

ADMINISTERING CLEARCASE LT

Release 4.1 and later

Windows/UNIX Edition

Rational[®]
the e-development company™

800-023901-000

Administering ClearCase LT
Document Number 800-023901-000 November 2000
Rational Software Corporation 20 Maguire Road Lexington, Massachusetts 02421

IMPORTANT NOTICE

Copyright Notice

Copyright © 1992, 2000 Rational Software Corporation. All rights reserved.
Copyright 1989, 1991 The Regents of the University of California
Copyright 1984–1991 by Raima Corporation
Copyright 1992 Purdue Research Foundation, West Lafayette, Indiana 47907

Trademarks

Rational, the Rational logo, Atria, ClearCase, ClearCase MultiSite, ClearCase Attache, ClearDDTS, ClearQuest, ClearGuide, PureCoverage, Purify, Quantify, Rational Rose, and SoDA 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.

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

Microsoft, MS, ActiveX, BackOffice, Developer Studio, Visual Basic, Visual C++, Visual InterDev, Visual J++, Visual Studio, Win32, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

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

Oracle and Oracle7 are trademarks or registered trademarks of Oracle Corporation.

Sybase and SQL Anywhere are trademarks or registered trademarks of Sybase Corporation.

U.S. Government Rights

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.

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.

Warranty Disclaimer

This document and its associated software may be used as stated in the underlying license agreement, and, except as explicitly stated otherwise in such 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 software and documentation is based in part on software written by Victor A. Abell while at Purdue University. We acknowledge his role 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

Preface	xxi
About This Manual	xxi
ClearCase LT Documentation Roadmap	xxii
Online Documentation	xxiii
.....	xxiii
Technical Support	xxiv

Administering the ClearCase Network

1. Understanding the ClearCase LT Network	1
1.1 Network Overview	1
1.2 ClearCase Hosts.....	2
Other Hosts	2
1.3 ClearCase Data Storage	3
Versioned Object Bases (VOBs).....	3
Views.....	4
1.4 ClearCase User Base	5
1.5 Tags for VOBs and Views	6
1.6 ClearCase Client/Server Processing	6
ClearCase Server Processes.....	7
Server Error Logs	7
1.7 ClearCase Startup and Shutdown	8
UNIX Startup and Shutdown.....	8
Windows NT Startup and Shutdown	8

2. Administering ClearCase LT Hosts	9
2.1 ClearCase Administration Console.....	9
2.2 ClearCase Data and Non-ClearCase Hosts.....	10
2.3 Administering Networkwide Release Areas on UNIX.....	11
Changing the Location of the Release Area.....	11
Renaming a UNIX Release Host.....	11
3. Understanding ClearCase Access Controls	13
3.1 Fundamentals of ClearCase Access Control.....	13
Users and Groups.....	13
Setting the Primary Group: Special Considerations on Windows	14
Privileged Users and Groups.....	15
User Processes.....	15
ClearCase Objects	16
Access to ClearCase Data	18
3.2 Access Control for VOBs and VOB Objects	19
Access Control for VOBs	19
Permission to Create VOBs	20
Permission to Delete VOBs	20
Permission to Read VOBs.....	20
Permission to Write VOBs.....	20
Permission to Execute VOBs.....	20
Access Control for Elements	21
Permission to Create Elements.....	21
Permission to Delete Elements	22
Permission to Read Elements	22
Permission to Write Elements	22
Permission to Execute Elements.....	23
Access Control for Other VOB Objects.....	23
Permission to Create Other VOB Objects	23
Permission to Delete Other VOB Objects.....	24
Permission to Read Other VOB Objects.....	24
Permission to Write Other VOB Objects	24

	Locks on VOB Objects	24
	Locking Type Objects	24
3.3	Access Control for Views and View Objects	25
3.4	ClearCase and Native File-System Permissions	25
4.	Administering Windows NT Domains and ClearCase	27
4.1	Configuring Domains for Use with ClearCase	27
	ClearCase Requirements for Users, Groups, and Domains	28
	ClearCase Users and User Groups	28
	ClearCase Restrictions	28
4.2	Creating a Domain	29
4.3	Establishing User Account and Group Assignments	30
	Setting Up Domain Accounts	30
	Setting a User's Primary Group	31
	Verifying a User's Primary Group	32
	Overriding a User's Primary Group	32
4.4	Multiple User Account Domain Support	33
	Creating Proxy Groups and Enabling Domain Mapping	33
	Setting VOB Element Permissions	35
	Setting VOB Storage ACLs	35
4.5	Nondomain Systems	36
5.	Administering ClearCase LT Licenses	37
5.1	Overview of ClearCase LT Licensing	37
	Floating and Node-Locked Licenses	38
	Temporary, Permanent, and Term Licenses	38
	ClearCase LT and Rational Suites Licenses	39
5.2	Installing and Configuring a License Server	39
5.3	Installing and Configuring Windows Client Licenses	39
	The License Key Administrator	39
5.4	Installing and Configuring UNIX Client Licenses	40

Administering VOBs

6. Understanding VOB Storage	43
6.1 Introduction to Versioned Object Bases	43
6.2 VOB Database	44
6.3 VOB Storage Pools.....	45
Source Storage Pools	45
Cleartext Storage Pools	46
Derived Object Storage Pools	46
The vob_server Process.....	46
6.4 Storage Pool Creation.....	47
7. Setting Up VOBs	49
7.1 ClearCase LT Server Configuration.....	49
7.2 Planning for One or More Additional VOBs	50
Planning for Release VOBs.....	52
7.3 Modifying a ClearCase LT Server on UNIX	52
UNIX Kernel Resources	52
7.4 Creating VOB Storage Locations.....	53
7.5 Creating a VOB	53
Linking a VOB to an Administrative VOB	54
Adjusting the VOB's Ownership Information	55
Case 1: One Group for All VOBs, Views, and Users	55
Case 2: Accommodating Multiple User Groups.....	55
7.6 Coordinating the New VOB with Existing VOBs	56
7.7 Populating a VOB with Data	56
Importing Data into a UCM Project.....	57
Example: Importing RCS Data.....	57

	Creating the Data File.....	57
	Running clearimport	57
	Example: Importing PVCS Data	58
	Creating the Data File.....	58
	Running the Conversion Scripts	59
7.8	Converting a SourceSafe Configuration	60
	Overview of Payroll Configuration.....	60
	Shares	61
	Branches	62
	Labels	62
	Pins	62
	Setting Your Environment	62
	Setting Environment Variables	62
	Setting Your SourceSafe Current Project.....	63
	Running clearexport_ssaf.....	63
	Using the Recursive Option.....	63
	Example.....	63
	Running clearimport.....	65
	Examining the Results	66
	Version Numbers	67
	Labels	67
	Branches	68
	Pins	68
	Shares	68
7.9	Upgrading a VOB to a New Release	68
	Upgrading the Feature Level of a VOB.....	69
	Reformatting a VOB.....	69
8.	Using Administrative VOBs and Global Types.....	71
8.1	Overview of Global Types	71
8.2	Why Use Global Types?	72
8.3	Working with Administrative VOBs.....	72
	Creating an Administrative VOB.....	72

	Linking a Client VOB to an Administrative VOB.....	73
	Administrative VOB Hierarchies	73
	Listing an AdminVOB Hyperlink	74
	Restrictions on Administrative and Client VOBs	75
	If an Administrative VOB Becomes Unavailable	76
	Breaking a Link Between a Client VOB and an Administrative VOB	76
	Removing the AdminVOB Hyperlink.....	76
	Removing All GlobalDefinition Hyperlinks	76
	Removing an Administrative VOB	77
8.4	Working with Global Types.....	78
	Creating a Global Type	78
	Auto-Make-Type Operations.....	79
	Describing Global Types	80
	Listing Global Types	80
	Listing History of a Global Type	81
	Changing Protection of a Global Type	82
	Locking or Unlocking a Global Type.....	82
	Changing the Type of an Element or Branch	83
	Copying a Global Type	83
	Renaming a Global Type	84
	Changing the Scope of a Type	84
	Removing a Global Type	85
	Cleaning Up Global Types	86
9.	Backing Up and Restoring VOBs.....	87
9.1	Choosing Backup Tools	87
	UNIX Backup Issues.....	87
	Windows Backup Issues.....	88
9.2	Backing Up a VOB.....	88
	Backing Up a VOB on UNIX.....	88
	Backing Up a VOB on Windows NT.....	89
	Choosing Between Standard and Semi-Live Backup	90
	Benefits of Semi-Live Backup	91
	Costs of Semi-Live Backup.....	91

	Enabling Semi-Live Backup.....	91
	Determining a VOB's Location	92
	Ensuring a Consistent Backup.....	92
	Locking and Unlocking a VOB	92
	Partial Backups	93
	Cleartext Pool Backup	94
	Administrative Directory Backup	94
	Incremental Backups of a VOB Storage Directory	94
9.3	Restoring a VOB from Backup with vob_restore	95
	vob_restore: Sample Session.....	96
	Target Prompt.....	97
	Storage Directory Prompt.....	97
	Snapshot Prompt.....	98
	Backup-Loaded Prompt	98
	Sample VOB Restoration Scenario.....	98
	vob_restore: Restoration Scenarios.....	104
	How vob_restore Determines the Scenario.....	105
	Restoration Rules and Guidelines	106
	vob_restore: In Place.....	107
	vob_restore: VOB Is Active.....	108
	vob_restore: Move VOB on Same Host	108
	vob_restore: Move VOB to New Host	109
	vob_restore: Unregistered	110
	vob_restore: Restoring with a Database Snapshot	110
9.4	Restoring a VOB from Backup Without vob_restore.....	112
9.5	Restoring an Individual Element from Backup	114
9.6	VOB and View Resynchronization.....	117
	Resynchronizing Views and VOBs.....	118
10.	Administering VOB Storage	119
10.1	VOB Storage Maintenance	119
	Scrubbing VOB Storage Pools	120
	Scrubbing VOB Databases	121

Database Scrubbing: Logical vs. Physical	121
10.2 User-Supplied Maintenance Procedures.....	121
10.3 Removing Unneeded Versions from a VOB.....	122
11. Moving VOBs.....	123
11.1 Moving a VOB on Windows NT	123
The Move VOB Procedure for Windows NT.....	124
11.2 Moving a VOB on UNIX.....	125
The UNIX Move VOB Procedure	126
12. Removing VOBs	129
12.1 Locking as an Alternative to VOB Deactivation	129
12.2 Taking a VOB Out of Service	129
Restoring the VOB to Service.....	130
12.3 Removing a VOB	130
13. Using checkvob.....	133
13.1 When to Use checkvob.....	133
13.2 Checking Hyperlinks	134
13.3 Checking Global Types.....	134
Fix Processing.....	134
Output Log for Global Type Checking.....	135
Example Check or Fix Scenario	135
13.4 Database or Storage Pool Inconsistencies	141
Updating the VOB Database.....	143
Requirements for Using checkvob	144
Running checkvob	144
Output Log for Pool Checking	145
Overview of checkvob Processing	149
Individual File Element Processing	150
Pool Mode (-pool Option) Processing: Overview	151
Force-Fix Mode	151
Pool Setup Mode	152

Descriptions of Storage Pool Problems	153
Source or Cleartext Pool: Bad Pool Roots	154
Description	154
Cause	154
Fix Processing	154
Source Pool: Misprotected Container on Windows NT	154
Description	154
Cause	155
Fix Processing	155
Missing and Unreferenced Data Containers	155
Source Pool: Missing Container	156
Source Pool: Unreferenced Container (Debris)	158
Source Pool: Corrupted Container	159
Description	159
Cause	159
Fix Processing	159
13.5 Sample Check and Fix Scenarios	159
Scenario 1: VOB Database Newer Than Storage Pools	160
Running checkvob	161
Scenario 2: Storage Pools Newer Than VOB Database	161
Running checkvob	161
13.6 Sample checkvob Runs	162
13.7 Database Newer Than Pools	162
13.8 Database Older Than Pools	162
13.9 Unreferenced Containers from Incremental Backup or Restore	163
14. Splitting VOBs with relocate	165
14.1 What Does relocate Do?	165
14.2 Element Relocation Illustrated	166
Cataloging in the Source VOB	169
Cataloging in the Target VOB	170
Relocating Borderline Elements	171
14.3 Before Relocating Elements	174

14.4	Common Errors During a Relocate Operation.....	175
	Errors Not Related to Source VOB Element Removal.....	176
	Errors During Source VOB Element Removal	176
14.5	After Relocating Elements.....	177
	Symbolic Links.....	177
	Upgrading Views That Rely on Symbolic Links.....	178
	Cleanup Guidelines.....	178
	Updating Directory Versions Manually.....	180
	Fixing Symbolic Links Created by relocate	180
	Modifying Old Target Directory Versions to See Relocated Elements.....	181
	Modifying Newest Version of Source Directory to See Relocated Elements.....	182

Administering Views

15.	Views and View Storage.....	185
15.1	ClearCase LT Views	185
15.2	View Contents.....	186
15.3	View Storage Areas	186
15.4	Backing Up a View	187
15.5	Restoring a View from Backup.....	188
15.6	Configuring Text Modes for Views	190
	Text Modes	190
	Determining a View's Text Mode	191
	Choosing a Text Mode for a View.....	192
15.7	File Naming Issues in Mixed Environments	192
	Case Sensitivity	192
	Character Sets.....	193

Administering Scheduled Jobs

16. Managing Scheduled Jobs	197
16.1 Tasks and Jobs	197
Task and Job Storage.....	198
Task and Job Database Initialization	199
Job Execution Environment	199
16.2 The Default Schedule.....	200
16.3 Managing Tasks.....	200
Creating a Task.....	201
Editing a Task	202
Deleting a Task	202
16.4 Managing Jobs	202
Creating a Job.....	203
Specifying a Job's Schedule	204
Specifying Job Notifications	205
Viewing Job Properties.....	206
Editing Job Properties.....	206
Running a Job Immediately	207
Deleting a Job.....	207
16.5 Managing the Scheduler Access Control List	208

Administering Web Servers

17. Configuring a Web Server for the ClearCase Web Interface	213
17.1 Configuration Planning.....	213
Web Administration Considerations	213
ClearCase Considerations	214

17.2	Configuring the Web Server	215
	Apache.....	216
	Microsoft Internet Information Server (IIS)	217
	Netscape Enterprise Server	218
18.	Configuring Integrations with Microsoft Web Authoring Tools	221
18.1	Overview of the Integration.....	221
	Server Setup Overview	222
	Client Setup Overview.....	223
18.2	Server Setup Procedure	223
	Step 1: Install IIS.....	224
	Step 2: Install FPSE or OSE.....	226
	Step 3: Install ClearCase LT.....	226
	Step 4: Run the Web Authoring Integration Configuration Wizard.....	226
18.3	Client Setup Procedure	229
	Step 1: Install the Client Application	229
	Step 2: Add Web to Source Control	229
	From FrontPage 98	229
	From FrontPage 2000	230
	From Visual InterDev 6.0	230
	Step 3: Verify That New Web Content Is Added to Source Control.....	231
	Step 4: Setting User Permissions	233
	FrontPage 98.....	233
	FrontPage 2000.....	233
	Visual InterDev 6.0.....	233
	Step 5: Local Mode Client Setup for FrontPage 2000	233
18.4	Web Folders Support in Office 2000 and Microsoft Internet Explorer 5.....	235
18.5	Updating the Shared View on the Web Server	235
18.6	Considerations for Migrating and Converting Data to the Integration	235
18.7	Accessing Help Information for the Integration	236

Tuning for Performance

19. Improving ClearCase LT Server Performance	239
19.1 Minimize Process Overhead	239
19.2 Maximize Disk Performance	240
19.3 Add Memory for Disk Caching on Windows NT	240
19.4 Tune Block Buffer Caches on UNIX	240
Block Buffer Cache Statistics.....	241
Flushing the Block Buffer Cache.....	241

Troubleshooting

20. Repairing VOB and View Storage Directory ACLs on Windows NT	245
20.1 ClearCase ACLs.....	245
20.2 Causes of Protection Problems.....	247
Copying the Storage Directory.....	247
Converting the File System from FAT to NTFS.....	248
Editing Permissions	248
20.3 Utilities for Fixing Protection Problems.....	249
fix_prot.....	249
lsacl	250
20.4 Fixing Protection Problems.....	251
Index	253

Figures

Figure 1	VOB Database and VOB Storage Pools	45
Figure 2	Linking Multiple VOBs into a Single Directory Tree	51
Figure 3	Sample SourceSafe Payroll Configuration	61
Figure 4	ClearCase Version Tree of \bugfix\mod_empl.c Element	67
Figure 5	Administrative VOB Hierarchy	73
Figure 6	Semi-Live Backup	90
Figure 7	VOB Restoration	106
Figure 8	Restore Scenario Summarized by Output from vob_restore	106
Figure 9	vob_restore: In Place.....	107
Figure 10	vob_restore: VOB Is Active	108
Figure 11	vob_restore: Move VOB on Same Host	109
Figure 12	vob_restore: Move VOB to New Host	110
Figure 13	VOB Database or Storage Pool Skew Associated with VOB Snapshots	111
Figure 14	Controlling VOB Growth.....	120
Figure 15	Pool Access Through vob_server and VOB Database Access Through db_server.....	144
Figure 16	checkvob Output Log: Summary File.....	146
Figure 17	checkvob Output Log: Condensed Transcript File	148
Figure 18	Common Scenarios in VOB Database or Storage Pool Synchronization...	160
Figure 19	Elements Cataloged by Directory Versions Before Relocate Operation....	168
Figure 20	Directory Version Cataloging After Relocate Operation	169
Figure 21	Destination VOB That Includes Multiple Versions	170
Figure 22	Source VOB that Includes a Borderline Element.....	172
Figure 23	Source and Destination VOBs with Borderline Element Relocated	173
Figure 24	Source and Destination VOBs with Borderline Element Not Relocated ...	174
Figure 25	Source VOB with Multiple Branches on Parent Directory	179
Figure 26	Destination VOB After Modifying Old Version of Destination Directory	182
Figure 27	Setting Up the Root Web in the IIS Installation.....	225
Figure 28	Setting Up the VOB Storage for the Integration.....	227
Figure 29	Setting Up the View Storage for the Integration.....	228

Figure 30	FrontPage Source Control Icons.....	231
Figure 31	Visual InterDev Source Control Icons.....	232

Tables

Table 1	Protection Mode for a ClearCase Object	17
Table 2	Protection Mode Digits for a ClearCase Object.....	18
Table 3	Importance of VOB Directories in Partial Backups	93
Table 4	Characters That Are Not Allowed in file names on Windows	193
Table 5	Access Types in Scheduler ACL Entries.....	208
Table 6	Supported Platforms for Web Servers	224

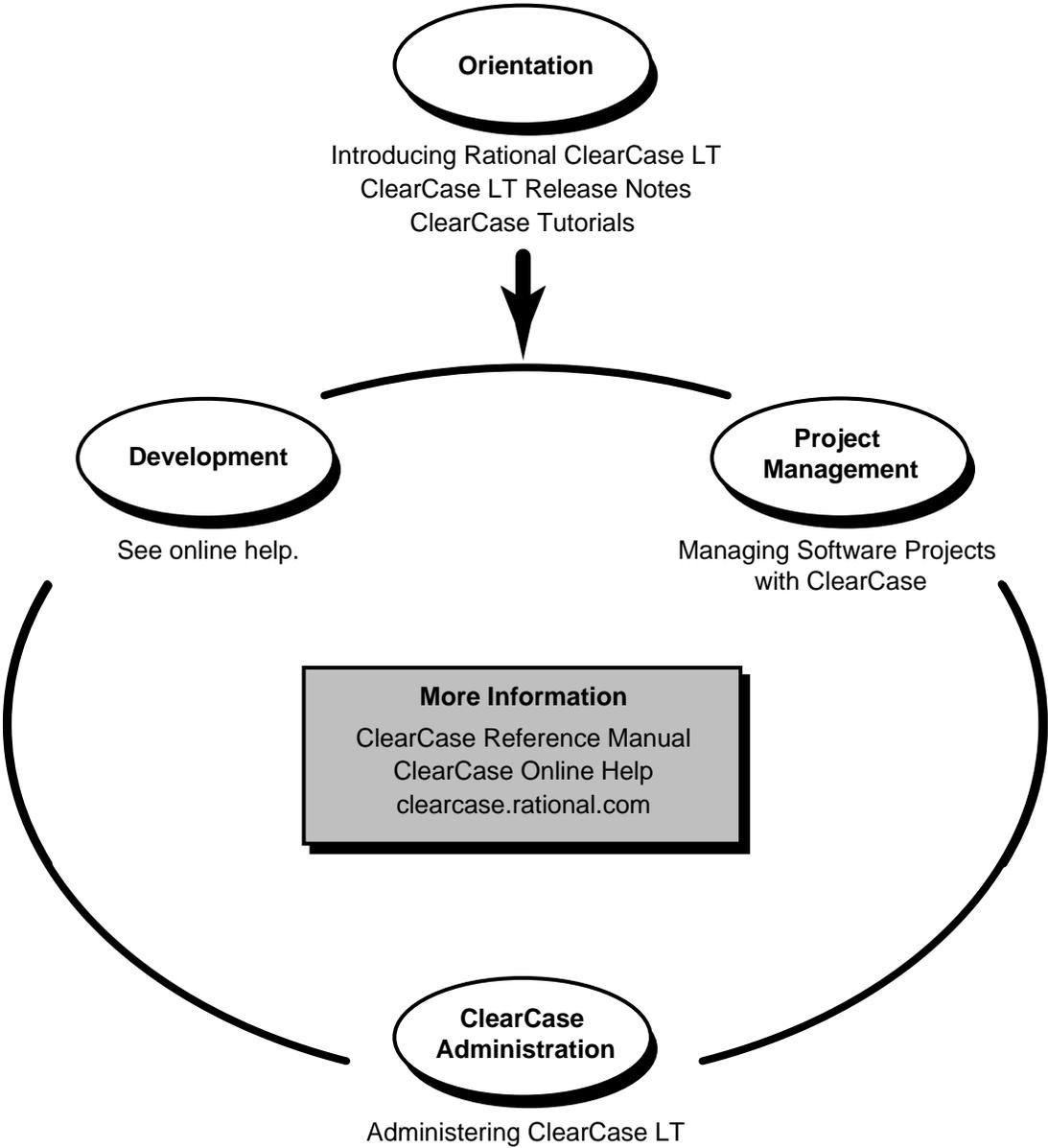
Preface

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. *Administering ClearCase 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

Understanding the ClearCase LT Network

1

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

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

- Windows 2000
- Windows NT
- Windows 95
- Windows 98

References to Windows NT also apply to Windows 2000 unless otherwise noted. References to Windows 95 or Windows 98 refer only to the specific operating system. References to UNIX refer to all UNIX and Linux platforms supported by ClearCase LT.

1.1 Network Overview

A ClearCase administrator has three principal concerns:

- **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 and provides client access to critical ClearCase data. ClearCase LT supports any combination of UNIX and Windows clients and servers. Any UNIX or Windows NT computer can be a

ClearCase LT server or client. Windows 95 and Windows 98 computers can only be ClearCase LT clients.

- ▶ **Data storage.** ClearCase data is stored in *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 used by individuals; however, it is likely that one or more shared views will be created, and these require some central administration.
- ▶ **User base.** ClearCase is used by a set of people, each of whom has a user name and is a member of one or more *groups*. Any number of people can use ClearCase on any number of hosts; the licensing scheme limits the number of concurrent users, but not the number of hosts.

1.2 ClearCase Hosts

ClearCase 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 client and the server.

- ▶ **ClearCase LT clients.** Each ClearCase LT user typically has one workstation. It is called a *client host* because it runs ClearCase client programs: the programs installed in *ccase-home-dir/bin*, including the **cleartool** command line facility and various graphical user interfaces (GUIs). 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.

Other Hosts

A ClearCase LT administrator may also be responsible for two other types of hosts. These hosts do not need to—and in most cases do not—have ClearCase LT installed. Nevertheless, they are important to the functioning of the ClearCase LT network.

- **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 ClearCase LT Licenses*, for more information on this topic.
- **Networkwide UNIX release host.** In a network that includes ClearCase LT client hosts on UNIX, one UNIX host in the network typically acts as the networkwide *release host*. This host stores an entire ClearCase release, exactly as it is supplied on the distribution medium. This release area is active storage, not archival storage; individual workstations can access ClearCase programs and/or data through UNIX symbolic links to this release area.

You may want to simplify your administrative chores by setting up the ClearCase LT server as the networkwide UNIX release host. Such a host actually does not need to have ClearCase LT installed, because it merely provides a network-accessible file system from which ClearCase LT clients can install the product. A networkwide UNIX release host must occasionally serve pages of executable files to ClearCase LT clients. This does not usually constitute a significant level of overhead for a properly configured computer, but if you want to minimize all non-ClearCase activity on the ClearCase LT server, you may want to assign networkwide release host duties to another UNIX computer.

1.3 ClearCase Data Storage

All ClearCase data is stored in Versioned Object Bases (VOBs) and views. These data repositories reside for the most part on the ClearCase LT server. Users of ClearCase LT clients see these structures as global resources; they access them through *VOB-tags* or *view-tags*.

Versioned Object Bases (VOBs)

The network's permanent ClearCase data repository is a centralized resource. Typically, this repository is distributed among multiple *versioned object bases (VOBs)* hosted on the ClearCase LT server. Each VOB is implemented as a *VOB storage directory* (actually a directory tree), which holds file-system objects and an embedded database.

Administration of a network's VOBs falls into these broad areas:

- **Registration.** All VOBs are listed in a networkwide *storage registry* on the ClearCase LT server. In a typical network, registry maintenance is minimal; certain ClearCase commands update the registry automatically. You must update the registry manually only if you move a VOB to another location (for example, to a different disk on the ClearCase LT server).
- **Backup.** VOBs have special backup and recovery requirements and must be backed up frequently and reliably. ClearCase does not include data-backup tools; use system-supplied or third-party tools. VOB backup and restore procedures are described in Chapter 9, *Backing Up and Restoring VOBs*.
- **Periodic maintenance.** Administering the central repository requires that you balance the need to preserve important data with the need to conserve disk space. ClearCase includes tools for collecting data on disk space used and for occasional *scrubbing* of unneeded data.

ClearCase includes a job scheduler that manages periodic execution of various administrative tasks, including disk-space data collection and VOB scrubbing. ClearCase installation sets up a predefined schedule of these maintenance operations. You can change this 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. As your organizational structure changes (for example, when a new project is launched), you may need to adjust a VOB's group list.
- **Growth.** As new projects are launched or existing projects are brought under ClearCase control, you must create new VOBs, define their accessibility to various groups, and incorporate them into your data backup and periodic maintenance schemes.
- **Reformatting.** From time to time, a new ClearCase 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 short-term storage for 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).

A ClearCase LT view is also known as a *snapshot view* because it keeps a “snapshot” of VOB data as ordinary files, usually on the ClearCase LT client’s local disk. A snapshot view consists of two directories, both of which are created as part of the view-creation process.

- The *view directory* contains files loaded from the VOB, along with view-private files that are generated during local operations (and may later be added to source 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 access and use for storage. You should back up view directories as you would back up and other directory that holds important files that change frequently, especially if these files are not regularly checked in to a VOB.
- 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.

Most VOBs 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 backups, periodic maintenance, and attention to access control issues.

1.4 ClearCase User Base

ClearCase does not maintain a database of its users. Any user who is logged on to a ClearCase host and can acquire a license is able to use the software. Successful use of ClearCase depends on networkwide user and group naming consistency on all platforms: UNIX users must have the same user IDs and group IDs on all hosts; Windows user and group names must be the same on all hosts. Such consistency is usually achieved by using networkwide databases at the operating system level. On UNIX, the NIS **passwd** and **group** maps provide a consistent set of user identity information. On Windows NT, *domains* are the most reliable way to provide this consistency.

Each user has a user ID, a principal group ID, and a supplementary list of group IDs. These identities control the user’s permission to read and create ClearCase data. Nearly all ClearCase 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 more information on this topic.

1.5 Tags for VOBs and Views

All ClearCase LT data storage areas—all VOBs and views—in a local area network are centrally registered on the ClearCase LT server in three *registries*:

- An object registry that records the physical storage locations of VOBs and views
- A tag registry that records names assigned to VOBs and views
- A storage location registry that records the names directories on the ClearCase LT server used for VOB and view storage

For example, an object registry entry may record the fact that a VOB's storage directory is located on ClearCase LT server host **neptune** at pathname **C:\vobstore\project1.vbs**; a corresponding tag registry entry may record the fact that on each developer's workstation, the VOB is referenced using the VOB-tag **\proj1**.

Similarly, an object registry entry may indicate that a view storage directory is **C:\viewstore\integ.vws** on ClearCase LT server host **neptune**; a corresponding tag registry entry may enable developers to access the view using the view-tag **integration**.

1.6 ClearCase Client/Server Processing

Because ClearCase is a distributed client/server application, multiple processes, running on multiple hosts, can play a role in the execution of ClearCase commands—even the simplest ones. 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 (RPC).
2. The checkout procedure requested in the RPC is handled by several server processes 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.

ClearCase Server Processes

Each ClearCase LT server runs an Atria Location Broker Daemon process, **albd_server**, which is usually started automatically at boot time; it can be started and stopped manually as needed by a user with adequate privileges. Other ClearCase server processes are started, as needed, by the **albd_server** process.

Most server processes manage a particular data structure; for example, a **view_server** process manages a particular view storage directory. These server processes all run on the ClearCase LT server, which hosts several processes of this type:

- **view_server**—Manages the view storage directory and database of a particular view.
- **vob_server**—Manages the storage pools of a particular VOB.
- **db_server**—Fields requests from one ClearCase client program, destined for one or more VOB databases on a host.
- **vobrpc_server**—Fields requests from one or more **view_server** processes, destined for a particular VOB database.

Server Error Logs

Each ClearCase server process maintains an error log on the ClearCase LT server.

- On UNIX hosts, these logs are stored in the directory **/var/adm/atria/log**.
- On Windows NT hosts, these logs are stored in the directory *ccase-home-dir***var**\log. In addition, most ClearCase server processes write error messages to the Windows NT event log.

Given the distributed architecture of ClearCase, an error resulting from a command entered on a ClearCase LT client will probably generate an error log entry on the ClearCase LT server, rather than on the client. In such cases, the user is directed to the appropriate log file on the ClearCase LT server. See the **getlog** and **errorlogs_ccase** reference pages in the *ClearCase Reference Manual* for details on these error logs and how to display them. In addition, users on Windows NT hosts can use the ClearCase Administration Console to access error logs on the ClearCase LT server, regardless of whether the server is running Windows NT or UNIX.

1.7 ClearCase Startup and Shutdown

ClearCase normally starts and stops when a ClearCase host starts up or shuts down, using the normal conventions for the operating system.

UNIX Startup and Shutdown

On UNIX, when a ClearCase LT server is bootstrapped, the ClearCase startup script is executed by **init(1M)**. The startup script does the following:

- ▶ Starts the **albd_server** process.
- ▶ Starts a **lockmgr** process, which arbitrates concurrent access to VOB databases by multiple client processes.
- ▶ Performs additional file-system setup tasks.

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

```
# ccase-home-dir/etc/atria_start stop
```

Stops all ClearCase processing on the ClearCase LT server.

```
# ccase-home-dir/etc/atria_start start
```

Restarts ClearCase processing on the ClearCase LT server.

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

Windows NT Startup and Shutdown

On Windows NT, ClearCase services are configured to start automatically when the computer is started. Unless configured to start manually, the lock manager and albd services are started at the same time. You can also start and stop the lock manager and albd services manually from the **ClearCase** program in Control Panel.

Administering ClearCase LT Hosts

2

This chapter discusses general issues in administering hosts on which ClearCase is installed. Other sections of this book discuss administration of hosts that function as servers for VOBs, views, licenses, and the ClearCase registry. For information on the ClearCase scheduler, which manages periodic jobs, see Chapter 16, *Managing Scheduled Jobs*.

2.1 ClearCase Administration Console

ClearCase on Windows NT 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 NT. 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 NT 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 NT. 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**.

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. For information on creating views, see the on-line help for *Developing Software with ClearCase*.
 - 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 NT 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

This section describes administration of the networkwide ClearCase release area on a UNIX host.

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 *ClearCase Product Family Installation Notes*.
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 *ClearCase LT Installation Notes*.
3. **Remove the old release area.** When all hosts have been reinstalled, you can remove the old release area.

Renaming a UNIX Release 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.

Understanding ClearCase Access Controls

3

This chapter describes how ClearCase 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 NT 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 NT, it is part of the Windows NT Server 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 NT, the primary group is assigned when the user's domain account is created. ClearCase considers both the user ID and primary group when determining a user's access rights to ClearCase objects.

Setting the Primary Group: Special Considerations on Windows

Because of a bug in Windows NT, a user who logs on to a domain account may not be assigned the primary group specified by the Windows NT domain account management tools. A correct, dependable primary group setting is critical for VOB access.

To work around this bug, we recommend that you set the user environment variable `CLEARCASE_PRIMARY_GROUP` to the correct primary group. For ClearCase purposes, the `CLEARCASE_PRIMARY_GROUP` environment variable offers a general alternative to primary group management through the User Manager for Domains (Windows NT) or the Active Directory Users and Computers MMC snap-in (Windows 2000). The `CLEARCASE_PRIMARY_GROUP` assignment has no security or access-control implications outside the context of VOB access. Users who do not have `CLEARCASE_PRIMARY_GROUP` set correctly are likely to have problems creating elements or otherwise accessing VOBs.

The value of this variable (*group-name* in the examples that follow) must be the name of an existing domain group that meets all of the following requirements:

- It includes the user as a member.
- It appears on the user's group list.
- It matches the user's primary group on UNIX.

To set this environment variable on a computer running Windows NT:

1. Click **Start>Settings>Control Panel**.
2. Run the System program and click the **Environment** tab.
3. Click any entry in the **User Variables** list.
4. In the **Variable** box, type `CLEARCASE_PRIMARY_GROUP`.
5. In the **Value** box, type *group-name*.
6. Click **Set**, and then click **OK**. This change does not take effect until the next logon.

To set this environment variable on a computer running Windows 2000:

1. Click **Start>Settings>Control Panel**.
2. Run the System program and click the **Advanced** tab.
3. On the **Advanced** tab, click **Environment Variables**.
4. In the **User Variables** section of the **Environment Variables** dialog box, click **New**.
5. In the **Variable Name** box, type `CLEARCASE_PRIMARY_GROUP`.
6. In the **Variable Value** box, type `group-name`.

On a computer running Windows 95 or Windows 98, you must set the `CLEARCASE_PRIMARY_GROUP` variable in the `AUTOEXEC.BAT` file.

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 ID that 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.
- On a ClearCase LT server running Windows NT, the privileged user is a member of a special domain group, the *ClearCase group*. See *Configuring Domains for Use with ClearCase* on page 27 for more information about the ClearCase group.

On either UNIX or Windows NT, the privileged user has the right to modify VOBs and the objects that VOBs contain.

User Processes

A user process is an application a user is running that requires access to ClearCase data. The application can be a ClearCase GUI, a command-line program like `cleartool`, or a non-ClearCase program such as a text editor that accesses ClearCase data in a dynamic view. When a user wants

to read or write ClearCase data, a process running on behalf of the user actually reads or writes the data.

Each user process has one of these properties, which are important for access control:

- **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

Some 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 208.

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. When ClearCase displays information about an object's protection mode, it usually uses a single-character abbreviation to display 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 (x) or No (-)	Yes (w) or No (-)	Yes (x) or No (-)
Member of Object's Group	Yes (x) or No (-)	Yes (w) or No (-)	Yes (x) or No (-)
Other	Yes (x) 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

Digit	Read Permission?	Write Permission?	Execute Permission?
0	No	No	No
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes
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 19 and *Access Control for Views and View Objects* on page 25.

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 or (on Windows NT) a *local Administrator* on the ClearCase LT server can use the **cleartool protectvob** command to change the VOB's owner, group, or supplemental group list.

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 the primary group ID of the process that creates the element. 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. A directory element initially has read, write, and execute permission for all user categories.
 - On Windows NT, a file element initially has only read permission for all user categories. 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 18.

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 exists:

- 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 exists:

- 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 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 18.

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.

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 on Windows NT, you may be able to repair the damage using ClearCase tools. See Chapter 20, *Repairing VOB and View Storage Directory ACLs on Windows NT*.

A 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.

Administering Windows NT Domains and ClearCase

4

A Windows NT domain is a unit of network administrative control. A domain contains a set of user accounts, groups, and computers. (It also contains other items that are not relevant to this discussion.)

Each domain contains a trust list, which is a list of all the other domains that it can work with. With the appropriate trust relationships in place, a user needs only one account in one domain. That user can then log on to a computer in another domain as long as that computer's domain has a trust relationship with the user's domain.

4.1 Configuring Domains for Use with ClearCase

To use ClearCase LT in a domain environment, configure the domains in one of the following ways:

- As a single Windows NT domain

Use one domain that contains all user accounts, group accounts, and computers.

- As separate logon and computer (that is, resource) domains

Use one domain that contains all user accounts and group accounts. Use one or more domains that contain only computers. Note that a computer can be a member of no more than one domain. We recommend, but do not require, that you put all ClearCase client and server computers in the same domain.

- As multiple domains for ClearCase users.

If your existing domain configuration requires it, you can use multiple domains that contain user and group accounts for ClearCase users, and use one or more domains that contain computers. Because this configuration is more complicated to set up and administer, you should avoid using it unless necessitated by overriding organizational or security concerns. See *Multiple User Account Domain Support* on page 33 for information on setting up user and group accounts in multiple domains.

ClearCase Requirements for Users, Groups, and Domains

The ClearCase security model sets requirements for users, groups, and domains. This section discusses the implications of ClearCase security for domains. For more information on access and permissions, see Chapter 3, *Understanding ClearCase Access Controls* and Chapter 20, *Repairing VOB and View Storage Directory ACLs on Windows NT*.

In this chapter, the term *community* refers to a set of users and computers that access a common ClearCase LT server. Large organizations can have multiple communities, perhaps at more than one site, all in a single Windows NT domain.

ClearCase Users and User Groups

Each VOB, view, and element has an associated owner and group. Initially, the owner is the user and the group is the primary group of the process that created the entity. Many ClearCase operations are restricted on the basis of the owner and group of the entity. For example, you cannot check out a version unless you are the owner of the element, a member of the element's group, or the owner of the element's VOB.

A community can have multiple user groups as part of its implementation of security policies. For example, one group may be allowed to check out certain files, and another group may be allowed only to read them. In this chapter, **clearusers** is the conventional name for the first, and perhaps only, ClearCase user group.

ClearCase Restrictions

The ClearCase security model sets certain requirements for ClearCase users, groups, and domains:

- All users who share ClearCase data must have domain accounts.

- All users who share ClearCase data must have the same primary group.
- There must be only one *ClearCase group*, typically called **clearcase**, over all the domains that are being used by a community. In other words, this single group must exist in only one domain.
- The domains that the ClearCase client and server computers belong to must trust the domains that contain the users.
- When ClearCase users who share data belong to more than one domain, additional restrictions apply. See *Multiple User Account Domain Support* on page 33.

4.2 Creating a Domain

To create a Windows NT domain:

1. Install the Server version of Windows NT on at least one system, and configure one Windows NT Server system to be the *domain controller*.

(At Windows NT Server installation time, you choose whether to make the system a domain controller. The only way to change a system's domain controller status is to reinstall the Windows NT Server software.)

2. After establishing the domain controller, add all other ClearCase client and server hosts to the domain.

If your environment already includes a domain and your Windows NT systems and users are using it, you do not need to create a new domain for ClearCase.

To verify or change a host's domain assignment:

- On Windows NT, open Control Panel and run the Network program. The host's current domain membership (if any) appears on the **Identification** tab. To change the host's domain membership, click **Change** and enter the new domain name.
- On Windows 2000, click **My Computer>Properties**. The host's current domain membership (if any) appears on the **Network Identification** tab. To change the host's domain membership, click **Properties** to open the **Identification Changes** dialog box. In this dialog box, you can change the host's domain.

NOTE: Although Windows 95 and Windows 98 computers cannot be members of a domain, users can log on to Windows 95 and Windows 98 computers using a domain account. ClearCase users on Windows 95 and Windows 98 should enable network login and log in using a domain account.

4.3 Establishing User Account and Group Assignments

If user accounts and group assignments are not yet set up correctly at your site, correct the problems before you install ClearCase LT. The remainder of this section describes how to perform these tasks.

Setting Up Domain Accounts

Every ClearCase user must have a domain account. In addition, users who share ClearCase data must have the same *primary group*.

You must do all of the following before you install ClearCase LT at a site:

1. Verify that each ClearCase user has an account in a Windows NT domain that includes other users who will share ClearCase data. If users have local, per-system accounts only, these accounts must be re-created as domain accounts.

Accounts created on the domain controller host become domain accounts automatically. Domain accounts can be created from a client host only by the domain's *Administrator* user or by a member of the domain's Administrators group.

2. Verify that all ClearCase users in a domain who are to share the same source-controlled files belong to the same primary group. If your network does not already have a domainwide group for all ClearCase users, create a group called **clearusers**.
 - > On Windows NT, click **Start>Programs>Administrative Tools >User Manager for Domains**. Click **User >New Global Group** and enter the group name.
 - > On Windows 2000, click **Start>Settings>Control Panel**. Open **Administrative Tools**, then open the **Active Directory Users and Computers** MMC snap-in. In the console tree, open the **Users** node and click **Action>New>Group**; then enter the group name.

3. Make **clearusers** (or other domainwide group) the primary group for each user (follow the steps in the section *Setting a User's Primary Group* on page 31).

NOTE: If ClearCase users at your site belong to more than one domain, see *Multiple User Account Domain Support* on page 33 for information about group membership and primary groups for these users.

4. Advise users that they must log on to the correct domain to use ClearCase.
5. (Optional) Create a password-protected ClearCase administrator account, **clearadm**, and make **clearusers** (or other domainwide group) the primary group of this account. This account must also be a member of the *ClearCase group*, typically called **clearcase**. We recommend that one or more ClearCase administrators use this single account to create views and VOBs and to implement VOB policies.
6. (Optional) Create one or more additional password-protected VOB owner accounts that are not members of the ClearCase group but are members of groups that need access to VOB objects. Designated members of groups that need VOB access can then log in using one of these VOB owner accounts to create views and VOBs and to implement VOB policies, but will not have broader ClearCase administrative powers. Before you create these users, read Chapter 3, *Understanding ClearCase Access Controls*.

Setting a User's Primary Group

Use the following procedure to set the primary group for a domainwide user account. All ClearCase users who share the same source-controlled files must have the same primary group.

1. Log on to the domain as *Administrator*.
2. Run the **User Manager for Domains** or **Active Directory Users and Computers** snap-in as described in Step #2 on page 29
3. Double-click the desired user name to open the **User Properties** dialog box.
4. In the **User Properties** dialog box, click **Groups**.
5. If the desired group is not in the **Member of** list, select the primary group (**clearusers**, for example) from the **Not Member of** list and click **Add**. In the **Member of** list, select the primary group and click **Set**.

CAUTION: Do not put the *ClearCase group*, typically called **clearcase**, in the group list of any users except a password-protected ClearCase administrator account.

If Windows NT clients will access UNIX VOBs. Each user's Windows NT primary group must match the user's primary group on UNIX as reported by the UNIX **id** command. If a user's primary group on UNIX is not in the Windows NT group list, you must create it.

- > On Windows NT, click **Start>Programs>Administrative Tools >User Manager for Domains**. Click **User >New Global Group**, and then enter the group name.
- > On Windows 2000, open **Control Panel**. Open **Administrative Tools**, and then open the **Active Directory Users and Computers** MMC snap-in. In the console tree, open the **Users** node and click **Action>New>Group**; then enter the group name.

See also Chapter 6, *Configuring ClearCase in a Mixed Network*.

NOTE: A bug in Windows NT may prevent you from making primary group assignments with predictable results. After you install ClearCase, users should verify their primary group assignment; see *Verifying a User's Primary Group* for details.

Verifying a User's Primary Group

After installing ClearCase, users should verify that their primary group is correct:

1. Log on under your normal user account, not as *Administrator*.
2. Run ClearCase Doctor by clicking **Start>Programs>Rational ClearCase LT>ClearCase Doctor**.

NOTE: ClearCase Doctor runs the first time you log on to your computer after installing ClearCase.

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

Overriding a User's Primary Group

If the primary group is incorrect, and your system administrator has made the correct group assignment, override the Windows NT primary group with the user environment variable

CLEARCASE_PRIMARY_GROUP. See *Setting the Primary Group: Special Considerations on Windows* on page 14 for details.

4.4 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 is more complicated to set up and administer, we recommend that you avoid using it unless organizational or security concerns require you to do so. If you need to use ClearCase in this environment, you must enable the ClearCase domain mapping feature as described in this section.

Creating Proxy Groups and Enabling Domain Mapping

Suppose that ClearCase 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 in this environment, create *proxy groups* and enable the domain mapping feature as follows:

1. Ensure that all computers running ClearCase are members of one or more resource domains that trust domains Atlanta, Boston, and Cupertino.
2. Create the *ClearCase group* (typically called **clearcase**) in one of the user account domains Atlanta, Boston, or Cupertino. (Create only one such group and user account, not one per domain.) The domain in which you create these accounts is the *primary ClearCase domain*. 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 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 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 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 host. Require each ClearCase user whose account is not in the primary ClearCase domain (see Step #2 of this procedure) to create a new value under the Windows registry key **HKEY_CURRENT_USER\SOFTWARE\Atria\ClearCase\CurrentVersion**. Use the following procedure:
 - a. Using a Windows NT registry editor such as **regedt32**, navigate to **HKEY_CURRENT_USER\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.

Windows 95 and Windows 98 computers must be rebooted before this change takes effect.

NOTE: Be sure to enable domain mapping on the Windows NT computer used by Windows 95 and Windows 98 computers as their credentials server.

6. Require each ClearCase user to set the per-user environment variable `CLEARCASE_PRIMARY_GROUP` to the value `Atlanta\clearusers`. See *Setting the Primary Group: Special Considerations on Windows* on page 14.

Setting VOB Element Permissions

All elements in any VOB that will be accessed by users from multiple account domains 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

You can restrict access to world-readable elements to a smaller set of users by setting the access control list (ACL) on the shared directory whose UNC path refers to the VOB storage directory on the ClearCase LT server. 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 allow access only by the groups `Atlanta\clearusers`, `Boston\clearusers_Boston`, `Cupertino\clearusers_Cupertino`, and the *ClearCase group*.

To change the ACL on a VOB storage directory:

1. In Windows Explorer on the ClearCase LT server, right-click the directory that corresponds to the shared directory to display the shortcut menu.
2. Click **Sharing**.
3. On the **Sharing** tab of the **Properties** dialog box, click the name of the share you want to change in the **Share Name** list.
4. Click **Permissions** and modify the permissions as needed.

4.5 Nondomain Systems

Windows NT systems that are not in a domain have extreme limitations:

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

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

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.

Administering ClearCase LT Licenses

5

This chapter provides an introduction to ClearCase_LT licensing and the FLEXlm License manager.

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 *ClearCase LT Installation Notes* include 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 user 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 license 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 Suites 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 both UNIX and Windows ClearCase LT clients.

- 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 *ClearCase LT Installation Notes*.

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 *ClearCase LT Installation Notes* for information about how to set up a FLEXlm license server host and obtain permanent ClearCase LT licenses.

Administering VOBs

Understanding VOB Storage

6

This chapter describes the on-disk data structures that implement *VOBs* on the ClearCase LT server. There is no VOB storage on ClearCase LT clients.

6.1 Introduction to Versioned Object Bases

The ClearCase data repository for a network is implemented as a set of *versioned object bases* (*VOBs*). Each VOB appears 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*.

A VOB itself exists as several directories in the file system. This directory structure is visible when you use native operating system utilities to list the *VOB storage directory* and its subdirectories, which include the following:

- **The VOB database.** The **db** subdirectory contains the binary files managed by an embedded database management system (DBMS). Each VOB has its own database; there is no central database that encompasses all VOBs. 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

- (Unified Change Management *Project VOBs* only) UCM objects: folders, projects, streams, activities, components, baselines, and so on

Actual file-system data (for example, the contents of version 3 of file **msg.c**) is not stored in the VOB database.

- **VOB storage pools.** The **c**, **d**, and **s** subdirectories contain the VOB's *storage pools*, each of which is a standard directory. The storage pools hold *data container* files, which store the VOB's file-system data: 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 *deltas* (version-to-version differences).
- **Identity directory.** On UNIX hosts, the **.identity** subdirectory contains files that establish the VOB's owner, its principal group, and its group list. These files determine the user ID, group ID, and group membership of the **vob_server** process. (On Windows NT, this information is included in the directory's access control list.)

The **vob** reference page provides a detailed description of the contents of a VOB.

ClearCase commands and utilities access a VOB by referring to its *VOB-tag*, which is a name with an associated global path to the VOB storage directory through which all hosts in a region can access the VOB.

6.2 VOB Database

Each VOB has its own database, implemented as a set of files in the **db** subdirectory of the VOB storage directory. ClearCase server programs are started as needed to access a VOB's database:

- A **db_server** process handles requests from a single ClearCase client program.
- A **vobrpc_server** process handles requests from one or more ClearCase **view_server** processes.

These server processes run on the ClearCase LT server.

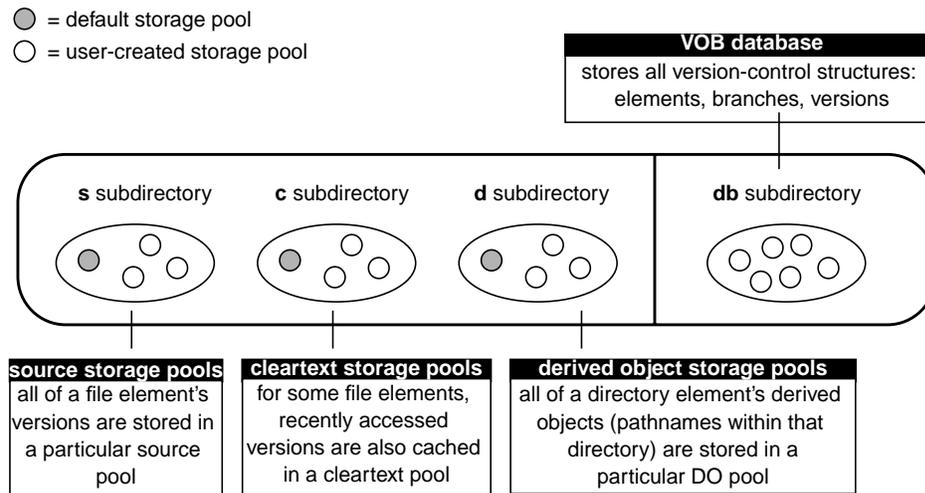
CAUTION: Moving a VOB is a complex procedure. You cannot simply use standard file-system utilities to copy or move the VOB database directory (**db**) to another host. See Chapter 11, *Moving VOBs*, for instructions. See Chapter 10, *Administering VOB Storage*, for steps you can take when a VOB database threatens to fill up its disk partition.

For the most part, ClearCase servers and the ClearCase scheduler manage routine VOB database maintenance automatically. See Chapter 10 for more on VOB maintenance.

6.3 VOB Storage Pools

Each VOB has a set of storage pools, which hold several different kinds of data containers. Each storage pool holds data containers of one kind (Figure 1).

Figure 1 VOB Database and VOB Storage Pools



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 reconstructing individual versions from *deltas* in the data container. Likewise, the type manager updates the data container when a new version is checked in.

Source pools are accessed by **cleartool** commands such as **checkout** and **checkin**, as well as by system utility programs like **type** or **cat(1)** that read the contents of elements that are not 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 elements for which all versions are stored in a single data container: compressed text files and text files.

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.

The **vob_server** Process

A ClearCase server program, the **vob_server**, provides most access to VOB storage pools. This process handles data-access requests from clients, forwarded to it by the **vobrpc_server**. As with these other servers, a **vob_server** runs on the host where the corresponding VOB storage directory resides. Each VOB on the ClearCase LT server has its own dedicated *vob_server* process.

6.4 Storage Pool Creation

When a new VOB is created, three subdirectories are created to hold *storage pools*:

c	Directory for all cleartext pools
c\cdft	Default cleartext pool
d	Directory for all derived object pools (not used by ClearCase LT)
d\ddft	Default derived object pool (not used by ClearCase LT)
s	Directory for all source pools
s\sdft	Default source pool

You cannot modify these pools or create new ones after the VOB has been created.

Setting Up VOBs

7

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

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

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

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 240.

- ▶ **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*.

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:

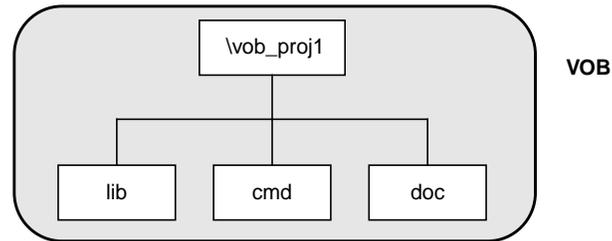
- ▶ Splitting data into several small VOBs increases your flexibility. It is easy to move an entire VOB to another disk. It is more difficult to split a large VOB into several new VOBs and move one or more of them to another disk.
- ▶ Typically, you have fewer performance bottlenecks when you use several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

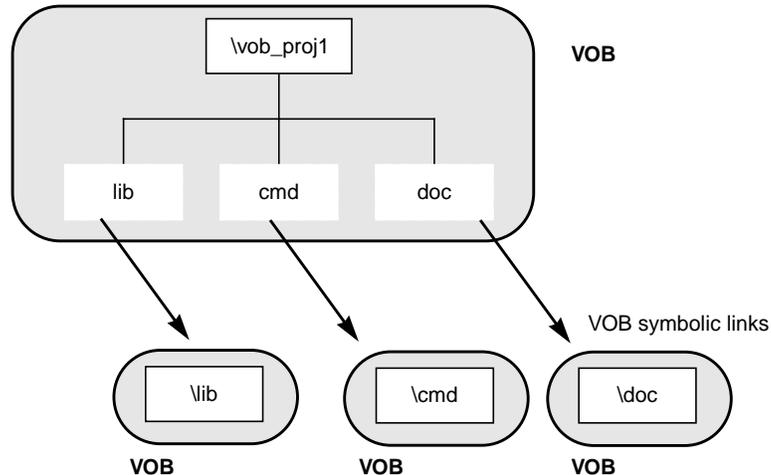
In your VOB planning, keep in mind that you can make several VOBs appear to be a single directory tree, using *VOB symbolic links* (Figure 2).

Figure 2 Linking Multiple VOBs into a Single Directory Tree

Directory Tree Implemented as One VOB



Directory Tree Implemented as Four VOBs



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  
Link created: "lib".
```

(..\vob_lib, not \vob_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 a battery of server processes, which run on the ClearCase LT server. These servers make significant demands on system resources. Accordingly, make sure that the system's configuration is adequate for ClearCase needs.

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 240.

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 NT 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 ClearCase 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 global *ClearCase group* (used mainly by ClearCase server programs) created at initial installation.

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 NT) 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 under *Creating VOB Storage Locations* in this chapter.

- > **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 NT.

The following command creates a VOB on Windows NT 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 NT, 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 73.

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 arise when group membership is used to restrict access to VOB data but 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

Small development organizations, and those in which data security is not a major issue, sometimes place all users in the same group. In such organizations, 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 53 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:

- a. The user's primary group is in the VOB's group list.

- b. 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. (See *Managing Software Projects with ClearCase* for an introduction to type objects.)

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 with ClearCase*.

Example: Importing RCS Data

A simple conversion scenario illustrates the migration of UNIX sources to ClearCase. In this example, we use an entire UNIX RCS source tree to populate a newly created ClearCase 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 the RCS-to-ClearCase export utility, **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 Windows sources to ClearCase. In this example, entire Windows PVCS source tree is used to populate a newly created ClearCase 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 the PVCS-to-ClearCase export utility, **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_ffile**, **clearexport_pvcs**, **clearexport_rcs**, **clearexport_sccs**, **clearexport_ssaf**, **clearexport_cvs**, and **clearimport** reference pages.

7.8 Converting a SourceSafe Configuration

This section uses a sample configuration to show how to use the **clearexport_ssaf** 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 3 Sample SourceSafe Payroll Configuration

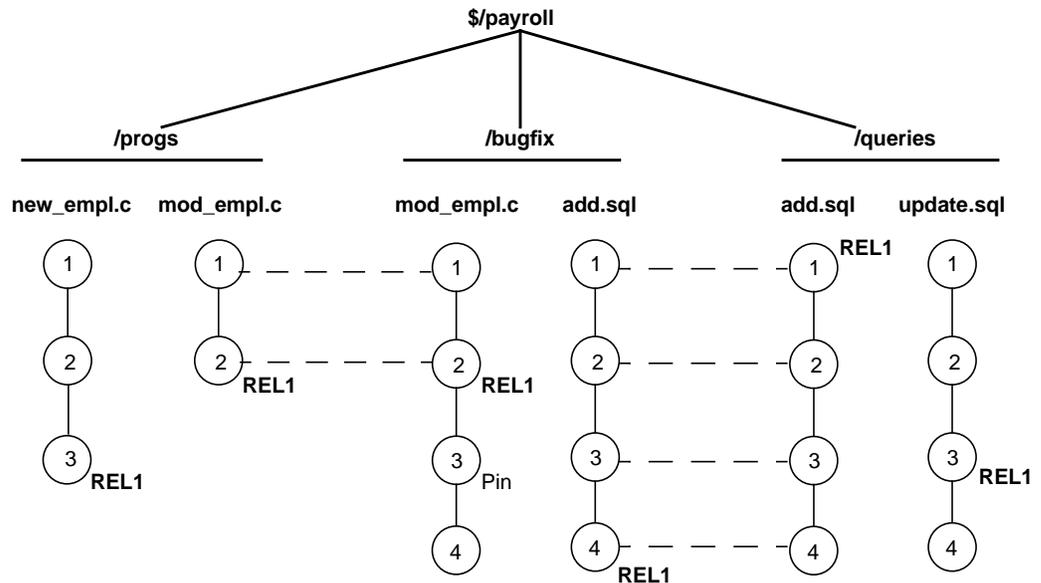


Figure 3 shows the configuration hierarchy in SourceSafe, which consists of these projects:

- **\$/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_ssaf**e 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_ssaf**e utility exports data for files and directories in your SourceSafe current project. Before running **clearexport_ssaf**e, verify that your SourceSafe current project is set so that **clearexport_ssaf**e 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_ssaf

The **clearexport_ssaf**e 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_ssaf**e 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_ssaf**e 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_ssaf**e 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_ssaf**e exports all files in the **/progs**, **/bugfix**, and **/queries** subprojects. The **-o** option directs **clearexport_ssaf**e 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"

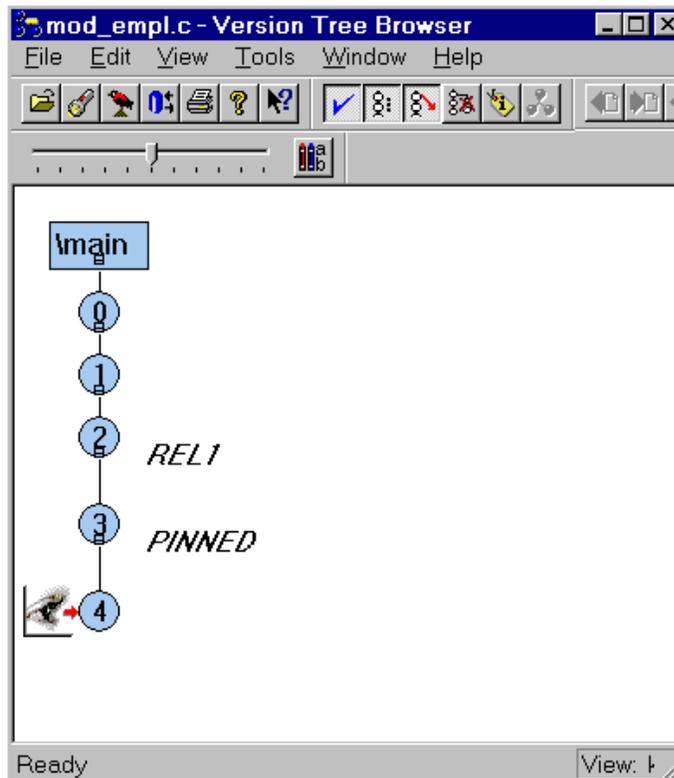
```

```
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 4 shows the version tree for the `\bugfix\mod_empl.c` element.

Figure 4 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.

7.9 Upgrading a VOB to a New Release

New ClearCase releases often introduce new features that require changes to the VOB database schema or other internal VOB data structures. To use these new features, you may need to take one or both of two actions to upgrade an existing VOB:

- Change the VOB's *feature level*.
- Reformat the VOB to change the VOB's database schema.

These actions are independent of each other. In general, you do not have to raise the feature level of a VOB unless you want the VOB to have the capabilities that require a higher feature level. Reformatting a VOB may be required, optional, or not applicable for a particular release. See *ClearCase LT Release Notes* for more information.

Upgrading the Feature Level of a VOB

A feature level is an integer that defines which ClearCase features a VOB supports. Whenever a ClearCase release introduces features that require support in the VOB server, you must raise the feature level of a VOB before clients can take advantage of the new features when accessing data in that VOB.

The **cleartool describe** command shows the feature level for a VOB. The **chflevel** command changes the feature level of a VOB. To raise the feature levels of unreplicated VOBs on a VOB server host:

1. Log on to the host that contains VOB storage directories for the VOBs you want to upgrade.
2. Issue the **chflevel** command with the **-auto** option. The command lists each VOB whose storage directory is located on 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 that is installed on the host.

Reformatting a VOB

A release of ClearCase may introduce a new schema, or format, for the VOB database. The new format may support new product features, enhance storage efficiency, or improve performance. Upgrading existing VOBs to use the new database format may be required or optional for a particular release. See *ClearCase LT Release Notes* for information about VOB database formats in the current release.

All VOBs on the ClearCase LT server must use the same database format. To change the database formats for VOBs:

1. Back up all VOBs on the ClearCase LT server. See Chapter 9, *Backing Up and Restoring VOBs*.
2. Ensure that the ClearCase installation on the VOB server host supports the new VOB database format. For more information, see *ClearCase LT Installation Notes*.
3. Run the **reformatvob** command for each VOB storage directory on the host. By default, the command dumps the existing VOB database into an ASCII representation, then loads that representation into a new database that uses the format corresponding to the release of ClearCase that is installed on the host.

For more information, see the **reformatvob** reference page in *ClearCase Reference Manual*.

Using Administrative VOBs and Global Types

8

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 the VOBs in your local area network use the same set of type objects by linking them all 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. ClearCase 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 creates a copy of the type object in the client VOB. This capability is called *auto-make-type*.
4. ClearCase 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 53.

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 73.
2. Create global types in it. See *Creating a Global Type* on page 78.

Linking a Client VOB to an Administrative VOB

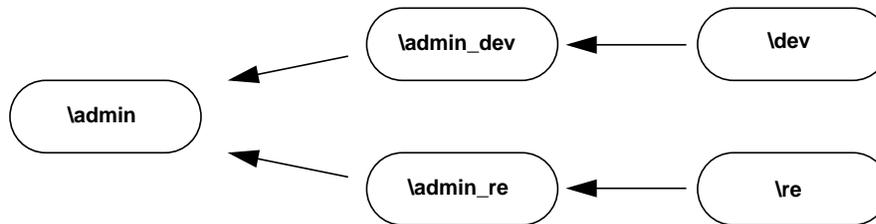
To associate a client VOB with an administrative VOB, create a hyperlink of type **AdminVOB** from the client VOB to the administrative VOB.

```
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 5 illustrates this example.

Figure 5 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 admin 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.

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 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 **mkatype**, **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 creates a local copy of the global type in the client VOB. Specifically:

- When you create attributes, branches, elements, hyperlinks, or labels, ClearCase creates a local copy of the global type.
- When a checkout operation causes the creation of a branch (auto-make-branch), ClearCase 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 creates the local copy if it does not already exist.
- If the global type has supertypes, ClearCase creates local copies of the supertypes and fires any **mk**type** triggers associated with them.

In addition, ClearCase 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 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 lctype: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 lctype:LABEL1@\dev
Changed protection on "LABEL1".
```

```
cleartool describe -local lctype: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 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 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 **mkatype**, **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 **mkatype**, **mkbtype**, **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 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 rmttype -nc lbtype:LABEL1@\dev
```

```
cleartool: Error: There are labels of type "LABEL1".
```

```
cleartool: Error: Unable to remove label type "LABEL1".
```

```
cleartool rmttype -nc -rsmall 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 **rmttype** command in a client VOB that does not contain a local copy of the global type, ClearCase 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 **rmttype** command again.

For more information on removing types, see the **rmttype** 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*.

Backing Up and Restoring VOBs

9

The most important maintenance task for a ClearCase LT administrator is to ensure frequent, reliable backups of essential ClearCase 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 187.

9.1 Choosing Backup Tools

ClearCase does not include any backup tools. All ClearCase 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 data structures.

Windows Backup Issues

Some common Windows backup utilities do not back up files that are open for writing. Because the VOB and view database files are typically open for writing while ClearCase is running, a backup operation skips these files, which makes the backup useless. To avoid this problem, do one of the following:

- Purchase and configure backup software to capture files that are open for writing.
- Lock the VOB and copy the storage directory to another on-disk location, and then back up the copy.
- Stop ClearCase before performing the backup.

9.2 Backing Up a VOB

Because of operating system differences, VOB backup operations are handled differently on UNIX and Windows NT. 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 NT

A VOB backup on Windows NT can be summarized as follows:

1. Lock the VOB.
2. If your backup software can be configured to capture files that are open for write access:
 - > Back up the VOB storage directory.
 - > Unlock the VOB.
3. If your backup software cannot process files that are open for write access, you have several alternatives:
 - > Copy the VOB storage directory as described in the section *Copying the Storage Directory* on page 247, unlock the VOB, and back up the copy.
 - > Copy the VOB database subdirectory (*vob-stg-dir\db*) to an on-disk location, back up the database copy plus the rest of the storage directory in place, and unlock the VOB.
 - > Use the semi-live backup strategy described in *Choosing Between Standard and Semi-Live Backup* on page 90.
 - > 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.

VOB storage on Windows NT is wholly contained in a single directory tree, which simplifies the backup process.

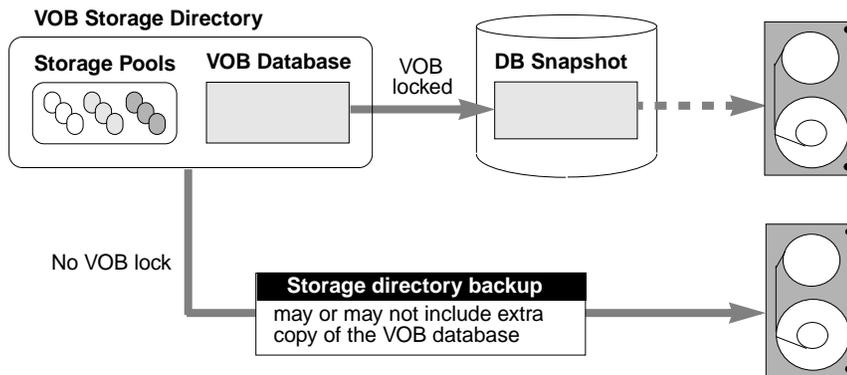
Choosing Between Standard and Semi-Live Backup

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 6, 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 6 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 110.

Enabling Semi-Live Backup

To enable database snapshots, run `vob_snapshot_setup` on a VOB. This command causes the ClearCase 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.

Determining a VOB's Location

To determine the location of a VOB storage directory, use the ClearCase Administration Console or the `cleartool lsvo` 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 lsvo -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 90), 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 NT 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\cdf`, 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 141) 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 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 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 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. **vob_restore** handles all of the following subtasks:

- Stops and restarts ClearCase
- Updates the ClearCase 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 can use **vob_restore** with or without a VOB snapshot. Any valid VOB backup (as defined in section 9.2) will work.

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 the *local Administrator* on Windows NT.
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 NT, 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 96
- *vob_restore: Restoration Scenarios* on page 104
- *vob_restore: Restoring with a Database Snapshot* on page 110

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 117.

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 restoration 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 104.

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 host'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>

Clear text 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 96.

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

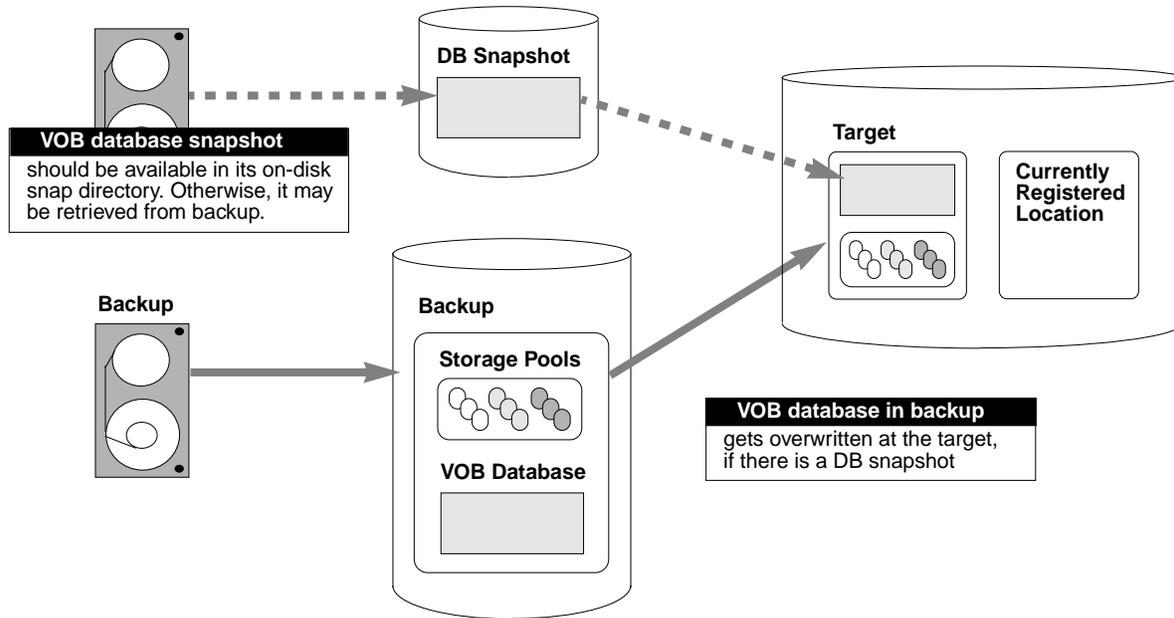
When you run **vob_restore**, it prompts for three critical pieces of information:

- 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 7, **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 7 VOB Restoration



vob_restore uses your input to describe the restore scenario. For example, Figure 8 shows the scenario from the sample restore session in the previous section.

Figure 8 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 108.
- 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.

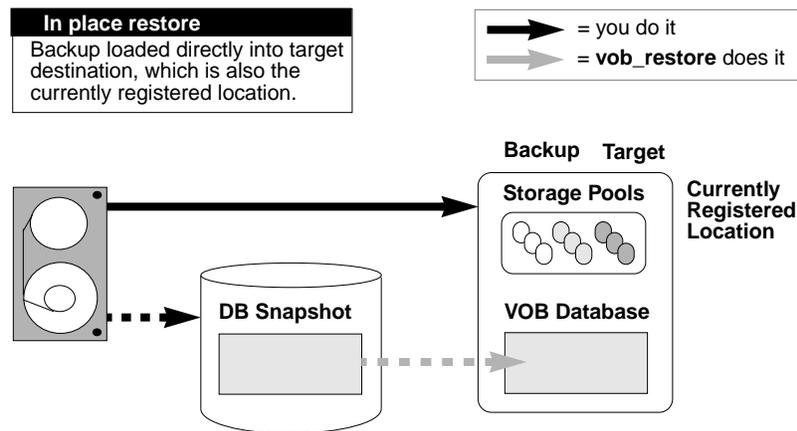
vob_restore: In Place

This scenario is illustrated in Figure 9

Formula. *backup = target = currently-registered-location*

Advice. Use this scenario whenever practical.

Figure 9 vob_restore: In Place



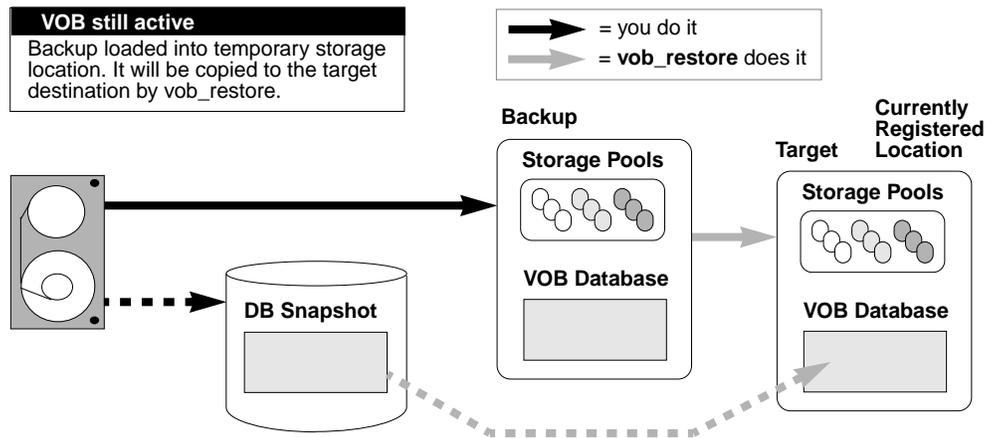
vob_restore: VOB Is Active

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 10.)

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 10 vob_restore: VOB Is Active



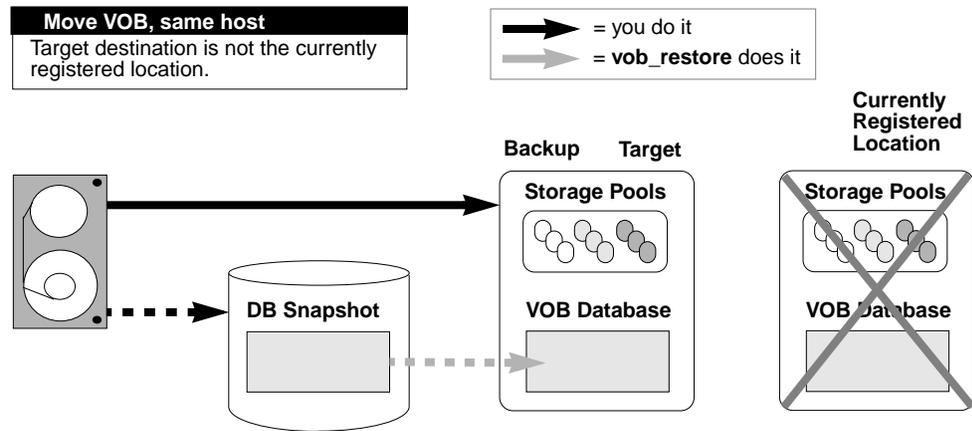
vob_restore: Move VOB on Same Host

In this scenario, the backup is restored to a different location on the same host. (See Figure 11.)

Formula. *target* different from *currently-registered-location*

Advice. Load the backup directly into the target destination.

Figure 11 vob_restore: Move VOB on Same Host



vob_restore: Move VOB to New Host

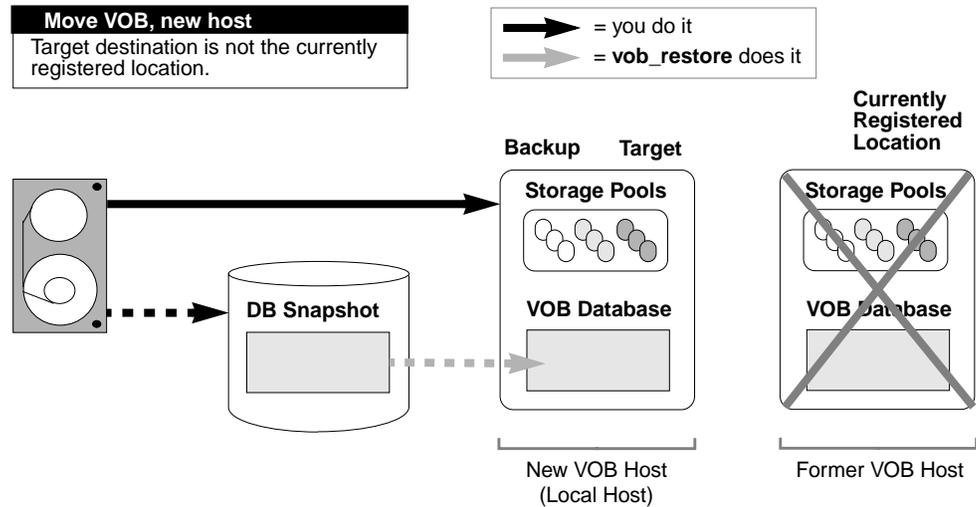
When moving a VOB as part of a **vob_restore** operation, the target host must have the same architecture (hardware and operating system). (See Figure 12.)

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 12 vob_restore: Move VOB to New Host



vob_restore: Unregistered

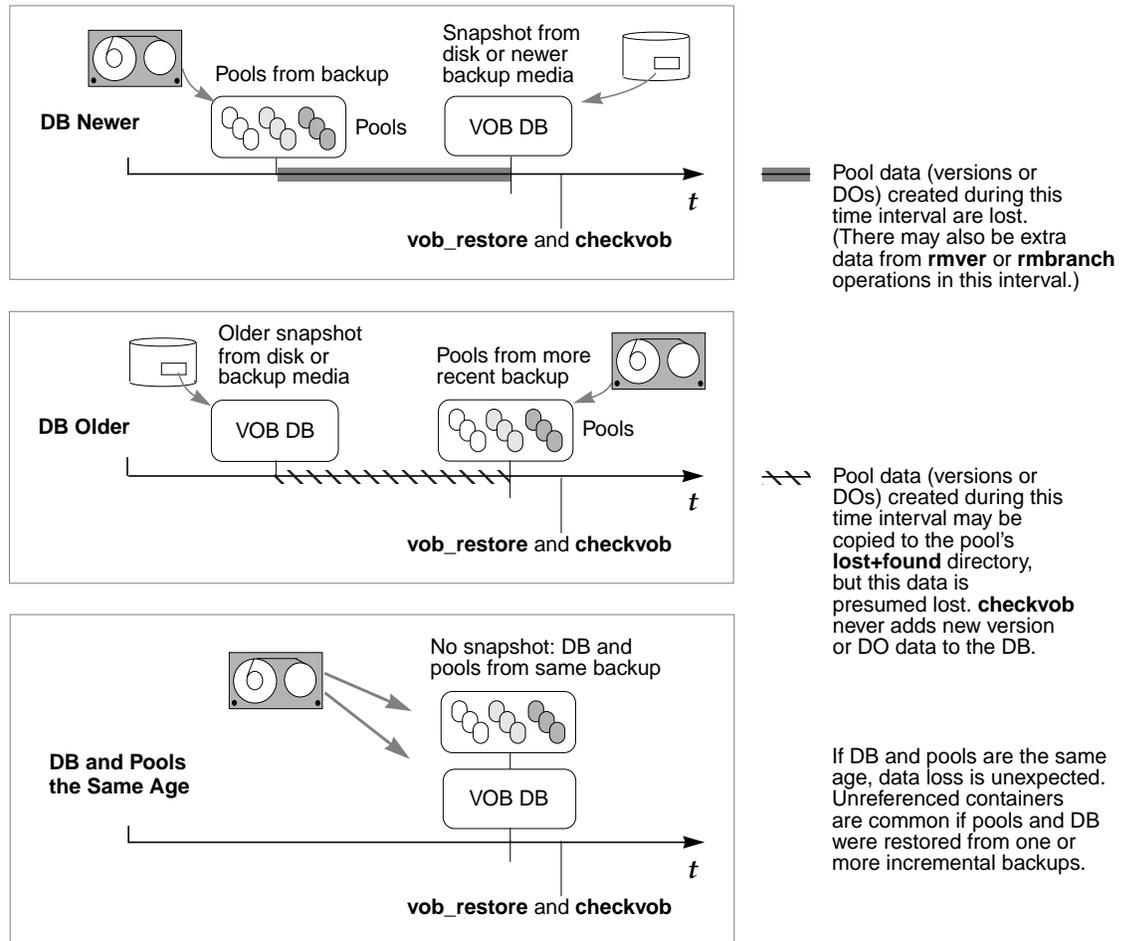
Formula. Same as in place, but no *currently-registered-location*

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 13 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 13 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 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 88.

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 the local *Administrator* on Windows NT.
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 NT, 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 NT 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 NT 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 NT*, 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 133, 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 117.

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 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 overall procedure for restoring a file element:

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 **clearexport_ccase** and **clearimport** to copy the element from the backup VOB to the temporary VOB.

6. Unregister the backup VOB.
7. Re-register the original VOB.
8. Use **clearexport_ccase** and **clearimport** to copy 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 ClearCase **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 112.
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. **Copy the element from the backup VOB to the temporary VOB.** Use the `clearexport_ccase` program to make a complete copy of the element. 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.

```
c:\> z:
z:\> cd oldproj\src
z:\oldproj\src> clearexport_ccase util.c
Converting element "util.c" ...
Extracting element history ...
.
.
Creating script file cvt_data...
z:\oldproj\src> cd \tmpvob
z:\tmpvob> clearimport \oldproj\src\cvt_data
Converting files from \oldproj\src to .
.
.
Checked in ".\util.c" version "\main\3".
Checked in ".\." version "\main\2".
```

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 Unified Change Management *Process VOB*, you must use the `-ucmproject` option with the `register` command.

- 10. Copy the element from the temporary VOB to the original VOB.** Use the `clearexport_ccase` program as in Step #6 to make a complete copy of the element. Create the data file in the temporary VOB, and run `clearimport` in the original VOB.

```
cd tmpvob
clearexport_ccase util.c
Converting element "util.c" ...
Extracting element history ...
.
.
Creating script file cvt_data ...
cd ..\proj\src
clearimport z:\tmpvob\cvt_data
Converting files from \tmpvob to .
.
.
Checked in ".\util.c" version "\main\3".
Checked in ".\" version "\main\5".
```

- 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.

This chapter discusses maintenance procedures for ClearCase VOB storage directories. Perform these procedures periodically to control the growth of these data repositories.

10.1 VOB Storage Maintenance

VOB administration involves a continual trade-off between these goals:

- Preserving critical data and metadata.
- Discarding data and metadata that is no longer important, to minimize the disk space required.

Figure 14 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.

There are several ways to display information about disk space used by the VOBs on the ClearCase LT server.

- In the ClearCase Administration Console, the VOB storage node for a VOB subnode of the ClearCase 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.

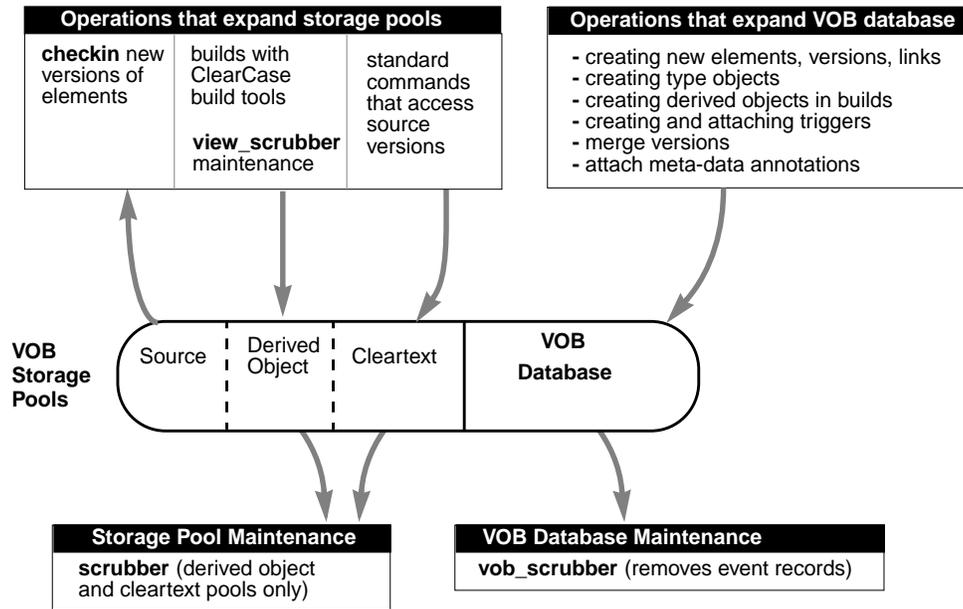
The ClearCase scheduler runs several jobs that gather data on disk space used by VOBs and that can reclaim excess disk space used by local VOBs:

- 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

For more information on the ClearCase scheduler, see Chapter 16, *Managing Scheduled Jobs*.

Figure 14 Controlling VOB Growth



Scrubbing VOB Storage Pools

On the ClearCase LT server, the ClearCase scheduler runs a daily job that scrubs the storage pools of all VOBs:

- Source pools are never scrubbed automatically. Source versions are too valuable to be routinely deleted. (But see *Removing Unneeded Versions from a VOB* on page 122.)

- 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 performance.

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.

Database Scrubbing: Logical vs. Physical

Deletion of event records from a VOB database by the **vob_scrubber** and **scrubber** utilities is logical, not physical. That is, the scrubbers do not reduce the size of any file in the VOB database subdirectory (**db**). Instead, they increase the amount of free space within these files, for use by newly created event records.

If you need to shrink a VOB's on-disk storage, use the **reformatvob** command, which discards all such free space. We don't recommend shrinking VOBs routinely; it's usually sufficient to use maintenance procedures that keep them from growing too fast.

10.2 User-Supplied Maintenance Procedures

The ClearCase scheduler runs default daily and weekly jobs. As shipped, these jobs do nothing, but you can customize them to run your own programs, including pool scrubbing. See *The Default Schedule* on page 200 for more on this topic.

10.3 Removing Unneeded Versions from a VOB

In general, approach the removal of source data from VOB storage with extreme caution. Removing entire elements, using **rmelem**, is particularly dangerous:

- Even if an element is no longer needed for the next release, you may need it to reproduce and maintain previous releases.
- **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. Renaming an element, using the **rmname** command, preserves its history without cluttering future projects.
- 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 114.)

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

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 locations. This chapter presents procedures for moving VOBs from one disk partition to another on the ClearCase LT server.

NOTE: Only a ClearCase LT server can be a VOB host.

11.1 Moving a VOB on Windows NT

This section explains how to move a VOB to another disk partition on the ClearCase LT server.

WARNING: When moving a VOB or view storage directory, use copy or backup software that preserves ownership and access control information.

For clarity, this section uses an example:

- The current location of the VOB storage directory to be moved is `c:\vobstore\libpub.vbs`, on a host named `sol`.
- The VOB is known to ClearCase LT clients as `\libpub`.
- The new location for the VOB storage directory is `c:\vobstore2\libpub.vbs`.

The Move VOB Procedure for Windows NT

Perform this procedure as the VOB owner:

1. **Tell users not to access the VOB.** This step is optional. Moving a VOB from which a view is being loaded or updated does not affect VOB data integrity, but could result in an incomplete load or update operation.
2. (If applicable) **Disable VOB snapshots on the ClearCase LT server.** If VOB database snapshots are enabled on the VOB, disable them with the following command

```
vob_snapshot_setup rmparam \libpub.
```

3. **Back up the VOB storage directory.** Use the procedures described in Chapter 9, *Backing Up and Restoring VOBs*
4. **Lock the VOB.**

```
cleartool lock vob:\libpub  
Locked versioned object base "\libpub".
```

5. **Stop ClearCase.** On the ClearCase LT server, open Control Panel. In the ClearCase program, click **Shutdown ClearCase**.
6. **Copy the VOB storage directory.** Make sure that the target location exists and is writable. Then, copy the entire VOB storage directory tree to the new location.

```
C:\vobstore> ccase-home-dir\etc\utils\ccopy libpub.vbs \vobstore2\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. Use of **ccopy** effectively grants 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):

```
scopy libpub.vbs w: /o /s
```

7. **Restart ClearCase.** In Control Panel on the ClearCase LT server, open the ClearCase program; then click **Start ClearCase**.
8. **Unlock the VOB.**

```
cleartool unlock vob:\libpub  
Unlocked versioned object base "\libpub".
```

9. **Ensure that the old VOB cannot be reactivated.** Remove it from the ClearCase storage registries. In the ClearCase Administration Console, you can use the VOB Tags node for the tag's regions to remove these VOB-tags. You can also use these commands:

```
cleartool unregister -vob \\sol\vobstore\libpub.vbs
cleartool rmtag -vob -all \libpub
```

10. **Make a new registry VOB object and VOB-tag for the VOB.** Use the ClearCase Administration console or the following commands:

```
cleartool register -vob \\sol\vobstore2\libpub.vbs
cleartool mktag -vob -tag \libpub \\sol\vobstore2\libpub.vbs
```

11. **Delete the old VOB storage directory.** If you did not overwrite the existing VOB storage directory, delete it after you verify that the VOB can be accessed at its new location.
12. (If applicable) **Enable VOB snapshots.** If you want to enable VOB database snapshots, run the `vob_snapshot_setup` command and supply the appropriate parameters.

11.2 Moving a VOB on UNIX

This section explains how to move a VOB to another disk partition on a ClearCase LT server running UNIX.

WARNING: When moving a VOB or view storage directory, use copy or backup software that preserves ownership information.

For clarity, this section uses an example:

- The current location of the VOB storage directory to be moved is `/vobstore/libpub.vbs`, on a host named `sol`.
- The VOB is known to ClearCase LT clients as `/libpub`.
- The new location for the VOB storage directory is `/vobstore2/libpub.vbs`.

The UNIX Move VOB Procedure

To move the VOB:

1. **Tell users not to access the VOB.** This step is optional. Moving a VOB from which a view is being loaded or updated does not affect VOB data integrity, but could result in an incomplete load or update operation.
2. (If applicable) **Disable VOB snapshots on the ClearCase LT server.** If VOB database snapshots are enabled on the VOB, disable them with `vob_snapshot_setup rmparam /proj/libpub`.
3. **Back up the VOB storage directory.** Use the procedures described in Chapter 9, *Backing Up and Restoring VOBs*.
4. **Lock the VOB.** Do this as the `root` user.

```
rlogin sol -l root
Password: <enter root password>
# cleartool lock vob:/proj/libpub
Locked versioned object base "/proj/libpub".
```

5. **Move 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.

```
# cd /vobstore
# tar -cf - libpub.vbs | ( 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.
6. **Ensure that the old VOB cannot be reactivated.** Remove it from the ClearCase storage registries.

```
# cleartool rmtag -vob -all /proj/libpub
# cleartool unregister -vob /vobstore/libpub.vbs
```

7. **Terminate the old VOB's server processes.** 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 **libpub.vbs**. Use **kill(1)** to terminate any such processes.
8. **Make a new registry VOB object and VOB-tag for the VOB.** Use the ClearCase Administration console or the following commands:

```
# cleartool register -vob /net/ccsvr04/vobstore2/libpub.vbs  
# cleartool mktag -vob -tag /libpub /net/ccsvr04/vobstore2/libpub.vbs
```
9. **Unlock the VOB.**

```
# rlogin ccsvr04  
Password: <enter root password>  
# cleartool unlock vob:/libpub  
Unlocked versioned object base "/libpub".
```
10. **Free any old or unneeded VOB storage.** If you did not reuse the existing VOB storage directory in Step #5, delete it to free the disk space it consumes.
11. (If applicable) **Enable VOB snapshots.** If you want to enable VOB database snapshots, do so with **vob_snapshot_setup modparam**, supplying the appropriate parameters.

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 Unified Change Management *Process 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.

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 90.) 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**.

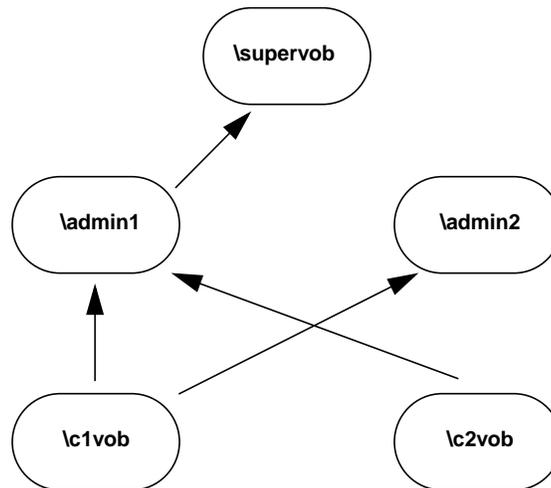
- Attempts to acquire eclipsing local copies and eclipsing ordinary types. **checkvob** queries for confirmation unless you specify **-force**. If a type being acquired is locked, the locks are discarded if there is a lock on the global type. If there is no lock on the global type, the lock information 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 "\clvob"
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 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:

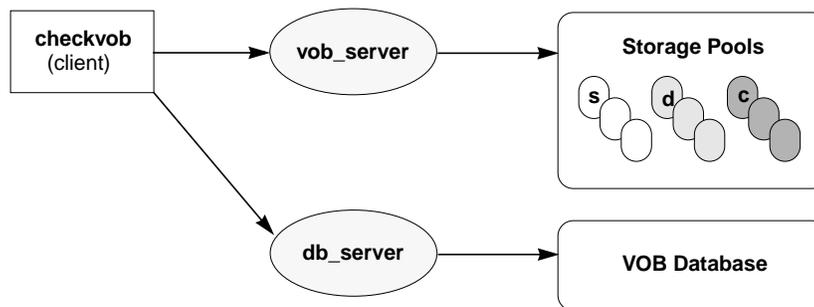
- It removes references to versions that are confirmed missing by **checkvob** and accepted as missing by the user (interactive **yes** or **-force -fix** option).
- After successfully reconstructing a missing container, in whole or part, **checkvob** adjusts the database to reference the newly reconstructed 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 15, the **vob_server**, which runs as the VOB owner on UNIX and as the `NT Authority\system` identity on Windows NT, is the only process that can create or delete data containers. Similarly, the **db_server** mediates all VOB database access.

Figure 15 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 151*).

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 16 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 16 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 17 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 17 checkvob Output Log: Condensed Transcript File

<pre> ===== Starting "source pool" processing at 11-Oct-98.18:16:50 ... Processing of misprotected containers is: ENABLED Processing of ndata containers is: ENABLED Processing of unreferenced containers is: ENABLED Fix processing mode: ENABLED ... Poolkind transcript log: .\11-Oct-98.18:16:50\poolkind_source\transcript ===== </pre>	<p>Header</p> <ul style="list-style-type: none"> VOB ID Pool ID Problem processing Output log dir
<pre> ===== Checking the installed type managers... ===== </pre>	<p>Check type managers</p>
<pre> ===== Checking "source pool" pool root storage areas... ===== </pre>	<p>Check pool roots</p>
<pre> ===== Starting "source pool" Referenced Container Checking at 11-Oct-98.18:16:52. ... ----- Container problem report: \vob_src\foo.c@ s\sdf\3d\38\0-77857609cb6711cfb17808009935288-2r (missing) ----- ... 1 containers checked; 1 broken (1 missing, 0 misprotected) ===== </pre>	<p>Find missing and misprotected containers</p> <ul style="list-style-type: none"> (optional: -data, -prot) Header, start-time Progress messages Check summary End-time
<pre> ===== Starting "source pool" Referenced Container Fixing at 11-Oct-98.18:16:53. There were missing and / or misprotected containers detected. This processing phase will attempt to correct those problems. ... Fix Processing Summary: ... ===== </pre>	<p>Fix missing and misprotected containers</p> <ul style="list-style-type: none"> (optional: -fix) Header, start-time Individual obj processing Fix summary
<pre> ===== Starting "source pool" Unreferenced Container Checking at 11-Oct-98.18:16:54. ... Check Processing Summary: ... ===== </pre>	<p>Find debris</p> <ul style="list-style-type: none"> (optional: -debris)
<pre> ===== Starting "source pool" Unreferenced Container Fixing at 11-Oct-98.18:16:55. ... Fix Processing Summary: ... ===== </pre>	<p>Fix debris</p> <ul style="list-style-type: none"> (optional: -fix)
<pre> "source pool" Processing Summary: ... The VOB's source pools are healthy. ===== </pre>	<p>Pool kind summary</p> <ul style="list-style-type: none"> Processing times Type manager status Pool root status Referenced containers Unreferenced containers Pool kind overall health

<pre> ===== Starting "source pool" processing at 11-Oct-98.18.16.50 ... Processing of misprotected containers is: ENABLED Processing of ndata containers is: ENABLED Processing of unreferenced containers is: ENABLED Fix processing mode: ENABLED ... Poolkind transcript log: checkvob.11-Oct-98.18.16.50\poolkind_source\transcript ===== </pre>	<p>Header</p> <hr/> <p>VOB ID Pool ID Problem processing Output log dir</p>
<pre> ===== Checking the installed type managers... ===== </pre>	<p>Check type managers</p> <hr/>
<pre> ===== Checking "source pool" pool root storage areas... ===== </pre>	<p>Check pool roots</p> <hr/>
<pre> ===== Starting "source pool" Referenced Container Checking at 11-Oct-98.18.16.52. ... ----- Container problem report: \job_src\foo.c@@ s\sdft\3d\38\0-77857609cb6711cfb17808009935288-2r (missing) ----- ... 1 containers checked; 1 broken (1 missing, 0 misprotected) ===== </pre>	<p>Find missing and misprotected containers</p> <hr/> <p>(optional: -data, -prot) Header, start-time Progress messages Check summary End-time</p>
<pre> ===== Starting "source pool" Referenced Container Fixing at 11-Oct-98.18.16.53. There were missing and / or misprotected containers detected. This processing phase will attempt to correct those problems. ... Fix Processing Summary: ... ===== </pre>	<p>Fix missing and misprotected containers</p> <hr/> <p>(optional: -fix) Header, start-time Individual obj processing Fix summary</p>
<pre> ===== Starting "source pool" Unreferenced Container Checking at 11-Oct-98.18.16.54. ... Check Processing Summary: ... ===== </pre>	<p>Find debris</p> <hr/> <p>(optional: -debris)</p>
<pre> ===== Starting "source pool" Unreferenced Container Fixing at 11-Oct-98.18.16.55. ... Fix Processing Summary: ... ===== </pre>	<p>Fix debris</p> <hr/> <p>(optional: -fix)</p>
<pre> ===== "source pool" Processing Summary: ... The VOB's source pools are healthy. ===== </pre>	<p>Pool kind summary</p> <hr/> <p>Processing times Type manager status Pool root status Referenced containers Unreferenced containers Pool kind overall health</p>

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 above in *Individual File Element Processing*.
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 NT

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* 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 151* 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\sdf\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

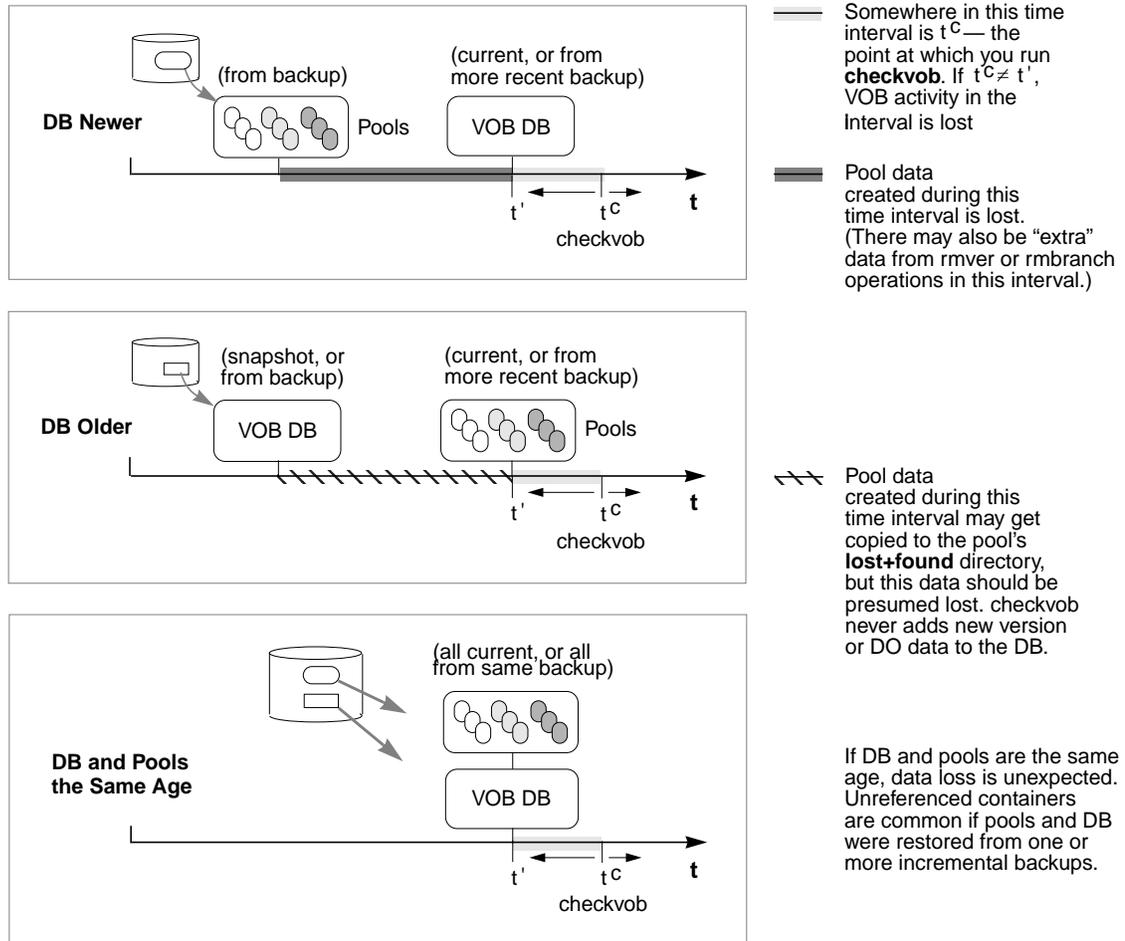
Fix Processing

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 18) illustrate some general guidelines for using **checkvob**.

Figure 18 Common Scenarios in VOB Database or Storage Pool Synchronization



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 88. 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 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**.

The ClearCase export/import utilities 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. **relocate** 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.)
- Move the current directory or the VOB root directory.

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:

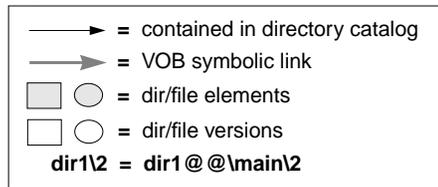
dir1\1 = **dir1@@\main\1**

Figure 19 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 19 Elements Cataloged by Directory Versions Before Relocate Operation

Key



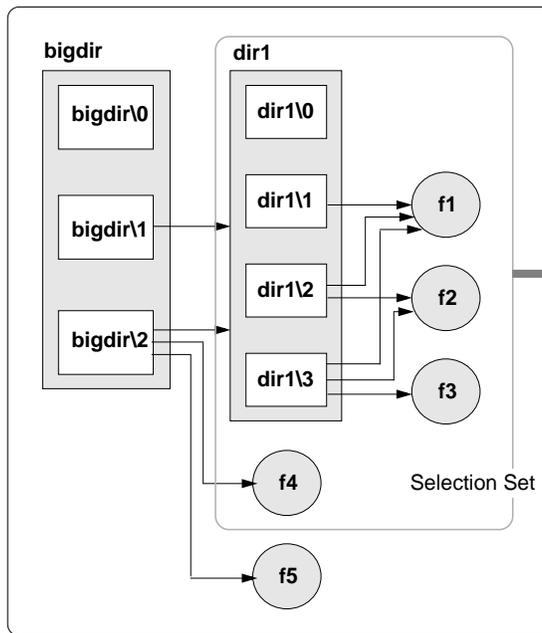
Directory cataloging

Since directory versions catalog elements, not versions, file versions are not shown. File element version trees are moved, intact, to the new destination VOB.

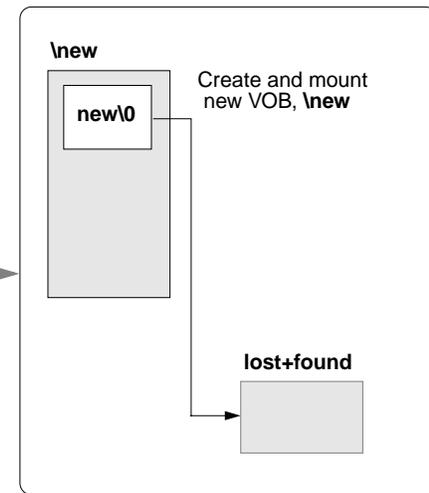
VOBs before command:

```
cleartool relocate dir1 f4 \new
```

“From VOB” — \bigdir



“To VOB” — \new



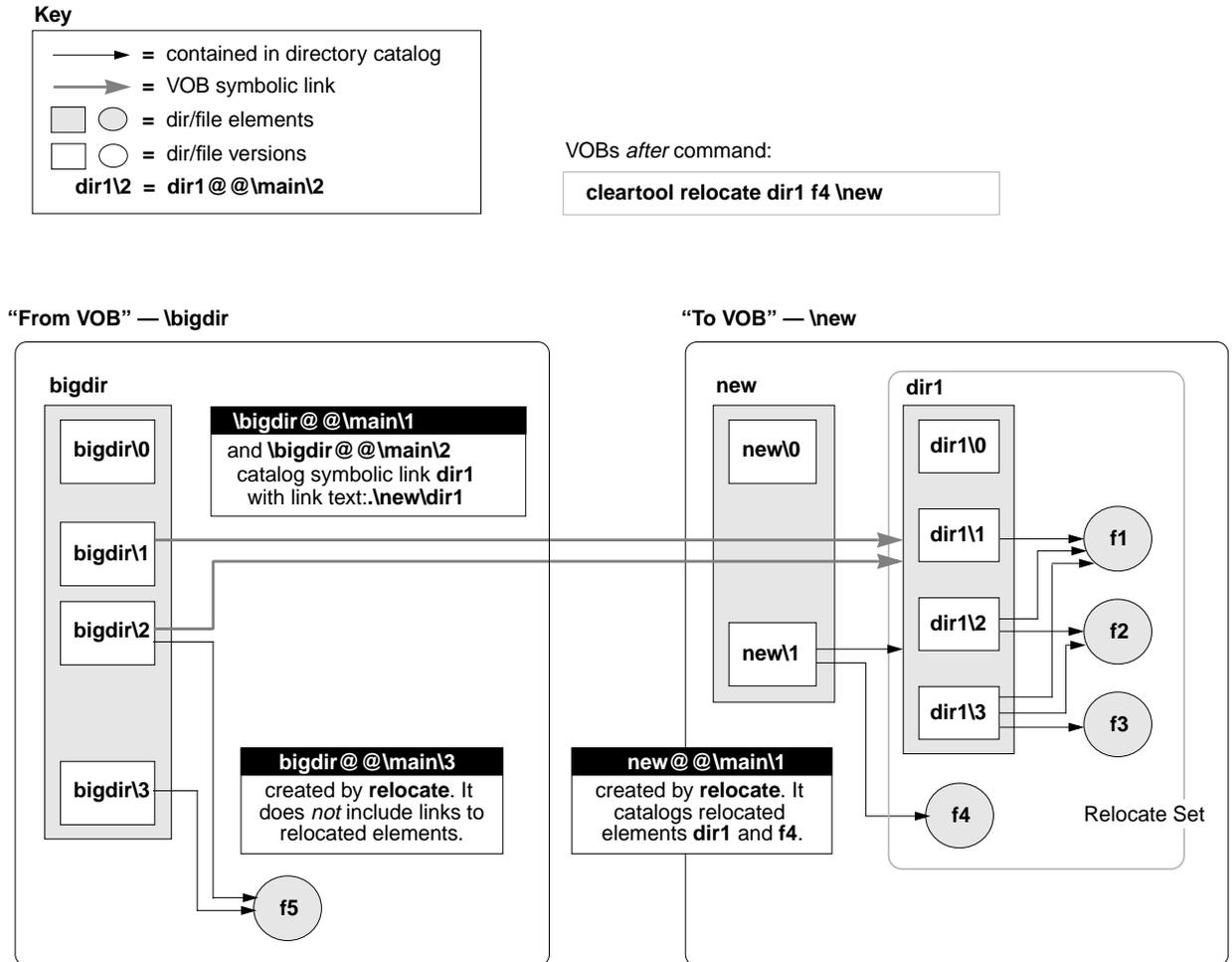
Version 0

on a directory element branch catalogs no elements

The **relocate** command defines a *selection set* that includes **dir1**, **f1**, **f2**, **f3**, and **f4**.

Figure 20 shows the VOBs after **relocate** runs.

Figure 20 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 20) 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.

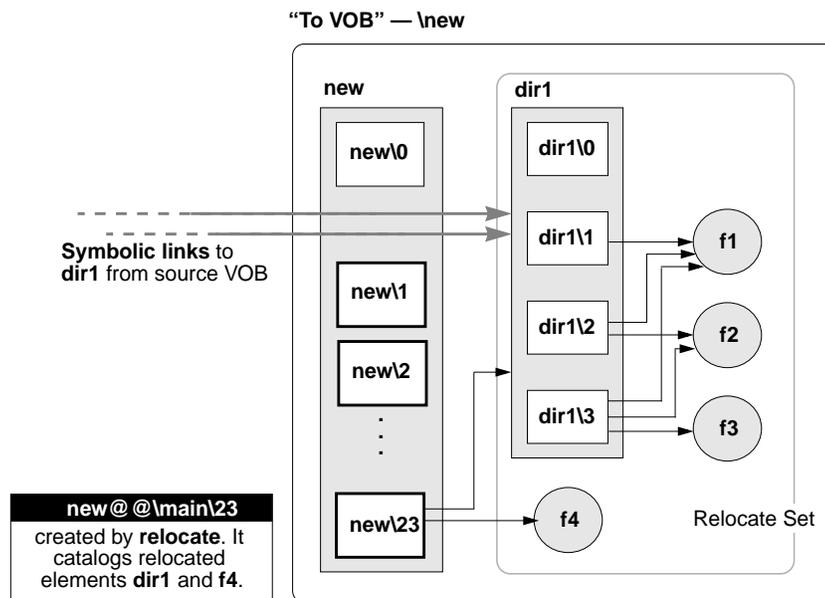
relocate checks in a new version of each relocated element’s containing directory (**bigdir@@\main\3** in Figure 20). The new directory version created by **relocate** does not catalog

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 177 for some reasons to avoid relying too heavily on VOB symbolic links. See *Updating Directory Versions Manually* on page 180 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 20 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 21 shows the destination VOB from Figure 20, expanded to include multiple versions of the target directory.

Figure 21 Destination VOB That Includes Multiple Versions

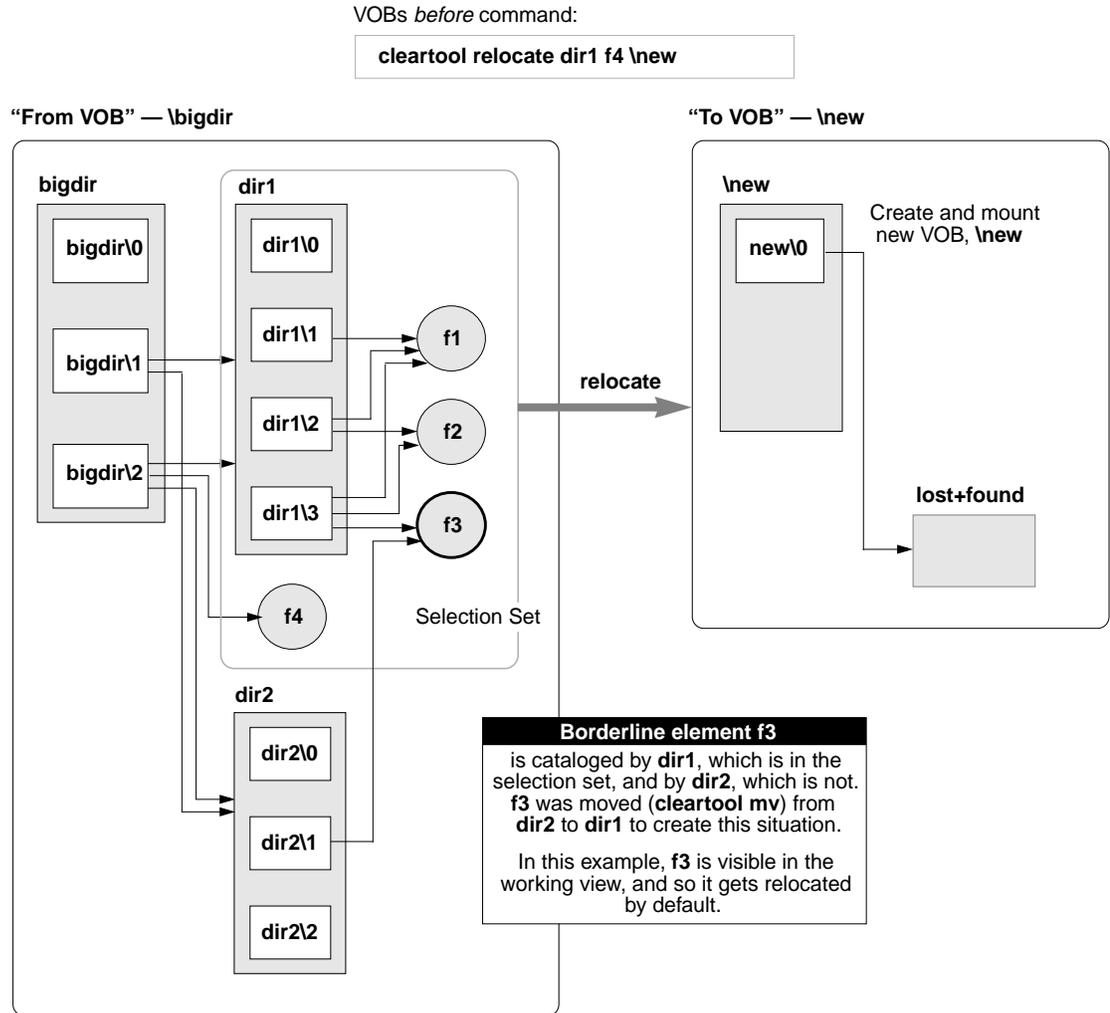


See *Updating Directory Versions Manually* on page 180 for information on adding symbolic links to relocated elements where circumstances warrant it.

Relocating Borderline Elements

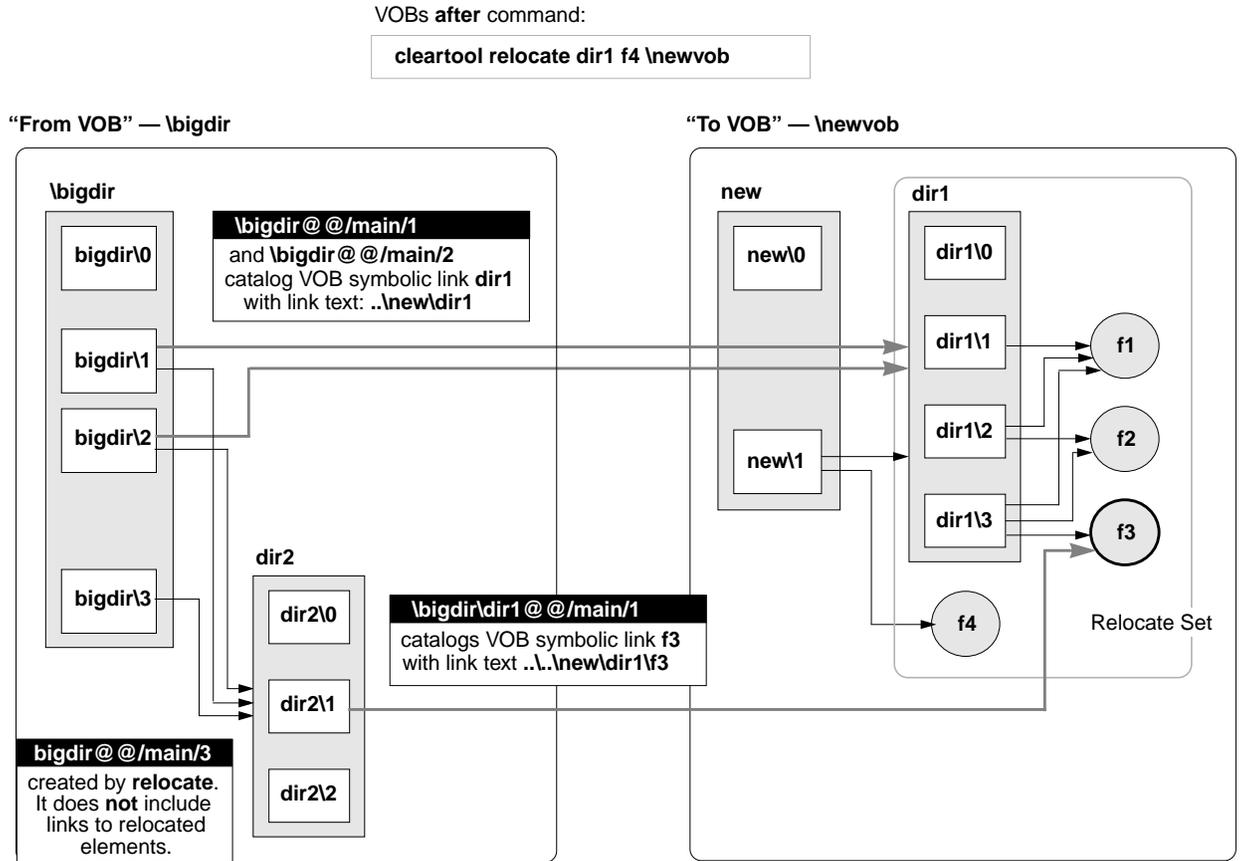
Figure 22 illustrates the relocate operation shown in Figure 19 and Figure 20, but this time the source VOB includes a *borderline element* (**f3**). This element is cataloged both in a version of a directory in the selection set and in some version of a directory outside the selection set (in a directory that stays behind). (See key in Figure 19 on page 168.)

Figure 22 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 23.

Figure 23 Source and Destination VOBs with Borderline Element Relocated

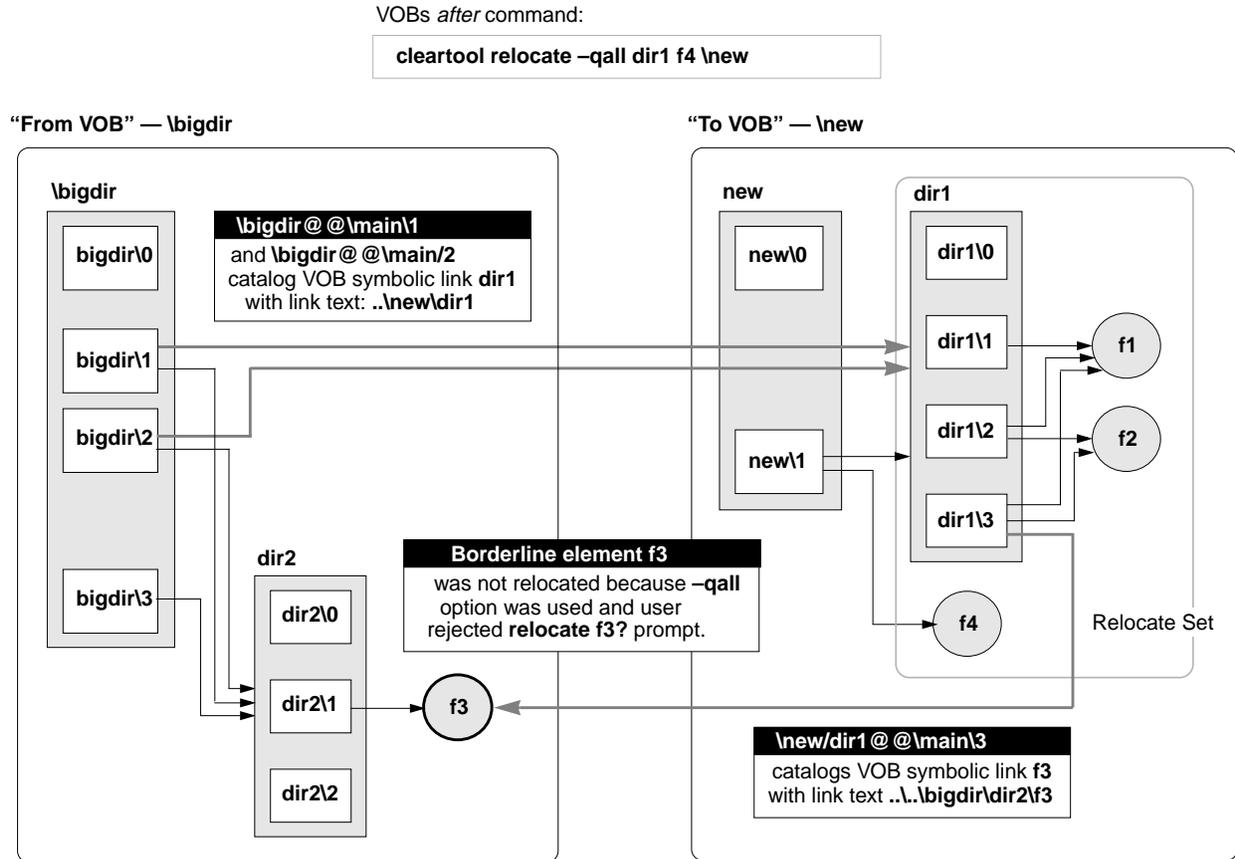


Element **f3** is relocated, and the directory version in the source VOB that referenced it (**dir2@ @\main\1**) is updated with a VOB symbolic link to its new location.

Figure 24 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 24 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 177.

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:

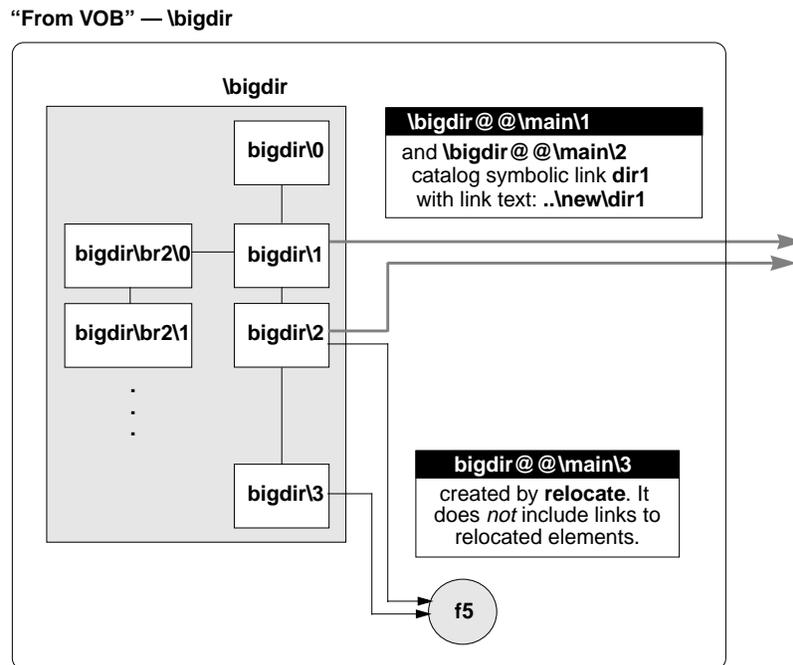
- **Use the view that was used for the relocate operation.** Before adjusting config specs, modifying build rules, and so on, activate the same view, or the same config spec, that was in effect at relocate time.

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 25 shows the source VOB from Figure 20, 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 19 on page 168.)

Figure 25 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 178 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 NT 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 21 on page 170, **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**.

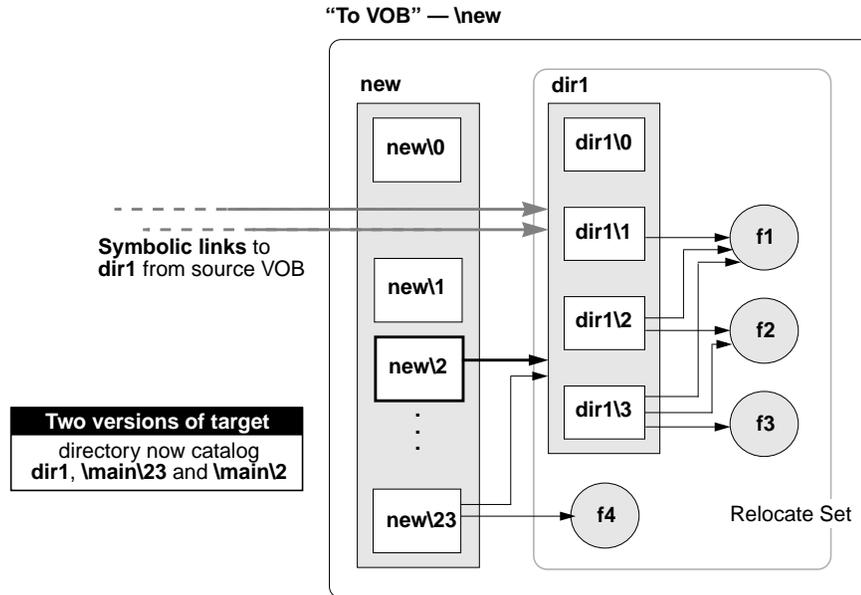
In a case like this, you can manually update previous versions of **relocate**'s destination directory to catalog relocated elements. For example, to add **dir1** to directory version **\new@@\main\2**:

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 26 shows the effect of the **cleartool ln -nco** command. (See the key in Figure 19 on page 168.)

Figure 26 Destination VOB After Modifying Old Version of Destination Directory



Modifying Newest Version of Source Directory to See Relocated Elements

In Figure 20 on page 169 and Figure 23 on page 173, 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 20) 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

This chapter describes ClearCase LT views. 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 elements and use other file-system objects that are outside ClearCase 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 **view** 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 database embedded in ClearCase. 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 scheduler runs by default generates this data periodically.

Each view has an associated **view_server** process, which runs on the ClearCase LT. 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*.

15.3 View Storage Areas

View storage must be created in a ClearCase 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 NT 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.

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, 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) must take pains to ensure that, for each 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 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 187.

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(1)** 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(1)** 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 following 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, ClearCase 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 **setsite** reference page in *ClearCase Reference Manual*.

You can create a view in any of three text modes:

- **transparent text mode.** In a view created in **transparent** text mode, ClearCase 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, deselect 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 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 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 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 the following 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 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 filenames 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 commands that take a filename 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 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

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 via 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 200 and *Managing Jobs* on page 202.

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 NT, 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 200.

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 202.

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 NT.
- The standard input stream is closed.
- 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 NT and for *root* on UNIX. In addition, on Windows NT, **ATRIAHOME** is set to *ccase-home-dir*.

16.2 The Default Schedule

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 (see the **errorlogs_ccase** reference page).
- 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 in *ClearCase Reference Manual* 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 199). 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 in the **schedule** reference page in *ClearCase Reference Manual*. 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 203. 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 in the **schedule** reference page in *ClearCase Reference Manual*.

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 206.

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 206. 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 NT 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 208), 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 208.

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 200.
- 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 204.
- Job-related events that trigger e-mail notifications and recipients for the notifications. See *Specifying Job Notifications* on page 205.
- 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 in the **schedule** reference page in the *ClearCase Reference Manual*. You cannot specify any read-only job properties, such as **LastCompletionInfo**. The job runs in the environment described in *Job Execution Environment* on page 199.

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 in the **schedule** reference page in *ClearCase Reference Manual*.

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 NT, 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 in the **schedule** reference page in *ClearCase Reference Manual*.

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 in the **schedule** reference page in *ClearCase Reference Manual*.

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 in the **schedule** reference page in *ClearCase Reference Manual*. 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 199.

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 NT domain for ClearCase LT servers running Windows NT and an NIS domain for ClearCase LT servers running UNIX. Each group and user is qualified by a domain name. In a Windows NT 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 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 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 in the **schedule** reference page in *ClearCase Reference Manual*.

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 Netscape. 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 ClearCase 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 NT computers)

- ▶ If the Web server on the system to be configured is a Netscape server, the configuration requires the name of the Netscape administration server. The URL for the Netscape administration server is identified during installation of the Netscape server.

Netscape 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 NT computers and **/usr/atria** on UNIX computers. This location is denoted by *ccase-home-dir* in the configuration instructions.

ClearCase installation directory:_____

Although the 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 NT 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 that contains ClearCase instead.)

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 view directories used by the Web interface to be created.

- ▶ 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, add the following line to 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.

17.2 Configuring the Web Server

Each of the supported Web servers has a different configuration mechanism.

Note that a Windows NT 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 Netscape Web servers; the instructions below for configuring an Apache Web server include this setup. If you start a Windows NT 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 NT does not recognize backslashes (\).
- The **ScriptAlias** directive must precede the **Alias** directive.

For example, on a Windows NT 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 NT, configure the Apache Web server as a Windows NT service. To do so, execute the **Install Apache as Service** entry in the Apache folder which was added to the **Start** menu during installation.

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.

To configure the ClearCase Web Interface in IIS:

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 NT **Challenge/Response** authentication allows Internet Explorer clients, if they are running on Windows NT or Windows 98 and logged on to a Windows NT 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. Windows NT Basic authentication must be enabled, so it can be used when needed.

9. Click OK to close the **Authentication Methods** dialog box. Then click OK to close the **Properties** sheet.

IIS is now configured for ClearCase Web access.

Netscape Enterprise Server

Configure the Netscape Web server by connecting to the Netscape administration server. The procedure is the same for both Windows NT and UNIX. Connect to the URL for the administration server with a browser.

To configure the ClearCase Web Interface in Netscape:

1. Select the server to configure by clicking the button labeled with the server name.
2. In the row of buttons at the top of the page, click **Programs**.
3. Select the **CGI Directory** entry at the left of the page.
4. 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.
5. Click **OK**, and then click **Save and Apply**. Close the message box.

6. In the row of buttons at the top of the page, click **Content Management**.
7. Click **Additional Document Directory** at the left of the page.
8. Enter the value of *ccbase-url* in the URL prefix box, and enter *ccase-home-dir/web* in the **Map To Directory** box.
9. Click **OK**, and then click **Save and Apply**. Close the message box.

The Netscape server is now configured for ClearCase Web access.

Configuring Integrations with Microsoft Web Authoring Tools

18

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, 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 supported server host for Webs under source control runs on a Windows NT version 4.0 or Windows 2000 Server with Internet Information Server (IIS). On Windows NT Workstation 4.0, IIS is called Personal Web Server (PWS); on Windows 2000 Workstation, the Web server is named

IIS. The integration with PWS or IIS on Windows 2000 Workstation is suitable for demo and evaluation purposes, but is not supported for production use.

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>Rational ClearCase LT Client >Integrations >Web Authoring Integration Configuration**.
 - > For ClearCase LT Server, click **Start>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 27 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 226.) 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 27.

Figure 27 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:

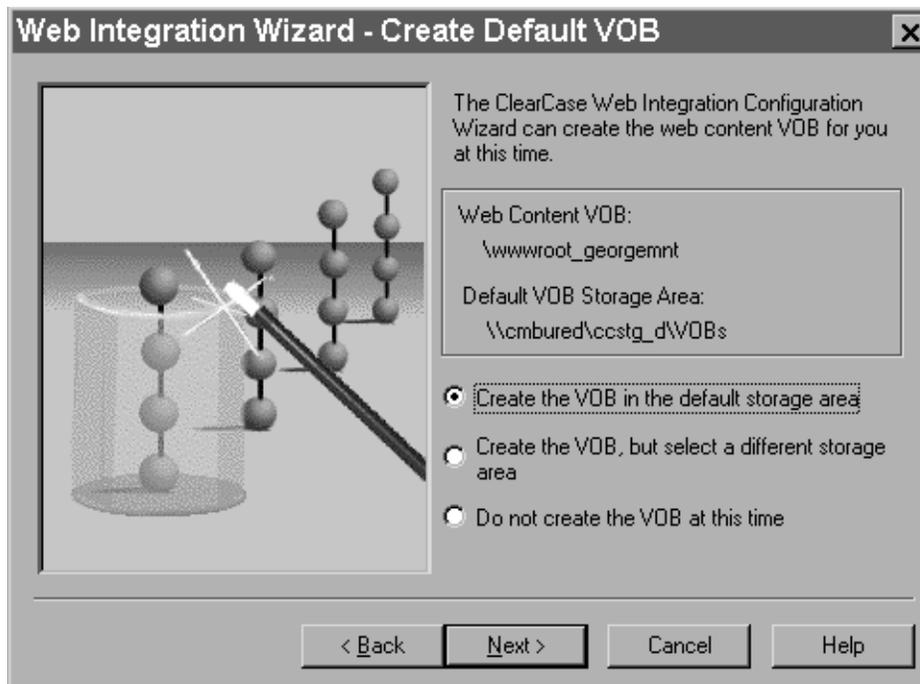
- ▶ For ClearCase LT client, click **Start>Rational ClearCase LT Client >Integrations >Web Authoring Integration Configuration**.
- ▶ For ClearCase LT server, click **Start>Rational ClearCase LT Server >Integrations >Web Authoring Integration Configuration**.

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.

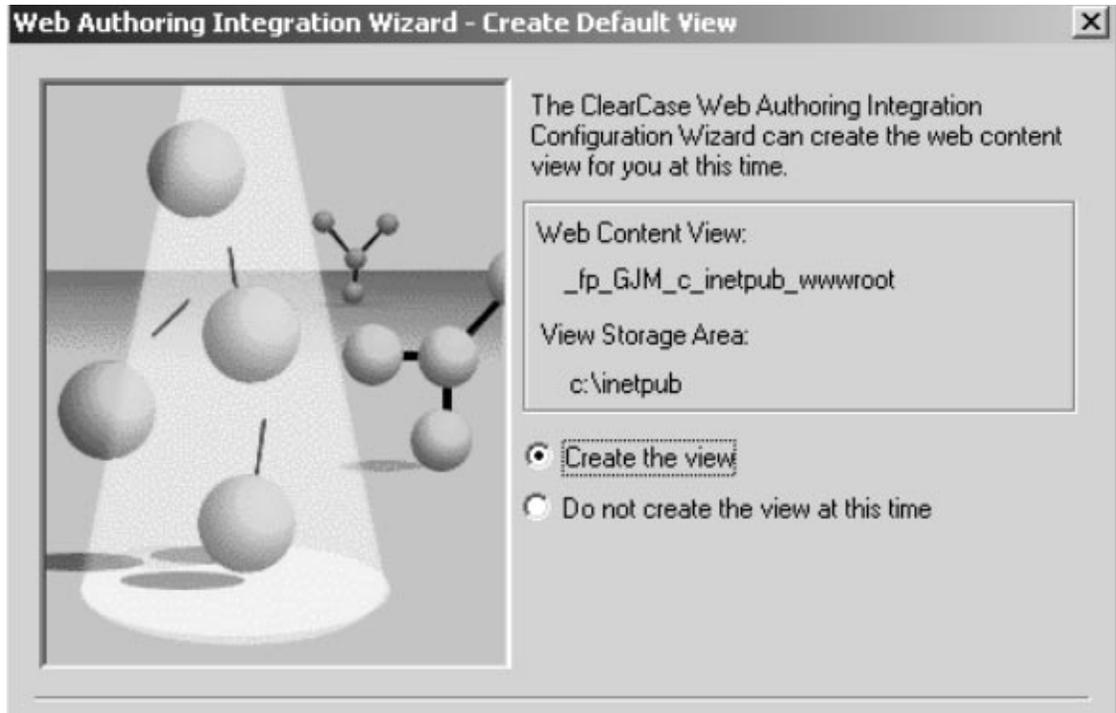
After the Web Authoring Integration Configuration Wizard checks the versions of IIS and server extensions on the Web server, you are prompted to create a content area VOB. You can accept the default VOB storage location, as in Figure 28, or use the wizard to select another VOB storage location.

Figure 28 Setting Up the VOB Storage for the Integration



The wizard then prompts you to create a content area view, as in Figure 29.

Figure 29 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 30); in Visual InterDev, they are locks (Figure 31).

Figure 30 FrontPage Source Control Icons

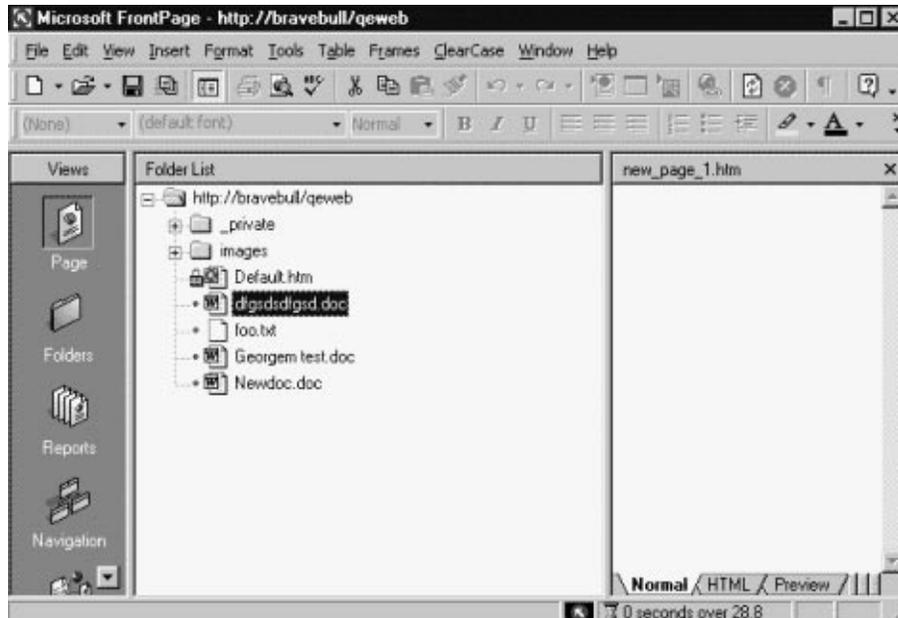
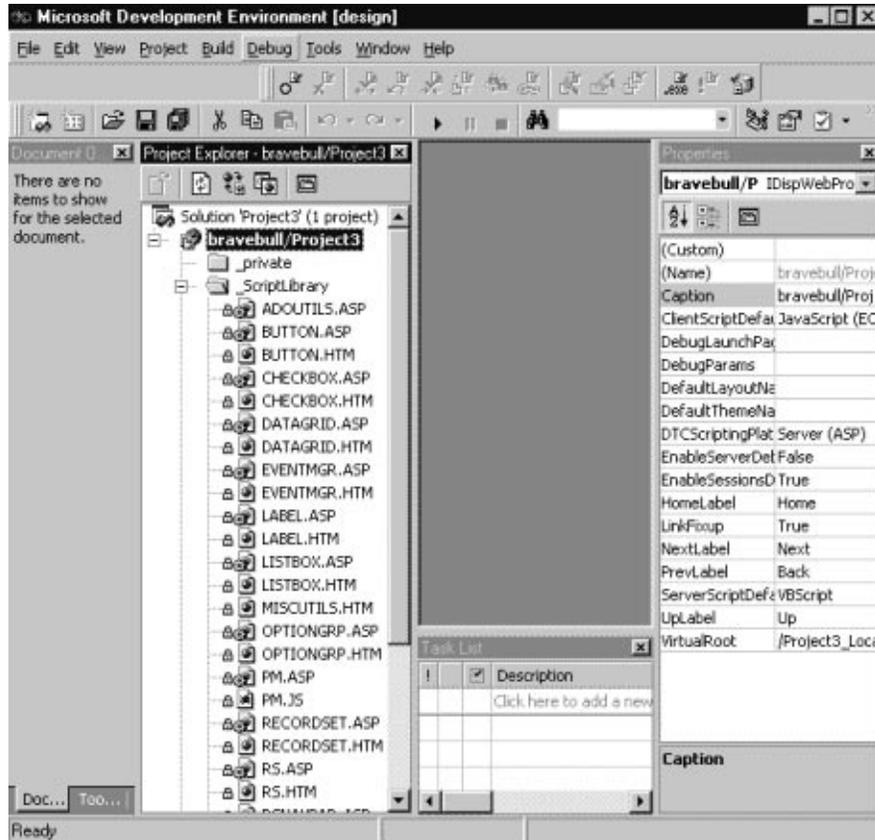


Figure 31 Visual InterDev Source Control Icons



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

Improving ClearCase LT Server Performance

19

This chapter presents techniques for improving ClearCase performance 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*.

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 NT.
- **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 striping and mirroring.

19.3 Add Memory for Disk Caching on Windows NT

Windows NT 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* on page 49.

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.

Troubleshooting

Repairing VOB and View Storage Directory ACLs on Windows NT

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 does not have a 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

ClearCase establishes ACLs for VOB and view storage directories when VOBs and views are created. These ACLs have a particular form that ClearCase 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_admin:(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_WEST\clearcase:(CI)F (clearcase group)
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_admin:(OI)(IO)(special access:) (VOB owner)
    DELETE
    WRITE_DAC
    WRITE_OWNER
    GENERIC_READ
    GENERIC_WRITE
    GENERIC_EXECUTE
```

```
NT_WEST\clearcase:(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 251.

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 251.

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 251. For more information about the options to the **checkvob** command, see the **checkvob** reference page.

20.3 Utilities for Fixing Protection Problems

This section discusses the **fix_prot** and **lsacl** utilities. The **fix_prot** program repairs permissions on ClearCase VOB and view storage directories. The **lsacl** program displays DACLs for file-system objects.

fix_prot

ClearCase includes the utility *ccase-home-dir\etc\utils\fix_prot.exe*.

```
fix_prot [-r [-type { d | f }]]  
          { -root -chown login-name -chgrp group-name  
            | [-chown login-name] [-chgrp group-name] [-chmod permissions] }  
          pname ...
```

Options:

- r**
Recursively fixes protections.
- type**
Specifies the type of file-system objects. Use **d** for directories and **f** for files.
- root**
Specifies that the pathname is a VOB-storage directory root. Causes the command to create the **identity.sd** and **groups.sd** 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).

NOTE: Current protections cannot be displayed. Thus, at least one of the **-chown**, **-chgrp**, or **-chmod** options is always required.

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 the object is in the VOB's source or derived object pool, please fix it  
with "cleartool checkvob -protections -pool".  
For other VOB objects, please fix it with "cleartool protectvob".  
Otherwise, please fix the storage directory with  
"%ATRIAHOME%\etc\utils\fix_prot".  
fix_prot: Error: Unable to change protection on "foo": Operation not  
permitted.
```

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

ClearCase includes the utility *ccase-home-dir\etc\utils\lsacl.exe*, which displays an NTFS file-system object's DACL.

```
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 247.

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

A

access control

See also permissions

- about 13
- algorithm used 18
- ClearCase permissions 5
- groups and group lists 4
- locking VOBs 129
- locks 24
- protection mode scheme 16
- protection problems, fixing (procedure) 251
- storage directories 25
- to VOBs for nongroup members 55
- views and contents 25
- VOBs and contents 19

access to ClearCase

- consistent user data 5
- non-ClearCase hosts 10

ACLs

- for storage directories 245
- managing for scheduler 208

Administration Console 9

administrative VOBs

- checking and fixing global types 134
- creating (procedure) 72
- removing 77
- restrictions 75
- unavailable 76

albd_server process

- how started 7

Apache server, configuring 216

B

backing up views 187

backing up VOBs

- about 87
- checkvob run on incremental backup 163
- finding pathname 92
- incremental backups 94
- locking the VOB 92
- partial backups 93

semi-live vs. standard procedure 90

storage pools 90

summary of procedure 89

tools for 4

vob_snapshot and 90

backup tools, recommendations 87

block buffer cache

how used 240

statistics 241

building software

when VOBs are locked 90

C

caches

cleartext storage pools 46

checkvob utility

broken hyperlinks 134

check/fix scenarios 159

detecting file protection problems 248

diagnostic uses of 144

force fix mode 151

log files 145

requirements for type manager 144

synchronizing database with storage pool 110

using -force -fix options 144

when to use 133

checkvob utility, sample runs

about 162

database newer than pools 162

database older than pools 162

incremental backup/restore 163

clearexport_rcs command 57

clearexport_ssaf command

conversion example 63

cleartext pools

about 46

backing up 94

scrubbing 121

client/server processing 6

D

DACLs, about 245

data containers

DO 46

examining with checkvob 133

fixing inconsistencies in 143

missing 156

types of 45

data loss, VOB restored from backup 91

db_server process 7, 44

directories

relocated, how cataloged in source VOB 169

relocated, how cataloged in target VOB 170

removing, effect on file elements 114

symbolic links to relocated 180

disk buffer cache, how used 240

disk space

conserving, and versions 122

required for VOB host 49

semi-live backup 91

DO pools

about 46

incorrect permissions 154

missing containers 159

documentation

online help description xxiii

domains

about 27

configuring 27

creating (procedure) 29

security requirements 28

setting up user accounts 30

user accounts across multiple 33

user accounts and groups 30

DOs (derived objects)

checkvob processing 150

data containers 46

E

elements

access control scheme 21

checkvob processing 150

directory, removing 114

moving to another VOB 165

moving to another VOB (illustrations) 166

relocating borderline 171

relocating, preliminary procedures 174

removing from VOBs 122

restoring removed 114

error logs 7

error logs, for servers 7

event records, scrubbing 121

F

FAT file system

converting to NTFS, security information 248

security of storage directories 245

feature levels 69

file descriptor table

modifying for VOB host 52

file elements

editing permissions, and ClearCase protections 248

location of source data containers 45

fix_prot utility 249

floating license scheme 37

G

global types

about 71

changing protection 82

changing scope 84

checking and fixing 134

copying 83

creating 78

describing 80

how they work 71

listing 80

listing history 81

locking and unlocking 82

removing 85

renaming 84

group IDs, consistency on ClearCase hosts 5

group lists

role in access control 4

groups

adjusting VOB identity information 55

H

hosts

See also license server hosts; registry server hosts; VOB hosts

consistency of user IDs 5

release, renaming 11

verifying domain assignment 29

hyperlinks, checking 134

I

.identity subdirectory
contents of 44

IIS, configuring 217

importing data to VOBs

about 56
from PVCS (example) 58
from SourceSafe (example) 60

init command, ClearCase startup script 8

installing ClearCase, changing release area location 11

L

licenses

adding information to database (procedure) 39
licensing scheme 37

locking VOBs

locking objects in 24
procedure 92
techniques to reduce duration 92

log files, checkvob 145

lost+found directory

files in deleted directory 114

lsacl utility 250

M

memory

required for VOB host 49

Microsoft Internet Information Server, configuring 217

moving VOBs

danger when moving database 44
moving elements to other VOBs 165
same architecture (procedure) 126

N

Netscape Enterprise Server, configuring 218

network

ClearCase components in 1
Windows NT domains 27

non-ClearCase hosts

access to VOBs 10

NTFS file system

converting from FAT, security information 248
security of storage directories 245

O

online help, accessing xxiii

P

performance

disk I/O on VOB servers 240
VOB hosts, improving 239
VOB memory and storage required 49

permissions

See also access control
editing, and ClearCase protections 248
how controlled 5
incorrect, on pools 154

pools

See storage pools

primary group

overriding (procedure) 14, 32
setting for user account (procedure) 31
verifying (procedure) 32

principal group

role in access control 4

process table

modifying for VOB host 52

processes

extraneous, on VOB host 239

protection mode 16

public VOBs

creating 54

PVCS, data conversion example 58

R

registry server hosts

about 6
VOB-tag password file location 54

release area, changing location 11

release hosts

about 3
renaming 11

relocate utility

about 165
sample operations (illustrations) 166

restoring VOBs from backup

checkvob run from incremental backup 163
costs of semi-live backup 91
database snapshot 110
procedure 95
rules and guidelines 106
sample session 96
scenarios 104
synchronizing views and VOBs 117

without vob_restore (procedure) 112

S

scheduler

- about 197
- creating jobs 203
- creating tasks 201
- default job schedule 200
- deleting jobs 207
- deleting tasks 202
- editing job properties 206
- editing task definition 202
- job notification mechanisms 205
- managing jobs 202
- managing tasks 200
- running jobs 207
- schedule types 204
- viewing job properties 206

scrubbing

- about 4
- impact on cache hits 46
- logical vs. physical 121
- storage pools 120
- VOB databases 121

Security Descriptor

- effect of file-system conversion 248

semi-live backup

- costs and benefits 91
- how it works 90

server processes 7

servers, error logs 7

shutdown script, how invoked 8

SMB server

- See* TAS SMB server

source pools

- about 45
- checkvob sample run 162
- corrupted containers 159
- debris 158
- incorrect permissions 154
- missing containers 156
- scrubbing 120
- unreferenced containers 158

SourceSafe, data conversion example 60

splitting VOBs

- about 165
- causes of common failures 175
- cleanup 177
- cleanup guidelines 178
- element removal errors 176
- handling unrelated errors 176
- preliminary procedures 174
- sample operations (illustrations) 166

startup script, how invoked 8

storage directories

- ACLs, form of (example) 245
- native file-system permissions 25
- retaining protection information of copied 247
- security on NTFS vs. FAT 245

storage pools

- See also* cleartext pools; DO pools; source pools
- about 45
- backing up 90
- checkvob processing 151
- cleartext, backing up 94
- default directories 47
- finding and fixing problems in 142
- inconsistent with VOB database 141
- misnamed 154
- scrubbing 120
- setup mode in checkvob 152
- synchronizing with VOB database 110

storage registries

- maintenance required 4

symbolic links

- for moved elements 166
- limitations 177

T

technical support xxiv

text modes

- configuring for mixed environment 190

trust lists 27

type managers

- about 45
- checkvob requirements 144

type objects

- changing 83
- coordinating for multiple VOBs 56
- global 71
- locking 24

U

user accounts

- in multiple domains 33

V

versions

- removing from VOB 122

view database

- about 5, 186
- skew after VOB restoration 117

- view storage directories**
 - requirements 186
 - view_server process**
 - about 7
 - views**
 - about 4, 185
 - access control scheme 25
 - backing up 187
 - configuring text modes for mixed environment 190
 - limitations of symbolic links 178
 - restoring from backup (procedure) 188
 - VOB database**
 - about 44
 - contents of 43
 - danger when moving 44
 - inconsistent with storage pools 141
 - location of 44
 - proportion of cache size 241
 - scope of checkvob updating 143
 - scrubbing, logical vs. physical 121
 - scrubbing 121
 - skew after VOB restoration 117
 - synchronizing with storage pools 110
 - updating schema 69
 - VOB hosts**
 - cache size to VOB database 241
 - criteria for selecting 49
 - improving performance 239
 - kernel resource adjustments 52
 - modifying for ClearCase access 52
 - VOB storage**
 - maintenance trade-offs 119
 - vob_restore**
 - about 95
 - database snapshot 110
 - restoration scenarios 104
 - sample session 96
 - vob_server process** 7, 46
 - vobrpc_server process** 7
 - VOBs**
 - about 3
 - access control scheme 19
 - administration tasks 3
 - configuring text modes for mixed environment 190
 - coordinating type objects 56
 - creating (procedure) 53
 - identity information adjustments 55
 - importing data 56
 - locking objects in 24
 - locking 129
 - public, creating 54
 - reformatting 69
 - removing elements from 122
 - removing temporarily (procedure) 129
 - restoring to service (procedure) 130
 - size and distribution trade-offs 50
 - structure of 43
 - upgrading to new release 68
 - VOB-tags**
 - creating 54
 - password file location 54
- W**
- Web servers**
 - configuration checklist 213
 - configuration procedures 215

