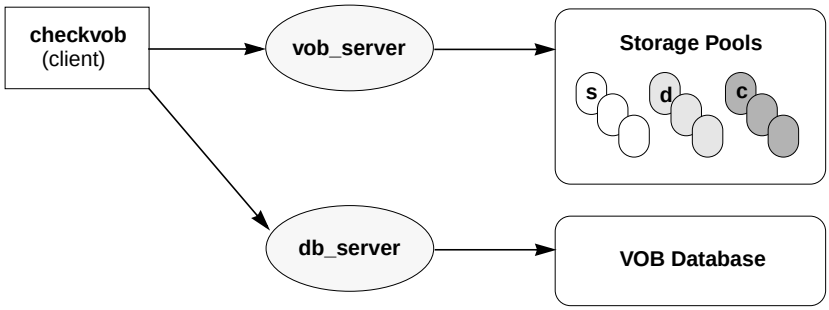
- It removes references to versions that are confirmed missing by **checkvob** and accepted as missing by the user (interactive **yes** or **–force –fix** option).
- After successfully reconstructing a missing container, in whole or part, **checkvob** adjusts the database to reference the newly reconstructed container, not the old, missing container.

Requirements for Using checkvob

If you use a customized *type manager*, it must include the **get_cont_info** method; if it does not, **checkvob** cannot repair data containers managed by this type manager. See the **type_manager** reference page for more information.

checkvob requires operational **vob_server** and **db_server** processes on the ClearCase LT server where it executes. As shown in Figure 14, the **vob_server**, which runs as the VOB owner on UNIX and as the **NT Authority\system** identity on Windows, is the only process that can create or delete data containers. Similarly, the **db_server** mediates all VOB database access.

Figure 14 Pool Access Through vob_server and VOB Database Access Through db_server



To understand the **checkvob** results, you must understand how ClearCase type managers operate on data containers. For example, changes to a data container (checkin, in particular) always result in a new, renamed container, regardless of the applicable type manager. See the **type_manager** reference page for more details.

Running checkvob

To accept as lost any pool data added in the interval between the pool and database reference times, run **checkvob –force –fix** and supply an appropriate time interval (see *Force-Fix Mode on page 193*).

To accept the default data loss interval of one day, run **checkvob –force –fix**. Then, run **checkvob** without **–force**, and fix any remaining elements with missing containers manually (as prompted for).

Output Log for Pool Checking

The output from **checkvob** is captured in a log file directory created each time **checkvob** runs. The default location of the log directory is *current-dir***checkvob.***date-time**,* but you can move it with the **–log** option. A **checkvob** log directory's contents:

.\checkvob.11-Oct-99.18.30.45
poolkind_source\transcript
poolkind_derived\transcript
poolkind_cleartext\transcript
summary

NOTE: The output log is a single **summary** file if you run **checkvob** on a single object.

Each **transcript** file contains a detailed report on check and fix processing for each existing pool kind. The **summary** file contains the header and tail sections from each **transcript** file.

Figure 15 shows a **summary** log file. This output was generated with a command like the following, on a VOB with one missing data container:

cleartool checkvob –pool –fix \\saturn\vobstore\src.vbs

Figure 15 checkvob Output Log: Summary File

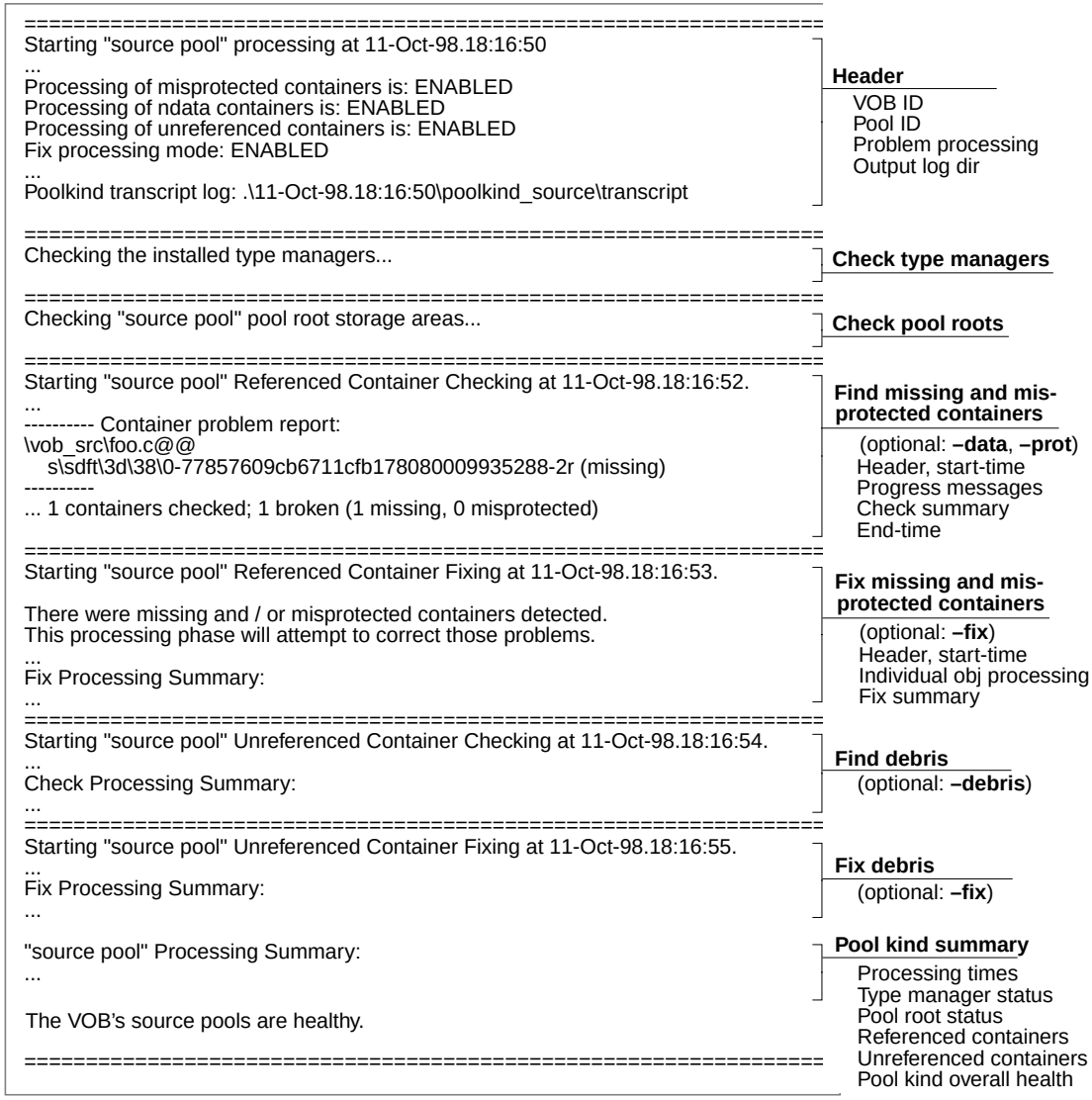
```
==========================================================================
Starting "source pool" processing at 11-Oct-98.18:16:50

Running from host: saturn
VOB hostname: saturn
VOB host storage pathname: \vobstore\src.vbs
VOB global storage pathname: \net\saturn\vobstore\src.vbs
VOB replica oid: 110a7bca.cb6711cf.b514.08:00:09:93:52:88
VOB host reference time: 11-Oct-98.18:16:50
Processing pools: sdft
Processing of misprotected containers is: ENABLED
Processing of ndata containers is: ENABLED
Processing of unreferenced containers is: ENABLED
Fix processing mode: ENABLED
Not allowing fixes involving missing data.

Poolkind transcript log: .\11-Oct-98.18:16:50\poolkind_source\transcript
==========================================================================

Completed "source pool" processing at 11-Oct-98.18:16:51

"source pool" Processing Summary:
Referenced Container Check Processing Time: 00:00:01
Referenced Container Fix Processing Time: 00:00:03
Unreferenced Container Check Processing Time: 00:00:00
*** 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:
   1 containers checked
         1 ndata         0 misprotected
   1 objects checked
         1 ndata         0 misprotected
Referenced container fix processing:
   1 Initial objects with problem containers.
         1 ndata         0 misprotected
   0 Final objects with problem containers.
         0 ndata         0 misprotected
   0 Fixed objects suffered data loss (0 lost data items).
Unreferenced container check processing:
   1 containers checked     (0 kbytes)
         0 unreferenced but under age      (0 kbytes)
         0 unreferenced but maybe needed   (0 kbytes)
   0 unreferenced containers (0 kbytes, 0 empty)

The VOB's source pools are healthy.

Poolkind transcript log: \11-Oct-98.18:16:50\poolkind_source\transcript
==========================================================================
```

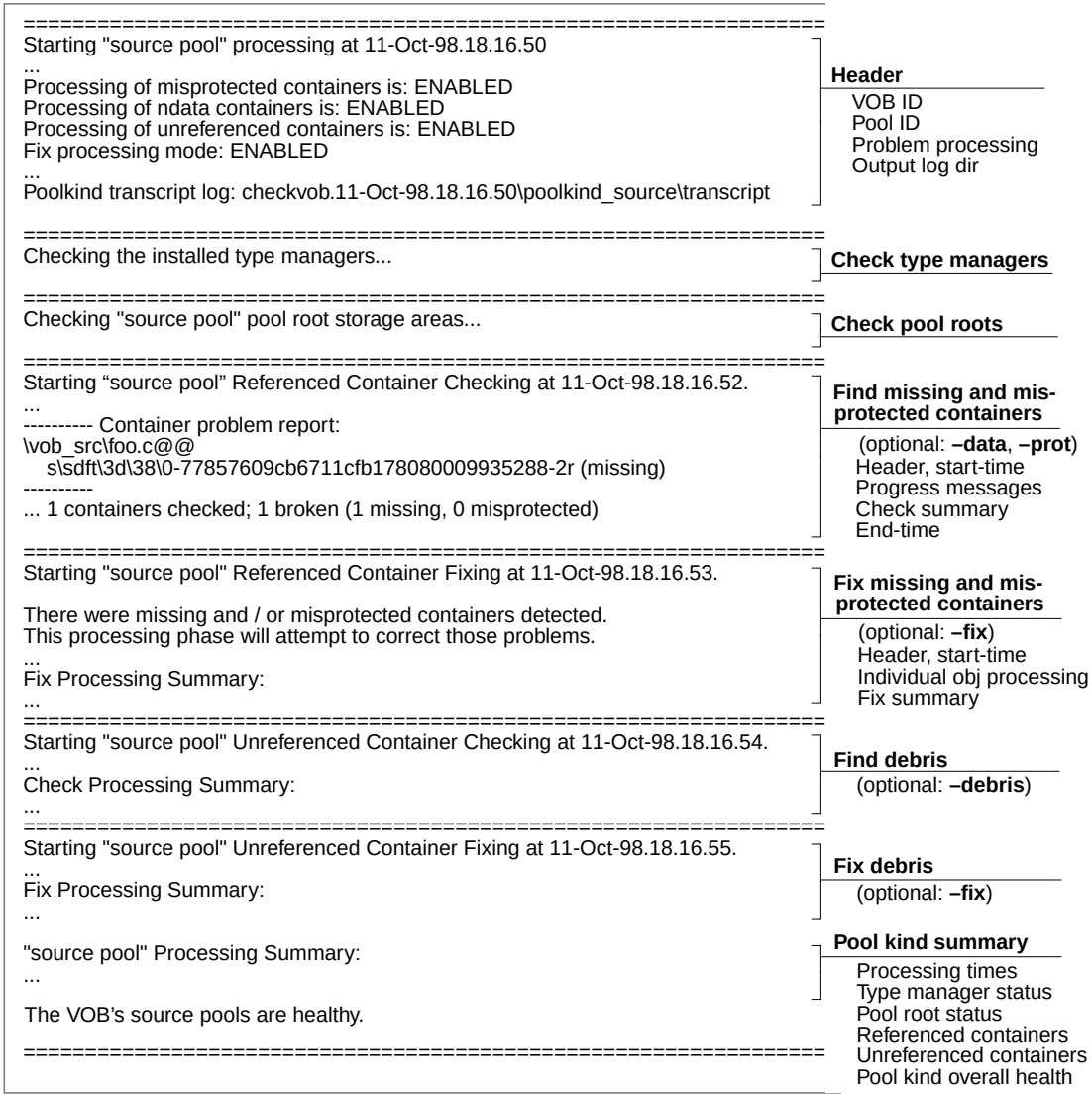
```
==========================================================================
Starting "source pool" processing at 11-Oct-98.18.16.50

Running from host: saturn
VOB hostname: saturn
VOB host storage pathname: C:\vobstore\src.vbs
VOB global storage pathname: \\saturn\vobstore\src.vbs
VOB replica oid: 110a7bca.cb6711cf.b514.08:00:09:93:52:88
VOB host reference time: 11-Oct-98.18:16:50
Processing pools: sdft
Processing of misprotected containers is: ENABLED
Processing of ndata containers is: ENABLED
Processing of unreferenced containers is: ENABLED
Fix processing mode: ENABLED
Not allowing fixes involving missing data.

Poolkind transcript log: checkvob.11-Oct-98.18.16.50\poolkind_source\transcript
==========================================================================

Completed "source pool" processing at 11-Oct-98.18.16.51

"source pool" Processing Summary:
Referenced Container Check Processing Time: 00:00:01
Referenced Container Fix Processing Time: 00:00:03
Unreferenced Container Check Processing Time: 00:00:00
*** 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:
   1 containers checked
         1 ndata         0 misprotected
   1 objects checked
         1 ndata         0 misprotected
Referenced container fix processing:
   1 Initial objects with problem containers.
         1 ndata         0 misprotected
   0 Final objects with problem containers.
         0 ndata         0 misprotected
   0 Fixed objects suffered data loss (0 lost data items).
Unreferenced container check processing:
   1 containers checked     (0 kbytes)
         0 unreferenced but under age      (0 kbytes)
         0 unreferenced but maybe needed   (0 kbytes)
   0 unreferenced containers (0 kbytes, 0 empty)

The VOB's source pools are healthy.

Poolkind transcript log: checkvob.11-Oct-98.18.16.50\poolkind_source\transcript
==========================================================================
```

Figure 16 shows a condensed **transcript** file for source pool check and fix processing. This output was generated with a command like the following, on a VOB with one missing data container:

cleartool checkvob –fix –force –pool –source \\saturn\vobstore\src.vbs

Figure 16 checkvob Output Log: Condensed Transcript File





Overview of checkvob Processing

The following sections describe **checkvob** actions.

Individual File Element Processing

When run in fix mode (**–fix**) against individual file-system objects (no **–pool** option), **checkvob** writes output like the following to standard output and also to *log-dir***transcript**. For example:

```
======================================================================
Processing element "\vob_src\open.c@@".
The element is now locked.
Checking status of 1 referenced containers in pool "s\sdft"...
...
    fix processing output
Initial container status: 0 missing, 0 misprotected.
Final container status: 0 missing, 0 misprotected.
The element is now unlocked.
======================================================================
```

File element processing:

1. Check the element for **–data** (missing container) and **–protection** problems.
2. If fix mode (**–fix**), fix protection and missing container problems:

   NOTE: Fix processing is blocked if the VOB, source pool, or element is locked.

   a. Lock the element.
   b. Fix missing protection problems.
   c. Fix missing data container problems by scanning the pool for missing or misplaced containers and reconstructing containers as necessary.
   d. Update the VOB database to reference the reconstructed containers.
   e. For missing version data, update the VOB database to dereference lost versions with the equivalent of **rmver -data**.
   f. Apply minor events to element's event history.
   g. Move alternate (unreferenced) containers for this element to pool's **lost+found** directory.
   h. Unlock the element.

Pool Mode (–pool Option) Processing: Overview

When run with the **–pool** option, **checkvob** examines some or all of the VOB's source, DO, and cleartext pools.

For each kind of pool (source, DO, cleartext):

1. Check pool roots. Check pool names, locations, and pool identity information (each pool must have a **pool_id** file, which stores the pool's OID). These problems cannot be fixed with **checkvob** and must be directed to Rational Technical Support.
2. For source pools, check the installed type managers for **checkvob** support (**get_cont_info** method).

For source and DO pools only:

3. Scan VOB database for missing (referenced, but not found) or misprotected containers.
4. Generate a list of problem VOB objects.
5. If **–fix** is specified, process each problem VOB object as described in *Individual File Element Processing* on page 192.
6. Scan for unreferenced data containers.
7. If **–fix** is specified, move each unreferenced container to the pool's **lost+found** directory.

Force-Fix Mode

If you use the **–force –fix** options, **checkvob** prevents you from unintentionally accepting data loss:

- **Do not allow removal of all non-*branch*\0 versions.** In force-fix mode, **checkvob** does not update the VOB database to reflect an element's missing data containers unless at least one version having a version number greater than 0 and its data remain. That is, you must run **checkvob** in **-fix** mode, without **–force**, and agree when prompted to accept a reconstructed element whose only remaining versions have version IDs like **\main\br1\0** and **\main\br1\br2\0**.
- **Prompts to allow data loss.** In force-fix mode, **checkvob** prints the following prompt:

```
Do you want to override the default and allow fixing of
elements involving missing version data? [no] n
```

If you answer **yes**, **checkvob** prompts you to specify a time interval for which data loss is allowable (or expected). For example, if you restored source pools from a backup with date-time **6-Oct-99.00:02:00**, and your VOB database is current, you can reasonably expect to lose version data created since that time. In this case, you can direct **checkvob** to allow the loss of data created and recorded in the VOB database after that time:

```
cleartool checkvob –force –fix –data –pool –source c:\vobstore\proj1.vbs
...
Allow missing data created since: [date-time, <CR>] 6-Oct-99.00:02:00
```

If **checkvob** cannot find an expected container with data that was created before **6-Oct-99.00:02:00**, it records this fact in the output log but does not update the VOB to dismiss the missing container; it does not accept the data loss. Run **checkvob** again without the **–force** option to process such elements individually, or adjust the allowed data loss time.

To silently accept (fix) all missing data containers without regard for creation time, use a very old date-time. The default time interval for allowed data loss is "since yesterday at 00:00:00." If you supply a date older than one week, **checkvob** forces you to confirm it.

checkvob log files do not capture the time-interval dialogue from **–force –fix** operations.

See the description of the *date-time* argument in the **lshistory** reference page for a list of acceptable values.

Pool Setup Mode

checkvob –setup –fix –pool is run by **reformatvob** when you upgrade a ClearCase LT server to a new ClearCase LT release. If this part of the **reformatvob** operation fails, you must run **checkvob –setup –fix** manually. This command must complete successfully to enable **checkvob** processing in the VOB's storage pools (which is a prerequisite to using the **vob_snapshot** utility successfully.)

checkvob –setup –fix –pool does the following:

- Checks pool roots—pool names, locations, and pool identity information. (Each pool must have a **pool_id** file, which stores the pool's OID). For any pool, if a missing **pool_id** file is the only detected error, **checkvob** adds the file.
- (Source pools only) Verifies that the installed type managers support use of **checkvob**. If a type manager does not support the **get_cont_info** method, **checkvob** cannot fix missing data containers managed by that type manager.

- (Source pools only) Converts source pool data container pathnames to the correct format. See the **type_manager** reference page for a description of the source container name format.

VOB, pool, or element locks prevent setup processing. In this case, do the following:

1. (Pool or VOB locks only) Log on as the VOB owner or a privileged user.
2. Replace the VOB lock:

   cleartool unlock vob:*vob-pname*
   cleartool lock –nusers *userid-that-will-run-checkvob* **vob:***vob-pname*

3. Replace pool locks:

   cleartool lock –replace –nusers *userid-that-will-run-checkvob locked-pools*

4. Replace element locks:

   cleartool lock –replace –nusers *userid-that-will-run-checkvob locked-elements*

5. Rerun **checkvob –setup –fix** manually.

Descriptions of Storage Pool Problems

The following sections describe how storage pool problems arise and how **checkvob** fixes them.

Notes on problem descriptions:

- **Unexpected events.** Many problems described here are often side effects of various infrequent or uncommon events. Such events include network failure, system crash, failed or aborted cleanup operations, operating system bugs, disk failure, network reconfiguration events, and so on. In addition, there is a class of events that includes accidental or malicious delete, move, rename, or change-protection operations on the actual containers. These are all varieties of unexpected events.
- **Major and minor problems.** The summary output from **checkvob** records the number of major and minor problems detected. Bad pool roots and missing data containers are considered major problems. All others are considered minor.
- **Reference times.** A pool or VOB database's *reference time* is the point at which VOB activity was last recorded there. This can be the current date-time, the date-time when a still-operational VOB was locked, or the date-time when a snapshot or backup operation

captured the pool or database. Expected output and fix processing from **checkvob** varies markedly depending on the relative reference times on the VOB database and storage pools being compared. There are three general cases:

- The database is newer.
- The pools are newer.
- The database and storage pools are synchronized: they're both current, or they were retrieved as a unit from a complete backup of the VOB storage directory.

- **Fix processing.** The following sections describe, under *Fix Processing* headings, how **checkvob** fixes the problems it finds. However, some descriptions include additional repair steps for the user.

Source or Cleartext Pool: Bad Pool Roots

Description

Misnamed or misidentified pools**.**

Cause

rename *pool* in the interval between database and storage pool reference times.

Fix Processing

Unless pool names and IDs match VOB database expectations exactly, abort processing for this pool kind (source or cleartext) and skip to the next pool kind.

Exception: If, for any pool, the only inconsistency is a missing **pool_id** file, create the **pool_id** file.

Source Pool: Misprotected Container on Windows

Description

FAT file system: incorrect RO attribute.
NTFS file system: incorrect RO attribute or ACL.

Cause

User copied or restored pools or containers without preserving protection information.

Fix Processing

Reset container's RO attribute and ACL as necessary. If the VOB owner's rights to pool contents are insufficient, **checkvob** reports, but cannot fix, container ACLs. If **checkvob** cannot repair protection problems, you must take the following steps:

1. Log on as a member of the ClearCase administrators group.
2. In Windows Explorer, click **File** > **Properties**. On the **Security** tab, take ownership of all files and directories in the VOB's storage directory tree.

   NOTE: If you have identified only a small number of affected files, take ownership of these files only, to avoid a time-consuming **checkvob** operation in Step #6.
3. For all files and directories in the VOB storage directory, use the **Security** tab to grant **full rights** to the **clearcase** and *vob-owner* accounts.
4. Log out. Log on as VOB owner.
5. Take ownership of all files and directories in the VOB storage directory tree.
6. Run **checkvob** to fix protections on storage pools:

   cleartool checkvob –force –fix –protection –pool c:\vobstore\myvob.vbs

Missing and Unreferenced Data Containers

checkvob works to reconcile the contents of VOB storage with the information in the VOB database. This reconciliation requires **checkvob** to categorize data containers based on their relationship to this information. There are two categories:

- **Missing data container.** A container is considered missing if it does not appear under the exact name and location recorded in the VOB database. This definition implies missing version data. During fix-mode processing, **checkvob** attempts to fix missing containers by scanning all storage pools, looking for alternate containers from which to reconstruct containers for the missing data.

- **Debris.** A container is considered debris if its pathname is not recorded in the VOB database. During **–fix –debris** processing, **checkvob** finds all debris in the source pools. It checks various aspects of an unreferenced container before moving it to the applicable pool's **lost+found** directory.

Whenever the VOB database and storage pools have different reference times (one is older than the other), **checkvob** is likely to find both missing and unreferenced containers. For example, consider a container whose location has changed as a result of a rename operation on a pool: it stores the data that the database is trying to reference, but at the wrong location in the storage pool. **checkvob** reports a missing container and an unreferenced container. Note that in fix mode, **checkvob** resolves these simple location problems.

Source Pool: Missing Container

Description

The VOB database references a source pool data container that does not exist at the expected location. A container is considered missing if it does not appear under the exact name and location recorded in the VOB database.

Causes

- (Database newer) Database records checkin not found in (older) pool.
- (Pool newer) Pool stores updated, renamed container with new checkin data, but (older) database references pre-checkin container name, which no longer exists.
- (Pool newer) Pool stores updated container to reflect versions removed with **rmver** or **rmbranch**, but the database references old container.
- (Pool or database reference times differ) A **chpool** operation on a file element in the interval between pool and database reference times leaves the database pointing at wrong pool.
- Unexpected events.

Fix Processing

During fix mode processing, **checkvob** uses the following algorithm to fix missing containers by scanning all storage pools for alternate containers from which to reconstruct missing data containers.

1. Scan pools for alternate containers.

In a previous pass over the pools, **checkvob** found all referenced containers. It now scans for unreferenced containers for that element, looking for alternate containers that can be used to reconstruct a replacement for the missing container by rebinding the alternate container to take the place of the missing one. There are two types of rebind operations:

Optimized rebind: find alternates maintained by the right type manager.

a. Find best match: identical, superset, or subset.

   identical—container with correct contents (user ran **chpool** during interval between pool and database reference times).

   superset—container with superset of versions expected by database. This is common when the pool is newer than the database.

   subset—container with subset of versions expected by database. This is common when the database is newer than the pool.

b. Clone and prune best match. Create a new container and copy the best alternate's contents. Delete extra version data from the new container.

Nonoptimized rebind: alternate containers with the right type manager not available (no "best match" found).

a. Construct container one version at a time from whatever sources are available: containers maintained by other type managers and cleartext pools.

2. If container reconstruction is incomplete, collect and report a list of missing versions.

3. If **–force** is not specified, prompt user to accept fix.

   If **–force** is specified, safety features prevent some kinds of inadvertent data loss. See *Force-Fix Mode on page 193* for details.

4. If **–force** is in effect, or if user accepted fix prompt, update VOB database:

   a. Adjust database to reference reconstructed containers.

   b. With the equivalent of **rmver –data**, adjust the database to dereference lost version data.

5. Move all of the element's alternate containers to pool's **lost+found** directory.

   NOTE: Because (unreferenced) alternate containers are moved to **lost+found** now, rather than during **checkvob**'s subsequent debris processing pass, you have an opportunity to reclaim

disk space from **lost+found** if the disk fills up during reconstruction. Reconstruction can consume substantial disk space.

Source Pool: Unreferenced Container (Debris)

Description

Source pool includes a data container that is not referenced by the VOB database. Such containers are tentatively classified as debris, but must pass several tests before being moved to the applicable pool's **lost+found** directory.

Causes

- (Database newer) Database references newer, post-checkin container name (which is *missing*), but pool stores older, unreferenced container.
- (Database newer) Database records **rmver** or **rmbranch** events, but older pool stores container with branches or versions still in place.
- (Pool newer) Pool has container with versions from one or more checkin operations not recorded in the older database.
- (Pool or database reference times differ) A **chpool** operation on a file element in the interval between pool and database reference times leaves the database pointing at the wrong pool. The containers, being at an unexpected location, are unreferenced.
- Restoring a pool from incremental backups.
- Unexpected events.

Fix Processing

checkvob usually moves an unreferenced container to the applicable pool's **lost+found** subdirectory (*vob-storage-dir***\s\sdft\lost+found**, by default). It leaves in place unreferenced containers that fit into these categories:

- **May be needed. checkvob** found, but did not fix, a *missing* container problem, and the pathname of the current *unreferenced* container suggests that it may be able to contribute to reconstruction of the missing container on a subsequent **checkvob** run. If you specify [**–force**] **–fix –debris**, without **–data**, **checkvob** performs **–data** check (not fix) processing to identify debris that may be needed.
- **Underage.** The container is less than one hour old. **checkvob** skips underage containers to avoid removing a newly created container before the VOB database has been updated to

reference it. Typically, the time between new container creation and database reference update is less than one second, but **checkvob** takes a conservative approach because of the critical nature of source containers.

- **Scheduled for deletion.** The VOB is configured for deferred source container deletion (see **vob_snapshot_setup** and **vob_server**), and the container is already marked for deletion.

NOTE: When the pool is newer than the database and includes more recent versions not recorded in the (older) database, **checkvob** does not salvage versions from the unreferenced containers and update the database. It uses the unreferenced containers to return the pool to the state expected by the database, and it moves the unreferenced containers (with the latest version data) to *pool-dir***lost+found**. These versions should be presumed lost. Contact Rational Technical Support for more information.

Source Pool: Corrupted Container

Description

File truncated and similar conditions

Cause

Unexpected events

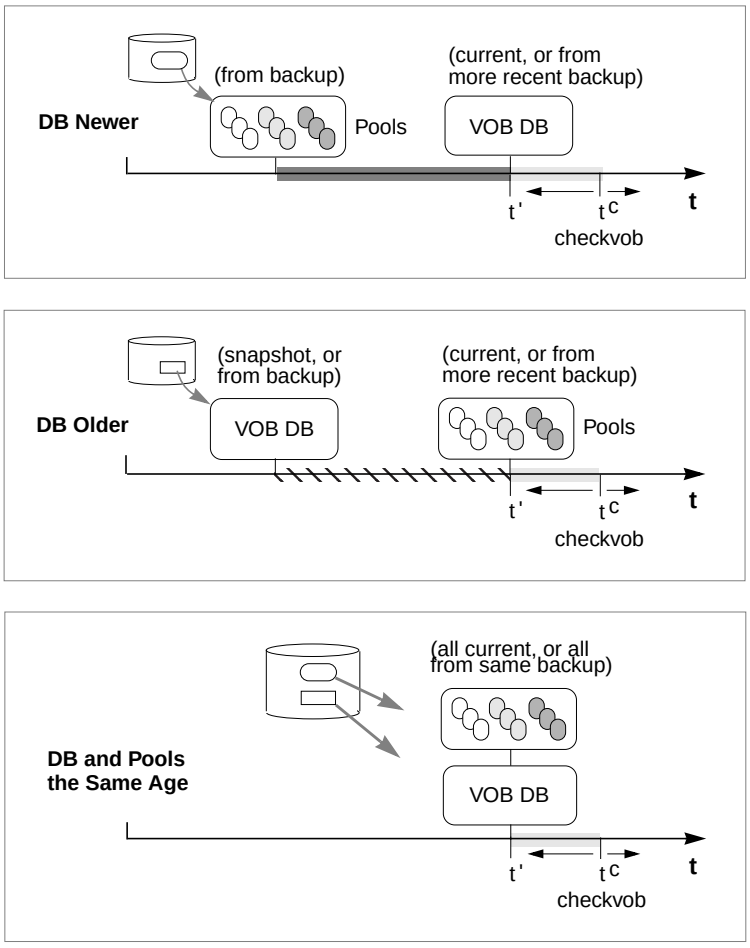
Fix Processing

None. **checkvob** does not find or fix corrupted data containers. A container with the right pathname is considered healthy.

13.5 Sample Check and Fix Scenarios

The following sample scenarios (see Figure 17) illustrate some general guidelines for using **checkvob**.

Figure 17 Common Scenarios in VOB Database or Storage Pool Synchronization



Somewhere in this time interval is t^c — the point at which you run **checkvob**. If $t^c \neq t'$, VOB activity in the Interval is lost

Pool data created during this time interval is lost. (There may also be “extra” data from rmver or rmbranch operations in this interval.)

Pool data created during this time interval may get copied to the pool’s **lost+found** directory, but this data should be presumed lost. checkvob never adds new version or DO data to the DB.

If DB and pools are the same age, data loss is unexpected. Unreferenced containers are common if pools and DB were restored from one or more incremental backups.

Scenario 1: VOB Database Newer Than Storage Pools

A disk crash destroyed remote storage pools, which have been retrieved from backup. The VOB database is current, and the VOB is locked against any further write operations.

- The VOB database records development activity that is not substantiated in the storage pools.Under these circumstances, **checkvob** ought to discover missing and unreferenced

data containers. Checkins have been recorded in the VOB database, but they do not appear in the storage pools. This represents real data loss. **checkvob** synchronizes the VOB database with the storage pool by executing **rmver -data** on missing versions.

Running checkvob

To accept as lost any pool data added in the interval between the pool and database reference times, run **checkvob –force –fix** and supply an appropriate time interval (see *Force-Fix Mode*).

To accept the default, one-day data loss interval, run **checkvob –force –fix**; then, run **checkvob** without **–force** and fix any remaining elements with missing containers manually (as you are prompted by **checkvob**).

Scenario 2: Storage Pools Newer Than VOB Database

The VOB database is corrupted. The storage pools are current, and you have a recent VOB database snapshot on disk. You have run **vob_restore** to recover the VOB database and storage pools. **checkvob** runs to complete the restoration process.

The VOB storage pools reflect development activity that has not been recorded in the VOB database. This scenario, like the previous one, is consistent with using a semi-live VOB backup scheme, as described in *Backing Up a VOB* on page 112. Under these circumstances, **checkvob** ought to find missing and unreferenced data containers. New checked-in versions have been written to VOB storage after the reference time on the available VOB database. Note that if any **rmver**, **rmbranch,** or **rmelem** events occurred during the time interval, recovering all versions is not possible (and, presumably, not desirable).

Running checkvob

Run **checkvob –force –fix** and supply an appropriate time interval (see *Force-Fix Mode*). Most new work (versions found in the newer pools, but unknown to the older database) will be captured in containers identified as debris and moved to the pool's **lost+found** subdirectory (from where you can, under some circumstances, retrieve it successfully). Note that under no circumstances does **checkvob** update a VOB database with "new" version data found in more recent pools. You may be able to construct versions from containers in **lost+found** and check them in as new versions, but **checkvob** cannot do so. The safety net features (see *Force-Fix Mode*) prevent large scale data loss. The only expected data loss reports should be the result of **rmver** or **rmbranch** operations no longer reflected by the (older) database.

13.6 Sample checkvob Runs

The remaining sections in this chapter describe some common problem scenarios for storage pools, and they refer to **checkvob** log files that can be found online in *ccase-home-dir***\doc\examples\checkvob_sample_logs**.

Notes on the sample logs:

- The **–force** option is not used in the sample runs. (Runs without **–force** do not illustrate the acceptable data-loss-interval dialogue.)
- The log transcripts do not capture user input.
- Several of the sample logs refer to derived objects (DOs). Derived objects are not supported in Rational ClearCase LT.

13.7 Database Newer Than Pools

Log: *ccase-home-dir***\doc\examples\checkvob_sample_logs\checkvob.DB_newer**

Highlights from this run:

- The database records a missing data container for a checkin (**foo.c**) that is newer than the source pool. A corresponding older unreferenced container appears in the source pool.

The only alternate container for element **foo.c** is missing version **\main\2** (a newer checkin). Note that nothing happened during the check and fix processing of unreferenced containers because the only unreferenced containers in the pool were for this broken element. These unreferenced containers were moved to the pool's **lost+found** directory.

13.8 Database Older Than Pools

Log: *ccase-home-dir***\doc\examples\checkvob_sample_logs\checkvob.DB_older**

Highlights from this run:

- An unreferenced container for **foo.c** stores the newest checkin data. There is a corresponding missing container, because the database is looking for a different, older one.
- An **rmver** operation took place in the source pool for **bar.c** but is not reflected in the database, which continues to look for a missing container.

The only alternate container for element **foo.c** has an extra, unreferenced version, **\main\3**. This version appears to have been checked in sometime in the future, from the database's point of view. The only alternate container for element **bar.c** is the missing version **\main\2**. It was removed in the future from the database point of view. Note that the unreferenced container check or fix phase found nothing to do, because the only unreferenced containers in the pool were for these broken elements, and these unreferenced containers were moved to the pool's **lost+found** directory.

NOTE: From **checkvob**'s perspective (and its output), there is no difference between the **foo.c** and **bar.c** cases; both containers are missing version data.

13.9 Unreferenced Containers from Incremental Backup or Restore

Log: *ccase-home-dir***\doc\examples\checkvob_sample_logs\checkvob.incremental**

In this case, a synchronized, healthy VOB underwent incremental backup and restore operations, resulting in unreferenced containers.

Highlights of this run:

- The backup captured two replaced containers each for **foo.c** and **bar.c**.

14 Splitting VOBs with relocate

The export/import utilities included with Rational ClearCase provide a way to copy elements between VOBs. To move directory elements, file elements, and VOB symbolic links from one VOB to another, use the **relocate** command. A relocate operation is appropriate if you need to reorganize VOB data to reflect changes in component architecture or organizational structure or if you need to groups of elements out of a VOB to reduce its size.

Relocating elements is a complex operation. All of the data and metadata associated with each relocated element must be preserved, and all clients must be able to access the relocated elements along with their metadata.

This chapter first describes how **relocate** works. Then, it examines what happens before, during, and after a relocate operation from an administrator's point of view.

NOTE: You cannot use **relocate** in a UCM VOB. Do not perform any **relocate** operation without also consulting the **relocate** reference page.

14.1 What Does relocate Do?

The **relocate** command moves a designated set of elements from one VOB (the source VOB) to another VOB (the target VOB)—typically, a new VOB created for this purpose. The **relocate** command does all of the following:

- Moves elements from one VOB to another, including these:
  - Data containers

- Event histories (some minor events are lost)
- Bidirectional hyperlinks (see also **mkhlink**)
- Related VOB database records

- Creates *VOB symbolic links* from the source VOB to the target directory, generally preserving the source VOB's namespace for views bound to specific, preexisting versions of relocated elements.
- Copies metadata types associated with moved elements to the target VOB, as necessary (label, attribute, element, trigger, and hyperlink types).
- Leaves behind an event history for each relocated element—a remove element event in the source VOB, and a relocate event in the target VOB.
- Creates a log of its activities—by default, in the file **relocate.log.***date-time* in the current directory.

relocate does not do the following:

- Relocate elements when either the source or the destination VOB is a UCM VOB.
- Copy elements; it moves them. (**relocate** is not a backup tool.)
- Move elements to a new location in the same VOB. (Use **cleartool mv** for this purpose.)

14.2 Element Relocation Illustrated

The following series of figures illustrates several variants of a comparatively simple relocate operation and emphasizes how the various versions of affected directories catalog elements after the move. The figures assume these conditions:

- The view used for the **relocate** operation selects **\main\LATEST** for all elements being relocated. This approach is recommended because it is least likely to generate confusing results for VOB users and administrators. (You can use a different branch, as long as the config spec selects the latest version on that branch.)
- The target VOB was created to accommodate the relocated elements. This approach is not mandatory, but we recommend it. It minimizes the chances of name collision between an

existing element and a relocated one and of encountering locked type objects in the target VOB.

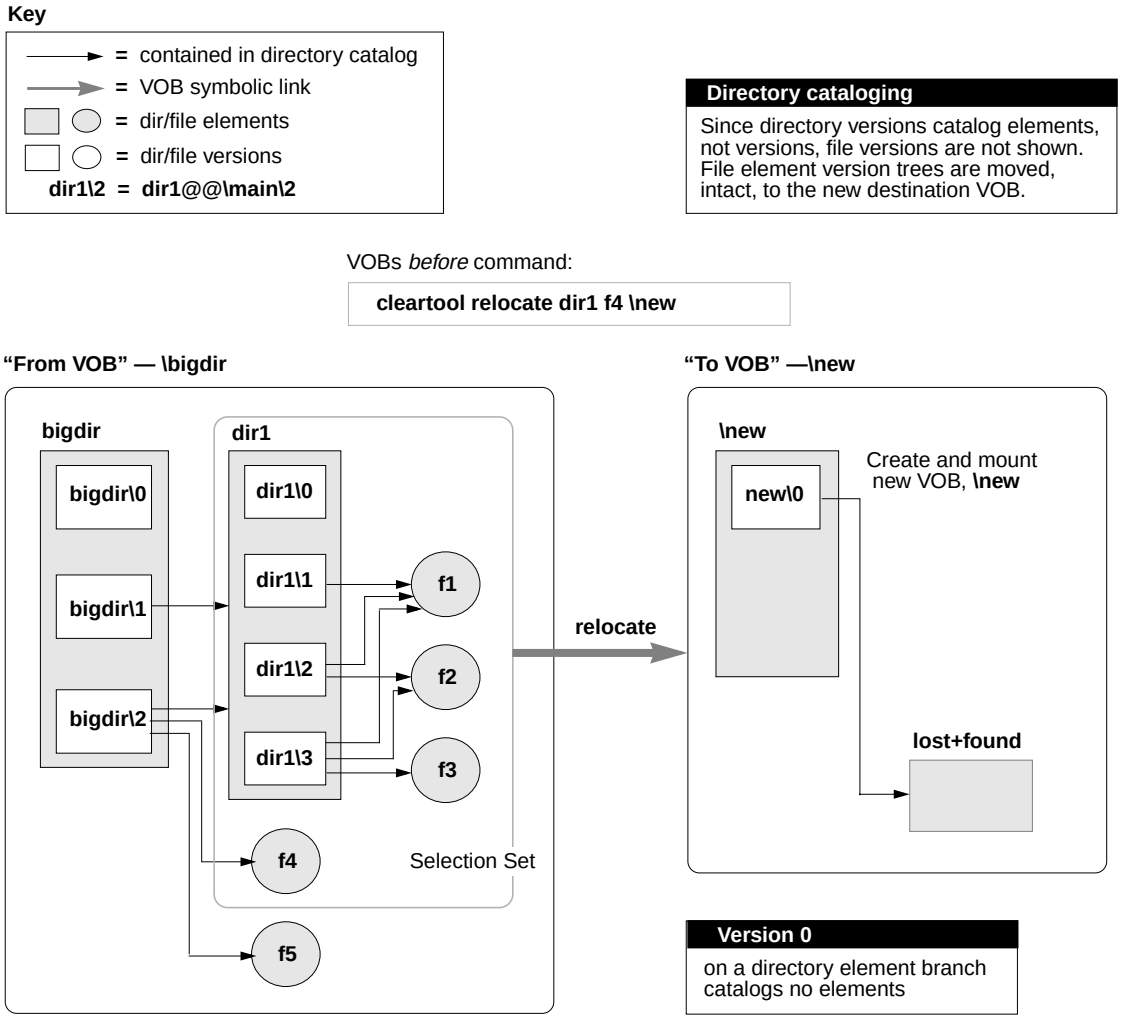
NOTE: All figures use the following shorthand notation for directory versions:

dir1\1 = dir1@@\main\1

Figure 18 shows the source and target VOBs, **\bigdir** and **\new**, before relocating one directory element (**dir1**) and four file elements (three contained by **dir1**, plus the single element **f4**). This is the applicable **relocate** command sequence:

```
c:\> net use v: \\view\main        (view selects \main\LATEST in source and target VOBs)
c:\> v:
v:\> cd \bigdir
v:\bigdir> cleartool relocate dir1 f4 \new
```

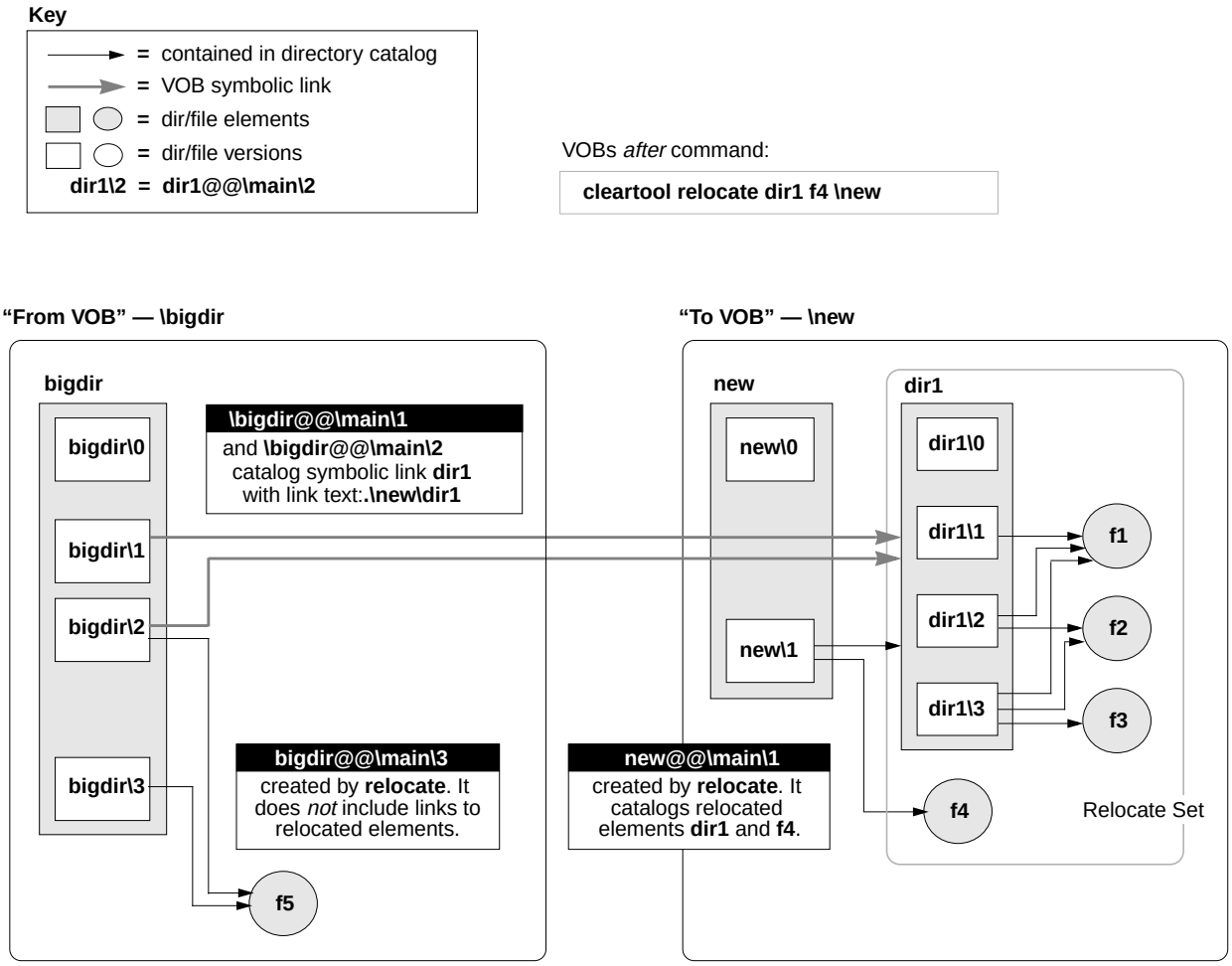
Figure 18 Elements Cataloged by Directory Versions Before Relocate Operation



The **relocate** command defines a selection set that includes **dir1**, **f1**, **f2**, **f3**, and **f4**.

Figure 19 shows the VOBs after **relocate** runs.

Figure 19 Directory Version Cataloging After Relocate Operation



Cataloging in the Source VOB

In the source VOB, after **relocate** completes, preexisting parent directory versions (**bigdir@@\main\1** and **bigdir@@\main\2** in Figure 19) include symbolic links to the moved elements. These links provide stability for historical views, which can continue to view the VOB as it existed before the relocate operation.

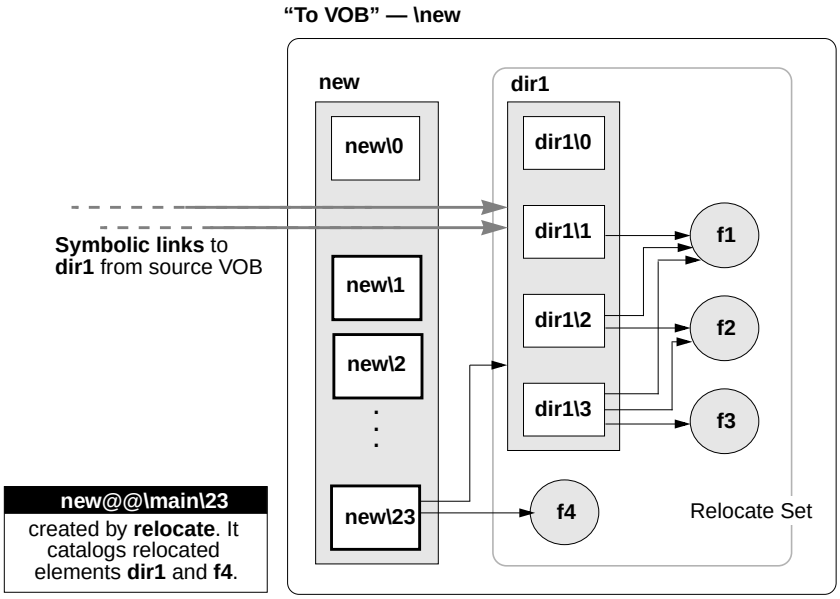
relocate checks in a new version of each relocated element's containing directory (**bigdir@@\main\3** in Figure 19). The new directory version created by **relocate** does not catalog

any relocated elements. In general, this result is desirable; it forces you to address any anomalies in your active development environment that may have resulted from the relocate operation. A guiding principle is that as many views, scripts, and so on, as possible find relocated elements at their new locations, rather than through VOB symbolic links left behind in the original VOB. See *Symbolic Links* on page 219 for some reasons to avoid relying too heavily on VOB symbolic links. See *Updating Directory Versions Manually* on page 222 for information on adding symbolic links to relocated elements where circumstances warrant it.

Cataloging in the Target VOB

In the destination VOB, only the latest version of the target directory catalogs relocated elements. Figure 19 illustrates the common, recommended scenario involving a new destination VOB. However, even if the **new** directory has many versions, only the version created automatically by **relocate**—on the target directory branch selected by the current config spec—catalogs the moved elements. Figure 20 shows the destination VOB from Figure 19, expanded to include multiple versions of the target directory.

Figure 20 Destination VOB That Includes Multiple Versions

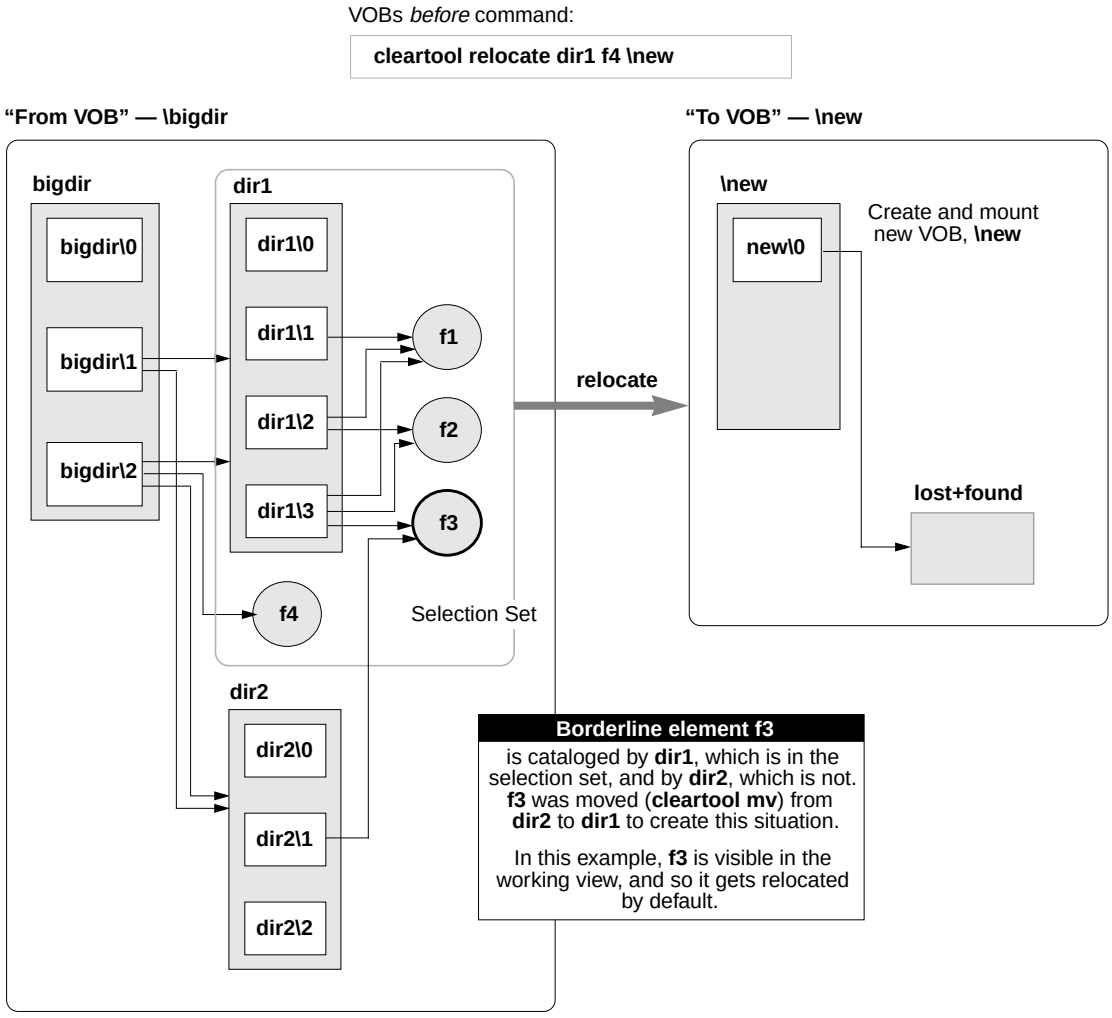


See *Updating Directory Versions Manually* on page 222 for information on adding symbolic links to relocated elements where circumstances warrant it.

Relocating Borderline Elements

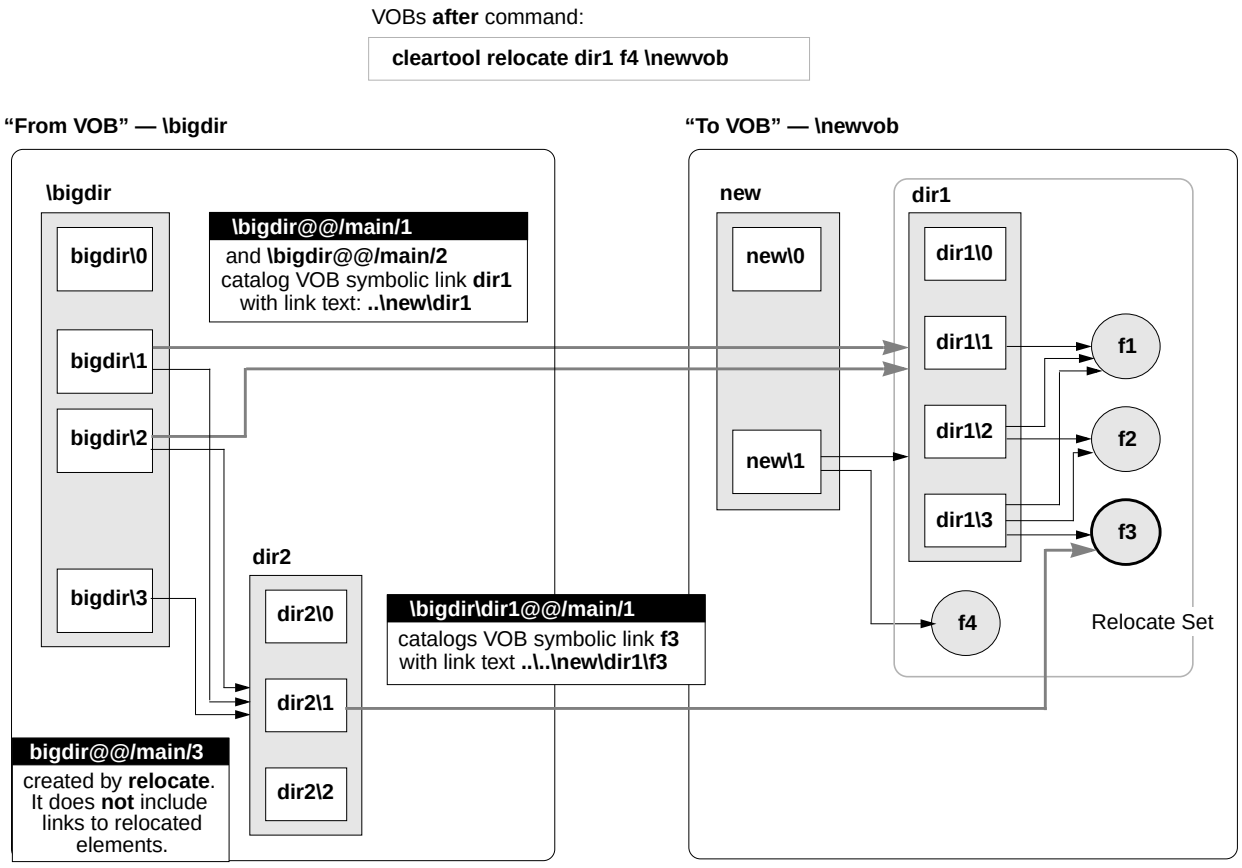
Figure 21 illustrates the relocate operation shown in Figure 18 and Figure 19, but this time the source VOB includes a *borderline element* (**f3**). This element is cataloged both in a version of a directory in the selection set and in some version of a directory outside the selection set (in a directory that stays behind). (See key in Figure 18 on page 210.)

Figure 21 Source VOB That Includes a Borderline Element



File element **f3** is cataloged in both **dir1** and **dir2**. **dir1** is in the selection set, but **dir2** is not. Assume for now that the config spec for the administrator's working view includes a **\main\LATEST** rule that makes **f3** visible as **\bigdir\dir1\f3**. Because **f3** is visible to the working view, **relocate** moves it by default, as shown in Figure 22.

Figure 22 Source and Destination VOBs with Borderline Element Relocated

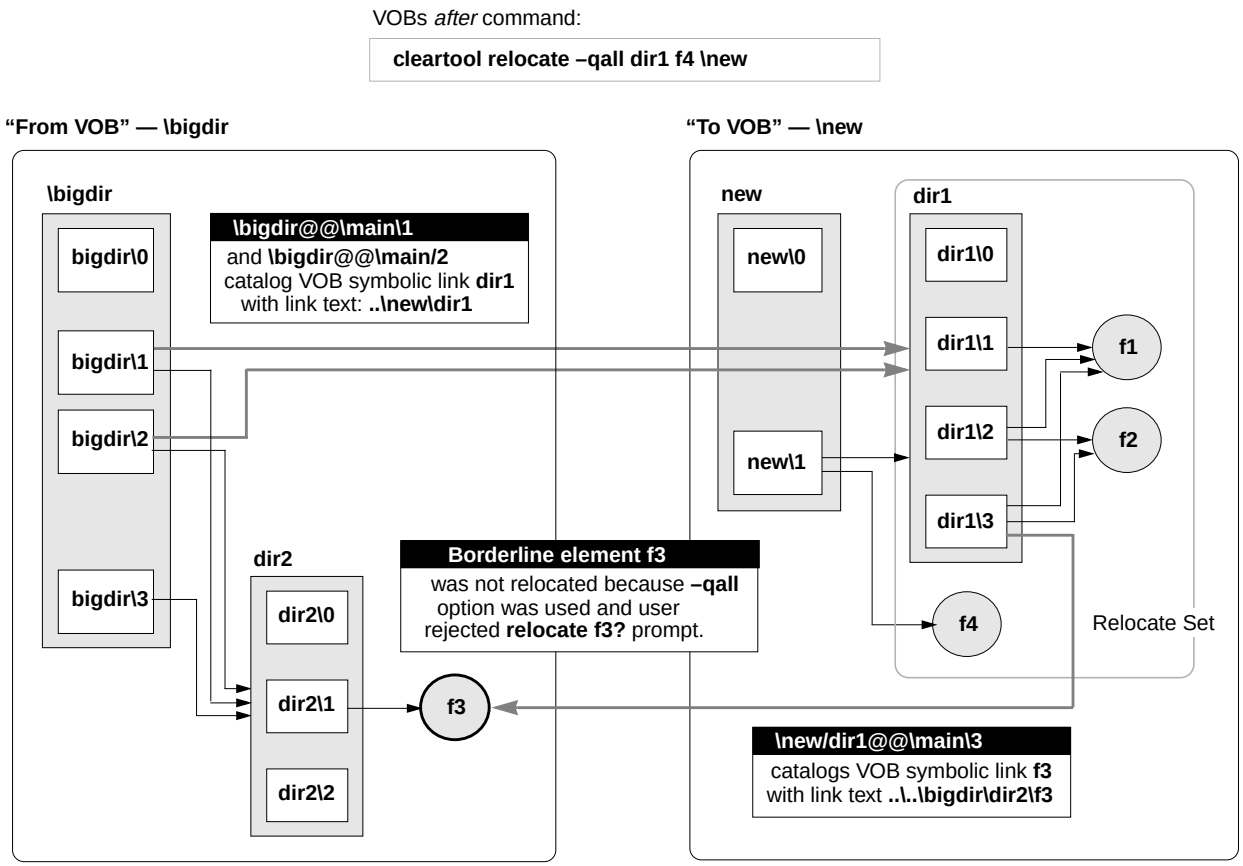


Element **f3** is relocated, and the directory version in the source VOB that referenced it (**dir2@@\main\1**) is updated with a VOB symbolic link to its new location.

Figure 23 shows borderline element **f3** left behind in the source VOB. These different scenarios may yield this result:

- The **–qall** option to **relocate** is used. In this case, each borderline element triggers the `relocate this element or not?` prompt.
- At relocate time, if the working view selects no version of **f3**, it stays behind by default. (In this case, using **–qall** yields an opportunity to relocate **f3** anyway.)

Figure 23 Source and Destination VOBs with Borderline Element Not Relocated



The next section follows on these examples to examine the administrator's role in a relocate operation.

14.3 Before Relocating Elements

Take the following steps before moving elements from one VOB to another:

1. **Inform users of your intentions**. The affected source and target VOB elements must be inactive during the relocate operation.

2. **Check for views with checkouts in the VOB.** Use the ClearCase Administration Console's VOBs node. This node has a Referenced Views subnode that lists all views with checkouts in the VOB and allows you to manage these objects from the console window. You can also use the **cleartool lscheckout** command to list checkouts on a per-view basis.

3. **Resolve any checkouts.** Resolving checkouts typically involves notifying the appropriate users about the problem. **relocate** aborts if it encounters an active checkout in any directory it is trying to move.

4. **Establish a working view.** Use a working view whose config spec selects the branch (typically **\main**) on which the move is to occur.

   This step is very important. Your view must be able to see and check out elements in both the source and destination VOBs. Therefore, a working view configured without a **CHECKEDOUT** rule, for example, is inappropriate. Also, run **relocate** with the same view or with the same config spec that you will use to adjust makefiles, rebuild libraries, reset config specs, modify development tools, or complete any other work that may accompany the relocate task. See also *After Relocating Elements* on page 219.

5. **Run relocate in test mode.** Run the intended **relocate** command and monitor its output, but stop short of moving any elements by responding **no** at this prompt:

   ```
   Do you want to relocate these objects? [no]
   ```

   You may want to include the **–qall** option in your test run, to examine **relocate**'s potential handling of borderline elements:

   cleartool relocate –qall dir1 \new

   When **relocate** encounters a borderline element, its output looks like this:

   ```
   Element "f3" is
     located outside the selection tree as "f3" in
     directory "\bigdir\dir2",
     located inside the selection tree as "f3" in
     directory "\bigdir\dir1".
   Do you want to relocate it? [no]
   ```

14.4 Common Errors During a Relocate Operation

The following conditions are the most frequent causes of failed relocate operations:

- Checked-out files in relocated directories
- Locked type objects in the target VOB
- Triggers on **rmelem** that prevent **relocate** from removing elements from the source VOB after they have been created in the target VOB

In general, you can restart **relocate**. You fix the reported error and restart **relocate** with the same command line. Errors that occur during source VOB element removal require manual repair.

Errors Not Related to Source VOB Element Removal

In the event of an error:

1. **Stop and fix the problem.** For example, if **relocate** reports a locked type in the target VOB, unlock it. If it reports a checked-out version in the relocate set, resolve it.

2. **Restart relocate.** When invoked with an identical command line, **relocate** resumes processing from the interrupt point, provided that these conditions are met:

   - You do not release source VOB element locks that **relocate** sets automatically.
   - No user modifies the elements being moved (new checked-in versions, new labels, and so on).
   - (**–qall** only) You answer all `relocate this object?` queries the same way.

   If any of these conditions is not met, **relocate** starts processing from the beginning, and it may encounter new problems that return you to Step #1 of this procedure.

Errors During Source VOB Element Removal

After **relocate** has re-created elements in the target VOB, it removes the elements from the source VOB. If this step fails (typically, due to a trigger on the **rmelem** operation), you must remove the elements manually, as follows:

1. **Find element OIDs from log file.** Examine the relocate log file (*view-stg-dir***relocate.log.***date-time*) and find the lines that specify the unique object IDs (OIDs) of the elements that **relocate** tried to remove but could not. You must remove these elements manually.

2. **Remove the elements.** For each element reported in the log file, run this command:

   cleartool rmelem oid:*OID-reported-in-relocate-output*

3. If necessary, remove the trigger that prevented **relocate** from removing the elements.

To avoid these errors, remove any **rmelem** triggers on elements before relocating them.

14.5 After Relocating Elements

Although **relocate** leaves behind symbolic links to keep VOB namespace consistent, you probably need to make some additional adjustments in your development environment. Check existing views (config specs), build scripts, triggers, and any other tools that may rely on access to the relocated elements. Update these tools to access relocated elements at their new location, rather than relying on symbolic links.

The section *Symbolic Links* addresses potential problems associated with symbolic links. The section *Cleanup Guidelines* outlines the steps you must take to stabilize the development environment. Finally, *Updating Directory Versions Manually* offers techniques for fine-tuning the symbolic links left behind by **relocate**.

Symbolic Links

VOB symbolic links provide a powerful mechanism for linking MVFS objects, but you must be aware of their limitations:

- In general, **cleartool** commands do not traverse VOB symbolic links. Instead, 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.
- Config specs do not follow symbolic links to other VOBs.
- Build scripts may include operations that fail on symbolic links.
- If you move a relocated element with **cleartool mv**, any symbolic links from the source VOB are broken.

- On UNIX hosts, broken symbolic links—those with nonexistent targets—are not visible to most file-system commands. These include those links that have accurate pathname information but point to elements not selected by the current view. The **cleartool ls** command lists these objects.

Upgrading Views That Rely on Symbolic Links

Views left to access relocated elements by means of symbolic links may have problems, even though they see relocated elements without difficulty. For example, you cannot check out a VOB symbolic link, even if it points to an element. There are several ways around this limitation, if a view with a more direct path to elements in the target VOB exists:

- Reset the view's config spec to use absolute pathnames to the new VOB. This is the most thorough and predictable approach. But it may not be practical when many elements have been relocated and the config spec needs rules that specify a large number of individual elements.
- Add labels to relocated versions, and configure the view to select these elements with a label-based rule.
- Apply a particular branch type to relocated elements, and configure the view to select this branch.

Cleanup Guidelines

To clean up after the relocate operation:

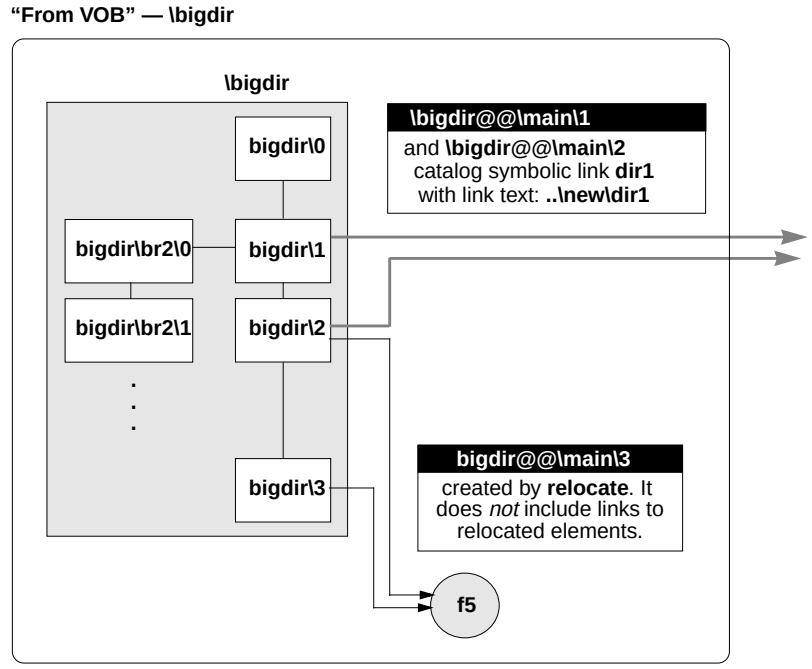
- **Use the view that was used for the relocate operation.** Before adjusting config specs, modifying build rules, and so on, activate the same view, or the same config spec, that was in effect at relocate time.

  In the source VOB, **relocate** completes its operation by checking in the parent directories of all relocated elements. Recall that this last checked-in directory version does not include symbolic links to relocated elements. Presumably, your view selects this version, in which relocated elements do not appear. Working in this view also allows you to track tools, build rules, and other potentially broken references to the relocated elements.

- **Check important and historical views to see whether symbolic links work.** Some kinds of config specs are likely to have trouble:

- Config specs with explicit pathname rules to relocated elements. (Config specs do not follow symbolic links to other VOBs.)
- Config specs that use predominantly label-based rules. In this case, you may choose to add labels to relocated elements and their containing directories in the target VOB.
- Config specs that look for relocated elements on a branch other than that on which the relocate operation was performed. Figure 24 shows the source VOB from Figure 19, expanded to include a second branch on the directory that contained the relocated elements. Note that **relocate** adds symbolic links only to versions on the branch used to select the element at relocate time. (See the key in Figure 18 on page 210.)

Figure 24 Source VOB with Multiple Branches on Parent Directory



- **Fix broken views.** After you find the problem config specs, use one of the techniques listed in *Upgrading Views That Rely on Symbolic Links* on page 220 to make relocated elements visible again.

Updating Directory Versions Manually

You can modify the results of a **relocate** operation by adding and removing VOB symbolic links to specific directory versions manually. However, these operations alter the intended results of **relocate**, and they are rarely necessary.

WARNING: The techniques described here are powerful and potentially dangerous. Do not use **cleartool ln –nco** or its companion command **rmname –nco** carelessly; they make permanent VOB changes without leaving behind an event history that you can trace easily. To use these commands, you must be the VOB owner or the privileged user.

Fixing Symbolic Links Created by relocate

In some cases, the VOB symbolic links that **relocate** creates automatically may have to be modified to point to their intended targets. An incorrect symbolic link in a specific directory version can be removed with **cleartool rmname –nco** and replaced with **cleartool ln –nco**.

relocate has to make an educated guess about the relationship between the source and target VOB roots and about the relationship between the source and target directories. The local host map, VOB mounting and naming conventions on UNIX, as well as drive assignment, disk sharing, and naming conventions on Windows can sometimes lead to an incorrect guess.

To help identify the sources and targets for symbolic links, **relocate** connects a symbolic link object to the target element object with a hyperlink of type **HyperSlink**. Use the **cleartool describe** command on the symbolic link to display information about this hyperlink, which can help to repair the symbolic link.

After you relocate elements, if you move any of them again with **cleartool mv**, symbolic links are not updated automatically. You can use this technique to update the links manually.

The following commands replace a relative symbolic link created by **relocate** with an alternative absolute pathname to the target VOB.

```
cd \bigdir
cleartool rmname –nco \bigdir@@\main\2\dir1
Modify non-checkedout directory version "\bigdir@@\main\2"? [no] yes
Link removed: "\bigdir@@\main\2\dir1"

cleartool ln –nco \new\proj2\dir1 \bigdir@@\main\2\dir1
Modify non-checkedout directory version "\bigdir@@\main\2"? [no] yes
Link created: "\bigdir@\main\2\dir1"
```

Modifying Old Target Directory Versions to See Relocated Elements

In Figure 20 on page 212, **relocate** checks in a new version of the destination directory, and only that version catalogs the relocated elements. Now, consider this scenario:

- You have a view, **alh_port**, that selects a previous version of the target directory (**\new@@\main\2**, for example).
- You want the **alh_port** view to be able to see the newly relocated **dir1** element at its new home, **\new\dir1**, rather than at its old location, **\bigdir\dir1**.

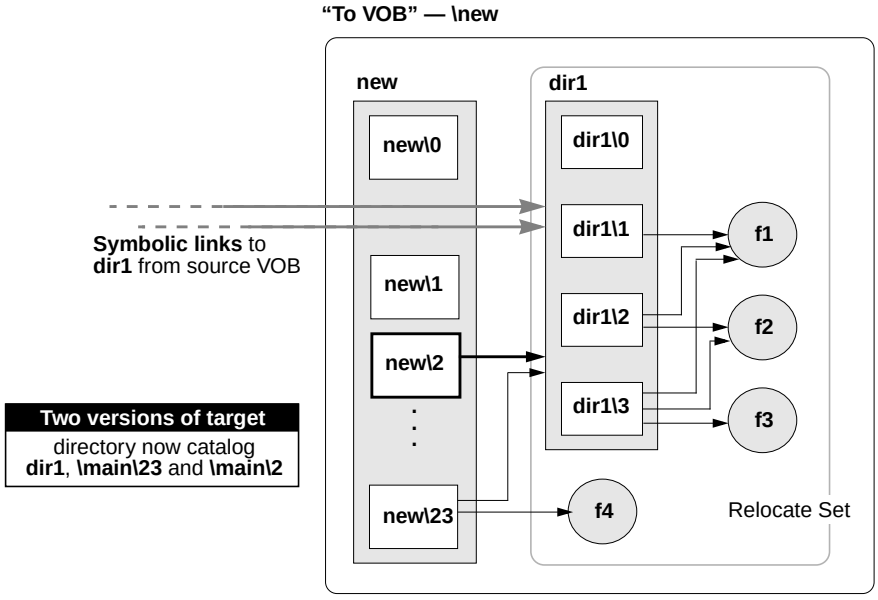
In a case like this, you can manually update previous versions of **relocate**'s destination directory to catalog relocated elements. For example, to add **dir1** to directory version **\new@@\main\2**:

1. Log on as the privileged user.
2. Create the VOB symbolic link. The following command requires special privileges because it modifies a directory version's contents without checking it out or changing its event history.

   ```
   cd \new                          (and your working view must select some version of dir1)
   cleartool ln –slink –nco dir1 \new@@\main\2\dir1
   Modify non-checkedout directory version "\new@@\main\2"? [no] yes
   Link created: "\new@@\main\2\dir1"
   ```

Figure 25 shows the effect of the **cleartool ln –nco** command. (See the key in Figure 18 on page 210.)

Figure 25 Destination VOB After Modifying Old Version of Destination Directory



Modifying Newest Version of Source Directory to See Relocated Elements

In Figure 19 on page 211 and Figure 22 on page 215, the latest version of a relocated element's parent directory does not catalog that relocated element. In some circumstances, you may want to add such cataloging manually, in the form of symbolic links. For example, the following command sequence updates the from-directory version **bigdir@@\main\3** (Figure 19) to see relocated directory **dir1** at its new location:

```
cd \new
cleartool ln –slink –nco dir1 \bigdir@@\main\3\dir1
Modify non-checkedout directory version "\bigdir@@\main\3"? [no] yes
Link created: "\bigdir@@\main\3\dir1"
```

Administering Views

Views and View Storage

15

This chapter describes ClearCase LT views. In Rational ClearCase LT, view administration is much simpler than VOB administration, involving little more than occasional backups, periodic maintenance, and attention to access control issues.

15.1 ClearCase LT Views

A ClearCase LT development environment can include any number of views. A typical view is created and used by a single user, or perhaps by a small group of users tackling a particular task as a team. A view provides a workspace where users access versions of ClearCase LT elements and use other file-system objects that are outside ClearCase LT control. The workspace is populated with versions of file and directory elements based on the contents of the view's configuration specification, or config spec.

Config specs can be created in many ways. They can be written by a developer or project leader and referenced by all views used in a project; they can also be created automatically by the Unified Change Management (UCM) tools. A default config spec is used if no other is specified.

File and directory elements can be loaded into the view from any VOB on the ClearCase LT server. A VOB namespace browser allows you to select the initial set of VOBs and subdirectories that will be loaded into a newly created view. A load rule editor allows you to modify these selections later to include new VOBs or subdirectories or to exclude VOBs and subdirectories that the view no longer needs.

The **mkview** reference page provides detailed information on views.

15.2 View Contents

A ClearCase LT view is also known as a *snapshot view* because it keeps a "snapshot" of VOB data as ordinary files on the ClearCase LT client's local disk or on another disk that the client can access over the network. A view consists of two directories:

- The *view directory* itself contains versions of file and directory elements loaded from the VOB, along with view-private files that are generated in the course of local operations (and may later be added to source control if appropriate). These copies are ordinary file-system objects—files and directories—with ordinary file-system protections and access control. The view directory can be located on a ClearCase LT client or on any disk in the network that a ClearCase LT client can use for storage.

- The *view storage directory* includes a view database and other system files that track the correspondence between certain VOB database objects and objects loaded into the view. The view storage directory must be located on the ClearCase LT server. The most significant contents of this directory are the following:

  - **View database.** The **db** subdirectory contains the binary files managed by the view's embedded database. The database keeps track of the loaded VOB objects and checked-out versions in the view.

  - **Administrative directory.** The **admin** directory contains data on disk space used by the view. A job that the ClearCase LT scheduler runs by default generates this data periodically.

Each view has an associated **view_server** process, which runs on the ClearCase LT server. View storage directories are always located on the ClearCase LT server. View directories can be located on any accessible storage, but are most often located on individual developers' workstations. It is important to know where the view directory and the view storage directory are located, because you need to back them both up together if you want to be able to restore them as a coherent unit. For more information, see *Backing Up a View* on page 229.

15.3 View Storage Areas

View storage must be created in a ClearCase LT server storage location. These storage locations are created when the ClearCase LT server is installed. You can add new server storage locations later using the **mkstgloc** command or the ClearCase Administration Console.

Normally, the view creation tools (**clearviewtool** on Windows or the **mkview** command) select an appropriate view storage area; but a user can override this default and select any advertised view storage area on the ClearCase LT server.

15.4 Backing Up a View

When you back up and restore a view, the utility you use for the backup must maintain the original modification times and ownership of all files and directories in the view. If it does not, loaded files become hijacked. The **ccopy** utility (*ccase-home-dir***\etc\utils\ccopy)** provides this capability. Although **ccopy** copies all the ownership information required by ClearCase LT, it does not copy the full security descriptor of an object. Use of **ccopy** has the side effect of granting the user who executes the command full access to the copied object. If you need to have all security descriptor information copied, use a copy utility that preserves this information (for example, the **scopy** command from the Windows NT Resource Kit).

Backing up views is less complicated, and in some ways less important, than backing up VOBs. If you already have a system in place that backs up files and directories on ClearCase LT client hosts, that system will also back up view directories located on those hosts. These directories contain the files that developers create and modify. If you need to recover one or more of these files from a backup, perhaps because a developer has inadvertently deleted something that was changed significantly since the last time it was checked in, you can usually do this in the same way that you recover any file from backup: simply recover the files to the view directory. The developer should be able to continue working on them and check them in when needed. If deleted file had been loaded into the view but not checked out, reloading the view recovers the file.

Any backup and recovery plan that includes the view storage directory (for example, a disaster recovery plan for the ClearCase LT server) must take pains to ensure that, for reach view, the view storage directory and the view directory are backed up at the same time. The view storage directory maintains detailed information about the checkin/checkout state of the files in the view directory; any skew between what the view storage directory records and the actual state of the view when both are restored will result in apparent anomalies such as files that appear to be hijacked and files that appear to be "checked out but removed."

When both the view and view storage directories reside on the ClearCase LT server, it is easier to ensure that both directories are backed up at the same time. When the directories are on different hosts (for example, the view directory is on a ClearCase LT client), you need to make sure that developers do not check in or check out files or directories while the backup is in progress.

To back up a view:

1. **Determine the location of the view storage directory.** Use the ClearCase Administration Console or run **lsview** to display view storage information. If your backup program runs over the network, you need the `Global path`. If your backup program runs locally, you need the `View server access path`:

   ```
   C:\> cleartool lsview –long akp_vu
   Tag: akp_vu
       Global path: \\neptune\home\akp\views\akp.vws .
    ...
       Region: dvt
    ...
   View on host: neptune
   View server access path:
   /home/akp/views/akp.vwsc:\home\akp\views\akp.vws .
    ...
   ```

2. **Ensure self-consistency of the backup**. To keep the view inactive while it is backed up, warn ClearCase LT users not to use it during the backup, or use the **chview** command to prevent users from updating the view database.

   ```
   cleartool chview –readonly akp_vu
   Properties: readonly
   ```

3. **Enter the backup commands.** Back up the entire view directory tree. If the view storage directory is outside the view's directory tree, be sure to back up both the view directory tree and the view storage directory.

4. If you made the view read-only in Step #2, make it writable.

   ```
   C:\ cleartool chview –readwrite akp_vu
   Properties: readwrite
   ```

5. If you renamed the view storage directory in Step #2, rename it to its registered location.

15.5 Restoring a View from Backup

Use the following procedure to restore a backup view storage directory. The view is the same one discussed in *Backing Up a View* on page 229.

If you are restoring a view, you must restore both the view directory tree and the view storage directory. If the view storage directory is located outside the view's directory tree, you must be sure to restore both.

NOTE: If you are restoring the view storage directory to a new location, you must re-register the view at its new location (Step #5). If you are restoring a view's directory tree to a new location, but the location of the view storage directory (outside the view directory tree) remains unchanged, you do not need to re-register the view.

1. **Log on to the ClearCase LT server.**

2. **Check available disk space**. Make sure that there is enough free space in the storage location to hold the restored copy. If necessary, create a new view storage location to hold the restored copy.

3. **Move the original view aside.** Rename or delete the view directory and the view storage directory:

   ```
   C:\home\akp\views> ren akp.vws akp.vws.OLD
   ```

4. **Load the backup.** Restore the view directory and the view storage directory from the backup medium.

   NOTE: For ClearCase LT servers on UNIX, each view storage directory includes a directory named **.identity**, which stores files with special permissions: the *setUID* bit is set on file **uid**; the *setGID* bit is set on file **gid**. You must preserve these special permissions when you restore a view backup:

   - If you used **tar** to back up the view, use the **–p** option when restoring the view. In addition, run the **tar** command as the *root* user. If you do not, the **–p** flag is ignored.

   - If you used **cpio** to back up the view, no special options are required in the **cpio** command that restores the backup data.

5. **Run cleartool recoverview.** Use the following command to update the view database:

   ```
   %C:\> cleartool recoverview –tag akp_vu
   ```

The view is now ready for use.

15.6 Configuring Text Modes for Views

UNIX and Windows observe different conventions for terminating lines in text files. UNIX systems normally terminate lines with a single **<LF>** (line feed, or new line) character and Windows systems terminate lines with a two-character **<CR><LF>** (carriage return, line feed) sequence. Some Windows applications can read and display files in either format, some Windows applications always write files using **<CR><LF>** format, and some Windows applications can be configured to determine which format to use.

Because of these different conventions, line-termination problems are a typical result of the use of text editors on files that must be edited on both UNIX and Windows platforms. For example, a file with the contents:

```
abc
def
ghi
```

would look like this if it were created by a typical Windows editor and read by a typical UNIX editor (for example, **vi**):

```
abc^M
def^M
ghi^M
```

The UNIX text editor renders the **<CR>** character as **^M**. The same file would look like this if it were created by a typical UNIX editor and read by a typical Windows editor (for example, Notepad):

```
abc■def■ghi
```

To better support parallel development in mixed operating system environments, Rational ClearCase LT provides a *text mode* setting for views that controls how line terminators are handled when text files are presented to applications.

Text Modes

Each view has a text mode setting that specifies how it handles line terminator sequences. This setting only applies to file elements whose element type is **text_file** or a subtype of type **text_file**. You determine a view's text mode when you create the view. You cannot change the text mode of a view after the view has been created.

A ClearCase LT site configuration parameter determines the default text mode for view creation. For details, see the reference page for the **setsite** command.

You can create a view in any of three text modes:

- **transparent text mode**. In a view created in **transparent** text mode, ClearCase LT does no line terminator processing. If no site default is specified, views are created in **transparent** text mode by default. To create a view in **transparent** text mode regardless of the site default, clear the **Use interop (insert_cr) text mode** check box on in the **Advanced** options of the **View Creation Wizard**, or use the **-tmode transparent** option to the **mkview** command.

  If all the developers at your site use the same development platform (Windows or UNIX) or use tools that are compatible with either line-termination convention, all views should be created in **transparent** text mode.

- **insert_cr text mode.** In a view created in **insert_cr** text mode, ClearCase LT inserts a **<CR>** character before every **<LF>** character. To create a view in **insert_cr** text mode, select the **Use interop (insert_cr) text mode** check box in the **Advanced** options of the **View Creation Wizard** or use the **-tmode insert_cr** option to the **mkview** command.

- **strip_cr text mode.** In a view created in **strip_cr** text mode, ClearCase LT strips the **<CR>** character from every **<CR><LF>** sequence. To create a view in **strip_cr** text mode, use the **-tmode strip_cr** options to the **mkview** command. You cannot create a view in **strip_cr** mode from the **View Creation Wizard**.

In a view created in either **insert_cr** or **strip_cr** text mode, ClearCase LT adds or removes the **<CR>** characters whenever it updates the view. It reverses the **<CR>** manipulation (adding or removing **<CR>** characters as appropriate) during the checkin process.

NOTE: You cannot create a view in **strip_cr** mode from any GUI.

Determining a View's Text Mode

If you do not know a view's text mode, you can find out in one of two ways:

- In Windows Explorer, right-click a drive that represents a view. Then click **ClearCase** > **Properties of View**. The view's text mode is displayed on the **Access** tab.

- Use the **cleartool lsview** command with the **–properties –full** options.

With these methods, **transparent** text mode is displayed as `unix`, **strip_cr** text mode is displayed as `strip_cr`, and **insert_cr** text mode is displayed as `msdos`.

Choosing a Text Mode for a View

ClearCase LT does not enforce any policy governing access to VOBs based on a view's text mode, and a user editing a file in a view that has the "wrong" text mode configuration can cause problems for other users who need to edit that file. Most sites with both Windows and UNIX development platforms should adopt a policy that allows users of the primary development platform to create views in **transparent** text mode and that limits the use of **strip_cr** or **insert_cr** text modes for the minority of platforms that require different line-termination conventions.

If most of your users are developing software on UNIX:

- UNIX clients should use views created in **transparent** text mode.
- Windows clients should use views created in **insert_cr** text mode.

If most of your users are developing software on Windows:

- Windows clients should use views created in **transparent** text mode.
- UNIX clients should use views created in **strip_cr** text mode.

No matter what policy you adopt, it is important to maintain a consistent combination of client platform, view text mode, and VOB. If a client uses a view in an interop text mode (**strip_cr** or **insert_cr**) to access a VOB, then later begins using a view in **transparent** text mode to access the same VOB, versions created by the views in different text modes will be difficult to compare or merge.

15.7 File Naming Issues in Mixed Environments

If your ClearCase LT community includes both UNIX and Windows computers, users should be aware of potential file access problems stemming from the different file naming conventions observed by each platform.

Case-Sensitivity

On UNIX, the native file system and commands are case-sensitive, and it is legal for two file names to differ only in the case of one or more of their characters.

On Windows, the native file system and commands are case-insensitive, though case is preserved when displayed. File names that differ only in the case of one or more characters are considered the same.

On both UNIX and Windows, most ClearCase LT commands are case-sensitive. ClearCase LT users on Windows must provide ClearCase LT commands that take a file name argument with the case-correct file name. As the following example shows, Windows commands can access a file regardless of the case in which you type its name. ClearCase LT commands require the case to be correct.

```
C:\myview>dir readme.txt
.
05/02/00  04:57p                 2,511 readme.txt
.
.
C:\myview>dir README.txt
.
05/02/00  04:57p                 2,511 readme.txt
.
.
C:\myview>cleartool lshistory README.TXT
cleartool: Error: Pathname not found: "README.TXT".

C:\myview>cleartool lshistory readme.txt
02-May.16:58   roc        create version "readme.txt@@\main\4"
.
.
```

Character Sets

Various characters that are allowed in UNIX file names are not allowed in Windows NT file names. File names that include any of these characters are recognized by Windows NT and cannot be loaded into a Windows snapshot view. Table 4 lists these characters.

Table 4 Characters That Are Not Allowed in file names on Windows

?	*	/	\	|	<	>

Administering Scheduled Jobs

16 Managing Scheduled Jobs

The ClearCase job scheduling service runs programs periodically. Using the scheduler, you can define jobs to be run one or more times at specified intervals or in specified sequences. The scheduler also provides the following features:

- E-mail notifications of job-related events
- Remote administration using the ClearCase Administration Console
- An access control list (ACL) that controls access to the schedule and to the ACL itself
- A predefined set of jobs for managing disk space used by VOBs and views

The scheduler runs on the ClearCase LT server.

16.1 Tasks and Jobs

The scheduler manages ClearCase jobs and arranges to run them at specified times. A job consists of an executable program, or task, that the scheduler runs one or more times with a given set of arguments. A task is a program that is available for scheduling. A job is a combination of a task with a schedule that specifies when and under what conditions the task actually runs. For a job to run, two conditions must exist:

- The task must be defined for the scheduler.
- A job that runs the task must be defined and given a schedule (and possibly other attributes).

This section discusses how ClearCase distinguishes and initializes tasks and jobs. For information on creating, editing, and deleting tasks and jobs, see *Managing Tasks* on page 242 and *Managing Jobs* on page 244.

Task and Job Storage

The scheduler relies on two data repositories:

- A database of tasks available for scheduling
- A database of jobs, or scheduled tasks

The database and the jobs, along with various other ClearCase administrative tools, are installed on the ClearCase LT server under the directory represented here as *ccase-var-dir*. On UNIX, this directory is **/var/adm/atria**. On Windows, the file is in *ccase-home-dir***var**.

A task must be defined in the task database before you can schedule it. The task database is a single text file *ccase-var-dir***scheduler\tasks\task_registry**. You can add task definitions to the task database by editing this file in a text editor. You must not change the definitions of standard ClearCase tasks, but you may add your own task definitions at the end of the file. For more information, see *Managing Tasks* on page 242.

Standard ClearCase tasks reside in the directory *ccase-home-dir***config\scheduler\tasks**. You cannot edit these tasks. Tasks that you define can reside anywhere in the file system, but the recommended location is the directory *ccase-var-dir***scheduler\tasks.** This directory initially contains two task placeholder scripts that run on a regular basis but by default do nothing. To change this default:

- Edit **ccase_local_day.[sh | bat]** to add any user-defined operations to be run daily.
- Edit **ccase_local_wk.[sh | bat]** to add user-defined tasks to be run weekly.

The database of jobs is a binary file (*ccase-var-dir***scheduler\db)** that you can read and edit only by using the Scheduled Jobs node of the ClearCase Administration Console or the **cleartool schedule** command. For more information, see *Managing Jobs* on page 244.

Task and Job Database Initialization

ClearCase installs a template for an initial task database, containing definitions for standard tasks, as the file *ccase-home-dir***\config\scheduler\tasks\templates\task_registry**. The **albd_server** uses this template to create the first version of the actual task database, *ccase-var-dir***\scheduler\tasks\task_registry**.

ClearCase installs templates for two customized tasks, **ccase_local_day.bat** and **ccase_local_wk.bat**, in the directory *ccase-home-dir***\config\scheduler\tasks\templates**. The **albd_server** uses these templates to create initial versions of these tasks in the directory *ccase-var-dir***\scheduler\tasks**.

ClearCase installs an initial set of job definitions as the text file *ccase-home-dir***\config\scheduler\initial_schedule**. These job definitions rely on task definitions in the task registry template. The **albd_server** uses these job definitions to create the first version of the job database, *ccase-var-dir***\scheduler\db**.

NOTE: Do not edit or delete any files on the ClearCase LT server in the directory tree whose root is *ccase-home-dir***\config\scheduler**.

Job Execution Environment

Each task runs in a separate process started by the **albd_server** on the ClearCase LT server. A task has the following execution environment:

- The task runs with the identity of **root** on UNIX and **NT Authority\system** on Windows.
- The standard input stream is set to an empty file.
- Standard output and error messages are redirected to a file and captured by the scheduler as part of the job's last completion information.
- The current directory is undefined.
- Environment variables are those in effect for the **NT Authority\system** identity on Windows and for **root** on UNIX. In addition, on Windows, **ATRIAHOME** is set to *ccase-home-dir*.

16.2 The Default Schedule

Rational ClearCase has a set of standard jobs, most of which manage disk space in VOB and view storage directories. Some of these jobs run daily. Others run weekly.

Daily jobs:

- Scrub cleartext storage pools of all VOBs, using **scrubber**.
- Copy the VOB database for all VOBs that are configured for snapshots, using **vob_snapshot**.
- Run user-defined daily operations in **ccase_local_day.[sh | bat]**.
- Generate and cache data on disk space used by all views, using **space**.
- Generate and cache data on disk space used by all VOBs, using **space**.

Weekly jobs:

- Scrub some ClearCase logs.
- Scrub the databases of all VOBs, using **vob_scrubber**.
- Run user-defined weekly operations in **ccase_local_wk.[sh | bat].**

For information about how these jobs operate, see the reference page for the command or utility that the job uses. For information on managing VOB storage, see Chapter 10, *Administering VOB Storage*.

16.3 Managing Tasks

A task has two components:

- A program (job) to be executed when the task runs
- A definition for the task in the scheduler's task database

ClearCase has a set of standard executable tasks and standard definitions for these tasks in the task database. You cannot create, change, or delete any standard ClearCase tasks or task

definitions. You can, however, define new tasks in the task database. You can also customize two predefined ClearCase tasks, one of which is run daily and the other weekly in the default schedule. You can add your own procedures to these tasks and can change their schedules.

To view all task definitions in the task registry, use the following command:

cleartool schedule –get –tasks

Creating a Task

To create a new task:

1. Create an executable program suitable to be run in the scheduler's execution environment (see *Job Execution Environment* on page 241). You can place the program anywhere in the ClearCase LT server file system, but the recommended location is the directory *ccase-var-dir***scheduler\tasks**.

2. If you want the program to run daily, you can run it from the existing task **ccase_local_day.[sh | bat]**. If you want the program to run weekly, you can run it from the existing task **ccase_local_wk.[sh | bat]**. Both tasks are in the directory *ccase-var-dir***scheduler\tasks.** If you add your program to one of these customizable tasks, you need take no further action.

3. If you prefer to run your program as a new task, you must add the task to the scheduler's task registry, *ccase-var-dir***/scheduler/tasks/task_registry**. To add a task to this file, use a text editor.

   Tasks are defined using a job-definition syntax documented on the reference page for the **schedule** command. The essential components of a task definition are the following:

   - A unique numeric ID used by a scheduled job to refer to the task
   - A unique name for the task
   - The pathname to the executable program

   WARNING: Place new task definitions at the end of the task registry file. Do not alter or delete any of the standard ClearCase tasks defined in that file.

4. If you have added a task to the scheduler's task registry, you must create a new job to run the task. See *Creating a Job* on page 245. You do not need to create a new job if you have added your program to an existing scheduled job such as **ccase_local_day.[sh | bat].**

Editing a Task

You may need to edit an existing task definition in the scheduler's task registry. For example, if you move the task's executable program to another directory, you must change the pathname in the task definition in the task registry.

WARNING: Edit only tasks that have been added at your site. Do not alter or delete any of the standard ClearCase tasks defined in the task registry.

To change a task definition, use a text editor to edit the task registry file, *ccase-var-dir***scheduler\tasks\task_registry**. When you edit a task, you must use the task-definition syntax documented on the reference page for the **schedule** command.

CAUTION: The scheduler uses a task definition's numeric ID to refer to the task when it runs a scheduled job that uses that task. If you change a task's numeric ID, you must change all references to the task in all scheduled jobs. See *Editing Job Properties* on page 248.

Deleting a Task

Before you delete a task definition, you must remove all references to the task in all scheduled jobs. See *Editing Job Properties* on page 248. To delete a task definition, use a text editor to edit the task registry file *ccase-var-dir***scheduler\tasks\task_registry**.

WARNING: Delete only tasks that have been added at your site. Do not alter or delete any of the standard ClearCase tasks defined in the task registry.

16.4 Managing Jobs

You can create, delete, or edit jobs run by the ClearCase scheduler. You can also run a scheduled job immediately. To manage a scheduled job on any Windows or UNIX host, use the ClearCase Administration Console. The ClearCase Server snap-in on the console has a Scheduled Jobs node from which you can manage jobs on the ClearCase LT server. If the schedule ACL supports write access (see *Managing the Scheduler Access Control List* on page 250), you can also use the **cleartool schedule** command on the ClearCase LT server.

To create, edit, delete, or run a scheduled job, you must have **Change** or **Full** access in the scheduler ACL. To view scheduled jobs, you must have **Read** access in the scheduler ACL. See *Managing the Scheduler Access Control List* on page 250.

Creating a Job

When you create a job, you supply the following information:

- A name and an optional description for the job.
- The task that the job runs. The task must already be defined in the task registry. For more information see *Managing Tasks* on page 242.
- Any arguments that the scheduler passes to the task when the task runs. For a standard ClearCase task that operates on VOBs or views, you can specify that the task operates on particular VOBs or views, or on all VOBs or views on the ClearCase LT server.
- The schedule on which the job runs. See *Specifying a Job's Schedule* on page 246.
- Job-related events that trigger e-mail notifications and recipients for the notifications. See *Specifying Job Notifications* on page 247.
- Whether the scheduler deletes the job after it runs.

To create a new job using the ClearCase Administration Console:

1. In ClearCase Administration Console, navigate to the Scheduled Jobs node for the ClearCase LT server.
2. Click **Action** > **New** > **Job**. This command opens a dialog box in which you supply the information needed to define a new job.

To create a new job using the command line, use the following command:

cleartool schedule –edit –schedule

This command opens in a text editor a file that contains definitions for all currently scheduled jobs. To create a new job, add a definition using the job-definition syntax documented on the reference page for the **schedule** command. You cannot specify any read-only job properties, such as **LastCompletionInfo**. The job runs in the environment described in *Job Execution Environment* on page 241.

Specifying a Job's Schedule

You can arrange for a job to run under two kinds of schedules:

- **Sequential**. The job runs immediately after another job finishes.
- **Periodic**. The job runs on specified days at specified times.

To specify a sequential job, you designate a scheduled job after which the current job is to run. To specify a periodic job, you designate the times when the job is to run. A periodic job can run at four kinds of intervals:

- **Run once**. Specify the date and time at which the job runs once only. A job runs only one time when you specify identical start and end dates. You can also use the Scheduled Jobs node on the ClearCase Administration Console to force immediate one-time execution of any scheduled job.
- **Run daily or every *n* days**. Specify the frequency of the job and the time of day the job starts.
- **Run weekly or every *n* weeks**. Specify the frequency of the job, the days of the week it runs, and the time of day the job starts.
- **Run monthly or every *n* months**. Specify the frequency of the job, the day of the month it runs, and the time of day the job starts.

For daily, weekly, and monthly schedules, you can also specify starting and ending dates for the job, and you can set the job to repeat at intervals during the day.

To specify the schedule for a job, use the ClearCase Administration Console:

1. Navigate to the Scheduled Jobs node of the ClearCase LT server.
2. To specify the schedule for a new job, click **Action** > **New** > **Job**. This command opens a dialog box in which you supply the information needed to define a new job. After you specify the task, click the **Schedule** tab to specify the schedule.
3. To specify the schedule for an existing job, select a job in the detail pane and click **Action** > **Properties**. This command opens a dialog box. Click the **Schedule** tab to specify the schedule.

Or run the following command on the ClearCase LT server:

cleartool schedule –edit –schedule

This command opens in a text editor a file that contains the definitions for all currently scheduled jobs. To specify the schedule for a new or existing job, edit the job's **Schedule** property using the job-definition syntax documented on the reference page for the **schedule** command.

Specifying Job Notifications

The scheduler can send e-mail notifications to recipients you specify. You can also determine particular events that trigger notifications, such as the start of a job or the end of a job that fails.

NOTE: Job notifications require the scheduler to contact an SMTP mail server. On Windows, you must specify the name of this server on the **Options** tab of the ClearCase program in Control Panel on the ClearCase LT server. On UNIX, the scheduler uses the **/bin/mail** program to send notifications.

To specify the notification information for a job, use the ClearCase Administration Console:

1. Navigate to the Scheduled Jobs node for the host on which you want to specify a job's notification information.
2. To specify the notification information for a new job, click **Action** > **New** > **Job**. In the dialog box, supply the information needed to define a new job. After you specify the task, click the **Settings** tab to specify notification events and recipients.
3. To specify the notification information for an existing job, select a job in the detail pane and click **Action** > **Properties**. In the dialog box, click the **Settings** tab to specify notification events and recipients.

Or use the following command:

cleartool schedule –edit –schedule

This command opens in a text editor a file that contains definitions for all currently scheduled jobs. To specify the notification information for a new or existing job, edit the job's **NotifyInfo** property using the job-definition syntax documented on the reference page for the **schedule** command.

Viewing Job Properties

To view properties of a scheduled job, use the ClearCase Administration Console:

1. Navigate to the Scheduled Jobs node for the ClearCase LT server.
2. Select a job in the detail pane and click **Action** > **Properties**. In the dialog box, you can view properties of the job.
3. To view messages and information such as time and status from the last execution of the job, select the job in the detail pane and click **Action** > **Show Completion Details**.

Or use the following commands on the ClearCase LT server:

cleartool schedule –get –schedule

To view the definition of a particular job, use the following command:

cleartool schedule –get –job *job-id-or-name*

To view messages and information such as time and status from the last execution of the job, use the following command:

cleartool schedule –status *job-id-or-name*

These commands display properties of jobs using the job-definition syntax documented on the reference page for the **schedule** command.

Editing Job Properties

To edit an existing job, use the ClearCase Administration Console:

1. Navigate to the Scheduled Jobs node for the ClearCase LT server.
2. Select a job in the detail pane and click **Action** > **Properties**. This command opens a dialog box in which you can edit properties of the job. You cannot edit any read-only job properties.

Or use the following command:

cleartool schedule –edit –schedule

This command opens in a text editor a file that contains definitions for all currently scheduled jobs. Edit the properties of the job using the job-definition syntax documented on the reference page for the **schedule** command. You cannot edit any read-only job properties, such as **LastCompletionInfo**.

If you have a text file of job definitions that uses the scheduler's job-definition syntax, you can replace the entire schedule with the job definitions in your file by running the following command, where *defn_file_pname* represents your file of job definitions:

cleartool schedule –set –schedule *defn_file_pname*

Running a Job Immediately

To run a scheduled job immediately, use the ClearCase Administration Console:

1. Navigate to the Scheduled Jobs node for the ClearCase LT server.
2. Select the job in the detail pane and click **Action** > **Run Now**.

Or use the following command:

cleartool schedule –run *job-id-or-name*

The job runs in the scheduler's execution environment. See *Job Execution Environment* on page 241.

Deleting a Job

To delete a scheduled job, use the ClearCase Administration Console:

1. Navigate to the Scheduled Jobs node for the ClearCase LT server.
2. Select a job in the detail pane and click **Action** > **Delete Job**.

Or use the following command:

cleartool schedule –delete *job-id-or-name*

16.5 Managing the Scheduler Access Control List

The scheduler maintains a single access control list that determines who is allowed access to the scheduler and to the ACL itself.

The ACL consists of a list of entries. Each entry assigns an *access type* to an *identity*. Four types of identity exist: **Everyone**, **Domain**, **Group**, and **User**. A domain is a Windows domain for ClearCase LT servers running Windows and an NIS domain for ClearCase LT servers running UNIX. Each group and user is qualified by a domain name. In a Windows domain, a group must be a global group, and a user must be a domain account.

NOTE: UNIX hosts that are not part of an NIS domain can use the string **<unknown>** in place of the domain name in an ACL entry.

Each identity has one of three access types. Table 5 shows the access types and their implications for access to the schedule and access to the ACL itself.

Table 5 Access Types in Scheduler ACL Entries

Access Type	Access to Schedule	Access to ACL
Read	Read only	Read only
Change	Read and write; can start jobs	Read only
Full	Read and write; can start jobs	Read and write

Each identity can have only one access type. However, access rights are inherited from **Everyone** to **Domain** to **Group** to **User** in such a way that each user has the least restrictive of all these access rights that apply to that user. For example, if a user's ACL entry specifies **Read** access but the ACL entry for the user's group specifies **Change** access, the user has **Change** access.

By default, everyone has **Read** access. On a local Windows host (the host where the scheduler is running), a member of the ClearCase administrators group always has **Full** access. On a local UNIX host, the **root** user always has **Full** access. On a remote host, access rights of a member of the ClearCase administrators group or the **root** user are determined by the ACL. Thus, to change the default ACL, you must be logged on to the ClearCase LT server, and you must be the privileged user.

To view or edit the scheduler's ACL, use the ClearCase Administration Console:

1. Navigate to the Scheduled Jobs node for the ClearCase LT server.
2. Click **Action** > **All Tasks** > **Edit Permissions**. This command opens a dialog box in which you can view or edit the scheduler's ACL.

Or use the following command to view the ACL:

cleartool schedule –get –acl

Use the following command to edit the ACL:

cleartool schedule –edit –acl

This command opens in a text editor a file containing a representation of the current ACL. You can edit the ACL using the ACL-definition syntax documented on the reference page for the **schedule** command.

If you have a text file containing ACL entries using the scheduler's ACL-definition syntax, you can use the following command to replace the entire ACL with the ACL entries in your file:

cleartool schedule –set –acl *defn_file_pname*

Administering Web Servers

Configuring a Web Server for the ClearCase Web Interface

17

This chapter explains how to configure a ClearCase Web server and to enable use of the ClearCase Web interface.

17.1 Configuration Planning

Configuring a Web server to run the ClearCase Web interface includes both ClearCase administration and Web administration tasks. In many cases, especially when setting up the interface for Internet access, the two types of tasks are not handled by the same person. As an aid to planning the configuration, this chapter includes lines for recording information that may need to be shared between the ClearCase administrator and the Web administrator. We recommend that you record information on the lines provided and share it as appropriate.

Web Administration Considerations

This section discusses decisions that a Web administrator must make before beginning configuration. It assumes that the Web administrator has experience setting up corporate Web servers and is familiar with such things as security issues.

- Running the ClearCase Web interface requires that ClearCase LT be installed on a system running a Web server. Supported Web servers are Apache, Microsoft Internet Information Server (IIS), and iPlanet. The server needs enough free disk space to load ClearCase LT plus

approximately 0.5 to 1.0 MB for each user who will use the Web interface through the server. Start by noting the name of the ClearCase LT server.

Server Name: _____

Server Type (UNIX or Windows) _____

The Server Name can be either a system name (for example, within an intranet) or a domain name (for an Internet server).

Approximate disk space required _____

- In a number of places, the installation requires a pathname beginning with the base URL for the interface. The base URL (referred to as *ccbase-url* in the instructions) can be any arbitrary string, for example, **ccaseweb** or **ccweb**. Select a base URL name.

  ClearCase Base URL: _____

  The full URL for the Web interface:

  http://*server-name*/*ccbase-url*/**bin/ccweb** (UNIX computers)
  http://*server-name*/*ccbase-url*/**bin/ccweb.exe** (Windows computers)

- If the Web server on the system to be configured is an iPlanet server, the configuration requires the name of the iPlanet administration server. The URL for the iPlanet administration server is identified during installation of the iPlanet server.

  iPlanet Administration Server _____

- If there are security concerns, you may want to provide a secure port through which to access ClearCase. The ClearCase Web interface can run through either a secure port using the Secure Socket Layer (SSL) or through a standard HTTP port. If the server is on a secure port, the URL begins with **https** rather than **http**.

ClearCase Considerations

This section discusses issues for consideration by the site's ClearCase administrator.

- In addition to the ClearCase base URL, configuration requires the location of the ClearCase installation directory. This is typically **C:\Program Files\Rational\ClearCase** on Windows computers and **/usr/atria** on UNIX computers. This location is denoted by *ccase-home-dir* in the configuration instructions.

ClearCase installation directory:_____

- The Web interface creates snapshot view directories for client users on the server.

  Although the snapshot view directories on the ClearCase LT server are typically empty (no files loaded), they are used for temporary storage of files that are checked out or that are to be created as elements. Therefore, they must be on a disk volume that has enough space for the number of users to be supported. Depending on how many files a typical user will have checked out at once, or how many new elements are to be created at once, allow for at least 0.5 MB to 1 MB of disk space for each user.

  By default, the Web interface creates those directories under what is known as the "host data" area for ClearCase. On UNIX systems, this is typically **/var/adm/atria**. On Windows systems, it is the **var** subdirectory under the ClearCase installation directory. A **ccweb** subdirectory is created in the host-data directory at ClearCase installation time, and all snapshot view directories are stored under that **ccweb** directory. (If ClearCase is installed in a pathname that contains spaces, such as **C:\Program Files**, the **ccweb** directory is created in the root of the drive on which ClearCase is installed.)

  However, if the default directory is not appropriate, you can select a different area by modifying the **ccweb.conf** file in *ccase-home-dir*/**config/ccweb**. Add the line

  `-view_storage` *pathname*

  to **ccweb.conf**, where *pathname* is the directory in which you want snapshot view directories used by the Web interface to be created. This pathname must be local to the Web server host.

  NOTE: Even though the Web interface may create a separate **ccweb** directory for view storage, it continues to keep administrative information in a **ccweb** subdirectory under the ClearCase installation directory (for example, **C:\Program Files\Rational\ClearCase\var\ccweb**).

- You may want to limit the size of files that can be uploaded to the Web server, to reduce the likelihood of denial of service attacks. If an extremely large file is uploaded and the Web server disk is filled, operation of the server computer may be disrupted, (depending on which disk it is).

  To limit the size of uploaded files, modify the following line in the **ccweb.conf** file:

  `-upload_limit` *size*

  where *size* is the approximate desired size limit in bytes. An attempt to upload a file that is too large results in an error message in the **Client Upload** output window.

- We recommend that you specify the primary group for all users who access ClearCase using this Web server by modifying the following line in the **ccweb.conf** file:

  `-primary_group` *group-name*

  where *group-name* is the name of the ClearCase users group. See *Setting the ClearCase Primary Group* on page 33 for more information on this topic.

- You can configure the session timeout interval, which controls how long a user login remains valid. The default value is 14400 seconds (four hours). You can change this default by modifying the line

  `-session_timeout` *seconds*

  in the **ccweb.conf** file, where *seconds* is an integer number of seconds between 600 (10 minutes) and 2147483647 (about 68 days). Values lower than 600 will be interpreted as 600.

- You can designate a directory where the ClearCase Web interface will store temporary files by adding a line of the form

  `-tmpdir` *directory-name*

  to the **ccweb.conf** file, where *directory-name* is a directory on the Web server host. If this line is not present in the **ccweb.conf** file, the ClearCase Web interface uses the value of the TMP or TEMP environment variables, if they exist. The ClearCase Web interface must be able to create and delete files in this directory regardless of the files' ownership or permissions.

- If users from multiple domains will access ClearCase using this Web server, you must enable domain mapping as described in *Using Proxy Groups and Domain Mapping in Windows NT Domains* on page 35. If you enable domain mapping, you must also specify the primary group of each user of the ClearCase Web interface by setting the **primary_group** in **ccweb.conf**.

  NOTE: When the Web server runs on a Windows computer, ClearCase Web interface users must be given permission to **Log On Locally** to the Web server host. Windows NT Server and Windows 2000 Server do not grant this permission by default. (Microsoft Internet Information Server can be configured in a way that does not require users to have this permission. See *Microsoft Internet Information Server (IIS)* on page 260 for more information on this topic.)

Browser Considerations

Because Windows 2000 and Windows XP require special privileges to download components through Internet Explorer, users accessing the ClearCase Web interface on one of these platforms must be members of the Power Users or local Administrators groups to download the Web interface applets. After the applets have been downloaded (the first time the Web interface is used), these privileges are no longer required, although new releases of ClearCase and ClearCase patches may change the applets, which will require them to be downloaded again.

17.2 Configuring the Web Server

Each of the supported Web servers has a different configuration mechanism.

Note that a Windows Web server must run as a service and must be configured to log on as the **System** account. This is the default for the Microsoft and iPlanet Web servers; the instructions below for configuring an Apache Web server include this setup. If you start a Windows Web server as a console application, rather than as a service, it is not likely to have the rights required to perform logons for client users.

Apache

Configure the Apache server by editing the **httpd.conf** text file in the **conf** subdirectory of the Apache installation area. Add the following two lines to the file, substituting appropriate values for *ccase-home-dir* and *ccbase-url*.

```
ScriptAlias /ccbase-url/bin/ "ccase-home-dir/web/bin/"
Alias /ccbase-url/ "ccase-home-dir/web/"
```

Note that:

- On Windows as well as UNIX, all pathname component separators must be slashes (/); Apache on Windows does not recognize backslashes (\).
- The **ScriptAlias** directive must precede the **Alias** directive.

For example, on a Windows system where *ccase-home-dir* is **C:\Program Files\Rational\ClearCase** and *ccbase-url* is **ccase**, add these lines to **httpd.conf**:

```
ScriptAlias /ccase/bin/ "C:/Program Files/Rational/ClearCase/web/bin/"
Alias /ccase/ "C:/Program Files/Rational/ClearCase/web/"
```

For a ClearCase LT server running UNIX where *ccase-home-dir* is **/usr/atria** and *ccbase-url* is **ccaseweb**, add these two lines to **httpd.conf**:

```
ScriptAlias /ccaseweb/bin/ "/usr/atria/web/bin/"
Alias /ccaseweb/ "/usr/atria/web/"
```

For a ClearCase LT server running Windows, configure the Apache Web server as a Windows service. To do so, execute the **Install Apache as Service** entry in the Apache folder which was added to the **Start** menu during installation. (Newer versions of the Apache Web server provide an install-time option to install the Web server as a service.)

Restart the Apache server if it is running. (Note that only Apache servers require restarting.)

Microsoft Internet Information Server (IIS)

Configure IIS by running the Internet Service Manager; you must be logged on using the same account that you use to perform standard Web administrative operations. There is both a Microsoft Management Console version and an HTML version of the interface. These instructions apply to the MMC version. The tasks are the same in the HTML version, but the navigation is somewhat different.

NOTE: Configuration options are slightly different for IIS4 (supported on Windows NT) and IIS5 (supported on Windows 2000 and Windows XP).

Configuration Steps for IIS4

To configure the ClearCase Web Interface in IIS4:

1. Start the Internet Service Manager.
2. From the list presented, select the Web site in which you want to place the ClearCase interface.
3. Click the **Action** button in the toolbar, and then click **New** > **Virtual Directory**. The New Virtual Directory Wizard starts.

4. In the New Virtual Directory Wizard, enter the value of *ccbase-url* (for example, **ccase**), and click **Next**.

5. Enter the pathname of the *ccase-home-dir***web** directory (for example, **C:\Program Files\Rational\ClearCase\web**). To find the directory, click **Browse**; then click **Next**.

6. Make sure that the **Allow Read Access** and **Allow Execute Access** check boxes are selected, and that **Allow Write Access** and **Allow Directory Browsing** are cleared. (It doesn't matter whether **Allow Script Access** is selected.) Click **Finish**.

7. The new virtual directory appears as a folder under the chosen Web site. Right-click the folder and select **Properties** from the shortcut menu. On the **Virtual Directory** page, under **Content Control**, make sure that the **Directory browsing allowed** and **Index this directory** check boxes are cleared.

8. Click the **Directory Security** tab, and then click **Edit**. In the **Authentication Methods** dialog box, clear **Allow Anonymous Access** and select **Basic Authentication**.

   Basic authentication requires client users to supply a user name and password to access the Web site. Windows **Challenge/Response** authentication allows Internet Explorer clients, if they are running on Windows and logged on to a domain, to access the Web site as the client user without having to specify a user name and password. Access to other network resources from the Web server is not allowed. ClearCase LT supports both forms of authentication.

   NOTE: ClearCase Web interface users do not need permission to **Log On Locally** to the Web server host if

9. Click OK to close the **Authentication Methods** dialog box. Then click **OK** to close the **Properties** sheet.

IIS4 is now configured for ClearCase Web access.

Configuration Steps for IIS5

To configure the ClearCase Web Interface in IIS5:

1. Start the Internet Service Manager.

2. From the list presented, select the Web site in which you want to place the ClearCase interface.

3. Click the **Action** button in the toolbar, and then click **New** > **Virtual Directory**. The New Virtual Directory Creation Wizard starts.

4. In the New Virtual Directory Creation Wizard, enter the value of *ccbase-url* (for example, **ccase**), and click **Next**.

5. Enter the pathname of the *ccase-home-dir***web** directory (for example, **C:\Program Files\Rational\ClearCase\web**). To find the directory, click **Browse**; then click **Next**.

6. Make sure that the **Read** and **Execute** check boxes are selected and the **Write** and **Browse** check boxes are cleared. (It doesn't matter whether **Run Scripts** is selected.) Click **Next** to get to the confirmation page, then click **Finish**.

7. The new virtual directory appears as a folder under the chosen Web site. Right-click the folder and select **Properties** from the shortcut menu. On the **Virtual Directory** page, make sure that the **Directory browsing** and **Index this resource** check boxes are cleared.

8. Click the **Directory Security** tab, and then click the **Edit** button in the **Anonymous access and authentication control** group. In the **Authentication Methods** dialog box, clear **Allow Anonymous Access** and select **Basic Authentication**.

   Basic authentication requires client users to supply a user name and password to access the Web site. **Integrated Windows authentication** allows Internet Explorer clients, if they are running on Windows and logged on to a domain, to access the Web site as the client user without having to specify a user name and password. Access to other network resources from the Web server is not allowed. ClearCase LT supports both forms of authentication.

   NOTE: ClearCase Web interface users do not need permission to **Log On Locally** to the Web server host if

9. Click **OK** to close the **Authentication Methods** dialog box. Then click **OK** to close the **Properties** sheet.

IIS5 is now configured for ClearCase Web access.

iPlanet Enterprise Server

Configure the iPlanet Web server by connecting to the iPlanet administration server. The procedure is the same for both Windows and UNIX. Connect to the URL for the administration server with a browser.

To configure the ClearCase Web Interface in iPlanet:

1. Select the server to configure from the drop-down list of servers, and then click the **Manage** button.
2. On the next page, click the **Class Manager** link in the upper right.
3. Click the **Programs** tab.
4. Make sure the **CGI Directory** option is selected.
5. In the **CGI Directory** form, enter the value of *ccbase-url*/**bin** as the URL prefix, and enter *ccase-home-dir*/**web/bin** as the CGI directory. For example, if *ccbase-url* is **ccase**, and *ccase-home-dir* is **/usr/atria**, enter **ccase/bin** as the URL prefix and enter **/usr/atria/web/bin** as the CGI directory.
6. Click **OK**, and then click **OK** in the confirmation dialog box.
7. In the row of tabs at the top of the page, click **Content Management**.
8. Click **Additional Document Directories** at the left of the page.
9. Enter the value of *ccbase-url* in the URL prefix box, and enter *ccase-home-dir*/**web** in the **Map To Directory** box.
10. Click **OK**, and then click **OK** in the confirmation dialog box.
11. Click the **Apply** link at the upper right of the page, and then click the **Apply Changes** button the next page.
12. Click the **OK** button in the confirmation dialog box.

The iPlanet server is now configured for ClearCase Web access.

18 Configuring Integrations with Microsoft Web Authoring Tools

This chapter explains how to configure an IIS Web server and enable use of the ClearCase integrations with Microsoft Web authoring tools.

18.1 Overview of the Integration

On Windows platforms, Rational ClearCase LT includes integrations with these Microsoft products:

- FrontPage 98
- FrontPage 2000
- Visual InterDev 6.0
- Office 2000 Web Folders functionality for Word, Excel, PowerPoint
- Internet Explorer 5.0 Web Folders

These integrations provide ClearCase configuration management capability for Web content and Web applications created using these tools. There are two installations, one on the server and one on the client.

The client is a Windows computer that runs a supported Web authoring application. Clients open Webs on the server using a URL and perform checkouts and checkins to a shared view that resides on the server. There can be multiple servers in your ClearCase region. Each server is

associated with a single Web content VOB using the default IIS alias. Multiple servers can be configured to share the same Web content VOB.

Server Setup Overview

A summary of the server installation is as follows. Detailed instructions for each step are in *Server Set-Up Details* on page 383.

1. Install IIS 4.0 (from Windows NT 4.0 Option Pack) or IIS 5.0 (from Windows 2000). IIS 5.0 is installed by default on Windows 2000 Server. If IIS is already installed on Windows NT 4.0, verify that the version of IIS is 4.0. If the IIS version is 2.0 or 3.0, upgrade to 4.0 by installing Windows NT 4.0 Option Pack. To determine the IIS version number, check the value of registry key **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC \Parameters\MajorVersion**.

2. Install FrontPage Server Extensions (FPSE) version 3.0.2 (from Windows NT 4.0 Option Pack) or version 4.0.2 (from Office 2000 CD3, Office Server Extensions, or Windows 2000). The FPSE are installed by default when Option Pack on Windows NT 4.0 Server and Windows 2000 Server are installed. If FPSE were installed along with IIS, you do not need to reinstall them. If version 3 was installed by a previous Option Pack install, you can upgrade to Office Server Extensions.

3. Install ClearCase LT (client or server, server recommended). This sets the value of the Windows registry key **HKEY_LOCAL_MACHINE\Software\Atria\ClearCase \CurrentVersion\ProductHome** to the directory in which you install the product.

4. Run the Web Authoring Integration Configuration Wizard and follow the instructions. To start the wizard:

   - For ClearCase LT Client, click **Start** > **Programs** > **Rational ClearCase LT Client** > **Integrations** > **Web Authoring Integration Configuration**.

   - For ClearCase LT Server, click **Start** > **Programs** > **Rational ClearCase LT Server** > **Integrations** > **Web Authoring Integration Configuration**.

   Online help is available. Specify server mode always. Specify local mode only if you have or plan to install FrontPage 2000 on the Web server.

Client Setup Overview

A summary of the client installation follows. Refer to the detailed instructions for each step in *Client Setup Procedure*.

1. Install one or more supported authoring tools:
   - FrontPage 98
   - FrontPage 2000
   - Visual InterDev 6.0
   - Internet Explorer 5.0 (for Web Folder functionality in Windows Explorer)
   - Office 2000 (for Save to Web Folder functionality in Word, Excel, PowerPoint)
2. Create and add Webs to source control using FrontPage 98, FrontPage 2000, or Visual InterDev 6.0.
3. Verify that the new Web content is added to source control.
4. Enable Author/Administrator privilege for users who need to deliver content using FrontPage 98, FrontPage 2000, or Visual InterDev 6.0.
5. Optionally, enable local mode support for FrontPage 2000 clients.

18.2 Server Setup Procedure

The supported server platforms and required software versions for the Web server are listed in Table 6.

Table 6 Supported Platforms for Web Servers

Operating System Platform	Web Server Software	FrontPage Server Extensions / Office Server Extensions
Windows NT 4.0	IIS 4.0	3.0.2.1105 (ships on FrontPage 98 CD)
Windows NT 4.0	IIS 4.0	3.0.2.1706 (ships on Windows NT 4.0 Option Pack CD)
Windows NT 4.0 or Windows 2000	IIS 4.0 or IIS 5.0	4.0.2.2717 (ships with Office 2000)
Windows 2000	IIS 5.0	4.0.2.3xxx (ships with Windows 2000)

Step 1: Install IIS

We recommend that IIS be the only Web server on the host and that the default port (80) be used. Check for other Web server software before proceeding to install IIS.

Note that if IIS version 2.0 or version 3.0 is installed, you must uninstall it before you install IIS version 4.0 from the Windows NT 4.0 Option Pack.

If you are using a Windows 2000 server and have taken all the defaults for the Windows 2000 server installation, IIS 5.0 and FrontPage 2000 Server Extensions are installed on your server. If you have not performed a default Windows 2000 server installation, use the Windows 2000 Control Panel to add the required components.

If you are using Windows NT 4.0 server, you need to install IIS 4.0 and either FrontPage Server Extensions or Office Server Extensions.

When installing IIS 4.0 from the Windows NT 4.0 Option Pack on Windows NT 4.0 or IIS 5.0 from the Windows 2000 Control Panel, the following components are required for the FrontPage/InterDev integrations. All other components are optional.

- Internet Information Server (when you select this component, you will also have to select a number of other components, like Windows NT Option Pack common files, on which this component depends)

- Internet Service Manager
- FrontPage Server Extensions (if you plan to install OSE later, installing FPSE at this time is optional)
- Windows Scripting Host

When you install IIS, a screen similar to Figure 26 appears. The default Web alias directory determines the VOB-tag of the Web content VOB that will be created in Step 4. (See *Step 4: Run the Web Authoring Integration Configuration Wizard* on page 270.) You need to choose the VOB-tag you want this Web server to access. By default, this is **\wwwroot** as shown in Figure 36. If this VOB-tag is already registered in this machine's ClearCase registry or if you want another name (for example, **wwwroot_<host>**, or a descriptive name, **\sales_Webs**) you must change the default now. The remainder of this pathname becomes a snapshot view root when the Web Authoring Integration Configuration Wizard is run, as shown in Figure 26.

Figure 26 Setting Up the Root Web in the IIS Installation

After IIS is installed, you may need to run Internet Service Manager to add IIS operators and enable Basic Authentication. If you use a local VOB and view on the Web server that you are setting up, you do not need to enable Basic Authentication.

If Office 2000 was installed on the Windows NT 4.0 server before IIS 4.0 was installed, the FPSE will not be installed on the Web server even though you chose to install FPSE from the Windows NT 4.0 Option Pack. In this case, we recommend that you install OSE after the Windows NT 4.0 Option Pack to upgrade to version 4.0 server extensions.

Step 2: Install FPSE or OSE

If FPSE were installed in *Step 1: Install IIS*, it is not necessary to reinstall. If version 3 FPSE were installed by a previous Option Pack install, you can upgrade to OSE, but it is not necessary.

If FPSE were not installed in *Step 1: Install IIS*, install FPSE version 3.0 (from NT Option Pack) or version 4.0 (from Office 2000 CD3, Office Server Extensions (OSE), or Windows 2000). The FPSE are installed when you install Windows NT 4.0 Option Pack on Windows NT 4.0 Server and Windows 2000 Server.

Step 3: Install ClearCase LT

Install ClearCase LT (server or client) software on the computer. It includes a Web Authoring Integration Configuration Wizard that is used in the Step 4.

Step 4: Run the Web Authoring Integration Configuration Wizard

Run the Web Authoring Integration Configuration Wizard and follow the instructions. To start the wizard:

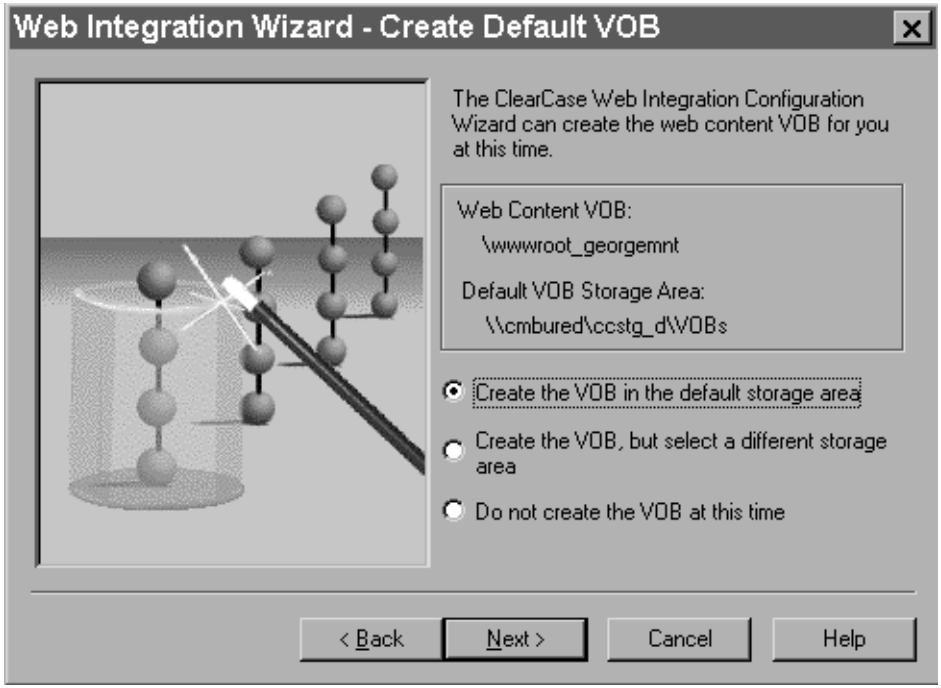
- For ClearCase LT client, click **Start** > **Programs** > **Rational ClearCase LT Client** > **Integrations** > **Web Authoring Integration Configuration**.
- For ClearCase LT server, click **Start** > **Programs** > **Rational ClearCase LT Server** > **Integrations** > **Web Authoring Integration Configuration**.

Online help is available for each wizard screen. When running the wizard, always specify **Server Mode Configuration.** Specify **Local Mode Configuration only** if you plan to install FrontPage 2000 client on the Web server.

If you have previously installed SourceSafe on your Web server, the wizard displays a **Replace SourceSafe** screen. To use the integration, you must click **Yes**. Doing so does not disable any part of SourceSafe except for the SourceSafe COM API.

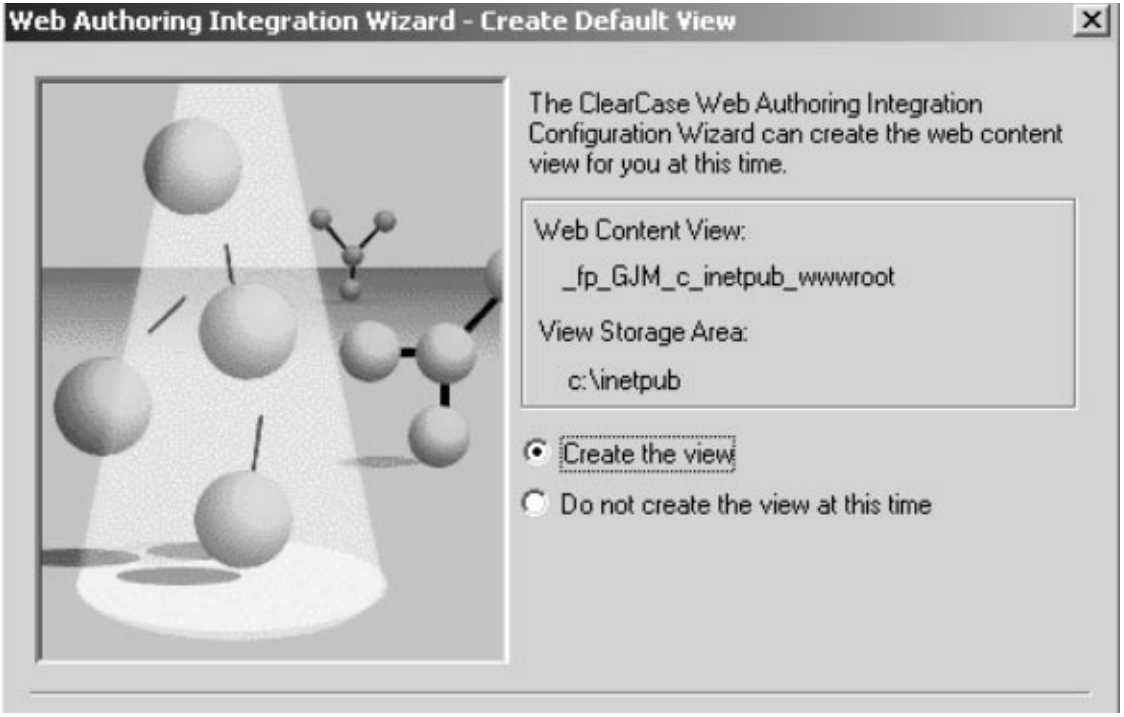
After the Web Authoring Integration Configuration Wizard checks the versions of IIS and server extensions on the Web server, you are prompted to create a content area VOB. You can accept the default VOB storage location, as in Figure 27, or use the wizard to select another VOB storage location.

Figure 27 Setting Up the VOB Storage for the Integration



The wizard then prompts you to create a content area view, as in Figure 28.

Figure 28 Setting Up the View Storage for the Integration



We recommend that you put the VOB on the Web server, which improves performance and avoids the use of basic authentication.

If you decide to use an existing VOB (on UNIX or Windows NT) as the Web content VOB, it must be created and registered with the expected VOB-tag as described in *Step 1: Install IIS*. If this VOB is created before the wizard is run, the wizard shows the existing VOB-tag and skips the VOB creation step.

Any VOB accessed through the integration has the same trigger restrictions that ClearCase Web client access imposes, that is, interactive triggers fail.

The integration can only use views that are created either by the wizard or in FrontPage 2000 (with **ClearCase** > **Create Snapshot View**). The integration uses views that have a different hijacked-file detection mechanism than do standard views. This is necessary because FrontPage and InterDev rewrite files frequently. Attempts to use views created in any other way will fail. The integration does not support UCM-enabled views.

The Web Authoring Integration Configuration Wizard maintains a log file at *ccase-home-dir***var****log****webintegration_log**. You can review this log in the event of unexpected behavior or errors.

18.3 Client Setup Procedure

This section explains how to set up a ClearCase Web client on Windows.

Step 1: Install the Client Application

Install one or more of the supported client applications on your client computers:

- FrontPage 98
- FrontPage 2000
- Visual InterDev 6.0
- Internet Explorer 5.0
- Office 2000

Step 2: Add Web to Source Control

Using FrontPage 98, FrontPage 2000, or Visual InterDev 6.0, you can add a new Web to source control on the IIS server.

When you add a new Web to source control, the integration expects the Web to be a top-level Web, for example, **http://***<cc Web server>***/***<Web to be placed under source control>*.

From FrontPage 98

1. Create a new Web. Click **File** > **New** > **FrontPage Web** and complete the New FrontPage Web Wizard.

2. Add the new Web to source control by clicking **Tools** > **Web Settings**. In the **FrontPage Web Settings** dialog box, click the **Configuration** tab and complete the **Source Control Project** box, using the name of your new Web in Step #1.

   The Web name in the **Source Control Project** box must start with **$/**. For example, if you created a Web at **http://ccWebs/sales_collateral**, the **Source Control Project** name must be **$/sales_collateral**. The Web name may be different from the actual Web directory. You must use the Web directory as the **Source Control Project** name.

From FrontPage 2000

1. Install ClearCase LT on your client computer.
2. Run the Web Authoring Integration Configuration Wizard.

   - For ClearCase LT Client, click **Start** > **Rational ClearCase LT Client** > **Integrations** > **Web Authoring Integration Configuration**.
   - For ClearCase LT Server, click **Start** > **Rational ClearCase LT Server** > **Integrations** > **Web Authoring Integration Configuration**.

   From the Web Authoring Integration Configuration Wizard, select **Local Mode Configuration**.
3. In FrontPage 2000, create a new Web. Click **File** > **New** > **Web** and complete the **New** dialog box.
4. Click **ClearCase** > **Add Web to source control** to add the new Web to source control.

From Visual InterDev 6.0

To use Visual InterDev 6.0 with the ClearCase integration, you do not need to install ClearCase LT on the client computer. If ClearCase is installed on the client, the online help for the Visual InterDev integration is enabled.

1. Click **File** > **New Project** to create a new Web project.
2. Select the project from the Visual InterDev Project Explorer and click **Project** > **Source Control** > **Add to Source Control** to open the **Initial Solution Add** dialog box.
3. To open the **Enable Source Control** dialog box, click **Selection**.

4. In the **Source control project name** box, delete the **_Web** suffix from the default text. For example, if the Web project is named **stock_trading** the default text is **$/stock_trading_Web** and the correct project name is **$/stock_trading**.

5. Click **OK**. The project is now added to source control on the IIS server.

Step 3: Verify That New Web Content Is Added to Source Control

After you add a new Web to source control in FrontPage 98, FrontPage 2000, or Visual InterDev 6.0, look for the special source control icons that indicate successful completion. In FrontPage, these icons are green dots (Figure 29); in Visual InterDev, they are locks (Figure 30).

Figure 29 FrontPage Source Control Icons

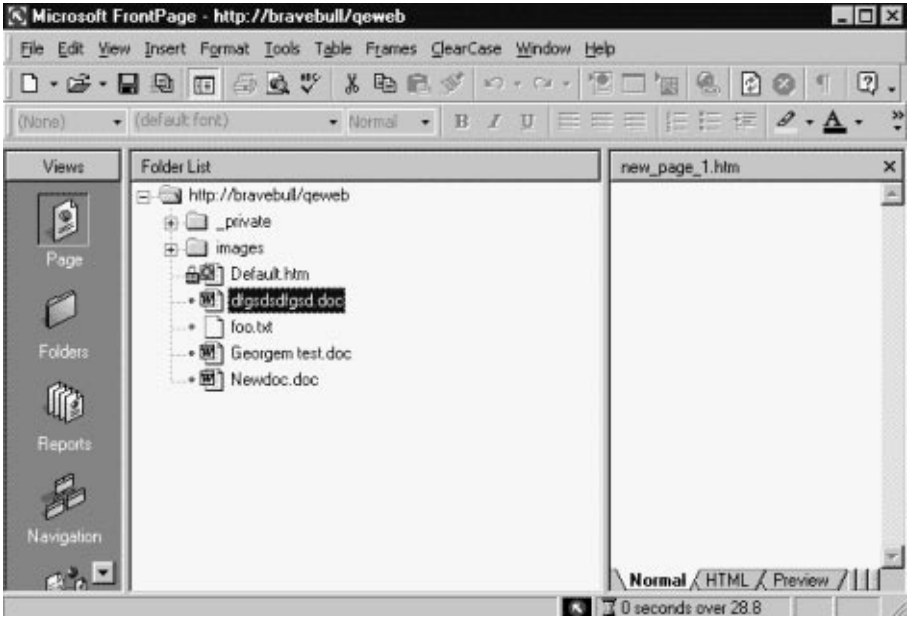
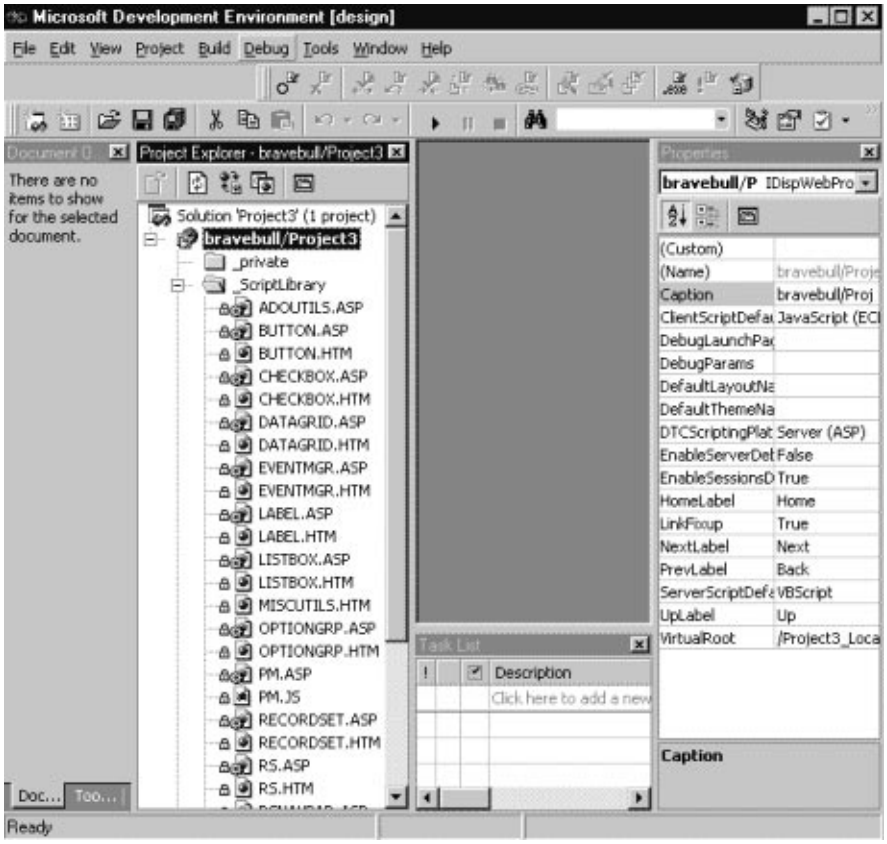


Figure 30 Visual InterDev Source Control Icons



If no source control icons appear in FrontPage or Visual InterDev, or if there are errors or other unexpected results during these procedures, the integration DLL maintains a log file at *ccase-home-dir***var\log\ssapi_log** that you can inspect to diagnose the error.

If you have existing content in flat files or on another Web server, you can also use the import features of FrontPage and Visual InterDev to pull additional content into your new Web and place it under source control.

Step 4: Setting User Permissions

FPSE maintain access control for Web authoring and administrative actions. To edit Web content, a user or group must have Author permission for the Web. Add all users and/or groups that need access to your Web content.

FrontPage 98

1. Click **File** > **Open FrontPage Web** to access the new Web.
2. Click **Tools** > **Permissions**. In the **Permissions** dialog box, click the **Users** tab or **Groups** tab to add user or group permissions.

FrontPage 2000

1. Click **File** > **Open Web** to access the new Web.
2. Click **Tools** > **Security** > **Permissions**. In the **Permissions** dialog box, click the **Users** tab or **Groups** tab to add user or group permissions.

Visual InterDev 6.0

1. Click **File** > **Open Project** to access the new Web.
2. Click **Project** > **Web Project** > **Web Permissions**. In the **Permissions** dialog box, click the **Users** tab or **Groups** tab to add user or group permissions.

In addition, ensure that you add any appropriate domain groups to the content VOB's group list. Values of **CLEARCASE_PRIMARY_GROUP** on client machines are not translated on the server end, so you need to add all groups at this time.

Every time an integration user opens a source control enabled Web, that user takes a ClearCase LT license.

Step 5: Local Mode Client Setup for FrontPage 2000

The integration supports two modes: server mode and local mode. Server mode operation is available on all the supported authoring tools. In server mode, the authoring tool runs locally, and all ClearCase operations run remotely on the ClearCase Web server. Webs are opened by accessing a URL. You do not need to install ClearCase on the client system, but doing so provides

the user with online help, as well as a limited ClearCase menu in FrontPage 2000 and a ClearCase toolbar in Visual InterDev.

Server mode operation provides access to a limited set of ClearCase features. When running in server mode, the interface to ClearCase resembles the interface between FrontPage or Visual InterDev and Microsoft Visual SourceSafe. All users share a single snapshot view on the Web server.

Local mode operation is supported only for FrontPage 2000; ClearCase must be installed on the same computer that is running FrontPage 2000. Local mode operation provides extra functionality, including a ClearCase menu that is available on the FrontPage 2000 GUI. Webs are opened by accessing a pathname (a disk-based Web) for a Web located in the user's own snapshot view. Multiple views and full access to ClearCase GUIs are available using local mode. Checkouts performed in local mode are unreserved by default.

To configure local mode for FrontPage 2000:

1. Install FrontPage 2000.
2. Install ClearCase LT.
3. Run Web Authoring Integration Configuration Wizard.

   - For ClearCase LT Client, click **Start** > **Rational ClearCase LT Client** > **Integrations** > **Web Authoring Integration Configuration**.
   - For ClearCase LT Server, click **Start** > **Rational ClearCase LT Server** > **Integrations** > **Web Authoring Integration Configuration**.

   From the Web Authoring Integration Configuration Wizard, select **Local Mode Configuration**.
4. In FrontPage 2000, click **ClearCase** > **Create Snapshot View** and complete the View Creation Wizard.

If you access Webs with FrontPage borders, you must reapply them in a local-mode view. We recommend not using FrontPage themes unless only server-mode access is used. Theme binding information is not stored under source control; reapplication of themes is time consuming, and many extra versions of essentially identical files are checked in.

Use of local mode requires any server-mode users to update the shared view periodically. If there are no local-mode users (or other write activity to the Web content VOB), updating the shared view is not necessary. You can mix and match local and server mode access in your installation.

18.4 Web Folders Support in Office 2000 and Microsoft Internet Explorer 5

The server mode integration option also enables you to configure your IIS Web server for the ClearCase integration with Office 2000 and Internet Explorer 5.0 Web Folders.

If you are using any of the Office 2000 applications or Internet Explorer 5.0, you can log on to an IIS Web server using the Web Folders feature and access the shared view of Web content as a Web address.

The ClearCase integration for Web Folders supports two cases:

- If the file is under source control, saving the file copies it to a Web Folder. ClearCase then checks in the file.
- If the file is not under source control, saving the file copies it to a Web Folder. ClearCase then adds the file to source control and checks it in.

18.5 Updating the Shared View on the Web Server

From time to time, users may want to update the shared Web content view. There is a script on the Web server that performs this function (*ccase-home-dir***etc****iisfix.bat**). Users can access this function from their client systems by opening the URL **http://**<*Web server name*>**/ccase_tools** and clicking **Update Shared View**. You may want to run this script on the IIS server periodically by using the Windows NT Task Scheduling service and specifying an appropriate user as the account to run the task. Using the Windows NT **at** command or the ClearCase Scheduler for this purpose is not supported.

18.6 Considerations for Migrating and Converting Data to the Integration

You can also use the integration with data from an existing Web. You may need to do so to migrate data from an existing VOB, or to convert data under source control from another source control vendor.

- Migration

The integration expects Webs under source control to be rooted in a subdirectory of the Web content VOB root. If you are relocating Webs from existing VOBs using **cleartool relocate** into the Web content VOB, relocate them to a VOB root subdirectory.

- Conversion

  The Web at **\source\qe_group\Webs\qe_Web** is currently in Visual SourceSafe. To convert it to ClearCase, run **clearexport_ssafe** on the VSS library where these files reside. Then use **clearimport**, specifying **\wwwroot\qe_Web** as the target VOB directory (assuming that the default Web content VOB-tag is **\wwwroot**).

In either scenario, it is necessary to run a script (*ccase-home-dir***\etc\iisfix.bat**) to install the FPSE and register the IIS alias and Source Control operations necessary to make such Webs visible and under source control in FrontPage and InterDev. After this has been done, the source control icons appear when the Web is opened in FrontPage and Visual InterDev.

NOTE: Use of VOB symbolic links in integration Webs is not supported.

18.7 Accessing Help Information for the Integration

When ClearCase LT is installed on client systems, online help for the integration is available. A ClearCase toolbar with a Help icon is available in Visual InterDev and a ClearCase menu, including a **Help** command, is available in FrontPage 2000. However, this online help is not available to users who have not installed ClearCase LT on their local systems. That is, if the elements are under ClearCase control, but ClearCase LT is not running on the local system, the only help available is the online help provided by the authoring tools. This online help on source control topics is for Visual SourceSafe.

Tuning for Performance

19 Improving ClearCase LT Server Performance

This chapter presents techniques for improving the performance of Rational ClearCase LT on the ClearCase LT server.

Your organization's VOBs constitute a central data repository. Good ClearCase LT server performance ensures that this centralized resource does not become a bottleneck.

The work of a ClearCase LT server involves both read and write access to the VOB database as well as periods of significant computation. VOB access patterns can greatly influence the number of concurrent users that a ClearCase LT server can support at a given level of performance. For example, many more users can read from a VOB directory at a level of good performance than can write to the same directory. For information about the characteristics of a good a ClearCase LT server, see Chapter 7, *ClearCase LT Server Configuration Guidelines*.

19.1 Minimize Process Overhead

The most effective measures for ensuring good performance from the ClearCase LT server are also the easiest to implement:

- **Keep non-ClearCase processes off the ClearCase LT server.** Do not use the ClearCase LT server to run another enterprise application (for example, a DBMS) or for a critical network resources such as an NIS server on UNIX or a Primary Domain Controller on Windows.
- **Keep ClearCase client processes off the ClearCase LT server.** Do not use the ClearCase LT server to perform ordinary activities (comparing and merging files, finding versions, and so on) that belong on developer's desktop system.

19.2 Maximize Disk Performance

Follow these recommendations to obtain the best I/O performance on the ClearCase LT server:

- Use disks with fast seek times and high rotational speeds.
- Where practical, dedicate a disk spindle for each VOB.
- For very busy VOBs, dedicate two disk spindles per VOB if the VOB server host provides such a feature: one for the database and source directories, and one for the cleartext and DO pools.
- Use stripe technology (RAID 0) for improved performance.
- Use mirroring (RAID 1) if high availability is desired.
- Avoid RAID 5, unless your benchmarks show equal performance to RAID (0+1).

Consult your system documentation for details about how to use disk striping and mirroring

19.3 Add Memory for Disk Caching on Windows

Windows systems have a dynamic disk buffer cache. As much main memory as possible is used to cache blocks of data files that have been updated by user processes. Periodically, the disk buffer cache is flushed to disk. The cache size increases when you add more memory to the host.

This feature speeds up disk I/O significantly; making full use of it is an important factor in good ClearCase LT server performance. An inadequate disk buffer cache causes thrashing of VOB database files. The result is a significant performance degradation. These are the symptoms:

- Extended periods required for **scrubber** and **vob_scrubber** execution
- ClearCase LT clients experience RPC time-out errors

There is additional information about the relationship of main memory to VOB size in *ClearCase LT Server Configuration Guidelines* on page 71.

19.4 Tune Block Buffer Caches on UNIX

Most of the UNIX operating systems that ClearCase LT supports have a dynamic block buffer cache. As much main memory as possible is used to cache blocks of data files that have been updated by user processes. Periodically, the block buffer cache is flushed to disk.

We recommend that the size of a ClearCase LT server's block buffer cache average roughly 50% of the total size of all VOB database directories (*vob-stg-dir***/db**) on the server. On most UNIX operating systems, the cache size increases when you add more main memory to the host.

If there is a substantial amount of non-ClearCase activity or ClearCase client activity on the ClearCase LT server, it will need even more main memory to assure good VOB database performance.

Block Buffer Cache Statistics

The standard UNIX SystemV **sar(1M)** utility reports block buffer cache activity. For example, this command reports activity over a 5-minute period, with a cumulative sample taken every 60 seconds:

sar –b 60 5

```
12:14:22 bread/s lread/s  %rcache bwrit/s lwrit/s  %wcache pread/s pwrit/s
12:15:22       0       1      100       1       1         0       0       0
12:16:23       1       1      -60       2       2         0       0       0
12:17:24       0       4      100       4      17        77       0       0
12:18:25       0       6      100       3     145        98       0       0
12:19:25      17      91       81      28     335        92       0       0

12:19:25 bread/s lread/s  %rcache bwrit/s lwrit/s  %wcache pread/s pwrit/s
Average        4      21       83       8     100        92       0       0
```

If your block buffer caches are sized correctly, cache reads are in the 90% to 95% range and cache writes are 75% or above.

Flushing the Block Buffer Cache

Interactive performance suffers considerably when the block buffer cache is flushed to disk. Most UNIX systems provide no user-level control over the frequency of flushing; HP-UX does, through the **syncer(1M)** utility. The larger the block buffer cache, the less frequently it should be flushed.

19.5 Modify Lock Manager Startup Options

Every ClearCase LT server must run a single lock manager process, which controls database access for every VOB on the system. The lock manager runs with a default set of startup options that are intended to deliver good performance under a wide range of loads. However, you may want to experiment with changing certain lock manager startup options on VOB server hosts that support many VOBs or many users.

Lock Manager Implementations

There are two implementations of the lock manager: one reads and writes data using a socket, the other, which is available on certain UNIX platforms, uses shared memory, which results in lower latency for each request. In many cases, the shared memory implementation can provide better performance on hosts that must support many VOBs and/or many concurrent users. You can tell which type of lock manager is on a system by examining its startup message in **lockmgr_log** file. The conventional lock manager prints the following lines:

```
db_VISTA Version 3.20
Database Lock Manager for UNIX BSD
```

The shared memory lock manager prints the following lines:

```
db_VISTA Version 3.20
Database Lock Manager for UNIX System V shared memory
```

Lock Manager Startup Options

On UNIX platforms, lock manager startup options are defined in the startup script *ccase-home-dir***/etc/atria_start.** To change these values, open the startup script in a text editor and edit the command that starts **lockmgr**:

```
${ATRIA}/etc/lockmgr ...
```

On Windows platforms, lock manager startup options are defined internally. To change these defaults, create the following Windows registry key as a REG_SZ value:

HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion

LockMgrCmdLine: REG_SZ : –a almd –f *num* –u *num* –q *num*

NOTE: If you change any lock manager startup option on UNIX or Windows, you must stop and re-start ClearCase on the host before the change will take effect.

SPECIFYING THE LOCKMGR SOCKET. *Default:* **almd**

–a almd

Specifies the name of the socket or shared memory file created by the **lockmgr**. Do not change this name.

SPECIFYING THE NUMBER OF FILES. *Default on Windows:* 128. *Default on* UNIX: 256.

–f *num*

Specifies the number of database files that can be open concurrently. Each VOB database consists of seven database files. The default startup values allow the lock manager to handle a maximum of 36 VOBs on a UNIX host and 18 on a Windows host. If there are more VOBs on the host, the lock manager will not be able to service requests for all of the VOBs concurrently. User response times will degrade, and the **db_server** or **vobrpc_server** log files will include messages of the form:

```
Error: Too many open databases on host (try increasing the -f argument
on lockmgr command line).
```

When this happens, you should either move some of the VOBs to another host, or increase the value of the **-f** option to 7**V* where *V* represents the number of VOBs on the host.

SPECIFYING THE NUMBER OF USERS. *Default on Windows:* 128. *Default on* UNIX: 256

–u *num*

Specifies the number of **db_server** or **vobrpc_server** processes that can concurrently request locks. Each active view requires one **vobrpc_server** process for each VOB that the view accesses. In addition, various operations that change VOB metadata cause a **db_server** process to access the VOB through the lock manager. Poor user response time and messages of the form

```
db_VISTA database -922: lockmgr is busy
```

in the **lockmgr_log** file may indicate that the value of the **-u** option should be raised.

On hosts that do not support the shared-memory lock manager, the **-u** option cannot be set higher than 1018. On hosts that support the shared-memory lock manager, the **-u** option is bounded only by available virtual memory.

You can compute an approximate worst case value for **-u** using the formula:

$V*(N/4 + 5)$

where *V* is the number of VOBs on the host, and *N* is the number of users who access those VOBs. For a more realistic value—one that will not cause the lock manager to consume unnecessary virtual memory on the VOB server host—monitor the total number of **vob_server** and **vobrpc_server** processes running on the VOB server host for an extended period of typical use (perhaps a week or two), then multiply the peak value by a factor that will accommodate growth (two, or perhaps a little more).

SPECIFYING THE SIZE OF THE REQUEST QUEUE. *Default on Windows:* 128. *Default on* UNIX: 1024.

−q *num*

Specifies the number of lock requests for locks to be queued. The lock manager delays queuing lock requests in excess of this value. Poor user response time and messages of the form

```
db_VISTA database -922: lockmgr is busy
```

in the **db_server_log** or **vobrpc_server_log** files (and, often, concurrently in a **view_log** file) may indicate that the value of the **-q** option should be raised. As a rule, this value should be no greater that five times the value of the **-u** option.

Troubleshooting

Repairing VOB and View Storage Directory ACLs on Windows

20

We recommend that you use NTFS-formatted disks for holding VOB and view storage directories on the ClearCase LT server. NTFS file-system objects are protected by Security Descriptors, which contain ownership information and discretionary access control list (DACL). (DACL and ACL are often used to mean the same thing.) FAT file-system objects are not protected in this way. Although the **readonly** attribute is available in both NTFS and FAT, it is not enforced; that is, it can be removed easily.

On NTFS, the ClearCase storage directory's ownership (its owner and primary group ID) is determined from the storage directory root's Security Descriptor. Because FAT file systems do not support the Security Descriptor, ClearCase creates the **identity.sd** file to store it. A copy of the storage directory root's Security Descriptor is always stored in the **identity.sd** file, regardless of the file system type, and the **groups.sd** file holds the supplementary VOB group list.

20.1 ClearCase ACLs

Rational ClearCase LT establishes ACLs for VOB and view storage directories when VOBs and views are created. These ACLs have a particular form that ClearCase LT relies on. The following example shows the correct ACL for a VOB storage directory, **myvob.vbs**, created by user **ccase_adm**, who has primary group **user**, in domain **nt_west**:

cacls c:\vobstore\myvob.vbs

```
NT_WEST\user:(CI)R                                  (VOB's principal group)
Everyone:(CI)R
NT_WEST\ccase_adm:(CI)(special access:)             (VOB owner)
        STANDARD_RIGHTS_ALL
        DELETE
        READ_CONTROL
        WRITE_DAC
        WRITE_OWNER
        SYNCHRONIZE
        STANDARD_RIGHTS_REQUIRED
        FILE_GENERIC_READ
        FILE_GENERIC_WRITE
        FILE_GENERIC_EXECUTE
        FILE_READ_DATA
        FILE_WRITE_DATA
        FILE_APPEND_DATA
        FILE_READ_EA
        FILE_WRITE_EA
        FILE_EXECUTE
        FILE_READ_ATTRIBUTES
        FILE_WRITE_ATTRIBUTES

NT AUTHORITY\SYSTEM:(CI)F                           (built-in server identity)
NT_WEST\user:(OI)(IO)(special access:)              (VOB's principal group)
        GENERIC_READ
        GENERIC_EXECUTE

Everyone:(OI)(IO)(special access:)
        GENERIC_READ
        GENERIC_EXECUTE

NT_WEST\ccase_adm:(OI)(IO)(special access:)         (VOB owner)
        DELETE
        WRITE_DAC
        WRITE_OWNER
        GENERIC_READ
        GENERIC_WRITE
        GENERIC_EXECUTE
```

```
NT AUTHORITY\SYSTEM:(OI)(IO)F
BUILTIN\Administrators:(OI)(special access:)
        DELETE
        READ_CONTROL
        WRITE_DAC
        SYNCHRONIZE
        FILE_GENERIC_READ
        FILE_GENERIC_WRITE
        FILE_GENERIC_EXECUTE
        FILE_READ_DATA
        FILE_WRITE_DATA
        FILE_APPEND_DATA
        FILE_READ_EA
        FILE_WRITE_EA
        FILE_EXECUTE
        FILE_READ_ATTRIBUTES
        FILE_WRITE_ATTRIBUTES
```

20.2 Causes of Protection Problems

This section describes typical scenarios that can lead to inappropriate storage protections and suggests how to avoid these situations.

Copying the Storage Directory

If you use **xcopy** or Windows Explorer to copy or restore the storage directory, Windows does not retain the Security Descriptor and its protection is incorrect. We recommend that you use one or more of the following procedures to copy the storage directory:

- Use **scopy** (Windows NT Resource Kit command)

  scopy *source destination* **/o /s**

  You must log on as a member of the Administrators or Backup Operators group. If you are copying across the network, your domain account must belong to one of these groups on both host machines.

- Use an offline backup tool

Use commercially available offline backup/restore tools for Windows, such as tape backup, which retain the Security Descriptors. You must log on as a member of the Administrators or Backup Operators group.

➤ Use **ccopy** (*ccase-home-dir***etc****utils****ccopy**)

ccopy *source destination*

You must log in as VOB owner.

NOTE: **ccopy** does not always preserve ownership when copying files. If this is a concern, use **scopy**.

For information about fixing protection problems caused by copying the directory, see *Fixing Protection Problems* on page 297.

Converting the File System from FAT to NTFS

If you create the storage directory in FAT and later convert that partition to NTFS, its protection will be incorrect. ClearCase reports a different VOB owner after the conversion. VOB and view servers do not start because the **identity.sd** file does not agree with the storage directory root's Security Descriptor. There is no way to avoid this behavior.

For information about fixing protection problems caused by converting from FAT to NTFS, see *Fixing Protection Problems* on page 297.

Editing Permissions

If you edit a file permission, for example, by using Windows Explorer or **cacls**, ClearCase users will begin experiencing access and protection problems.

WARNING: Never click **OK** in the **File Permissions** dialog box if the changes will affect VOB and/or view storage. Doing so changes DACL (the order of access control entries) even if you have not changed anything. To detect protection problems, run this command:

cleartool checkvob –protections –pool *vob-stg-pname*

For information about fixing protection problems caused by editing permissions, see *Fixing Protection Problems* on page 297. For more information about the options to the **checkvob** command, see the **checkvob** reference page.

20.3 Utilities for Fixing Protection Problems

ClearCase includes three utility programs for fixing protection problems on Windows:

- *ccase-home-dir***etc\utils\vob_sidwalk.exe** repairs permissions on VOB storage directories. It can also be used to change ownership on VOB objects. For details, see *Using vob_sidwalk to Change or Update VOB Users and Groups* on page 47.
- *ccase-home-dir***etc\utils\fix_prot.exe** repairs permissions on VOB or view storage directories
- *ccase-home-dir***etc\utils\lsacl.exe** displays NTFS DACLs for file-system objects

fix_prot

fix_prot [**-f**·**orce**] { **-root** [**-r**·**ecurse**] [**-recover** {**-chown** *user* | **-chgrp** *group* } | **-replace**·**_server_process_group** |
[**-r**·**ecurse**] [**-type** { **d** | **f** }] [**-chown** *user*] [**-chgrp** *group*] [**-chmod** *permissions*] } *pname* ...

Options

–f·**orce**
Do not prompt for confirmation.

–r·**ecurse**
Recursively fix protections.

–root
Specifies that *pname* is a VOB or view storage directory root.

–type
Specifies the type of file-system objects to fix. Use **d** for directories and **f** for files. If **-type** is not specified, **fix_prot** operates on both directories and files.

–chown
Specifies the new owner. Mandatory with **–root**.

–chgrp
Specifies the new primary group. Mandatory with **–root**.

–chmod
Specifies the new access rights: `owner`, `group`, `other` (world). Both symbolic and absolute codes are valid, such as `go-x` (symbolic) or `0644` (absolute).

-recover
(Windows only.) Restores correct file system ACLs in a view storage directory based on the information in the view's **identity.sd** and **groups.sd** files. Not valid on VOB storage (use **vob_sidwalk -recover_filesystem** instead).

-replace_**_server_process_group**
(Windows only.) Replaces ACL entries for the ClearCase administrators group. Use this option if you have changed the ClearCase administrators group name or have moved a view to a new domain that has a different SID for this group.

Examples

To create a UNIX **.identity** directory after moving a VOB from Windows to UNIX. The VOB has been moved to the storage directory **/vobstg/winvob.vbs**. The new owner is **vobadm** and the new group is **ccusers**. You must run this command as **root**.

*ccase-home-dir***/etc/utils/fix_prot -root -recurse -chown vobadm -chgrp ccusers /vobstg/winvob.vbs**

To repair ACLs damaged when a view storage directory was coped to **C:\ClearCaseStorage\integration.vws** using a copy utility that did not preserve Windows ACLs:

*ccase-home-dir***\etc\utils\fix_prot -root -recover C:\ClearCaseStorage\integration.vws**

To display the protection modes Clearcase associates with the directory **E:\vobstg\myvob**:

*ccase-home-dir***\etc\utils\fix_prot E:\vobstg\myvob**
```
drwxr-xr-x      MYDOMAIN\me      MYDOMAIN\mygroup     E:\vobstg\myvob
```

NOTE: **fix_prot** may return an error message that begins with the following text:

```
fix_prot: Error: unknown style protections on foo: The data is invalid.
```

If it does, you must rerun **fix_prot** and specify all of the **–chown**, **–chgrp**, and **–chmod** options with the absolute codes. For example:

*ccase-home-dir***etc\utils\fix_prot –chown** *owner* **–chgrp** *group* **–chmod 0666** *pname*

lsacl

lsacl [**–s** | **–l**] [**–n**] [**–f**] *path-name*

Options:

–s | **–l**
: Specifies short or long format; displays generic rights, by default.

–n
: Specifies that the numeric security ID (SID) is not to be translated into the user's name. Use this option if the domain controller is down or if the user's account has been removed.

–f
: Reads a security descriptor from a file; allows you to display the contents of the **identity.sd** and **groups.sd** files.

Note that you can also use **%SystemRoot%\system32\cacls** to display a DACL, but **cacls** cannot read a security descriptor from a file.

20.4 Fixing Protection Problems

The following sections describe how to fix the protection problems described in *Causes of Protection Problems* on page 293.

To fix most protection problems:

1. Log on as a member of the Administrators or Backup Operators group.
2. If the **groups.sd** file exists in the storage directory root, run this command:

*ccase-home-dir***etc\utils\lsacl –f** *stg-pname***groups.sd**

Note the supplementary group list. The following is sample output:

```
===== stg-pname\groups.sd
Owner: FOO\bob (User) (non-defaulted)              (Owner)
Group: FOO\usersnt (Group) (non-defaulted)         (Primary group)
ACL (revision 2):
0: allowed
SID: FOO\user (Group)                              (Supplementary group)
rights (00000000)

1: allowed
SID: FOO\tester (Group)                            (Supplementary group)
rights (00000000)

===== stg-pname\groups.sd
Owner: FOO\bob (User) (non-defaulted)              (Owner)
Group: FOO\usersnt (Group) (non-defaulted)         (Primary group)
ACL (revision 2):
Empty ACL: all access denied                       (No supplementary group)
```

3. Issue the following command:

   *ccase-home-dir***etc\utils\fix_prot –r –root –chown** *owner* **–chgrp** *group stg-pname*

   fix_prot –root removes the supplementary group list.

   If you are fixing view storage, you are finished. There should be no supplementary groups for the view storage directory.

4. If you are fixing VOB storage and your VOB had the supplementary group list, run this command:

   *ccase-home-dir***bin\cleartool protectvob –add_group** *group-name[,...] vob-stg-pname*

5. Remove the **cleartext** containers. To do so, log on as the VOB owner and run this command:

   *ccase-home-dir***bin\scrubber –e –k cltxt** *vob-stg-pname*

   This step is necessary because **cleartool checkvob** cannot fix the **cleartext** containers.

6. Fix the storage pool's protections. Log on as the VOB owner and run this command:

   *ccase-home-dir***bin\cleartool checkvob –force –fix –protections –pool** *vob-stg-pname*

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W