ADMINISTERING CLEARCASE

Release 4.1 and later

Windows/UNIX Edition

Rational®
the e-development company™

800-023707-000

Administering ClearCase
Document Number 800-23707-000 August 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.

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. 5,574,898 and 5,649,200 and 5,675,802. 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/).

Rational®
the e-development company™

Contents

Preface xxxi
About This Manual xxxi
ClearCase Documentation Roadmap xxxii
Typographical Conventions xxxiii
Online Documentation xxxiv
Technical Support xxxiv

Administering the ClearCase Network

1. Understanding the ClearCase Network 1
1.1 Network Overview 1
1.2 ClearCase Hosts 2
1.3 ClearCase Data Storage 4
Versioned Object Bases (VOBs) 4
Views 5
1.4 ClearCase User Base 6
1.5 Registries for VOBs and Views 6
Network Regions 7
1.6 ClearCase Client/Server Processing 8
ClearCase Servers 9
Server Error Logs 10
1.7 ClearCase Startup and Shutdown 11
UNIX Startup and Shutdown 11
Windows NT Startup and Shutdown 11

2. Administering ClearCase Hosts 13
2.1 ClearCase Administration Console 13
Controlling Remote Administration 14
2.2 Using DHCP with ClearCase 15
2.3 ClearCase Data and Non-ClearCase Hosts 15
2.4 Using Non-ClearCase Access on UNIX Hosts 17
Restrictions on Use 17
Using automount with Non-ClearCase Access 18
Problems with Non-ClearCase Access: NFS Client Caching 18
Problems with Non-ClearCase Access: NFS Locking 19
2.5 Recording Multiple Network Interfaces on UNIX Hosts 20
2.6 Using automount with ClearCase on UNIX 20
Automounter Maps 21
Using the –hosts Map 21
Not Using the –hosts Map 21
Specifying a Nonstandard Mount Directory 21
2.7 Administering Networkwide Release Areas on UNIX 22
Changing the Location of the Release Area 22
Renaming a UNIX Release Host 22

3. Understanding ClearCase Access Controls 23
3.1 Fundamentals of ClearCase Access Control 23
Users and Groups 23
Setting the Primary Group, Special Considerations on Windows 24
Privileged Users and Groups 25
User Processes 25
ClearCase Objects 26
Access to ClearCase Data 29
3.2 Access Control for VOBs and VOB Objects 30
Access Control for VOBs 30
Permission to Create VOBs 30
Permission to Delete VOBs 30
Permission to Read VOBs 31
Permission to Write VOBs 31

Permission to Execute VOBs 31
Access Control for Elements 31
Permission to Create Elements 32
Permission to Delete Elements 32
Permission to Read Elements 32
Permission to Write Elements 32
Permission to Execute Elements 33
Access Control for Other VOB Objects 33
Permission to Create Other VOB Objects 34
Permission to Delete Other VOB Objects 34
Permission to Read Other VOB Objects 34
Permission to Write Other VOB Objects 34
Locks on VOB Objects 34
Locking Type Objects 35
3.3 Access Control for Views and View Objects 35
Access Control for Dynamic Views 36
Permission to Create Views 37
Permission to Delete Views 37
Permission to Read Views 37
Permission to Write Views 37
Permission to Execute Views 38
Access Control for View-Private Files 38
Permission to Create View-Private Files 39
Permission to Delete View-Private Files 39
Permission to Read View-Private Files 39
Permission to Write View-Private Files 40
Permission to Execute View-Private Files 40
Access Control for Derived Objects 40
3.4 ClearCase and Native File-System Permissions 41

4. Administering Windows NT Domains and ClearCase 43
4.1 Configuring Domains for Use with ClearCase 43
ClearCase Requirements for Users, Groups, and Domains 44

ClearCase Users and User Groups....44
ClearCase Group....44
ClearCase Server Process User....45
ClearCase Restrictions....45
4.2 Creating a Domain....46
4.3 Establishing User Account and Group Assignments....47
Setting Up Domain Accounts....47
Setting a User's Primary Group....48
Verifying a User's Primary Group....49
Overriding a User's Primary Group....49
4.4 Defining the ClearCase Server Process User and ClearCase Group....50
Creating the Account and Group Manually....50
4.5 Multiple User Account Domain Support....52
Creating Proxy Groups and Enabling Domain Mapping....52
Setting VOB Element Permissions....54
Setting VOB Storage ACLs....54
4.6 Nondomain Systems....55

5. Understanding the ClearCase Multiversion File System....57
5.1 What is the MVFS?....57
Known Limitations of the MVFS on Windows NT....58
5.2 Issues with Using the MVFS on Windows NT....59
Problems with Case-Sensitivity....59
Interaction with NFS Products....60
Using Links with the MVFS....60
Using the VOB Root Directory....60
Running Executables in the MVFS....61
5.3 MVFS Performance Issues....61

Administering a UNIX/Windows Network

6. Configuring ClearCase in a Mixed Network 65

6.1 When to Use ClearCase in a Mixed Network Environment 65

When Not to Use a Mixed Environment 66

6.2 ClearCase Capabilities in a Mixed Environment 66

Windows NT 67

Windows 95 and Windows 98 67

UNIX 67

Constraints in a Mixed Environment 68

6.3 Identifying License and Registry Servers 69

6.4 Managing User Accounts 69

Checking User and Group Assignments 69

ClearCase and Nondomain Accounts 70

UNIX VOB Group Lists and Registered User Groups 71

6.5 ClearCase Access Control on UNIX and Windows 72

6.6 Case-Sensitivity 73

General Recommendations 73

Case Considerations On UNIX 74

Case Considerations On Windows 74

When to Use Case-Sensitive Mode 75

NFS Client Products and Case-Sensitivity 76

7. Configuring UNIX File Access for Windows Computers 77

7.1 ClearCase File Server 78

Enabling and Disabling CCFS 79

7.2 NFS Client Products 79

Disabling Automatic Case Conversion 80

Microsoft SFU and Intergraph DiskAccess 80

Hummingbird NFS Maestro 80

Setting an NFS Client's Default Protection 81
Microsoft SFU or Intergraph DiskAccess 81
Hummingbird NFS Maestro 81
Setting the Correct Logon Name 82
Microsoft SFU or Intergraph DiskAccess 82
Hummingbird NFS Maestro 82
Hummingbird NFS Maestro: Disabling DOS Sharing 82
Automounting and NFS Client Software 83
Microsoft SFU or Intergraph DiskAccess: Setting Up the ClearCase Server Process User and ClearCase Group 84
Setting Up the UNIX Account 84
Preparing the Windows NT Client 85
Alternative Setup: Administrative Option 85
Microsoft SFU: Configuring the Default LAN 86
7.3 SMB Server Products 87
Upgrading from a Previous Version of TAS 87
Disabling Opportunistic Locks on ClearCase Volumes 87
TAS Upgrade Does Not Require Reconfiguration 87
Installing TAS 88
Accessing TNAS 88
Performing Initial Setup of TAS 89
General TAS Settings 89
Enabling and Configuring the CIFS Realm 89
Configuring TAS to Support ClearCase 90
Creating a TAS Username Map for clearcase_albd 90
Creating a Volume 91
Configuring the File Service 92
Start Services and Accept Service Connections 93
Configuring ClearCase to Support TAS 94
Testing the TAS Configuration 94
Testing the TAS Configuration on Non-ClearCase Files 95
Testing the TAS Configuration with ClearCase 95

8. Configuring VOB and View Access in Mixed Environments 97
8.1 Preparing the UNIX VOB or View Host 98
8.2 Creating a New Network Region 98
Assigning Computers to the New Network Region 98
8.3 Creating VOB-Tags and View-Tags in the New Region 99
Using the Region Synchronizer 99
8.4 Re-Creating an Incorrect VOB-Tag or View-Tag 99
8.5 Windows Tags for UNIX VOBs with Symbolically Linked Storage 100
Mapping Storage Pools for an Existing VOB-Tag 101
8.6 Configuring Text Modes for Views 103
Text Modes 104
Determining a View's Text Mode 105
Choosing a Text Mode for a View 105
Enabling Interop Text Mode Support in VOBs 106
Determining Whether a VOB Supports Interop Text Modes 106
Special Procedure for MultiSite Users 107

Administering VOBs

9. Understanding VOB Storage 111
9.1 Introduction to Versioned Object Bases 111
9.2 VOB Database 112
9.3 VOB Storage Pools 113
Source Storage Pools 113
Cleartext Storage Pools 114
Derived Object Storage Pools 114
The vob_server Process 115
9.4 Storage Pool Creation 115

Remote Storage Pools on UNIX....115
9.5 Elements' Source Pool Assignments....117
9.6 Tools for Working with Storage Pools....117
cleartool Subcommands....117
Utility Commands....118

10. Setting Up VOBs....119
10.1 Selecting a VOB Host....120
10.2 Planning for One or More VOBs....121
Planning for Release VOBs....123
10.3 Modifying a UNIX VOB Host for ClearCase....123
UNIX Kernel Resources....123
Optional Software Packages....124
10.4 Creating VOB Storage Locations....124
10.5 Creating a VOB....124
Linking a VOB to an Administrative VOB....126
Creating a VOB on a Remote Host....127
Adjusting the VOB's Ownership Information....127
Case 1: One Group for All VOBs, Views, and Users....127
Case 2: Accommodating Multiple User Groups....128
Ensuring Global Access to the VOB—Special Cases for UNIX....128
Guess Was Wrong, But Global Pathname Does Exist....129
Network Requires Multiple Global Pathnames....129
Enabling Setuid and Setgid Mounting of the Viewroot and VOB File Systems on UNIX Hosts....130
Creating Remote Storage Pools on UNIX Hosts....130
10.6 Coordinating the New VOB with Existing VOBs....131
10.7 Populating a VOB with Data....131
Importing Data into a UCM Project....132
Example: Importing RCS Data....132
Creating the Data File....132
Running clearimport....133
Example: Importing PVCS Data....133
Creating the Data File....133

Running the Conversion Scripts 134
10.8 Converting a SourceSafe Configuration 135
Overview of Payroll Configuration 135
Shares 136
Branches 137
Labels 137
Pins 137
Setting Your Environment 137
Setting Environment Variables 137
Setting Your SourceSafe Current Project 138
Running clearexport_ssafe 138
Using the Recursive Option 138
Example 138
Running clearimport 140
Examining the Results 141
Version Numbers 142
Labels 142
Branches 143
Pins 143
Shares 143
10.9 Upgrading a VOB to a New Release 143
Upgrading the Feature Level of a VOB 144
Reformatting a VOB 144

11. Using Administrative VOBs and Global Types 147
11.1 Overview of Global Types 147
11.2 Why Use Global Types? 148
11.3 Working with Administrative VOBs 148
Creating an Administrative VOB 148
Linking a Client VOB to an Administrative VOB 149
Administrative VOB Hierarchies 149
Listing an AdminVOB Hyperlink 150
Restrictions on Administrative and Client VOBs 151

If an Administrative VOB Becomes Unavailable 152
Using Administrative VOBs with MultiSite 152
Breaking a Link Between a Client VOB and an Administrative VOB 153
Removing the AdminVOB Hyperlink 153
Removing All GlobalDefinition Hyperlinks 154
Removing an Administrative VOB 155
11.4 Working with Global Types 155
Creating a Global Type 155
Auto-Make-Type Operations 156
Auto-Make-Type of Shared Global Types 157
Describing Global Types 158
Listing Global Types 160
Listing History of a Global Type 160
Changing Protection of a Global Type 161
Locking or Unlocking a Global Type 161
Changing Mastership of a Global Type 162
Changing the Type of an Element or Branch 163
Copying a Global Type 163
Renaming a Global Type 164
Changing the Scope of a Type 164
Removing a Global Type 166
Cleaning Up Global Types 166

12. Backing Up and Restoring VOBs 167
12.1 Choosing Backup Tools 167
UNIX Backup Issues 167
Windows Backup Issues 168
12.2 Backing Up a VOB 168
Backing Up a VOB on UNIX 168
Backing Up a VOB on Windows NT 169
Choosing Between Standard and Semi-Live Backup 170
Benefits of Semi-Live Backup 171
Costs of Semi-Live Backup 171

Enabling Semi-Live Backup 171
Determining a VOB's Location 172
Ensuring a Consistent Backup 172
Locking and Unlocking a VOB 173
Partial Backups 173
DO Pool Backup 174
Cleartext Pool Backup 175
Administrative Directory Backup 175
Incremental Backups of a VOB Storage Directory 175
12.3 Backing Up a UNIX VOB with Remote Storage Pools 176
12.4 Restoring a VOB from Backup with vob_restore 177
vob_restore: Sample Session 179
Target Prompt 180
Storage Directory Prompt 180
Snapshot Prompt 180
Backup-Loaded Prompt 180
Sample VOB Restoration Scenario 181
vob_restore: Restoration Scenarios 187
How vob_restore Determines the Scenario 188
Restoration Rules and Guidelines 189
vob_restore: In Place 189
vob_restore: VOB Is Active 190
vob_restore: Move VOB on Same Host 191
vob_restore: Move VOB to New Host 192
vob_restore: Unregistered 192
vob_restore: Restoring with a Database Snapshot 192
12.5 Restoring a VOB from Backup Without vob_restore 194
12.6 Restoring an Individual Element from Backup 196
12.7 VOB and View Resynchronization 200
Resynchronizing Views and VOBs 202
Reestablishing Consistency of a View's Derived Object State 203

13. Administering VOB Storage 205
13.1 VOB Storage Maintenance 205
Scrubbing VOB Storage Pools 207
Scrubbing VOB Databases 207
Database Scrubbing: Logical vs. Physical 208
13.2 User-Supplied Maintenance Procedures 208
13.3 Removing Unneeded Versions from a VOB 208
13.4 Creating Additional Storage Pools for UNIX VOBs 209
Caution on Remote Source Pools 210
Example: Assigning All Files in a Directory to a New Pool 210
Example: Moving an Existing Storage Pool to Another Disk 211
13.5 Adjusting Storage Pool Scrubbing 213
Scrubbing Derived Objects More Often 213
Fine-Tuning Derived Object Scrubbing 214
Scrubbing Less Aggressively 215

14. Moving VOBs 217
14.1 Restrictions on Moving a VOB 217
14.2 Special Considerations for Replicated VOBs 218
14.3 Moving a VOB on Windows NT 219
The Move VOB Procedure for Windows NT 219
14.4 Moving a VOB on UNIX (Same Architecture) 222
The UNIX Move VOB Procedure 222
14.5 Moving a VOB Between UNIX Hosts (Different Architectures) 225

15. Removing VOBs 229
15.1 Locking as an Alternative to VOB Deactivation 229
15.2 Taking a VOB Out of Service 229
Restoring the VOB to Service 230
15.3 Removing a VOB 231

16. Using checkvob 233
16.1 When to Use checkvob 233
16.2 Checking Hyperlinks 234

16.3 Checking Global Types 234
Fix Processing 234
Output Log for Global Type Checking 235
Example Check or Fix Scenario 235
16.4 Database or Storage Pool Inconsistencies 241
Updating the VOB Database 243
Requirements for Using checkvob 244
Replicated VOB Considerations 245
Running checkvob 245
Output Log for Pool Checking 246
Overview of checkvob Processing 250
Individual File Element or DO Processing 251
Pool Mode (–pool Option) Processing: Overview 252
Force-Fix Mode 252
Pool Setup Mode 253
Descriptions of Storage Pool Problems 254
Source, DO, or Cleartext Pool: Bad Pool Roots 255
Description 255
Cause 255
Fix Processing 255
Source or DO Pool: Misprotected Container on Windows NT 256
Description 256
Cause 256
Fix Processing 256
Missing and Unreferenced Data Containers 257
Source Pool: Missing Container 257
Source Pool: Unreferenced Container (Debris) 259
Source Pool: Corrupted Container 260
Description 260
Cause 261
Fix Processing 261
DO Pool: Missing Container 261
Description 261

Causes 261
Fix Processing 261
DO Pool: Unreferenced Container (Debris) 261
Description 261
Causes 262
Fix Processing 262
DO Pool: Corrupted Container 262
Description 262
Causes 262
Fix Processing 262
16.5 Sample Check and Fix Scenarios 262
Scenario 1: VOB Database Newer Than Storage Pools 263
Running checkvob 264
Scenario 2: Storage Pools Newer Than VOB Database 264
Running checkvob 265
16.6 Sample checkvob Runs 265
16.7 Database Newer Than Pools 265
16.8 Database Older Than Pools 266
16.9 Unreferenced Containers from Incremental Backup or Restore 266
16.10 Pool Root Check Failure 267
Fixing Pool Roots: Getting Started 267
Fixing Pool Roots: The Most Common Problems 267
Pool Skew Caused by Addition of New Pool 268
Pool Skew Caused by Pool Deletion 268
Pool Skew Caused by Renamed Pool 268
A More Complex Pool Skew Scenario 268
How to Re-Create a Pool's pool_id 269

17. Splitting VOBs with relocate 271
17.1 What Does relocate Do? 271
17.2 Element Relocation Illustrated 272
Cataloging in the Source VOB 275
Cataloging in the Target VOB 276

Relocating Borderline Elements 277
17.3 Before Relocating Elements 280
17.4 Common Errors During a Relocate Operation 282
Errors Not Related to Source VOB Element Removal 282
Errors During Source VOB Element Removal 283
17.5 After Relocating Elements 283
Symbolic Links 284
Upgrading Views That Rely on Symbolic Links 284
Cleanup Guidelines 285
Updating Directory Versions Manually 287
Fixing Symbolic Links Created by relocate 287
Modifying Old Target Directory Versions to See Relocated Elements 288
Modifying Newest Version of Source Directory to See Relocated Elements 289

Administering Views

18. Understanding View Storage 293
18.1 ClearCase Views 293
18.2 Dynamic Views 293
View Database 294
View's Private Storage Area 295
Remote View Storage on UNIX 295
18.3 Snapshot Views 296

19. Setting Up Views 299
19.1 Setting Up an Individual User's View 299

View Storage Requirements 300
View Database 300
View's Private Storage Area 300
19.2 Setting Up a Shared View 301
19.3 Setting Up an Export View for Non-ClearCase Access 302
Exporting Multiple VOBs 304
Multihop Export Configurations 304
Restricting Exports to Particular Hosts 305

20. Backing Up and Restoring Views 307
20.1 Backing Up a View 307
20.2 Restoring a View from Backup 309

21. Administering View Storage 313
21.1 View Storage Maintenance 313
Scrubbing View-Private Storage 314
21.2 Cleaning Up a View Manually 315

22. Moving Views 319
22.1 Moving a View 319
Moving a View on UNIX 320
Moving a View on Windows NT 322
22.2 Moving a View to a UNIX Host with a Different Architecture 324
22.3 Moving a Dynamic View's Private Storage Area on UNIX 327

23. Removing Views 329
23.1 Taking a View Out of Service 329
Restoring the View to Service 329
23.2 Permanent Removal of a View 329

Administering ClearCase Licenses

24. Administering ClearCase Licenses 333
24.1 Floating License Architecture 333
License Priorities 334
License Expiration 334
License Report Utility 334
24.2 Setting Up a License Server 335
Adding New Licenses to an Existing License Server Host 335
Setting Up Additional License Server Hosts 335
24.3 Moving Licenses to Another Host 337
24.4 Renaming a License Server Host 338

Administering the ClearCase Registry

25. Understanding the ClearCase Registry 341
25.1 Storage Directories and Access Paths 341
Distributed VOBs and Views on UNIX 342
25.2 Storage Registries 342
25.3 Object Registries 342
25.4 Tag Registries 343
Tag Registries on UNIX 343
Tag Registries on Windows NT 343
25.5 Networkwide Accessibility of VOBs and Views 346
Public and Private VOBs 346
25.6 Managing VOB and View Registry Entries 347
Viewing VOB and View Registry Entries 347

Creating VOB and View Registry Entries 349
Creating VOB-Tags and View-Tags 349
25.7 Creating Server Storage Locations 350
25.8 Registering Site-Wide Properties 351
25.9 Registry Guidelines 352
Multiple Registries 353

26. Administering Regions 355
26.1 Network Regions 355
Registries in a Multiple-Region Network 358
Tag Registry Implementation 359
Establishing Network Regions 361
26.2 Adding a Network Region 362
When to Create Additional Regions 362
Multiple Regions Vs. Multiple Registries 363
A Sample Network Partition 363
Procedure for Adding a Network Region 364
To Create a New Region 365
To Move a Host into a New Network Region 365
To Create VOB-tags and View-tags for a New Network Region 366
If the New Region Is Served by a Separate Registry Host 367
Guidelines for Multiple Network Regions 368
26.3 Removing a Network Region 368

27. Moving, Renaming, and Backing Up the ClearCase Registry 369
27.1 Moving the ClearCase Registry: Registry Switchover 369
Moving the Registry to an Active Backup Registry Host 369
Moving the Registry to a Host Not Configured for Registry Snapshots 371
No Backup Host: Primary Registry Host Is Available 371
No Backup Host: Primary Registry Host is Down 371
27.2 Renaming the Registry Server Host 372
27.3 Changing the Backup Registry Host 372

Changing Backup Registry Host Using rgy_switchover....373
Changing Backup Registry Host Without rgy_switchover....373
27.4 Renaming a VOB or View Host....373

Administering Scheduled Jobs

28. Managing Scheduled Jobs....377
28.1 Tasks and Jobs....377
Task and Job Storage....378
Task and Job Database Initialization....379
Job Execution Environment....379
28.2 The Default Schedule....380
28.3 Managing Tasks....381
Creating a Task....381
Editing a Task....382
Deleting a Task....382
28.4 Managing Jobs....383
Creating a Job....383
Specifying a Job's Schedule....384
Specifying Job Notifications....385
Viewing Job Properties....386
Editing Job Properties....386
Running a Job Immediately....387
Deleting a Job....387
28.5 Managing the Scheduler Access Control List....388

Administering Web Servers

29. Configuring a Web Server for the ClearCase Web Interface 393
29.1 Configuration Planning 393
Web Administration Considerations 393
ClearCase Considerations 394
29.2 Configuring the Web Server 396
Apache 396
Microsoft Internet Information Server (IIS) 397
Netscape Enterprise Server 398

30. Configuring Integrations with Microsoft Web Authoring Tools 401
30.1 Overview of the Integration 401
Server Setup Overview 402
Client Setup Overview 403
30.2 Server Setup Procedure 403
Step 1: Install IIS 404
Step 2: Install FPSE or OSE 406
Step 3: Install ClearCase 406
Step 4: Run the Web Authoring Integration Configuration Wizard 406
30.3 Client Setup Procedure 409
Step 1: Install the Client Application 409
Step 2: Add Web to Source Control 409
From FrontPage 98 409
From FrontPage 2000 410
From Visual InterDev 6.0 410
Step 3: Verify That New Web Content Is Added to Source Control 410
Step 4: Setting User Permissions 413
FrontPage 98 413
FrontPage 2000 413
Visual InterDev 6.0 413

Step 5: Local Mode Client Setup for FrontPage 2000 413
30.4 Web Folders Support in Office 2000 and Microsoft Internet Explorer 5 415
30.5 Updating the Shared View on the Web Server 415
30.6 Considerations for Migrating and Converting Data to the Integration 415
30.7 Accessing Help Information for the Integration 416

31. Using Dynamic Views to Develop and Deliver Web Content 417
31.1 Overview of Using Dynamic Views on a Web Server 417
31.2 Example Scenario 418
VOB and Branch Configuration 418
View Configuration for Tasks 420
Developing New Content 420
Merging and Testing Approved Changes 420
Viewing and Testing Content 420
Accessing Content from a Web Browser 421
Implementing Policies 421
Testing Source Files Before Checkin 421
Restricting the Users Who Can Approve Changes 422
Labeling Approved Sources 422
Synchronizing the Massachusetts and California VOB Replicas 422
Copying Files to the Web Server 423
Rolling Back to Previously Published Versions 423
31.3 Configuring the Web Server 423
Configuring the Apache Web Server 424
To Configure an Apache Web Server on Windows 425
To Configure an Apache Web Server on UNIX 425
Configuring the Microsoft Internet Information Server 426
Configuring the Netscape Enterprise Web Server 427

Tuning for Performance

32. Improving VOB Host Performance 431
32.1 Minimize Process Overhead 431
32.2 Maximize Disk Performance 432
32.3 Add Memory for Disk Caching on Windows NT 432
32.4 Tune Block Buffer Caches on UNIX 433
Block Buffer Cache Statistics 433
Flushing the Block Buffer Cache 434

33. Improving View Host Performance 435
33.1 Obtaining View Cache Information 435
Analyzing the Output 436
33.2 Reconfiguring a View 437

34. Improving Client Host Performance 439
34.1 Increasing System Resources 439
34.2 Creating Remote Storage Pools for UNIX VOBs 440
Caution on Remote Source Pools 440
34.3 Examining and Adjusting MVFS Cache Size 441
Real-Time Updating of MVFS Cache Sizes 444
Adjusting the MVFS Memory Initialization Factor 445
Setting Individual Caching Parameters on UNIX 447
Setting Individual Cache Sizes on Windows NT 448
Minimizing Attribute Cache Misses 448
Attribute Cache Total Misses 448
Close-to-Open Misses 448
Generation Misses on UNIX 449
Cache Timeout Misses 449
Cache Fill Misses 450
Event Time Misses 450

Troubleshooting

35. Determining a Data Container's Location 453
35.1 Scenario 453
35.2 Determining the ClearCase Status of Files 453
35.3 Determining the Full UNIX Pathnames of Files 454
35.4 Where Is the VOB? 454
35.5 Where Is the View? 455
35.6 Where Are the Individual Files? 455
Locating a Checked-Out Version 456
Locating a Checked-In Version's Cleartext Container 457
Locating a Checked-In Version's Source Container 457
Locating a View-Private File 457
Issues with Nonlocal UNIX Storage 457
Links and Directories on UNIX 458

36. Repairing VOB and View Storage Directory ACLs on Windows NT 459
36.1 ClearCase ACLs 459
36.2 Causes of Protection Problems 461
Copying the Storage Directory 461
Converting the File System from FAT to NTFS 462
Editing Permissions 462
36.3 Utilities for Fixing Protection Problems 463
fix_prot 463
lsacl 464
36.4 Fixing Protection Problems 465

37. Preventing Accidental Deletion of Data by crontab Entries 467
37.1 Preventing Recursive Traversal of the Root Directory 467
crontab Modification During ClearCase Installation 468
Modifying a crontab Entry 468

Index 471

Figures

Figure 1 ClearCase Storage Registries 8
Figure 2 Client/Server Processing 10
Figure 3 VOB Database and VOB Storage Pools 113
Figure 4 Local and Remote VOB Storage Pools 116
Figure 5 Linking Multiple VOBs into a Single Directory Tree 122
Figure 6 Sample SourceSafe Payroll Configuration 136
Figure 7 ClearCase Version Tree of \bugfix\mod_empl.c Element 142
Figure 8 Administrative VOB Hierarchy 149
Figure 9 Replicated Administrative VOBs. 153
Figure 10 Semi-Live Backup 170
Figure 11 VOB Restoration 188
Figure 12 Restore Scenario Summarized by Output from vob_restore 189
Figure 13 vob_restore: In Place 190
Figure 14 vob_restore: VOB Is Active 191
Figure 15 vob_restore: Move VOB on Same Host 191
Figure 16 vob_restore: Move VOB to New Host 192
Figure 17 VOB Database or Storage Pool Skew Associated with VOB Snapshots 193
Figure 18 Controlling VOB Growth 206
Figure 19 Pool Access Through vob_server and VOB Database Access Through db_server 244
Figure 20 checkvob Output Log: Summary File 247
Figure 21 checkvob Output Log: Condensed Transcript File 249
Figure 22 Common Scenarios in VOB Database or Storage Pool Synchronization 263
Figure 23 Elements Cataloged by Directory Versions Before Relocate Operation 274
Figure 24 Directory Version Cataloging After Relocate Operation 275
Figure 25 Destination VOB That Includes Multiple Versions 276
Figure 26 Source VOB that Includes a Borderline Element 278
Figure 27 Source and Destination VOBs with Borderline Element Relocated 279
Figure 28 Source and Destination VOBs with Borderline Element Not Relocated 280
Figure 29 Source VOB with Multiple Branches on Parent Directory 286

Figure 30 Destination VOB After Modifying Old Version of Destination Directory 289
Figure 31 Export View for Non-ClearCase Access 303
Figure 32 ClearCase Object and Tag Registries (Single Network Region) 345
Figure 33 cleartool Commands and the ClearCase Storage Registry 350
Figure 34 Network with Global Naming 356
Figure 35 Network Regions and Their Tag Registries 360
Figure 36 Sample Network with Two Regions 364
Figure 37 Setting Up the Root Web in the IIS Installation 405
Figure 38 Setting Up the VOB Storage for the Integration 407
Figure 39 Setting Up the View Storage for the Integration 408
Figure 40 FrontPage Source Control Icons 411
Figure 41 Visual InterDev Source Control Icons 412
Figure 42 Development and Publishing Branches 419
Figure 43 VOB and Web Server Configuration 419
Figure 44 Directory as a Super-Root 468

Tables

Table 1 Protection Mode for a ClearCase Object 27
Table 2 Protection Mode Digits for a ClearCase Object 28
Table 3 Characters Not Allowed in Windows File Names 68
Table 4 Mechanisms for File Access Between ClearCase Clients and VOB and View Servers 78
Table 5 Importance of VOB Directories in Partial Backups 174
Table 6 Access Types in Scheduler ACL Entries 388
Table 7 Supported Platforms for Web Servers 403
Table 8 Supported Web Server Platforms 423
Table 9 MVFS Cache Information 442
Table 10 Cache Sizes Corresponding to mvfs_largeinit or Scaling Factor Settings 446
Table 11 Storage Locations of MVFS Files 456

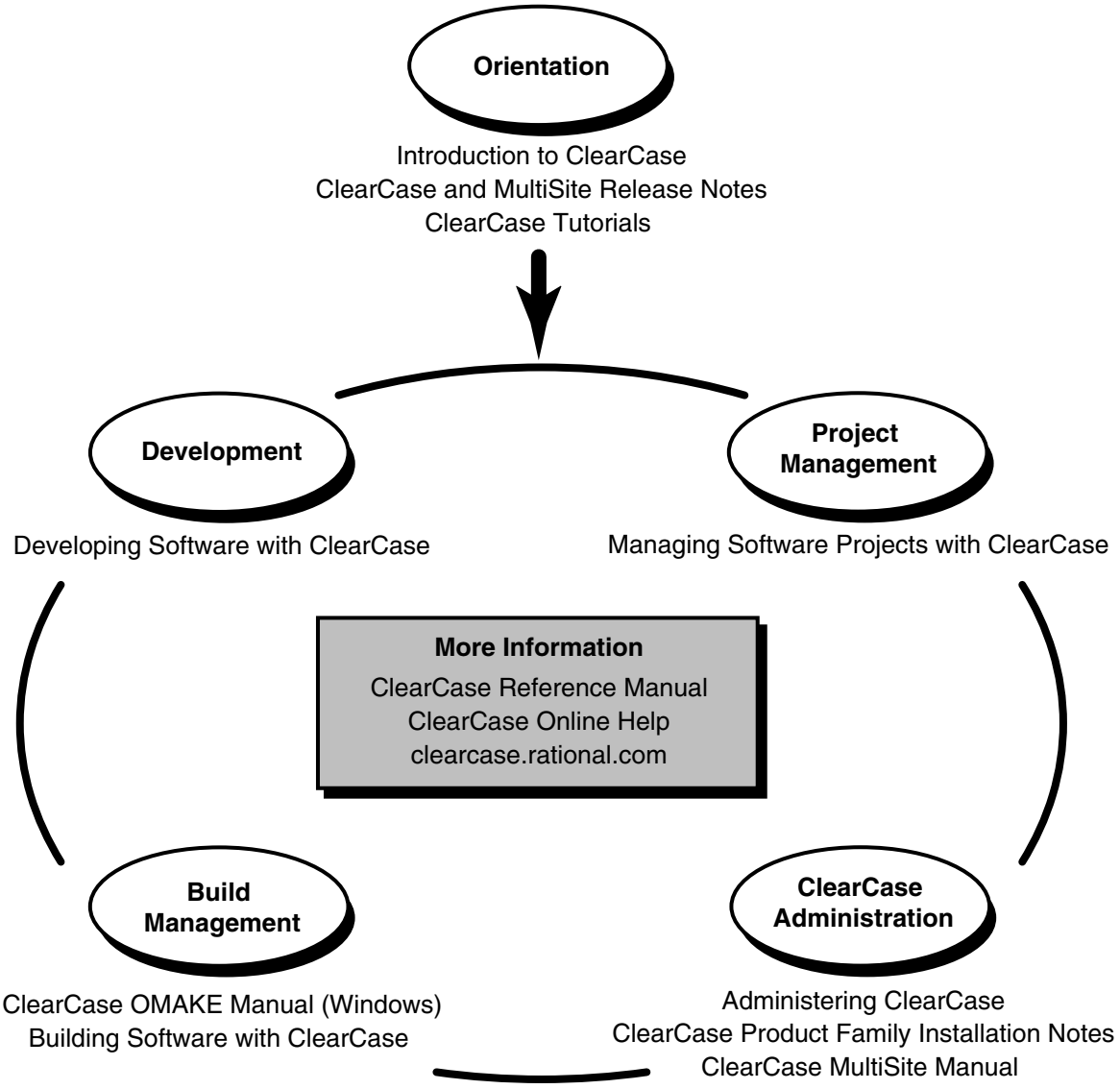
Preface

About This Manual

This manual is for ClearCase 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* discusses these subjects in depth:

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

ClearCase Documentation Roadmap



Typographical Conventions

This manual uses the following typographical conventions:

- *ccase-home-dir* represents the directory into which the ClearCase Product Family has been installed. By default, this directory is **/usr/atria** on UNIX and **C:\Program Files\Rational\ClearCase** on Windows.
- *attache-home-dir* represents the directory into which ClearCase Attache has been installed. By default, this directory is **C:\Program Files\Rational\Attache**, except on Windows 3.*x*, where it is **C:\RATIONAL\ATTACHE**.
- **Bold** is used for names the user can enter; for example, all command names, file names, and branch names.
- *Italic* is used for variables, document titles, glossary terms, and emphasis.
- `A monospaced font` is used for examples. Where user input needs to be distinguished from program output, **bold** is used for user input.
- Nonprinting characters are in small caps and appear as follows: **<EOF>**, **<NL>**.
- Key names and key combinations are capitalized and appear as follows: SHIFT, CTRL+G.
- [ ] Brackets enclose optional items in format and syntax descriptions.
- { } Braces enclose a list from which you must choose an item in format and syntax descriptions.
- | A vertical bar separates items in a list of choices.
- ... In a syntax description, an ellipsis indicates you can repeat the preceding item or line one or more times. Otherwise, it can indicate omitted information.

  NOTE: In certain contexts, ClearCase recognizes "**...**" within a pathname as a wildcard, similar to "*" or "?". See the **wildcards_ccase** reference page for more information.
- If a command or option name has a short form, a "medial dot" (·) character indicates the shortest legal abbreviation. For example:

  lsc·heckout

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

Online Documentation

The ClearCase graphical interface includes a help system.

There are three basic ways to access the online help system: the **Help** menu, the **Help** button, or the F1 key. **Help**> 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 lsprivate –help

```
Usage: lsprivate [-tag view-tag] [-invob vob-selector] [-long | -short]
                  [-size] [-age] [-co] [-do] [-other]
```

Additionally, the online *ClearCase Tutorial* provides important information on setting up a user's environment, along with a step-by-step tour through ClearCase's most important features. To start the *ClearCase Tutorial*

Technical Support

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

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

Administering the ClearCase Network

1 Understanding the ClearCase Network

This chapter presents a system administrator's overview of a local area network of computers running ClearCase and MultiSite. 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.

Also, this manual refers to VOBs and views by the operating system on which the storage directories are located (for example, *UNIX views*, *Windows VOBs*, and so on).

1.1 Network Overview

A ClearCase administrator has three principal concerns:

- **Hosts.** ClearCase can be installed and used on any number of hosts in a network. Different hosts use ClearCase software in different ways. For example, one host may be used only to

store version-controlled data; another may be used only to run ClearCase software development tools.

- **Data storage.** ClearCase data is stored in *VOBs* and *views*, which can be located on any or all the hosts where ClearCase is installed. A VOB or view hosted on UNIX can have auxiliary data storage on other UNIX hosts where ClearCase is not installed; such storage is accessed through UNIX symbolic links, which are not a feature of Windows NT.

  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, requiring 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. This means that any particular development task (for example, execution of a single ClearCase command) may involve programs and data on several hosts. It is important to classify hosts by the roles they play, because different kinds of hosts require different administrative procedures. But keep in mind that a ClearCase host may play different roles at different times or several roles at once.

- **License server hosts.** One or more hosts in the network act as ClearCase *license server hosts*, authorizing and limiting ClearCase use according to the terms of your license agreement. Each host on which ClearCase is installed is assigned to a particular license server host and communicates with that host periodically. (The **albd_server** process on a license server host acts as the license server process.)

- **Registry server host.** One host in the network acts as the ClearCase *registry server* host. Each host is assigned to a particular registry server host at ClearCase install time. This host stores a set of files that contain essential access-path information concerning all the VOBs and views in the network. ClearCase client and server programs on all other hosts occasionally communicate with the registry server host, to determine the actual storage location of ClearCase data. (The **albd_server** process on the registry server host acts as the registry server process.)

  You can construct a network with multiple registry hosts, maintaining separate clusters of ClearCase hosts that, in general, do not share VOBs and views. Because this approach

complicates the administration process, it is not recommended unless specific circumstances require it.

- **Backup registry server host.** On the primary registry host, you can designate another host as a backup registry server. A backup registry host takes periodic snapshots of the ClearCase registry files on the primary registry server. (See the **rgy_backup** reference page in the *ClearCase Reference Manual*.) If the primary server fails, you can switch registry server functions to the backup host (see **rgy_switchover**). Do not designate a backup registry host that is unsuitable to serve as primary registry host in an emergency.

- **Client hosts.** Each 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 **cleartool**, **clearmake**, and various graphical user interfaces (GUIs).

  ClearCase must be installed on each client host; installation can include the ClearCase *multiversion file system* (MVFS), which extends the native file system to provide support for ClearCase *dynamic views*. If a client uses only *snapshot views*, it does not need the MVFS.

- **Server hosts**. Some hosts may be used only as data repositories for VOB storage directories and/or view storage directories. Such server hosts run ClearCase server programs only: **albd_server**, **vob_server**, **view_server**, and other programs.

  ClearCase must be installed on each server host.

- **ClearCase Web server hosts.** If you want to use the ClearCase Web interface, you'll need at least one ClearCase Web server host. See Chapter 29, *Configuring a Web Server for the ClearCase Web Interface* for more detail.

  If your ClearCase network includes UNIX computers, there may be other types of ClearCase hosts in it too:

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

- **Non-ClearCase UNIX hosts.** ClearCase may not be installed on every host in your network. In fact, it is not possible to install it on hosts whose architectures ClearCase does not yet support. Such hosts cannot run ClearCase programs, but they can access ClearCase data, through standard UNIX network file system facilities. You administer these export (or share) mechanisms using standard UNIX tools.

1.3 ClearCase Data Storage

All ClearCase data is stored in VOBs and views. These data structures can be distributed throughout the local area network; even an individual VOB or view can be distributed. Users 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, however, this repository is distributed among multiple *versioned object bases* (*VOBs*), located on multiple hosts. 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. In a typical network, registry maintenance is minimal; certain ClearCase commands update the registry automatically. You must update the registry manually if you move a VOB to another location (for example, to another host). You may also need to update the registry if different pathnames must be used on different hosts to access the same VOB storage directories.

- **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 12, *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. You can specify what is unneeded on a per-VOB basis.

  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, which you can change as needed. For more information on creating and managing scheduled jobs, see Chapter 28, *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 begins), you may need to adjust a VOB's group list.

- **Growth.** As new projects begin (or existing projects are placed under ClearCase control), you must create new VOBs, define VOB access rights of 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 checked-out versions of file elements, *view-private files* that have no counterpart in the VOB (for example, text editor backup files), and newly built derived objects.

Developers think of views and VOBs as being very different: data resides in a VOB and is accessed through a view. From an administrator's standpoint, views and VOBs are quite similar. Each view is implemented as a *view storage directory* (actually a directory tree), which holds an embedded database. In a dynamic view, the view storage directory also holds developers' file-system objects. A snapshot view has a separate snapshot view root directory that holds copies of files maintained in VOBs. View administration is similar to that for VOBs, including registration, backup, periodic maintenance, and access control.

A view also includes both a storage area for file-system data and an associated database:

- In a *dynamic view*, the view's *private storage area* (subdirectory **.s** of the top-level view storage directory) holds all view-private objects. It also holds the data files (data containers) for derived objects built in the view. In a *snapshot view*, view-private objects reside in the view's directory tree.
- The *view database* tracks the correspondence between certain VOB database objects and view-private objects.

Most VOBs are long-lived structures, created by an administrator; views tend to be shorter-lived structures, created by individual developers.

1.4 ClearCase User Base

ClearCase does not maintain a registry of its users. Any user who is logged in to a ClearCase host and can acquire a license is able to use the software. (See also Chapter 24, *Administering ClearCase Licenses*.) Successful use of ClearCase depends on networkwide consistency in the user base: 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. Consistency is usually achieved by using networkwide databases maintained by the operating system such as the NIS **passwd** and **group** maps on UNIX or, on Windows, Windows NT *domains.*

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.

Many **cleartool** commands check the user's identity before granting access to particular objects—element, version, and so on. See Chapter 3, *Understanding ClearCase Access Controls* for detailed information on this topic. Standard non-ClearCase commands and programs that access ClearCase data in a dynamic view must also go through the ClearCase *MVFS* and are subject to access control.

1.5 Registries for VOBs and Views

All ClearCase data storage areas—all VOBs and views—in a local area network are centrally registered on the ClearCase *registry server host*. This host has two kinds of registries:

- The object registry records the physical storage locations of VOBs and views.
- The tag registry records globally valid pathnames to VOBs and views.

For example, an object registry entry may record the fact that a VOB's storage directory is located on 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 to be activated (*mounted*) at the location specified by its VOB-tag, in this case **\proj1**.

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

NOTE: ClearCase administrators in charge of mixed networks of UNIX and Windows hosts must be aware of several issues regarding VOB and view access in this environment. See Chapter 8, *Configuring VOB and View Access in Mixed Environments*, for more on this topic.

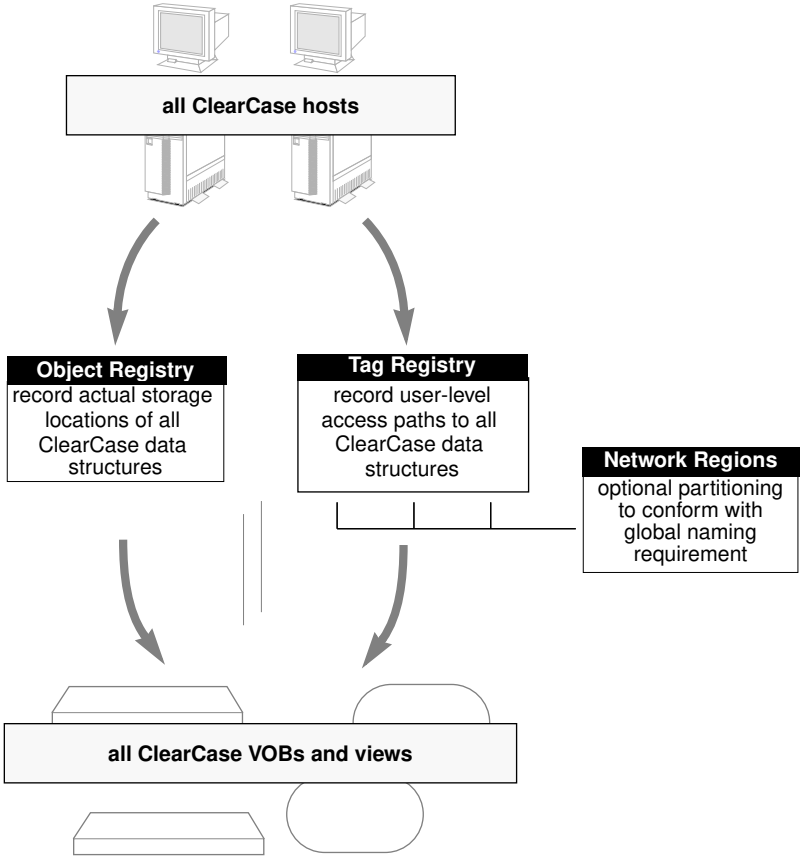
Network Regions

In an ideal network, all hosts access ClearCase data storage areas using identical global pathnames. Many networks, particularly those with mixed machine architectures and operating system software, fall short of this ideal. You can logically partition a network into multiple *network regions*. Conceptually, each region has its own tag registry; ClearCase data structures can be accessed with different global pathnames in different regions.

Figure 1 illustrates how registries mediate access to ClearCase data structures. Networkwide registries offer these administrative benefits:

- Centralized control over all components of the network's distributed data repository (VOBs and views)
- Independence from operating-system-specific mechanisms for mounting file systems
- Ability to accommodate heterogeneous networks and networks in which hosts have multiple names and/or multiple network interfaces
- Making VOBs and views globally accessible

Figure 1 ClearCase Storage Registries



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*, which run on the host where the VOB in which the checkout-requested element resides.

3. A view-private copy is made of the version being checked out. Making this copy involves the **view_server** process that manages the user's view. It can also involve other hosts:

   - The user's client process and the view may be on different hosts.
   - The VOB storage pool that holds the version being checked out may be located on a different host from the VOB storage directory.
   - On UNIX, the view may have a private storage area that is located on a different host from the view storage directory.

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 Servers

Each ClearCase host runs an Atria Location Broker Daemon process, **albd_server**, which is usually started at boot time, though it can be started and stopped manually as needed by a user with adequate privileges (see *ClearCase Startup and Shutdown* on page 11.) 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. Such servers always run on the host where that data structure resides. ClearCase has the following servers 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

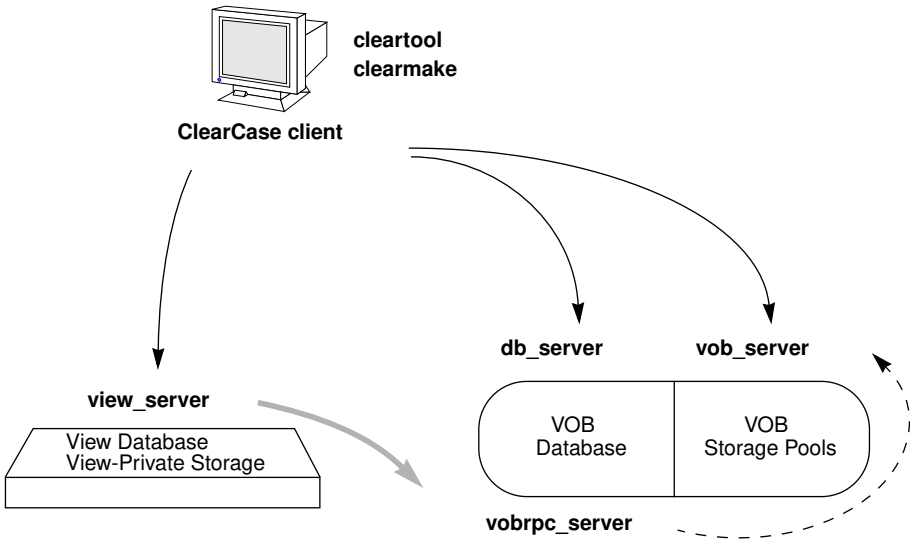
db_server
: Fields requests from one ClearCase client program, destined for one or more VOB databases on a host

vobrpc_server

Fields requests from one or more **view_server** processes, destined for a particular VOB database

Figure 2 shows the communications paths that connect a client process with server processes.

Figure 2 Client/Server Processing



Server Error Logs

Each ClearCase server maintains an error log on the host where it executes.

- 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, many ClearCase servers writes error messages to the Windows NT event log.

Given ClearCase's distributed architecture, an error resulting from a command entered on one host can generate an error log entry on another host. In such cases, the user is directed to the appropriate log file on the appropriate host. See the **getlog**, **cleargetlog**, and **errorlogs_ccase** reference pages in the *ClearCase Reference Manual* for details on the error logs and how to display them. In addition, Windows NT hosts can access error logs for all ClearCase hosts (Windows and UNIX) using the ClearCase Administration Console.

1.7 ClearCase Startup and Shutdown

ClearCase is normally started and stopped when a ClearCase hosts starts up or shuts down, using the normal conventions for the platform type.

UNIX Startup and Shutdown

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

- Starts the host's **albd_server** (Atria Location Broker Daemon) process.
- Starts a **lockmgr** process if there are any VOBs on the host. This process arbitrates concurrent access to VOB databases by multiple client processes.
- Performs additional file-system setup tasks, such as mounting *public* VOBs.

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

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

Stops all ClearCase processing on a host: terminates the **albd_server** and **lockmgr** processes, along with any other ClearCase server processes running on the host. Unmounts VOBs and unmounts the *viewroot directory*, **/view**. User processes that are set to views on that host will also be terminated.

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

Restarts ClearCase processing on a host by starting an **albd_server** process and a **lockmgr** process, mounting VOBs, and mounting the viewroot directory, **/view**.

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

Windows NT Startup and Shutdown

If the ClearCase MVFS is installed, the system starts it at system startup time. Unless explicitly configured to be started 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. To stop and restart the MVFS, you must shut down and restart your computer.

2 Administering ClearCase Hosts

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 28, *Managing Scheduled Jobs*.

NOTE: Sections 2.4 through 2.7 of this chapter apply only to ClearCase on UNIX hosts.

2.1 ClearCase Administration Console

ClearCase on Windows NT has an administration console that centralizes administration of views, VOBs, scheduled jobs, server logs, and the ClearCase registry 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 ClearCase on all platforms, and it provides a single point from which most ClearCase administration functions can be managed even in a large, complex network.

The ClearCase Administration Console manages VOB and view storage, scheduled jobs, and server logs for all ClearCase hosts known to the ClearCase registry server for the local host or to any other accessible ClearCase registry server. It also manages registry information on the registry server host and provides easy access to the ClearCase customer Web site. To start the ClearCase Administration Console on a Windows NT host, click **Start**>**Programs**>**Rational ClearCase Administration**>**ClearCase Administration Console**.

A restricted version of the console, ClearCase Host Administration, manages VOB and view storage, scheduled jobs, and server logs for the local host. To start ClearCase Host

Administration on a Windows NT host, click **Start**>**Programs**>**Rational ClearCase Administration**>**ClearCase Host Administration**.

In this document, we usually suggest using the ClearCase Administration Console to perform any administrative operation of which it is capable. We also provide information on using the command line. Any procedure described in this book that uses ClearCase Administration Console can also use ClearCase Host Administration if the procedure is confined to operating on the local host.

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.**

Controlling Remote Administration

The ClearCase Administration Console is a powerful tool for remote administration of ClearCase on Windows NT and UNIX. You may want to prevent remote administrative access to certain hosts (for example, critical VOB servers), because it is possible for an inexperienced or malicious user to impair or disrupt ClearCase operations by changing various host properties.

Every ClearCase host can be configured to allow or disallow remote administration. If a host allows remote administration, a user who is a member of the *ClearCase group* can use the host node of the ClearCase Administration Console to change some ClearCase properties of the host. These properties include registry regions, license server, registry server, and MVFS cache sizes. If the host does not allow remote administration, only a user who is logged on to that host can change these properties. (This setting also affects use of ClearCase Host Administration for local administration.)

Whether a host allows remote administration is initially determined when ClearCase is installed. To change this configuration after ClearCase is installed on a Windows host:

1. Click **Start**>**Settings**>**Control Panel**.

2. In the ClearCase program, click the **Options** tab and select the **Allow ClearCase Remote Administration** check box. Then click **OK**.

To change this configuration after ClearCase is installed on a UNIX host:

1. As the *root* user, make the file **/var/adm/atria/config/admin.conf** writable.
2. Edit **/var/adm/atria/config/admin.conf** with a text editor and find the line containing the `DISALLOW_REMOTE_ADMIN` parameter.
3. To enable remote administration, change the value of `DISALLOW_REMOTE_ADMIN` to **0**.
4. To disable remote administration, change the value of `DISALLOW_REMOTE_ADMIN` to **1**.
5. Save **/var/adm/atria/config/admin.conf** and make the file read-only.

2.2 Using DHCP with ClearCase

The Dynamic Host Configuration Protocol (DHCP) is an Internet standard protocol that allows a host to use a temporary, dynamically assigned IP address. This standard is widely implemented on Windows computers and is also available on some UNIX computers. Because ClearCase caches IP addresses for an extended period of time, if the computers at your site that run ClearCase use DHCP, you must understand potential interactions between DHCP and ClearCase.

It is possible to configure DHCP in a variety of ways. In some configurations, the IP address of a computer may change at boot time, which can present problems for ClearCase. Fortunately, in most common scenarios (for example, when the DHCP lease time is set to be at least a few days, and no computers that run ClearCase are down for more than half the lease time), IP addresses do not change when the computer is rebooted, and you should have no problems running ClearCase in a DHCP-based environment.

2.3 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, you cannot use version-extended or view-extended pathnames, and you cannot run ClearCase tools such as **clearmake** on the non-ClearCase host.

  To use this method:

  a. On a ClearCase host, create a snapshot view and load into it the files you want to access from a non-ClearCase host. For information on creating snapshot views, see *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 snapshot view directory resides. For a Windows NT host, share the drive or directory on which the snapshot view directory resides.

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

  d. To modify VOB data, check out and check in versions in the snapshot 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 29, *Configuring a Web Server for the ClearCase Web Interface*.

- **Use non-ClearCase access** (UNIX only). On UNIX hosts, you can use this method when the non-ClearCase host implements the NFS client protocol. Performance is slower than accessing a snapshot view through NFS because the MVFS, a view server, and the disk-based file system take part in each file access. It also cannot faithfully implement some parts of the NFS protocol. Because of these limitations, snapshot views and the ClearCase Web interface, when available, are preferable methods for accessing VOB data. For information on non-ClearCase access, see *Using Non-ClearCase Access on UNIX Hosts*.

2.4 Using Non-ClearCase Access on UNIX Hosts

A UNIX host on which ClearCase has not been installed can use non-ClearCase access to read VOB data from a UNIX VOB server. Typically, the technique is as follows:

- A UNIX host running ClearCase must export a view-extended pathname to the VOB mount point (for example, **/view/exportvu/vobs/vegaproj**). Edit the file **/etc/exports.mvfs** to specify this pathname.
- One or more non-ClearCase hosts access the VOB through a view-extended pathname. For example, a host may have an entry in its file-system table that begins

  ```
  mars:/view/exportvu/vobs/vegaproj /usr/vega nfs ...
  ```

For information on setting up an export view, see *Setting Up an Export View for Non-ClearCase Access* on page 302.

Restrictions on Use

Non-ClearCase access is restricted to UNIX computers, and carries several restrictions:

- **VOB access:** Users on the non-ClearCase host can only read data from VOBs on UNIX VOB server hosts configured for non-ClearCase access; they cannot modify the VOB in any way. They are also restricted to using the element versions selected by the specified view. They cannot use version-extended or view-extended pathnames to access other versions of the VOB's elements.
- **Building:** Although users cannot modify VOBs that are mounted through a view, they can write to view-private storage. Users can modify these view-private files with an editor and build them with a native **make** program or with scripts, though not with **clearmake**. Files created by such builds do not become derived objects; they are view-private files, unless developers take steps to convert them. (For more on this topic, see *Building Software with ClearCase*.)

  Because **clearmake** does not run on the non-ClearCase host, configuration lookup and derived object sharing are not available on these hosts.

Using automount with Non-ClearCase Access

After a ClearCase view/VOB pair has been exported from a ClearCase system via NFS, any properly authorized NFS client system can access the files within that view/VOB pair. If the NFS client can mount the view/VOB pair directly with a **mount** command, you can also put that view/VOB pair (explicitly or implicitly) into a map used by the NFS client's **automount** daemon. Explicit entries name the exported view/VOB pair directly. Implicit entries may arise from wildcard syntax or other advanced **automount** features.

For example, using the typical **automount** wildcard syntax, suppose an indirect map is configured at **/remote/***viewname* with a map file listing **server:/view/***viewname***/vobs/&**. This means that when a process on the NFS client accesses a subdirectory of **/remote/***viewname*, the automount process performs an NFS mount from the corresponding subdirectory of **server:/view/***viewname***/vobs**.

NOTE: Listing the directory **/remote/***viewname* usually shows only active mounts, not all possible mounts. This is similar to the result of listing **/net** for a hosts map.

If this type of map does not work correctly, verify that an explicit **mount** command works properly. If it does, then it is likely that the problem lies in the client automounter. Please consult your NFS client's documentation for full details on map syntax.

NOTE: Using the **–hosts** map for automount access does not work properly if the root file system and a view/VOB pair are exported on a ClearCase server. Suppose an NFS client host tries to access **/net/***cchost***/view/***viewname***/vobs/***vobpath*. The automounter mounts the server's root directory on **/net/***cchost*, then tries to mount the view/VOB on **/net/***cchost***/view/***viewname***/vobs/***vobpath*. However, **/net/***cchost***/view** has no subdirectories, because NFS exports do not follow local file-system mounts such as **/view**. This mount fails because the local client is unable to find a directory on which to mount the view/VOB pair.

For more information on automounting, see *Using automount with ClearCase on UNIX* on page 20.

Problems with Non-ClearCase Access: NFS Client Caching

Most NFS client implementations include caches to speed up access to frequently used file data and metadata. Newer client implementations typically cache more aggressively than older ones. When the NFS client believes its cache is valid, but something in the view or VOB has changed so that it is inconsistent with the cached data, the client may access the wrong file from the VOB.

A common inconsistency arises when a file is checked in from another view, or when the exporting view's config spec is changed. If, as a result, the view selects a new version of a file, the NFS client may not notice the change. The NFS client expects that any change in the name-to-file binding changes the time stamp of the directory that contains the file. In this case, the directory in the exporting view has not changed, but the file cataloged in that directory has changed versions. The NFS client may not revalidate its cached name-to-file binding (the association of the name with a certain version of the file) until it believes the directory has changed or the entry is pushed out of the cache because of capacity constraints.

Most NFS clients consider a cache to be valid for only a short period, typically 60 seconds or less. If the cache is not updated in a short time, you can use one of the following methods to work around this restriction:

- Create and remove a dummy file from the containing directory. This changes the directory time stamp, which invalidates the client's cache and forces the client to look up the new file version.
- Disable the client's attribute cache (usually with the **noac** mount option). However, our testing indicates that this works only for some NFS V2 clients, and that it will increase the network traffic between the NFS client and the exporting view server. If your client uses NFS V3 by default, and you want to use **noac**, we recommend that you edit the mount options to request NFS V2.
- As *root* user, unmount the file system, and then mount it. This flushes the NFS client cache for the file system. (Even if the unmount fails, it flushes the cache.)

We also recommend that you limit the dynamic nature of non-ClearCase access by using config specs that do not continually select new versions of files. For example, you can change the config spec for an exported view to contain a label-based rule rather than the **/main/LATEST** rule.

Problems with Non-ClearCase Access: NFS Locking

Because non-ClearCase access does not support NFS file locking for its files, application packages that use and require file locking do not work properly on files accessed with non-ClearCase access. Though file locking may work for view-private files on some UNIX operating systems running the MVFS, it may not work for VOB files and it does not work at all on some operating systems. An application package can hang if it insists on retrying lock requests until it can obtain a lock, and it can also be subject to file corruption if it continues when it cannot obtain a lock and multiple clients are modifying the same file. If your application requires file locking, use snapshot views or the ClearCase Web interface for access to VOB data. (See *ClearCase Data and Non-ClearCase Hosts* on page 15.)

2.5 Recording Multiple Network Interfaces on UNIX Hosts

If any ClearCase host (client or server) has two or more network interfaces (two or more separate lines in the **/etc/hosts** file or the **hosts** NIS map), it must have a file called **/var/adm/atria/config/alternate_hostnames**, which records its multiple entries. For example, suppose that the **/etc/hosts** file includes these entries:

```
 .
 .
159.0.10.16 widget sun-005 wid
 .
159.0.16.103 widget-gte sun-105
 .
```

In this case, the **alternate_hostnames** file must contain these entries:

```
widget
widget-gte
```

Note that only the first host name in each **hosts** entry must be included in the file. In general, the file must list each alternative host name on a separate line. There is no commenting facility; all lines are significant. If a host does not have multiple network interfaces, this file must not exist on that host.

2.6 Using automount with ClearCase on UNIX

This section discusses use of the standard UNIX **automount(1M)** program with ClearCase. Implementations of the facility vary from architecture to architecture; consult the documentation supplied by your hardware vendor.

For information on automounting and access to ClearCase from non-ClearCase hosts, see *Using automount with Non-ClearCase Access* on page 18.

Automounter Maps

You can use any **automount** maps, including both direct and indirect maps, to access remote disk storage where VOB storage areas reside. Keep in mind that within each *network region*, VOB mount points must be consistent across all of the region's hosts.

Using the –hosts Map

ClearCase looks for symbolic links to the mount points created through the **–hosts** map in any of these directories:

/net *(the automount default)*
/hosts
/nfs

If your site uses another directory for this purpose (for example, **/remote**), create a UNIX symbolic link to provide access to your directory through one of the expected pathnames. For example:

```
# ln –s /remote /net
```

Not Using the –hosts Map

If a host does not use the **–hosts** map to make remote VOB and view storage directories accessible, you must be careful when executing **mkvob** and **mkview** on that host. The heuristics that these commands use to guess a networkwide pathname for the new storage directory will fail. (These heuristics are described in *Ensuring Global Access to the VOB—Special Cases for UNIX* on page 128.) On such hosts, you must use the **–host**, **–hpath**, and **–gpath** options to **mkvob** and **mkview** to ensure that valid information is recorded in the ClearCase storage registries.

Specifying a Nonstandard Mount Directory

By default, **automount** mounts directories under **/tmp_mnt**. If a ClearCase host uses another location for a host's automatic mounts (for example, you use **automount –M**), you must specify it in the file**/var/adm/atria/config/automount_prefix**. For example, if your automatic mounts take place within directory **/autom**, place this line in the **automount_prefix** file:

```
/autom
```

2.7 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 host 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 Product Family 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.

3 Understanding ClearCase Access Controls

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 log-in 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 does 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 the **Environment Variables** button.
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, or a view that contains view-private files, 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 UNIX hosts, the privileged user is the *root* user.
- On Windows NT hosts, the privileged user is a member of a special domain group, the *ClearCase group*. See *Configuring Domains for Use with ClearCase* on page 43 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
- In dynamic views, view-private files, view-private directories, and derived objects

NOTE: The ClearCase scheduler maintains its own access control mechanism. See *Managing the Scheduler Access Control List* on page 388.

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 (`r`) or No (–)	Yes (`w`) or No (–)	Yes (`x`) or No (–)
Member of Object's Group	Yes (`r`) or No (–)	Yes (`w`) or No (–)	Yes (`x`) or No (–)
Other	Yes (`r`) or No (–)	Yes (`w`) or No (–)	Yes (`x`) or No (–)

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

cleartool describe myproj

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

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

Table 2 Protection Mode Digits for a ClearCase Object

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 30 and *Access Control for Views and View Objects* on page 35.

Sometimes ClearCase grants a process access to an object only if the process has access to one or more containing objects as well. For example, suppose that user sue wants to use a text editor to create a view-private file in a dynamic view. The text editor's process must have write permission for both the directory in which sue wants to create the file and the view itself.

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 host where the VOB storage directory resides 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 view-private file, the element has the same protection mode as the view-private 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 29.

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 29.

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
- Derived Objects

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, a UCM object, or a storage pool.

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 or a storage pool.

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 dynamic view, permissions on elements and versions interact with permissions on views and view-private files or directories to control access to both VOB and view data.

For example, you must check out a version of an element to modify the element. The element must grant permission to check out a version. In a dynamic view, checking out a version creates a view-private file. You must have permission to create the view-private file in both the view and the directory that contains the file. The containing directory, in turn, can be either an element version or a view-private directory.

In general, access to ClearCase data in a dynamic view requires a process to pass a series of tests:

- It must have access to the view
- It must have access to the containing directory
- It must have access to the element

In a 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.

Access Control for Dynamic Views

A dynamic view has these properties that are important for access control:

- **Owner.** The initial owner is the user of the process that creates the view.
- **Group.** The initial group is the primary group of the process that creates the view.
- **Protection mode.** The initial protection mode for a view is determined one way on UNIX hosts and another way on Windows hosts.
  - On Windows NT, a view initially has read, write, and execute permission for the owner and group, and it has read and execute permission for others. You can use the **Properties of View** dialog box to display the owner, group, and protection mode for a view. You cannot change the owner and group after the view is created. You can use the **chview** command to change the protection mode to read/write or read-only.
  - On UNIX, the initial protection mode depends on the umask of the user who creates the view. A umask is a UNIX setting that specifies that some permissions are *not* granted when the user creates a file. (For details, see the **umask(1)** reference page.) When a user creates a view, ClearCase begins with read, write, and execute permissions for all users and then removes the permissions specified by the user's umask. For example, if the user's umask is 002, ClearCase removes write permission for others.

    To locate the view storage directory and display the owner, group, and protection mode for a view, use the **cleartool lsview** command with the **-properties** and **-full** options:

    ```
    cleartool lsview -properties -full my_view
    * my_view          /net/myhost/views/myview.vws
    .
    .
    .
    Owner: sue        : rwx (all)
    Group: clearusers : rwx (all)
    Other:            : r-x (read)
    ```

    Use the UNIX **ls** command to list the view storage directory:

    ```
    cd /net/myhost/views
    ls –ld myview.vws
    drwxrwxr-x  6  sue  clearusers  512  Nov 10 11:29 myview.vws
    ```

The output of this command shows the view's owner (`sue`), group (`clearusers`), and protection mode (`drwxrwxr-x`). Depending on your UNIX system, you may need to use **ls -g** to view the group.

Permission to Create Views

Any user can create a view.

Permission to Delete Views

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

Permission to Read Views

A process must have read permission for both a dynamic view and a file or directory in the view to read the file or directory. To read a version of a file or directory element, the process must have read permission for the element. See *Permission to Read Elements* on page 32. To read a view-private file or directory, the process must have read permission for the view-private file or directory. See *Permission to Read View-Private Files* on page 39.

ClearCase uses an algorithm that considers the process's user and group and the view's owner, group, and protection mode to determine whether to grant read permission for a view. See *Access to ClearCase Data* on page 29.

Permission to Write Views

A process must have write permission for a view to perform some operations that change the view itself, such as setting its config spec.

A process must have write permission for both a dynamic view and a containing directory in the view to create or delete a file or directory in the containing directory. If the containing directory is an element version, the process must have write permission for the element. See *Permission to Write Elements* on page 32. If the containing directory is a view-private directory, the process must have write permission for the view-private directory. See *Permission to Write View-Private Files* on page 40.

ClearCase uses an algorithm that considers the process's user and group and the view's owner, group, and protection mode to determine whether to grant read permission for a view. See *Access to ClearCase Data* on page 29.

Permission to Execute Views

A process must have execute permission for both a dynamic view and a file or directory in the view to execute the file or directory. To execute a version of a file or directory element, the process must have execute permission for the element. See *Permission to Execute Elements* on page 33. To execute a view-private file or directory, the process must have execute permission for the view-private file or directory. See *Permission to Execute View-Private Files* on page 40.

ClearCase uses an algorithm that considers the process's user and group and the view's owner, group, and protection mode to determine whether to grant execute permission for a view. See *Access to ClearCase Data* on page 29.

Access Control for View-Private Files

This section discusses access control for view-private files in dynamic views. In a snapshot view, native file-system permissions on directories and files in the snapshot view directory tree determine access to those directories and files.

A view-private file or directory has these properties that are important for access control:

- **Owner.** The initial owner is the user of the process that creates the file or directory.
- **Group.** The initial group is the primary group of the process that creates the file or directory.
- **Protection mode.** The initial protection mode for a view-private file is determined one way on UNIX hosts and another way on Windows hosts.
  - On UNIX hosts, the initial protection mode depends on the umask of the user who creates the file or directory. A umask is a UNIX setting that specifies that some permissions are *not* granted when the user creates a file. (For details, see the **umask(1)** reference page.) When a user creates a view-private file or directory, ClearCase begins with a set of permissions that depend on how the file or directory is created. ClearCase then removes the permissions specified by the user's umask. For example, if the user's umask is 002, ClearCase removes write permission for others.

    You can use the **cleartool describe** command or the UNIX **ls** command to display the owner, group, and protection mode for a view-private file or directory. You can use the UNIX **chown** command to change the owner, the **chgrp** command to change the group, and the **chmod** command to change the protection mode.

- On Windows hosts, a view-private file or directory initially has read, write, and execute permission for all users.

  You can use the **cleartool describe** command to display the owner, group, and protection mode for a view-private file or directory. The **ClearCase**>**Properties of File** or **ClearCase**>**Properties of Directory** dialog box displays the owner and group. The **Read-only** check box in either dialog box indicates whether all users have write permission.

  You cannot change the owner or group of a view-private file or directory. You can use the **Read-only** check box in the **ClearCase**>**Properties of File** or **ClearCase**>**Properties of Directory** dialog boxes or the **attrib +R** and **attrib –R** commands to specify whether all users have write permission. You cannot change any other permissions.

Permission to Create View-Private Files

A process must have write permission for both the view and a containing directory in the view to create a file or directory in the containing directory. For view permissions, see *Permission to Write Views* on page 37.

If the containing directory is an element version, the process must have write permission for the element. See *Permission to Write Elements* on page 32. If the containing directory is a view-private directory, the process must have write permission for the view-private directory. See *Permission to Write View-Private Files*.

Permission to Delete View-Private Files

A process must have write permission for both the view and a containing directory in the view to delete a file or directory in the containing directory. For view permissions, see *Permission to Write Views* on page 37.

If the containing directory is an element version, the process must have write permission for the element. See *Permission to Write Elements* on page 32. If the containing directory is a view-private directory, the process must have write permission for the view-private directory. See *Permission to Write View-Private Files*.

Permission to Read View-Private Files

A process must have read permission for both the view and a view-private file or directory in the view to read the file or directory. For view permissions, see *Permission to Write Views* on page 37.

ClearCase uses an algorithm that considers the process's user and group and the view-private file or directory's owner, group, and protection mode to determine whether to grant read permission for the file or directory. See *Access to ClearCase Data* on page 29.

Permission to Write View-Private Files

A process must have write permission for both the view and a view-private file or directory in the view to write the file or directory. For view permissions, see *Permission to Write Views* on page 37.

ClearCase uses an algorithm that considers the process's user and group and the view-private file or directory's owner, group, and protection mode to determine whether to grant write permission for the file or directory. See *Access to ClearCase Data* on page 29.

Permission to Execute View-Private Files

A process must have execute permission for both the view and a view-private file or directory in the view to execute the file or directory. For view permissions, see *Permission to Write Views* on page 37.

ClearCase uses an algorithm that considers the process's user and group and the view-private file or directory's owner, group, and protection mode to determine whether to grant execute permission for the file or directory. See *Access to ClearCase Data* on page 29.

Access Control for Derived Objects

A *derived object* (DO) is a file created in a dynamic view by **clearmake** or during a **clearaudit** session. These commands do not create derived objects in snapshot views.

A derived object is at first like a view-private file. You must have the same permissions to create a DO as to create a view-private file. A DO has an owner, group, and protection mode, determined initially in the same way as those of a view-private file. See *Access Control for View-Private Files* on page 38.

A shareable derived object is one that other dynamic views can use by *winking in* the DO. When a shareable DO is winked in for the first time, it is *promoted* from the view in which it was created to become an object in the containing VOB. This VOB object is a shared DO.

A shared DO has an owner, group, and protection mode. The owner and group are initially those of the shareable DO at the time it is promoted to the VOB. The group of a shareable DO must be one of the VOB's groups for the DO to be promoted to the VOB.

A shared DO's owner, the VOB owner, or the privileged user can use the **cleartool protect** command to change the DO's owner, group, or protection mode.

A process that winks in a shared DO to a dynamic view must have read permission for the DO. ClearCase uses an algorithm that considers the process's user and group and the DO's owner, group, and protection mode to determine whether to grant read permission for the DO. See *Access to ClearCase Data* on page 29.

A winked-in DO is like a copy-on-write view-private file in the dynamic view that winks it in. It has the owner, group, and protection mode of the shared DO. If you change a winked-in DO in the view, such as changing its permissions or writing it, ClearCase converts the DO to a view-private copy.

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 each VOB and view host. 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 36, *Repairing VOB and View Storage Directory ACLs on Windows NT*.

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

add or remove group write permission for files in a snapshot view directory that are not under ClearCase control.

Like a dynamic view, a snapshot 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 for a snapshot view, just as it does for a dynamic view. Never change native file-system permissions on the view storage directory for a snapshot view.

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 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 52 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 36, *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 set of VOBs and views. 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, element, and view-private file 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 Group

The *ClearCase group* (also called the *ClearCase server process group*) is a Windows NT global group used to enable ClearCase server processes to access all VOB and view storage directories within a community. Each community has exactly one ClearCase group. ClearCase creates all VOB and view storage directories with ACLs that assign sufficient access rights to the ClearCase group.

Do not confuse this group with the ClearCase user groups discussed in *ClearCase Users and User Groups*. The ClearCase group is used only by ClearCase server processes and ClearCase administrators. ClearCase user groups are used by members of the community to control access to ClearCase data such as individual elements.

NOTE: Windows NT defines the concepts of global group and local group. Be sure that the ClearCase group is created as a global group.

By convention, the name of the ClearCase group is **clearcase**. A community can choose a different name as long as all ClearCase hosts in the community define the ClearCase group as that group name. If multiple communities share a single Windows NT domain, each community must have a unique name for its ClearCase group.

WARNING: ClearCase grants the members of the ClearCase group special administrative privileges (for example, the ability to delete ClearCase elements without being the element or VOB owner). Therefore, only ClearCase administrative user accounts and the ClearCase server process user ought to be members of this group.

ClearCase Server Process User

ClearCase server processes run as a Windows NT user account. This user account is called the *ClearCase server process user*. By convention, the name of the user account is **clearcase_albd**. However, a community can choose a different account name as long as all ClearCase hosts in the community define the ClearCase server process user as that account name.

This user must be a member of the *ClearCase 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 **albd_server** on each ClearCase system must run as the *ClearCase server process user*, typically called **clearcase_albd**. There must be only one ClearCase server process user; it

must exist in only one domain. This account must be a member of the single *ClearCase group*.

- The domains that the ClearCase client and server computers belong to must trust the domains that contain the users.
- The domains that the ClearCase client and server computers belong to must trust the domain that contains the *ClearCase server process user*.
- When ClearCase users who share data belong to more than one domain, additional restrictions apply. See *Multiple User Account Domain Support* on page 52.

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 the 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. 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 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, use the **User Manager for Domains** (**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 48).

   NOTE: If ClearCase users at your site belong to more than one domain, see *Multiple User Account Domain Support* on page 52 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 46

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 **clearcase_albd** and 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, use the **User Manager for Domains** (**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 Administration**>**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 24 for details.

4.4 Defining the ClearCase Server Process User and ClearCase Group

The *ClearCase server process user* and *ClearCase group* must be defined in a Windows NT domain (that is, they cannot be local to the computer on which you are running ClearCase Site Preparation). In multidomain environments, this Windows NT domain must be trusted by all domains that contain computers in the ClearCase community.

Typically, the ClearCase administrator defines the default ClearCase server process user and ClearCase group for the community by running the ClearCase Site Preparation Wizard.

If the user account and group for the community have already been created by running the ClearCase Site Preparation Wizard, those values are used as the default when you rerun the wizard to create a new release area for the current ClearCase release.

If the user account and group do not yet exist (as is the case when you are running ClearCase Site Preparation for the first time), the wizard attempts to create them. In this case, you must have domain *Administrator* privileges when running ClearCase Site Preparation. If you do not have these privileges, the wizard cannot create the account and group; someone with the appropriate privileges must create the account and group manually before you run ClearCase Site Preparation. (See *Creating the Account and Group Manually*.) After the account and group exist, you can specify them in the ClearCase Site Preparation Wizard (for security purposes, you must supply and confirm the password for the user account).

The account and group names specified during ClearCase Site Preparation are presented as the defaults when users run the ClearCase Installation program.

Creating the Account and Group Manually

The ClearCase installation program on Windows NT creates a ClearCase server process user and a ClearCase global group automatically. If you need to create this user and group manually, follow the procedure in this section. You must have domain *Administrator* privileges to create this user account and group.

To create the account and group:

1. Run the appropriate management tool:

   - On Windows NT, use the **User Manager for Domains** (**Start**>**Programs**>**Administrative Tools** >**User Manager for Domains**). Select **User** >**New Global Group**, 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, click **Action**>**New**>**Group**, and enter the group name.

2. Create a new **global** group called **clearcase** (or another group name the community has chosen). In the **Description** box, type the following text:

   Used exclusively by the ClearCase server process user and ClearCase administrative users.

3. Create a new user, **clearcase_albd** (or another user name the community has chosen), and put the group name defined in Step #2 in the new user's group list.

   - Select the **Password never expires** check box.

   - In the **Description** box, type the following text:

     Used exclusively by ClearCase servers

   - Give the **clearcase_albd** user the right to **Log on as a service**.

     On Windows NT, open the User Manager for Domains and click Policies to open the User Rights Policy editor. Click **Add**, then select **clearcase_albd** from the list of users in the **Add Users and Groups** dialog box. Select the **Show Advanced Rights** check box and select **Log on as a service** from the list of rights. Click **OK** to grant the **clearcase_albd** user the right to log on as a service.

     On Windows 2000, click **Control Panel**>**Administrative Tools** and run the **Local Security Policy** management console. Click **Local Policies**>**User Rights Assignment**. From the list displayed, right-click **Log on as a service** and select the **Security** task to open the local security policy setting dialog box. Click **Add** and select the **clearcase_albd** user from the list of users displayed. Click **OK** to grant the **clearcase_albd** user the right to log on as a service.

4.5 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. 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**) and the *ClearCase server process user* (typically called **clearcase_albd**) 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. Configure the Atria Location Broker service on every ClearCase host in each of these domains to log in as the **clearcase_albd** user in the primary ClearCase domain (in this case, **Atlanta\clearcase_albd**).

4. Create two more domain global groups, one in each of the other two 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).**

5. 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**.

6. 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** as the Value Name and select REG_DWORD as the value type.

   d. Click **OK** to open 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.

7. 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 24.

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 that is used in the UNC path that refers to the VOB storage directory. For example, if a VOB is registered with the global path **\\myserver\vobstorage\src_vob**, you can set the ACL on the **vobstorage** share to allow access only by the groups **Atlanta\clearusers**, **Boston\clearusers_Boston**, **Cupertino\clearusers_Cupertino**, and the *ClearCase group*.

To change the ACL on VOB storage directory:

1. In the Windows Explorer on the computer that hosts the share, 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.

4.6 Nondomain Systems

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

- They cannot host ClearCase VOBs or views used by other ClearCase hosts.
- They cannot access VOBs and views hosted on other Windows NT computers.

Nondomain systems can function as ClearCase clients in networks where all VOB and view storage resides on UNIX hosts.

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

For information about administering nondomain systems, see Chapter 6, *Configuring ClearCase in a Mixed Network*.

5 Understanding the ClearCase Multiversion File System

This chapter presents basic information about the ClearCase *multiversion file system* (MVFS). The information in this chapter applies only to use of ClearCase in dynamic views; ClearCase does not use the MVFS in snapshot views.

See Chapter 34, *Improving Client Host Performance* for information about improving performance of a ClearCase client host at the MVFS level.

5.1 What is the MVFS?

The MVFS is an extension to your native operating system. Most access by client programs to ClearCase data (in VOBs and views) goes through the host's MVFS when you use dynamic views. File-system objects stored in views and VOBs are called MVFS objects.

On UNIX computers, code that implements the MVFS is linked with a host's operating system. It can be linked statically, which requires generating a new version of the operating system that includes the MVFS, or dynamically, which means the MVFS code is loaded at system startup time. How the MVFS is linked depends on the type and version of the operating system.

On Windows NT computers, the MVFS is a file system driver. It is loaded by the Service Control Manager at system start time; an *Administrator* can also start the MVFS from the **ClearCase** program in Control Panel.

User-level software (for example, a compiler) references a pathname. The ClearCase MVFS, which processes all pathnames within VOBs, passes the pathname on to the appropriate

view_server process. Before accessing ClearCase data using the MVFS, you must activate a view and mount one or more VOBs. The VOB is mounted as a file system of type MVFS.

The MVFS is designed to present a selected combination of local and remote files as if they were stored in the native file system when accessed by your application. The selected files are versions of VOB files and view-private files.

The MVFS supports the following file types:

- Files
- Directories
- Symbolic links

You cannot create other file types within a VOB.

During build script execution, **clearmake** works with the MVFS to *audit* low-level system calls performed on ClearCase data, recording every instance when a file or directory is created, deleted, or opened for read access. Calls involving these objects are monitored:

- Versions of elements used as build input
- View-private files used as build input—for example, the checked-out version of a file element
- Files created within VOB directories during the build

NOTE: If the MVFS does not support all file-system semantics you need in your build environment, we recommend that you consider using snapshot views as an alternative. See *Developing Software with ClearCase* for more information.

For information about which on-disk and network file systems ClearCase supports for use with dynamic views, see *ClearCase and MultiSite Release Notes*.

Known Limitations of the MVFS on Windows NT

The MVFS has the following known limitations:

- File attributes such as System and Hidden are not supported.
- OS/2 Extended Attributes (EAs) are not supported.
- DOS sharing modes are not supported.

- If your application queries an MVFS volume for the amount of disk space, the MVFS always returns the value 512 MB.

  The reason for this behavior is as follows: the application may be about to perform operations that require space in the view storage directory or the VOB storage directory. The MVFS does not know whether the application needs to know how much disk space is available for view storage or for VOB storage. Therefore, the MVFS always returns a constant equal to 512 MB, so that the application always tries the operation. Thus, MVFS never causes an operation that may succeed to fail.

- Alternate, or short, names inside a VOB on an MVFS file system on Windows NT are not always valid.

  The MVFS cannot guarantee that a filename mapping from a particular long name to a particular short name (a name that is 8.3 compliant) is always valid. For example, a short name may already exist in another version of the given directory. In this case, ClearCase may return the wrong file data. As a result the MVFS does not return a short name for file names that are not 8.3 compliant.

  You can run applications that require short names on MVFS in one of the following ways:

  - By using pathnames that are entirely 8.3 compliant, including view and VOB tags and directory names

  - By creating relative symbolic links with short versions of the long names that are not 8.3 compliant

5.2 Issues with Using the MVFS on Windows NT

When you use the MVFS on Windows NT you must be aware of the issues described in this section.

Problems with Case-Sensitivity

On Windows NT, applications generally expect file systems, such as FAT and NTFS, to perform case-insensitive file lookup. Therefore, the ClearCase MVFS uses case-insensitive name lookup by default on Windows NT.

When you use MVFS on networks accessing both UNIX and Windows NT files, follow these guidelines:

- Configure ClearCase to use case-insensitive file lookup, which is the default setting.
- Disable automatic case conversion for your NFS client product.
- Do not rely on case-sensitive names for elements that you store in a VOB and that will be accessed by Windows NT clients.

You may also consider enabling the MVFS Case Preserving option, especially if you are using a Java compiler or other application that expects object file names in mixed case. To change this option, click **Start**>**Settings**>**Control Panel**. Open the **ClearCase** program and click the **MVFS** tab. Click **Help** for more information.

See also *Case-Sensitivity* on page 73.

Interaction with NFS Products

The MVFS requires some special configuration setup of NFS client products used to access UNIX VOBs and views from Windows. See Chapter 7, *Configuring UNIX File Access for Windows Computers* for more information.

Using Links with the MVFS

We strongly recommend that you use relative symbolic links, rather than absolute symbolic links. Absolute links require a full pathname of *\viewtag\vobtag\pathname*, that is, the absolute path you supply when in the *dynamic-views drive* (usually the **M:** drive). Absolute links work only in the view where they are created.

Using the VOB Root Directory

The VOB root directory has special properties that prevent it from caching the ENOENT entries of file names that do not exist in a directory. Performance may suffer if you put regular files into the VOB root directory. Therefore, we recommend that the VOB root be used primarily as a directory level where subdirectories are created and not as a direct storage location for individual files.

This recommendation is especially important for source files, include files, libraries, or executables that are in a search path and cause the VOB root to be searched frequently for file names that may not exist in it.

Running Executables in the MVFS

After you add an executable to a VOB, you must give it ClearCase execute permission to be able to run it from inside the MVFS. Do so with this command:

cleartool protect –chmod +x *executable_filename*

5.3 MVFS Performance Issues

Because it supports multiple versions of a file, the MVFS has a longer file name lookup procedure than typical native file systems, such as the Berkeley Fast File System (UFS or FFS) on UNIX or the FAT file system on Windows. The DNLC (Directory Name Lookup Cache) is important to improving the lookup speed of frequently accessed files.

You can improve performance, as follows:

- Monitor effectiveness using the **mvfsstat** and **mvfstime** commands.
- Monitor the number of **include**, **lib**, and paths directories. Consider using relative soft links to consolidate multiple include directories to reduce the number of **–I** directories in a build rule.

You may also improve performance by adjusting the cache size. See *Examining and Adjusting MVFS Cache Size* on page 441 for more details.

Administering a UNIX/Windows Network

6 Configuring ClearCase in a Mixed Network

If your network environment includes a mixture of hosts running Windows and UNIX operating systems, you can share ClearCase data across these platforms. In general, ClearCase hosts on Windows can access VOBs and views that are stored on ClearCase hosts running either Windows NT or UNIX. ClearCase hosts on UNIX can access VOBs stored on Windows NT hosts, but only from snapshot views. Dynamic view access from a ClearCase host on UNIX to a VOB on Windows is not supported. UNIX computers can also use the ClearCase Web interface to access data stored on Windows computers. This chapter assumes use of the native ClearCase interface.

This chapter describes the circumstances under which you may want to share ClearCase data across platforms and any constraints on cross-platform source code access and functionality.

6.1 When to Use ClearCase in a Mixed Network Environment

Working across platforms provides flexibility and may save time and development resources. It also requires extra planning and maintenance. The scenarios that follow represent situations in which it is beneficial to use ClearCase in a mixed environment:

- A group of Windows developers and a group of UNIX developers work on the same source code.

  To allow users on both Windows and UNIX to access the source code, the Windows developers use UNIX VOBs. However, the Windows developers have no need to see view-private changes to the UNIX source code, so they use Windows views for optimal performance.

- One group of developers maintains a code base for both UNIX and Windows.

  In this scenario, developers on Windows use the UNIX VOB in which the source code resides. They also use their UNIX views from their Windows computers so they have access to the same view-private files on both platforms.

- A group of Windows developers works primarily on a separate source code base but keeps the source code on a UNIX VOB for administrative purposes (for example, the backup procedure resides on UNIX).

  In this case, the developers use UNIX VOBs and Windows views.

When Not to Use a Mixed Environment

When you run ClearCase on Windows, you can access views and VOBs on both Windows and UNIX. If your developers do not need to access ClearCase data from UNIX computers, and you have no administrative reasons to maintain UNIX VOBs and views (for example, backup requirements), we recommend that you use Windows views and VOBs. This strategy yields better performance than does accessing UNIX VOBs and views from Windows computers

6.2 ClearCase Capabilities in a Mixed Environment

This section describes the access to ClearCase data that is possible between Windows and UNIX computers. The description assumes that you have configured the computers and your network to support cross-platform access to ClearCase data structures. Specifically:

- You have configured your ClearCase installation to enable all view- and VOB-access capabilities possible for that computer's operating system.

  The view- and VOB-access capabilities of a given ClearCase computer often are a function of how the computer was configured during ClearCase installation. This is especially true of ClearCase Windows computers. See *ClearCase Product Family Installation Notes* for details.

- You have installed a cross-platform file access product where necessary.

  If you are using snapshot views on either UNIX or Windows, you can enable ClearCase File Service (CCFS) to allow these views to access VOBs on either UNIX or Windows. You need not install a third-party file access product.

If you are using Windows NT dynamic views to access files and directories in UNIX VOBs or want to access UNIX dynamic views, you must install a third-party file-access product, such as an NFS client product or an SMB server product.

See Chapter 7, *Configuring UNIX File Access for Windows Computers* for details about CCFS and third-party file-access products.

Windows NT

Windows NT computers can access Windows snapshot and dynamic views and UNIX dynamic views. They cannot access UNIX snapshot views.

Windows NT computers can access Windows VOBs using Windows snapshot and dynamic views. They can access UNIX VOBs using Windows snapshot and dynamic views, and also using UNIX dynamic views.

NOTE: Although Windows NT computers can access UNIX dynamic views, they cannot use those UNIX views to access Windows NT VOBs.

Windows 95 and Windows 98

Windows 95 and Windows 98 computers can access Windows snapshot views whose view storage directories are located on Windows NT computers. They cannot access Windows NT dynamic views or UNIX snapshot or dynamic views.

Windows 95 and Windows 98 computers can access Windows NT and UNIX VOBs using Windows snapshot views.

UNIX

Using dynamic views, UNIX computers can access UNIX VOBs. Using snapshot views, UNIX computers can access UNIX VOBs or Windows NT VOBs. UNIX computers cannot access Windows snapshot or dynamic views.

UNIX computers can also use the ClearCase Web interface to access VOBs on Windows NT. For more information, see Chapter 29, *Configuring a Web Server for the ClearCase Web Interface*.

Constraints in a Mixed Environment

This section lists specific constraints to consider when using ClearCase in a mixed network environment:

- To access UNIX views, Windows NT computers must be configured to support *dynamic views*.
- ClearCase supports dynamic views only on UNIX and Windows NT computers; you cannot create or access dynamic views on a Windows 95 or Windows 98 computer.
- To avoid incompatibilities caused by differences in the way the Windows and UNIX operating systems address case-sensitivity, you may need to perform some additional configuration steps. See *Case-Sensitivity* on page 73 for details.
- You cannot move VOBs or views between UNIX and Windows computers.

  NOTE: You can use MultiSite facilities or the **clearexport_ccase** and **clearimport** commands to move ClearCase data from a Windows NT VOB to a UNIX VOB.
- You cannot create a UNIX VOB or view from a Windows computer. You cannot create a Windows VOB or view from a UNIX computer.
- You cannot use the **cleartool relocate** command to move elements between UNIX VOBs and Windows NT VOBs.
- A Windows computer cannot use a UNIX view that has symbolically linked view-private storage (**mkview –ln**).
- Various characters that are legal in UNIX file names are illegal in Windows NT file names. File names that include these characters are not recognized by the MVFS on Windows NT and cannot be loaded into a Windows snapshot view. VOB element names that include these characters are not visible in Windows views. Table 3 lists these characters.

Table 3 Characters Not Allowed in Windows File Names

?	*	/	\	|	<	>

6.3 Identifying License and Registry Servers

On all platforms, the ClearCase installation procedure prompts you to name a *license server host* and a *registry server host*. The license server and registry server can be any combination of UNIX and Windows NT computers, and either kind of server can service any combination of UNIX and Windows computers.

In general, a network has a single ClearCase registry server. Multiple license servers are common. The procedure for adding licenses on UNIX and Windows license servers is similar. For more information, see Chapter 24, *Administering ClearCase Licenses*.

NOTE: If the ClearCase license server is on a Windows NT computer, each user who accesses a UNIX view from a Windows computer requires two ClearCase licenses. This situation arises because of differences between UNIX and Windows in the implementation of user identities. If the requirement for multiple licenses is a problem, you can move the license server to a UNIX computer. If the ClearCase license server is on a UNIX computer, each user who accesses a UNIX view from a Windows computer requires only one ClearCase license. For more information on moving licenses to another server, see Chapter 24, *Administering ClearCase Licenses*.

6.4 Managing User Accounts

On UNIX, ClearCase relies on group IDs to implement workgroup sharing and to maintain strict access control. For details, see Chapter 3, *Understanding ClearCase Access Controls*. To access UNIX VOBs and views correctly:

> **Each Windows user's username and primary group name must match that user's username and primary group name on UNIX.**

Checking User and Group Assignments

Individual users can check the validity of their current user and group assignments with the **credmap** and **creds** utilities in *ccase-home-dir***etc\utils**. If the primary group is incorrect, follow the procedure described in *Setting the Primary Group, Special Considerations on Windows* on page 24 to set the **CLEARCASE_PRIMARY_GROUP** environment variable.

After your Windows primary group is established, you can verify that it matches the corresponding UNIX user and group IDs with *ccase-home-dir***etc\utils\credmap**. The **credmap**

utility takes one argument: a target UNIX computer that is running the VOB or view server. The following command checks the user and group IDs for user **anne** on Windows against their counterparts on UNIX computer **saturn**:

ccase-home-dir\etc\utils> **credmap saturn**

```
Identity on local Windows system:
  User: anne (0x1003f2)
  Primary group: user (0x1003ff)
  Groups:
    Administrators (0x20220)
    Domain Users (0x100201)

Identity on host "saturn":
  User ID: 1149 (0x47d)
  Primary group ID: 20 (0x14)
  Group ID list:
    -2 (0xfffffffe)
```

In this example, if Anne is unsure whether the UNIX user ID and primary group ID values are correct (they must correspond to UNIX user **anne** and group **user**), she can use the **id** command on a UNIX system:

id

```
uid=1149(anne) gid=20(user)
```

ClearCase and Nondomain Accounts

A Windows NT *domain* is a set of Windows systems that share a common user account database. The account database resides on one or more Windows NT Server systems. Each Windows NT system belongs to at most one domain. Windows NT systems that do not belong to any Windows NT domain have restricted access to other Windows NT systems.

Because domain membership is required to support Windows VOB and view access, we strongly recommend that all Windows NT systems running ClearCase belong to a Windows NT domain. Nevertheless, you can access a UNIX VOB or view from any Windows NT system, regardless of whether the system is a member of a Windows NT domain.

Each Windows NT system, regardless of whether it belongs to a domain, has a local, single-system user account database. Each user account in this database is assigned primary group **None**. Users who do not log on to domain user accounts (because the system is not in a

domain, or because the users choose to use a single-system account) must set **CLEARCASE_PRIMARY_GROUP** before they try to access a UNIX VOB or view. For more information, see *Checking User and Group Assignments* on page 69 and *Setting the Primary Group, Special Considerations on Windows* on page 24.

UNIX VOB Group Lists and Registered User Groups

As it manages VOB access, ClearCase must routinely resolve VOB group list entries on both Windows NT and UNIX computers. Therefore, the Windows NT *administrator* must use the **User Manager for Domains** program to add the primary and additional group names stored with each applicable UNIX VOB to the Windows NT domain.

For example, the following command, run on a UNIX host, displays the group names stored with UNIX VOB **/vobs/libvob2**:

cleartool describe vob:/vobs/libvob2

```
versioned object base "/vobs/libvob2"
  created 01-Feb-96.10:54:35 by vobadm.user@saturn
  "runtime libraries"
  VOB storage host:pathname "venus:/usr1/vobstore/libvob.vbs"
  VOB storage global pathname "/net/venus/usr1/vobstore/libvob.vbs"
  VOB ownership:
    owner: vobadm
    group: user
  Additional groups:
    devel
    gui
```

If the VOB is accessible from Windows, all three groups—**user**, **devel**, and **gui**—must be valid Windows NT domain groups, and Windows NT users must belong to at least one of these groups.

Note the following about group and user access between Windows and UNIX:

- Although you can define an unlimited number of groups on Windows NT, only the first 32 Windows NT domain-level groups for each user are considered by UNIX.
- In a Windows NT domain, user and group names are in the same namespace. Therefore, if a UNIX VOB uses the same name for both a user and group name, a ClearCase Windows computer has trouble accessing this VOB.

For example, if a UNIX VOB allows user access for a **vobadm** user and group access for a **vobadm** group, a ClearCase Windows computer cannot access this VOB.

6.5 ClearCase Access Control on UNIX and Windows

This section presents the differences between the UNIX and Windows implementations of ClearCase security features. For more information on ClearCase access control, see Chapter 3, *Understanding ClearCase Access Controls*.

- **View protection mode**—A Windows view always has read, write, and execute permission for the view's owner and group, and it has read and execute permission for others. On UNIX, a view's protection mode is determined by the umask of the view's creator.

- **View-private file protection mode**—In a Windows dynamic view, view-private files (including checked-out files) initially have read, write, and execute permission for all users. You can later specify whether all users have write permission, but you cannot change any other permissions. On UNIX, a view-private file's initial protection mode is determined by the umask of the file's creator, and you can change it later.

- **Reporting protection modes for elements**—On Windows, you cannot use standard operating system commands to view ClearCase protection modes for elements. Instead, in Windows Explorer, click **ClearCase**>**Properties of Element** on an element's shortcut menu, and then click the **Protection** tab.

- **Adding executable files to source control**—When you add an executable file to source control on Windows, ClearCase makes the file read-only by setting its protection to 0444. To make the file executable, run the following command:

  cleartool protect –chmod +x *filename*

  Alternately, from Windows Explorer, select the executable, and click **ClearCase**>**Properties of Element** from the shortcut menu. On the **Protection** tab, select **Execute** under **Owner**, **Group**, or **Other**, as appropriate.

6.6 Case-Sensitivity

In general, you can work with ClearCase on either operating system with little consideration for case-sensitivity issues. In the event you encounter unexpected behavior, read this section carefully.

This section applies only to the ClearCase multiversion file system (MVFS), and thus applies only to dynamic views.

General Recommendations

To avoid common problems, follow these general recommendations:

- Click **Start**>**Settings**>**Control Panel**. Open the **ClearCase** program. On the **MVFS** tab, verify that ClearCase is using **Case-Insensitive MVFS** file lookup. This is the default setting.
- Disable automatic case conversion for your NFS client or SMB server product.
- Do not rely on case-sensitive names (names that differ only in the case of their characters) for elements that you store in a VOB.
- Consider enabling the **Case Preserving** option (on the **MVFS** tab of the **ClearCase** Control Panel program), especially if you are using a Java compiler or other application that expects object file names in mixed case.

  WARNING: If you change the setting of the **Case Preserving** option, you may encounter problems when you write new versions of files that were created when the previous setting was in effect. For example, conflicts may arise if a development environment automatically generates files with mixed-case and uppercase names.

The remainder of this section explains the pertinent case-sensitivity issues on both UNIX and Windows and discusses each recommendation in more detail.

Case Considerations On UNIX

On UNIX, the native file system and all ClearCase components (MVFS, **cleartool**, and so on) are case-sensitive. The MVFS uses case-sensitive file lookup and performs no automatic case conversion of any kind.

Case Considerations On Windows

Regardless of how you configure ClearCase, all ClearCase applications, including **cleartool**, are case-sensitive on both UNIX and Windows. Generally speaking, so is **clearmake**. (For more information on building with **clearmake** on Windows, see *Building Software with ClearCase*.)

The remainder of this section discusses issues of case-sensitivity for dynamic views on Windows. Snapshot views use native Windows file systems (FAT, NTFS, and so on) to read and write files. These file systems perform case-insensitive file lookup and generally preserve case when writing files. We recommend that when using snapshot views, you avoid using VOB elements whose names differ only in capitalization.

Because native Windows file systems perform case-insensitive file lookup, the ClearCase MVFS defaults to case-insensitive name lookup on Windows.

By default, native Windows NT file systems preserve case on file-creation operations. The default MVFS behavior is to convert file names to lowercase.

On Windows NT, the MVFS options for file name case (on the **MVFS** tab in the **ClearCase** Control Panel program) are as follows:

- When you select the **Case Insensitive MVFS** check box (the default and recommended setting), the MVFS looks up files without regard to case. With this setting, when the MVFS writes view-private files, capitalization of file names depends on the setting of the **Case Preserving** check box:
  - When you select the **Case Preserving** check box, the MVFS preserves the case of the names of new view-private files.
  - When you clear the **Case Preserving** check box, the MVFS converts the names of new view-private files to lowercase. This is the default setting.
- When you clear the **Case Insensitive MVFS** check box, the MVFS uses case-sensitive file lookup. With this setting, the MVFS preserves the case of the names of all files it creates.

NOTE: If you change any MVFS options, the new settings do not take effect until you restart your computer.

In the default mode (**Case Insensitive MVFS** enabled, **Case Preserving** disabled), the MVFS converts to lowercase the names of view-private files it creates using non-ClearCase commands, tools, and utilities (for example, **.obj** files and editor backup files). This behavior can combine with **cleartool** case-sensitivity to produce results such as the following:

```
cl –c util.c                                    (Compiler creates util.OBJ)
dir
...
02/24/95  12:52p    12,765   util.obj           (The MVFS uses lowercase for util.OBJ)
...
cleartool checkin util.OBJ                      (cleartool is case-sensitive; checkin fails)
cleartool checkin util.obj                      (Correct; checkin succeeds)
```

Keep this example in mind when coding complex sequences of file-processing commands in makefiles. Make sure that any **cleartool** commands that are run from a **makefile** and that operate on derived objects use names in the correct case.

If you clear **Case-Insensitive MVFS** and select **Case-Preserving**, the results are as follows:

```
cl –c util.c                                    (Compiler creates util.OBJ)
dir
...
02/24/95  12:52p    12,765   util.OBJ           (The MVFS preserves uppercase for util.OBJ)
...
cleartool checkin util.obj                      (cleartool is case-sensitive; checkin fails)
cleartool checkin util.OBJ                      (cleartool is case-sensitive; checkin succeeds)
```

When to Use Case-Sensitive Mode

In general, we recommend that you use case-insensitive mode whenever possible. Use case-sensitive mode only if VOB files have names that differ only in capitalization. In this mode, many Windows applications may fail when they try to access files using incorrectly capitalized names. We recommend that for files you plan to access from Windows computers, you not rely on names that differ only in capitalization. A case-insensitive MVFS on Windows NT cannot distinguish between the two files. Under these circumstances, the results of any file access are undefined.

NFS Client Products and Case-Sensitivity

NFS client software for Windows NT computers affects the behavior that ClearCase users see. Most NFS client products can convert to lowercase the file names they create, and some do so by default. To avoid file access problems caused by case-conversion conflicts, we strongly recommend that you *disable automatic case conversion for NFS client products*.

Be aware that if you disable case conversion and create UNIX files without MVFS intervention (that is, you create files outside any VOB namespace), you may end up with UNIX files with names like **UTIL.OBJ**, because some PC tools generate uppercase names. This does not happen for a view-private object that you create in a VOB's namespace if the **Case Preserving** option is disabled (the default setting). In this case, the MVFS converts the name to lowercase when it creates the file

.

7 Configuring UNIX File Access for Windows Computers

To allow Windows computers to access UNIX VOBs and views, ClearCase supports three methods:

- **ClearCase File Server (CCFS)** — The ClearCase File Server is a TCP/IP-based mechanism for file transfers between UNIX VOB servers and ClearCase clients on Windows.

  CCFS only supports snapshot views.

  CCFS is the sole file transfer mechanism supported by ClearCase for Windows 95 and Windows 98 clients.

  For information about CCFS, see *ClearCase File Server* on page 78.

- **NFS client products** — You install an NFS client product on each Windows NT computer from which you want to access UNIX VOBs and views.

  For more information about which NFS client products ClearCase supports and how to configure them, see *NFS Client Products* on page 79.

- **SMB server products** — You install an SMB server product on each UNIX VOB or view server you will access from a Windows NT client.

  For more information about which SMB server product ClearCase supports and how to configure it, see *SMB Server Products* on page 87.

Table 4 lists supported file-access mechanisms between ClearCase client platforms and VOB and view servers.

Table 4 Mechanisms for File Access Between ClearCase Clients and VOB and View Servers

Client Platform	File Access to Windows NT VOB and View Servers	File Access to UNIX VOB Servers	File Access to UNIX View Servers
Windows 95, Windows 98	Native SMB	CCFS	Unsupported
Windows NT (using dynamic views)	Native SMB	Third-party NFS or TotalNET SMB	Third-party NFS or TotalNET SMB
Windows NT (using snapshot views)	Native SMB	CCFS, third-party NFS, or TotalNET SMB	Third-party NFS or TotalNET SMB
UNIX (using dynamic views)	Unsupported	Native NFS	Native NFS
UNIX (using snapshot views)	CCFS	Native NFS	Native NFS

NOTE: You must evaluate which mechanism for file access best meets your specific requirements. Each solution has its own strengths, which vary depending on many factors, including these:

- The characteristics of a site's networks
- The speed and relative locations on the networks of the client and server systems
- The capacity and load of those systems
- The size of data being transferred
- Developer behavior

7.1 ClearCase File Server

The ClearCase File Server is a TCP/IP-based mechanism for file transfers between UNIX VOB servers and ClearCase clients on Windows.

If a ClearCase client on Windows NT has enabled CCFS and uses only snapshot views, no third-party NFS client-based or SMB server-based product is necessary to access UNIX VOBs. Dynamic views on Windows NT computers still require a supported NFS client or SMB server product to access UNIX VOBs.

ClearCase clients running Windows 95 or Windows 98 use CCFS as their sole transfer mechanism when accessing UNIX VOBs. Because Windows 95 and Windows 98 computers cannot use dynamic views or run view servers, you must have at least one Windows NT computer that will run the view server process and contain the view storage directory for snapshot views created on Windows 95 and Windows 98 computers.

When CCFS is enabled, file transfers between Windows clients and UNIX VOB servers in ClearCase client applications and snapshot view operations (such as check out, check in, snapshot view creation, and update) take place over a standard TCP/IP connection. File transfers between Windows NT view servers and UNIX VOB servers also use this TCP/IP connection.

Enabling and Disabling CCFS

To enable and disable CCFS on a Windows NT computer:

1. Click **Start**>**Settings**>**Control Panel**. Open the ClearCase program.
2. On the **Options** tab, select the **Use CCFS to access UNIX VOBs** check box to enable use of CCFS. Clear this check box to disable use of CCFS.

   Click **OK**.
3. Shut down and restart the computer to ensure that the change takes effect for all processes.

NOTE: When CCFS is disabled (which is the default setting), you must have a supported NFS client or SMB server product to access UNIX VOBs.

7.2 NFS Client Products

ClearCase supports these NFS client products on Windows NT computers:

- Microsoft Windows NT Services for UNIX Client for NFS Products (SFU)

- Intergraph DiskAccess
- Hummingbird NFS Maestro

If you are using an NFS client product, you must install it on each Windows NT client that will access VOBs or views located on UNIX servers. You must install the product correctly and completely; in particular, you must assign and configure the NFS daemon and authentication process.

NOTE: The *READ ME FIRST* chapter in *ClearCase and MultiSite Release Notes (Windows Edition)* contains last-minute information about NFS client products, including which versions of those products ClearCase supports. Make sure that you read the remainder of *ClearCase and MultiSite Release Notes (Windows Edition)* for other information about configuring NFS client products.

The rest of this section describes configuration procedures specific to using an NFS client product with ClearCase. *Read and perform all procedures recommended for your product.*

Disabling Automatic Case Conversion

Some NFS client products change the case of file names by default, typically by converting to lowercase. Because ClearCase is case-sensitive, you need to disable case conversion, as described later in this section.

NOTE: Typically, you can use a command-line option to disable case conversion for a particular NFS mount. However, ClearCase can *automount* remote storage directories. See *Automounting and NFS Client Software* on page 83. For correct behavior on these mounts, configure NFS mount drive options to disable case conversion.

Microsoft SFU and Intergraph DiskAccess

To disable automatic case conversion:

1. Start the Client for NFS (for SFU) or DiskAccess (for DiskAccess) Control Panel program.
2. On the **Filenames** tab, click **Preserve Case (no conversion)**.

Hummingbird NFS Maestro

To disable automatic case conversion:

1. Click **Start**>**Settings**>**Control Panel**. Open the Network program.

2. On the **Services** tab, select **NFS Maestro for Windows NT Client**.

3. Click **Properties** to open the client configuration dialog box.

4. Under **Filename Capitalization**, click **Preserve Case**.

Setting an NFS Client's Default Protection

If you plan to work in a shared UNIX view, configure your NFS client with a default protection that grants group write access. Without this permission, other developers cannot modify view-private files that you have created.

Microsoft SFU or Intergraph DiskAccess

To set the default protection:

1. Start the Client for NFS (for SFU) or DiskAccess (for DiskAccess) Control Panel program.

2. On the **File Access** tab, ensure **User** is `RWX`, **Group** is `RWX`, and **Other** is `RX`.

Hummingbird NFS Maestro

To set the default protection:

1. Click **Start**>**Settings**>**Control Panel**. Open the Network program.

2. On the **Services** tab, select **NFS Maestro for Windows NT –Client**.

3. Click **Properties** to open the client configuration dialog box.

4. Under **Default Protection**, specify the protections as follows:

```
User Group Other
RWX  RWX   RWX
xxx  xxx   x x
```

Setting the Correct Logon Name

To avoid VOB and view access permission problems, do not log on to an NFS server as user **nobody** or with any user or group ID that does not match your Windows NT user and primary group IDs.

To verify that your Windows NT user name/UID and group name/GID match their UNIX counterparts, pass the name of a UNIX NFS server to *ccase-home-dir***\etc\utils\credmap**. For example:

*ccase-home-dir***\etc\utils\credmap saturn**

After you confirm your user name/UID and group name/GID, supply the user name as an NFS logon parameter.

Microsoft SFU or Intergraph DiskAccess

To set your logon user name:

1. Start the Client for NFS (for SFU) or DiskAccess (for DiskAccess) Control Panel program.
2. On the **Authentication** tab, type the correct **User Name**, **Password**, and **PCNFSD Server**.
3. Click **OK**; your logon session is validated.

Hummingbird NFS Maestro

To set your logon user name; at the command prompt, run the **nfs register** command:

nfs register *username*

This command prompts for a password.

Hummingbird NFS Maestro: Disabling DOS Sharing

The Maestro **DOS Sharing** option is incompatible with ClearCase use. When using Hummingbird NFS Maestro with ClearCase, you must disable this mode. If you do not, you may encounter MVFS log errors when attempting to open MVFS files, such as

```
ZwOpenFile returned status 0xc0000043
```

which indicates a sharing violation.

To disable DOS Sharing:

1. Start the Network program in Control Panel.
2. On the **Services** tab, select **NFS Maestro for Windows NT –Client**.
3. Click **Properties** to open the client configuration dialog box.
4. Under **Default Links**, clear the **DOS-Style Sharing** check box.

Automounting and NFS Client Software

When you mount a UNIX VOB or start a UNIX dynamic view, ClearCase needs to access the *VOB storage directory* or *view storage directory* on the UNIX file system partition. In the ClearCase program in Control Panel, the setting of the **Enable automatic mounting of NFS storage directories** check box determines how ClearCase accesses those directories when you use NFS client products.

All supported NFS client products can process UNC names. If you are using one of these products, clear the **Enable automatic mounting of NFS storage directories** check box. ClearCase then uses UNC names to access UNIX VOB and view storage directories. We recommend that you configure ClearCase hosts in this way.

If you have been using ClearCase with the **Enable automatic mounting of NFS storage directories** check box selected, you can continue to do so. ClearCase then maps Windows drive letters to UNIX VOB and view storage directories and accesses the directories through those drive letters.

NOTE: These drive letters are for internal ClearCase use. They are different from the drive letters you can assign and use for your own work within views. In particular, do not confuse them with the drive letters you can assign to dynamic views when you start those views.

We recommend that you disable automounting when using any supported NFS client product. If you install Microsoft SFU or Intergraph DiskAccess before you install ClearCase, the ClearCase installation procedure disables automounting for you.

If you are using Hummingbird NFS Maestro, or if you install a supported NFS product after you have installed ClearCase, you can disable automounting using the ClearCase program in Control Panel.

1. Click **Start>Settings>Control Panel.** Open the ClearCase program.
2. On the **Options** tab, clear the **Enable automatic mounting of NFS storage directories** check box.
3. Click **OK**.

Microsoft SFU or Intergraph DiskAccess: Setting Up the ClearCase Server Process User and ClearCase Group

SFU and DiskAccess use a scheme to map Windows NT user credentials to NFS user and group ID. This scheme requires that you perform an additional setup procedure for the special ClearCase server process user account, as described in *Defining the ClearCase Server Process User and ClearCase Group* on page 50.

The setup procedure enables ClearCase services to access remote NFS view and VOB storage directories with the proper privileges, which enables operations such as cleartext construction.

Cleartext construction is the process by which data is extracted from the VOB storage area source container by a ClearCase type manager. This type manager then constructs or creates a cleartext container the first time a version is accessed. For faster access, subsequent reads of that version access the cleartext file directly.

Setting Up the UNIX Account

On UNIX, the group ID (GID) is used for the permission to construct cleartext. The UNIX primary group name must match the user's Windows NT primary group name (see *Managing User Accounts* on page 69).

NOTE: By convention, the name of the ClearCase group is **clearcase**. A community can choose a different name as long as all ClearCase hosts in the community define the ClearCase group as that group name. If multiple communities share a single Windows NT domain, each community must have a unique name for its ClearCase group.

As part of setting up SFU or DiskAccess, the administrator needs to supply the ClearCase server process user with a valid UNIX name and password for a user account that belongs to the correct primary group. The administrator can identify the UNIX account by performing one of the following tasks:

- Add a UNIX account that matches the Windows NT name of the ClearCase server process user account.

- Add a new UNIX account that does not match the Windows NT name.
- Use an existing UNIX account.

Preparing the Windows NT Client

Set up the ClearCase server process user account (**clearcase_albd**) on every ClearCase Windows NT client that will access remote NFS views or VOBS. For the first client, follow the steps below. For subsequent clients, either follow the steps here or perform the steps in *Alternative Setup: Administrative Option*.

1. Install SFU or DiskAccess, restart, and set up your regular user account, from which you will log on to SFU or DiskAccess.
2. Log off from your current user session.
3. Log on as the ClearCase server process user on your system.
4. Start the Client for NFS (for SFU) or DiskAccess (for DiskAccess) Control Panel program.
5. On the **Authentication** tab, specify the UNIX name and password that you established in *Setting Up the UNIX Account*.
6. Click **OK**.

   When you exit the Client for NFS (for SFU) or DiskAccess (for DiskAccess) program, read the confirmation dialog box. Verify that you are currently logged on with the desired UID and GID.

   If there is an error, or if you are logged on with UID and GID of -1 and -2, repeat Steps 3 through 6 in this section.
7. Log off.

Alternative Setup: Administrative Option

Instead of performing the steps in *Preparing the Windows NT Client*, you may instead perform the steps in this section. For example, it may be inconvenient to log on as the ClearCase server process user on every client system. As an alternative, perform the following steps:

1. Log on as the ClearCase server process user (**clearcase_albd**) on one system.
2. Run the Windows Registry Editor (type **regedit** in the **Run** dialog box or from a command prompt).

3. Carefully save the *security-id* key under:

   HKEY_LOCAL_MACHINE\SOFTWARE\Intergraph\DiskAccess\CurrentVersion\Users

   Select the *security-id* subkey whose **Name** key value is your ClearCase server process user account name. To find the correct subkey, type the following command, and look on the line in the output that starts with `SID`:

   *ccase-home-dir***\etc\utils\creds clearcase_albd**

   NOTE: If your community uses a ClearCase server process user account name other than **clearcase_albd**, use that name instead.

4. Save the key to a registry file by clicking **Registry**>**Export Registry File** in the Registry Editor.

   To share the registry file, mail it to users or place it on a shared directory. When users want to load the file into the registry, they double-click the file name in a file browser such as Windows Explorer. The system displays a confirmation message after loading the file.

   All users who use the same ClearCase server process user account and the same primary Windows NT group that maps to the UNIX GID to which it was logged on can share this file.

Microsoft SFU: Configuring the Default LAN

Browsing of UNIX resources from Windows NT is not enabled by default on SFU. You must enable this feature manually for it to work. Otherwise, users cannot browse for UNIX resources using either **net view** at the command prompt or **Network Neighborhood** on the desktop. After configuring the **Client for NFS**, take the following steps:

1. Start the Client for NFS Control Panel program.

2. On the **Configured NFS LANs** tab, click **Add**, enter a name for the LAN (for example, Default_LAN), click **Specify LAN to browse**, and then click **OK**.

3. On the **Configured NFS LANs** tab, click **Edit**, and in the **Broadcast Address** box enter 255.255.255.255. You can either accept the defaults or customize the rest of the broadcast parameters.

4. Click **OK** in the **Broadcasting** dialog box; then click **OK** in the **Client for NFS** dialog box.

7.3 SMB Server Products

ClearCase supports the Syntax TotalNET Advanced Server (TAS) SMB server product.

If you are using an SMB server product, you must install and configure it on each UNIX VOB and view server that you want to access from a Windows NT client.

NOTE: Using TAS to provide mixed-environment file access for ClearCase is supported only on Solaris 2.5.1 or later and HP-UX 11.00 or later. A VOB or view server must be on a Solaris or HP-UX computer if TAS is to provide access from Windows NT clients. (See the Rational ClearCase Web page (**http://clearcase.rational.com**) for current UNIX platform support for ClearCase and TAS.) If a VOB or view server is on another UNIX platform not yet supported for TAS, use an NFS client product, as described in *NFS Client Products* on page 79.

The *READ ME FIRST* chapter in *ClearCase and MultiSite Release Notes (Windows Edition)* contains last-minute information about SMB server products, including which versions of those products ClearCase supports. Make sure that you read the remainder of *ClearCase and MultiSite Release Notes (Windows Edition)* for other information about configuring SMB server products.

Upgrading from a Previous Version of TAS

If you are upgrading from one version of TAS to another, follow the instructions in *TotalNET Advanced Server Release Notes*.

Disabling Opportunistic Locks on ClearCase Volumes

After you have upgraded to the new version of TAS using the Syntax instructions, ensure that opportunistic locks are disabled for each TAS volume that contains ClearCase storage. (The **Support opportunistic locks** check box in the volume definition should be cleared.) See *Administering Volumes* in the *General Administrative Tasks* chapter of *TotalNET Advanced Server Administration Manual* for details.

TAS Upgrade Does Not Require Reconfiguration

Aside from the exceptions noted in this section, the Syntax upgrade procedures preserve the previous configuration, including existing TAS volumes and file services supporting ClearCase. (To create new TAS volumes or file services, see *Creating a Volume* on page 91 or *Configuring the File Service* on page 92 for details.)

The remainder of this section describes how to perform an initial TAS installation, including how to configure TAS and ClearCase to support mixed-environment file access. If you are upgrading a TAS environment that was previously configured to support ClearCase, you can ignore these instructions.

Installing TAS

Follow the instructions in the appropriate platform-specific installation section of *TotalNET Advanced Server Release Notes* to install TAS on each VOB and view server requiring access from Windows NT.

Accessing TNAS

You can configure and administer TAS using the TotalNET Administration Suite (TNAS) Web interface. See the *TAS, TNAS, and TotalAdmin* chapter in *TotalNET Advanced Server Administration Manual* for details.

To access the TNAS Web interface:

1. Type a URL of this format in a Web browser:

   http://*computer***:***port#*

   where

   - *computer* represents the TAS server
   - *port#* represents the TNAS port number (the default is 7777)

   The TotalNET Advanced Server (TNAS) page appears.

2. Click **TotalNET Administration Suite (TNAS)**; a TNAS logon program appears.

3. Log on as **root**, using the **root** password for the TAS server. The TNAS interface now appears in your browser.

4. Click **TAS Configuration and Administration** in the *sphere frame* (that is, the frame at the upper right of the interface).

The TAS configuration and administration menu now appears in the *menu frame* (that is, the frame at the lower left of the interface).

Performing Initial Setup of TAS

After you have installed TAS on a server, you must perform an initial setup on that TAS installation. See the *Initial Setup Steps* chapter in *TotalNET Advanced Server Administration Manual* for details.

Click **Initial Setup** in the menu frame of the TNAS Web interface, and follow the instructions in the Syntax documentation, subject to the changes noted in these sections that are specific to use of TAS with ClearCase.

General TAS Settings

Accept the defaults for **Admin user**, **Admin group**, and so on in the **General TAS Settings** pane.

Enabling and Configuring the CIFS Realm

In the **Select Realms to Configure** pane, enable the CIFS realm, and click **Next**; the **CIFS Realm Configuration** pane appears.

NOTE: ClearCase does not require that the NetWare and AppleTalk realms be enabled.

Configure the CIFS realm as follows:

- **Server name** — Type the name of the VOB or view server, if it is not already the default.
- **Workgroup** — Type the name of the Windows NT domain to which your ClearCase clients belong.
- **Transport List** — Select the protocols appropriate for your site.
- **Device for NetBEUI** — Accept the default.
- **WINS Server** — If you are using proxy server authentication mode for CIFS file services (see *Configuring the File Service* on page 92), you may have to specify the IP addresses of the WINS servers for the network on which the authentication proxy server resides.

See the *Updating Realm Configuration* section under *Services for the CIFS Realm* in the *Configuring Services* chapter of *TotalNET Advanced Server Administration Manual* for details about configuring the CIFS realm.

Configuring TAS to Support ClearCase

After initial setup, configure the TAS server to support ClearCase, using the TNAS Web interface.

Creating a TAS Username Map for clearcase_albd

Create a TAS username map from the Windows NT ClearCase server process user account (see *Defining the ClearCase Server Process User and ClearCase Group* on page 50) to a UNIX user account whose primary group ID (GID) can access all VOBs and views that will be accessed by TAS file services.

To create the TAS username map:

1. Click **TAS System** in the menu frame; the **TAS System Configuration and Administration** pane appears.
2. Click **Username Maps**; the **Username Maps** pane appears. Make these changes to support ClearCase:

   - In the text box, type the name of a UNIX domain user account whose principal group has the appropriate privileges for VOB/view storage access and click **Create**.

     See Chapter 3, *Understanding ClearCase Access Controls*for details about group- and user-level access to ClearCase data.

   - In **List of client accounts**, type **clearcase_albd**.

     NOTE: If the Windows NT ClearCase server process user account is configured to be a user name other than **clearcase_albd**, type that user name instead.

   Click **Submit** at the bottom of the form; then click **OK** in the confirmation message.

See Chapter 4, *Administering Windows NT Domains and ClearCase* for details about the ClearCase server process user. See *Administering Username Maps* in the *General Administrative Tasks* chapter of *TotalNET Advanced Server Administration Manual* for details about creating user name mappings in TAS.

Creating a Volume

Create a TAS volume that exports the directory where the VOB and/or view storage are physically located. Clients use the volume name to represent the path to the physical VOB or view storage location.

NOTE: We recommend that you test the TAS installation and configuration using non-ClearCase files before attempting to use TAS to access VOBs and views. See *Testing the TAS Configuration* on page 94 for details.

The procedure required to support ClearCase is summarized here:

1. Click **TAS System** in the menu frame; then click **Volumes** in the **TAS System Configuration and Administration** pane.

2. Type a name (for example, **ccstore**) in the text box.

   Ensure that the volume name is of a form that is acceptable for all realms that will access it. For example, some realms do not accept names longer than 12 characters.

   NOTE: The text box contains a symbolic name for the volume, not the pathname to the volume storage. However, it is a good idea to specify TAS volume names that correlate to the VOB and view storage paths. (For example, a TAS volume named **ccstore** may be associated with **/ccstore** on the UNIX computer.) If these names do not correlate, examine the volume properties to determine which pathnames are associated with which volumes.

3. Click **Create**; a **New Volume Definition** pane appears. Make these changes to support ClearCase:

   - **Pathname** — Type the pathname to the virtual root of the storage area. This pathname is the root of the VOB or view storage areas for the VOB or view server. In other words, all VOB or view storage areas must be located below this pathname (but they need not be direct subdirectories of this pathname).

     For example, if you type **/ccstore**, legal VOB and view storage names for this volume are **/ccstore/vobstore**, **/ccstore/home/vobstore**, and **/ccstore/home/project/viewstore**.

   - **Volume umask** — Type **002**.

   - **Filename Case** — Select **preserve**.

   - **Support opportunistic locks** — Clear the check box.

   Click **Submit** at the bottom of the form; then click **OK** in the confirmation pane.

See the *Administering Volumes* section in *TotalNET Advanced Server Administration Manual* for details about creating and administering volumes.

Configuring the File Service

To configure the TAS file service to support ClearCase:

1. Access the file service:

   a. Click **CIFS Realm** in the menu frame.

   b. Click **Manage CIFS File Services**; a list of the file services appears.

   c. Click the file service that corresponds to your TAS server; then click **Administer**. A menu of file service operations appears.

2. Click **Configuration**; an update file service form appears. Make these changes to support ClearCase:

   - **Volume references** — Select the TAS volumes this file service references and exports.
   - **Browse master** — Select **off**.
   - **Umask** — Type **002**.
   - **Freespace report method** — Select **root**.
   - **Windows 95 logon server** — Clear this check box.
   - **Windows NT logon server** — Clear this check box.

     NOTE: You cannot use the **Windows NT Logon Server** feature if the TAS volumes are to include ClearCase storage.

   Click **Submit** at the bottom of the form; then click **OK** in the confirmation pane to return to the menu of file service operations.

3. Click **Authentication and Service Mode Options**; the **Authentication Mode** form appears. Select either the **Local authentication** or **Authentication proxy servers** check box and click **Submit**. An authentication mode configuration pane appropriate for the mode selected appears.

   NOTE: You cannot use **Share mode** authentication if the TAS volumes are to include ClearCase storage.

See your system administrator for assistance in determining the authentication mode for your site. See *Configuring Security* under *Services for the CIFS Realm* in *TotalNET Advanced Server Administration Manual* for details about file server authentication.

If you select **Authentication proxy servers**, configure the authentication mode pane as follows:

- **Authentication proxy servers** — Type the name of the proxy server in this text box.

  NOTE: You may need to specify in the CIFS realm the IP addresses of the WINS servers for the network on which the authentication proxy server resides. (See *Enabling and Configuring the CIFS Realm* on page 89.)

- **Username map** — Select this check box to ensure that the file service references the **clearcase_albd** username map specified in *Creating a TAS Username Map for clearcase_albd* on page 90.

If you select **Local authentication**, configure the authentication mode pane as follows:

- **Password encryption** — Select this check box.

  NOTE: If you select **Local authentication**, you must enter every user that will access TAS file services in a local password encryption database on the server supporting those file services. If your CIFS realm contains multiple servers supporting TAS file services, you must configure a local password encryption database on each server.

- **Username map** — Select this check box to ensure that the file service references the **clearcase_albd** username map specified in *Creating a TAS Username Map for clearcase_albd* on page 90.

Click **Submit** at the bottom of the authentication mode configuration form. Then click **OK** in the confirmation pane to return to the menu of file service operations.

See the *Creating and Modifying File Services* section in *TotalNET Advanced Server Administration Manual* for details about modifying file services.

Start Services and Accept Service Connections

To start the TAS file services and accept service connections:

1. Click **TAS System** in the menu frame and then click **TAS System Administration**.

2. Click **Start Services** in the **TAS System Administration** pane.

   Click **OK** in the **Confirmation** pane; then click **OK** to return to the **TAS System Administration** pane.

3. In the **TAS System Administration** pane, click **Accept Service Connections**.

   Click **OK** in the **Confirmation** pane; then click **OK** to return to the **TAS System Administration** pane.

See the *Controlling the Server System* section in the *General Administrative Tasks* chapter of *TotalNET Advanced Server Administration Manual* for details about starting TAS services and accepting service connections.

At this point, TAS is configured to support ClearCase. You can exit the TNAS Web interface.

Configuring ClearCase to Support TAS

For all ClearCase clients on Windows NT that have the MVFS installed and that will access TAS volumes, change the **MVFS Performance** settings in the ClearCase program in Control Panel as follows:

1. Click **Start>Settings>Control Panel**. Open the ClearCase program.
2. On the **MVFS Performance** tab:
   - Select **Override** for both **Maximum number of mnodes to keep on the free list** and **Maximum number of mnodes to keep for cleartext free list**.
   - Set the value for both to 800.
3. Click **OK** to apply the changes and close the dialog box.
4. Restart the Windows NT client.

Testing the TAS Configuration

This section summarizes how to test a TAS installation and configuration both using non-ClearCase files and using ClearCase VOBs and views.

Testing the TAS Configuration on Non-ClearCase Files

We recommend that you test the TAS installation and configuration using non-ClearCase files and directories before attempting to use TAS to provide file access to VOBs and views, as follows:

1. Create a directory structure on your TAS server (for example, **/tasstore/testdir**) and a test file in that directory (for example, **/tasstore/testdir/testfile**).
2. Install and configure TAS as described in this chapter, using **tasstore** as the volume name and **/tasstore** as the path name for the volume.
3. From a Windows NT client, create a file in the TAS volume. Then verify that the UNIX user and group settings for that file are correct.
4. Verify that all Windows NT clients can access the TAS volume, including testing permission and access restrictions, until you are confident that TAS is working properly.

Testing the TAS Configuration with ClearCase

To test that ClearCase and TAS are working together properly:

1. On a UNIX VOB or view server, install and configure TAS as described in this chapter, creating volumes containing VOB and/or view storage.
2. Verify that your ClearCase user and group assignments are appropriate. To do so, use the tests described in the section *Checking User and Group Assignments* on page 69.
3. Log on to a ClearCase client on Windows NT. Use the Region Synchronizer to import views, VOBs, or both from the UNIX TAS server.
4. Ensure that you can use any UNIX views or VOBs that you imported from the TAS server.

8 Configuring VOB and View Access in Mixed Environments

This chapter describes the tasks required to make UNIX VOBs and views accessible by Windows computers, and to make Windows NT VOBs accessible use by UNIX computers using snapshot views. UNIX computers cannot access Windows NT VOBs using dynamic views and cannot access dynamic views on Windows NT computers. Only Windows NT computers that are set up to support *dynamic views* can access UNIX dynamic views.

These are the most significant tasks associated with enabling cross-platform VOB and view access:

- Creating a new region in the ClearCase registry. UNIX and Windows have different network file naming conventions, so each VOB or view must have two tags: one used by UNIX computers and another used by Windows computers.

  NOTE: If you have not already done so, read about network regions in Chapter 26, *Administering Regions*.

- Creating an additional tag for each VOB or view that must be accessible to a computer that is not running the same operating system (Windows or UNIX) as the computer that hosts the VOB or view.

- Establishing conventions for the treatment of platform-specific text-file line-terminator conventions by views and VOBs.

8.1 Preparing the UNIX VOB or View Host

To prepare the UNIX VOB or view host so that you can use its VOBs or views on Windows:

1. If you have not already done so, install patches as instructed in the ClearCase customer area of the Rational Web site (**http://clearcase.rational.com**).

2. Determine whether any UNIX VOBs use symbolically linked storage pools (such pools are created with **mkpool –ln**). *Windows Tags for UNIX VOBs with Symbolically Linked Storage* on page 100 describes how to make this determination and how to perform a required extra step before registering such VOBs for a Windows region.

   NOTE: If a UNIX *view* has symbolically linked private storage (**mkview –ln**), you cannot use the view from a Windows computer if you access that view using an NFS client product. You can use that view if you are accessing the view using the Syntax TAS SMB server product.

8.2 Creating a New Network Region

Because Windows and UNIX have different network file naming conventions, Windows and UNIX computers each need to use platform-specific VOB- and view-tags that conform to those naming conventions. Whenever you must create two tags that refer to the same VOB or view, each of the tags must be created in a separate ClearCase network region. Creating a new network region is a software procedure; it implies no change in hardware configuration, and the region need not reflect physical locations or geographic regions.

Like VOBs and views, network regions are registered in the ClearCase registry. To create a new region, use the **cleartool mkregion –tag** *region-name* command. From a ClearCase host running Windows NT, you can also create a region using the ClearCase Registry snap-in to the ClearCase Administration Console. See *Adding a Network Region* on page 362 for more information on this subject.

Assigning Computers to the New Network Region

Once the new region has been created, follow the procedures in *To Move a Host into a New Network Region* on page 365 to add computers to it.

8.3 Creating VOB-Tags and View-Tags in the New Region

Use the Regions subnode of the ClearCase Registry node of the ClearCase Administration Console to create tags in the new region for VOBs and views. You can drag tags from one region into another, or copy them from one region and paste them into another. After you have copied a tag from one region to another, you must edit the tag's name and other properties so that they conform to the network file naming requirements of the platform for which the region was intended to serve.

You can also use the ClearCase Administration Console to create new tags that have the correct global access information. Or you can use the **mktag** command on either UNIX or Windows.

Administering Regions on page 355 has more information on creating VOB- and view-tags in multiple regions.

Using the Region Synchronizer

You can also use the Region Synchronizer on Windows to import the UNIX VOB-tags and view-tags into the ClearCase region to which you have assigned Windows computers.

To use the Region Synchronizer, click **Start>Programs>Rational ClearCase Administration>Region Synchronizer**. For help on using the Region Synchronizer, click **Help** in the **Synchronize ClearCase Regions** dialog box that opens when you start the Region Synchronizer.

NOTE: The Region Synchronizer expects a single ClearCase registry—one registry server. If you are registering UNIX VOBs and views for a network region that is managed by a separate registry server, use **cleartool mktag** to register VOB-tags and view-tags.

8.4 Re-Creating an Incorrect VOB-Tag or View-Tag

If you need to re-create an incorrect VOB-tag or view-tag, you can use the ClearCase Administration Console on a ClearCase host running Windows NT to change properties of the tag:

1. Navigate to one of the following nodes:

- The Tags subnode of the VOB or view storage node for the host where the VOB or view storage directory resides
- The VOB Tags or View Tags subnode for the tag's region in the ClearCase Registry node

2. Select a VOB-tag or view-tag in the Details pane.
3. Click **Action**>**Properties**. This command opens a dialog box in which you can edit properties of the tag if you are a member of the *ClearCase group*.

You can also use the command-line interface to re-create a tag:

- Use **cleartool mktag**, with the **–replace** option.

  For example:

  cleartool mktag –vob –tag \libvob –replace \\saturn\vobstore\libvob.vbs

- Use **cleartool rmtag** to remove the erroneous tag, and then use the Region Synchronizer to create a new one.

  The following command removes the tag for view **anne_main**:

  cleartool rmtag –view anne_main

8.5 Windows Tags for UNIX VOBs with Symbolically Linked Storage

NOTE: This section applies only if you are using an NFS Client product to access VOBs with symbolically linked storage pools; if you are using a supported SMB server product, you need not follow the instructions in this section.

If a UNIX VOB includes one or more symbolically linked storage pools, the VOB requires special handling when you register it in a Windows network region. Before registering a UNIX VOB in a Windows region, check for linked storage pools. If you discover you have already registered a VOB without accounting for its linked storage pools, see *Mapping Storage Pools for an Existing VOB-Tag* on page 101.

To check a VOB for linked storage pools, move to a UNIX computer and type a command similar to this one:

cleartool lspool –long –invob /vobs/libvob

```
 ...
pool "libvob"
 18-Jun-97.17:00:06 by VobAdmin(VobAdmin.sys@starfield)
  kind: source pool
  pool storage link target pathname "/net/gamma/pools/libvob.1"
  pool storage global pathname "/net/io/vb_store/myvob/s/sdft"
 ...
```

If the output includes one or more `pool storage link ...` lines, follow this procedure when you create the VOB-tag with the Region Synchronizer:

1. Select the VOB-tag, and click **Import**.
2. In the **Create VOB Tag** dialog box, click **Show Mount Options**.
3. Under **NT-Only Options**, in the **Split Pool Map** box, supply a one-line text string that specifies all remote storage pools. For example, the following line (which is broken, illegally, so you can read it) defines three remote pools, separated by vertical bars:

   ```
    s\sdft\=\\gamma\pools\s\libvob.1\
   |c\cdft\=\\gamma\pools\c\libvob.1\
   |d\ddft\=\\gamma\pools\d\libvob.1\
   ```

   In this example, the VOB storage directory is on **io** but includes symbolic links to source, cleartext, and derived object pools on **gamma**.

   NOTE: Pathnames are specified with UNC names, a backslash (\) terminates each path name, and vertical bars (|) separate individual pool mappings.

Mapping Storage Pools for an Existing VOB-Tag

If you discover that you have already registered a VOB without accounting for its linked storage pools, use the ClearCase Administration Console on a ClearCase host running Windows NT to change properties of the tag:

1. Navigate to one of the following nodes:
   - The Tags subnode of the VOB storage node for the host where the VOB storage directory resides
   - The VOB Tags subnode for the tag's region in the ClearCase Registry node

2. Select a VOB-tag in the details pane.

3. Click **Action**>**Properties**. This command opens a dialog box in which you can edit properties of the tag if you are a member of the *ClearCase group*.

4. In the dialog box, click the **Mount Options** tab. Enter a pool map string in the **Split Pool Map** box using the syntax described in *Windows Tags for UNIX VOBs with Symbolically Linked Storage* on page 100.

You can also delete the VOB-tag with **cleartool rmtag** and re-create it with the Region Synchronizer.

Alternately, you can re-create the VOB-tag with **mktag –replace**. A sample **mktag** command follows:

```
cleartool mktag –vob –tag \libvob –region dev_nt –replace   ^
More? –options poolmap=s\sdft\=\\gamma\pools\s\libvob.1\   ^
More?                  ^|c\cdft\=\\gamma\pools\c\libvob.1\   ^
More?                  ^|d\ddft\=\\gamma\pools\d\libvob.1\    ^
More? –host io   ^
More? –hpath  /usr1/vb_store/libvob.vbs   ^
More? –gpath  \\io\usr1\vb_store\libvob.vbs   ^
More? \\io\usr1\vb_store\libvob.vbs
```

This command illustrates several important rules to follow when composing the command line:

- A **poolmap** string commonly specifies multiple pools. Use vertical bars (|) to separate individual pool specifications. Escape each vertical bar with a caret (^).
- Use UNC names to specify pool locations.
- Specify all of the **–host**, **–hpath**, and **–gpath** arguments.
- Supply a UNC name to the VOB storage directory as the **–gpath** argument.

Here is formal syntax for each pool specification in a **poolmap** mount option to **cleartool mktag**:

pool-spec := *symlink-source*\=*symlink-target*\

symlink-source
The symbolic link to the remote pool, relative to the VOB storage directory.

symlink-target
The full pathname, in UNC format, of the linked pool. The pool must reside somewhere

in a UNIX directory subtree that has been mounted on the local Windows NT computer using an NFS product. The path name must be valid on all computers in the Windows NT network region that access the VOB.

8.6 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) character 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.

As a result of these differing 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 site configuration parameter in the ClearCase registry 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 other 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. (Previous versions of ClearCase documentation and interfaces refer to this mode as **unix** text mode.)

  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. (Previous versions of ClearCase documentation and interface refer to this as **interop** text mode or **msdos** text mode.)

- **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. **strip_cr** text mode is a new feature of ClearCase for release 4.1. You cannot create a view in **strip_cr** mode from the **View Creation Wizard**.

In a snapshot view created in either **insert_cr** or **strip_cr** text mode, ClearCase adds or removes the <CR> characters whenever it updates the view. In a dynamic view, ClearCase adds or removes the <CR> characters as you open and read files. For both snapshot views and dynamic views, ClearCase 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.

Enabling Interop Text Mode Support in VOBs

VOBs created with Clearcase 4.1 or later are compatible with views in any text mode. VOBs created with earlier versions of ClearCase are only compatible with views in transparent text mode until you run the **msdostext_mode** command on the VOB. Earlier versions of Clearcase documentation and interfaces refer to VOBs on which **msdostext_mode** has been run as *interop-enabled* VOBs. Beginning with Clearcase 4.1, we further generalize this by saying the **msdostext_mode** enables VOBs to support access from views in either transparent or interop text modes.

If you need to enable interop text mode support in a VOB created by an earlier version of ClearCase, use the **msdostext_mode** command. Note that **msdostext_mode** does not convert or modify files in any way. It affects only the information recorded for text file versions in the VOB database.

Only the VOB owner the privileged user can run **msdostext_mode.** The command syntax is

*ccase-home-dir***/etc/utils/msdostext_mode [–d]** *vob-storage-pname*

With no options, **msdostext_mode** does the following:

- For all versions of all file elements whose element type is text_file or a subtype of type text_file, generates and stores in the VOB database the information required to support access to these versions by views created in **strip_cr** or **insert_cr** text modes.
- Turns on interop text mode support, so that this information can be recorded for newly created versions.

With the **–d** option, **msdostext_mode** disables support for interop text modes.

Determining Whether a VOB Supports Interop Text Modes

To determine whether a entire VOB supports interop text modes, use the following command on either UNIX or Windows:

cleartool dump vob:*vob-tag*

If the `flags` line in the output contains the string `pc_line_count`, the VOB supports interop text modes.

For example, to determine whether the VOB **\pc_src** supports interop text modes:

cleartool dump vob:\pc_src

If the output of this command contains a line similar to the following, the VOB supports interop text modes.

```
flags: predefined, pc_line_count, unrestricted
```

Special Procedure for MultiSite Users

ClearCase MultiSite does not replicate a VOB's text mode support characteristics. If any replica in a VOB family has been enabled for interop text mode support, all replicas in that VOB family must be individually enabled at their local sites using **msdostext_mode** as follows:

1. Synchronize all VOB replicas.
2. Lock all replicas, using this command:

   cleartool lock –nuser root vob:*pname-in-vob*
3. Run **msdostext_mode** on all replica storage directories.
4. Unlock all replicas.

Administering VOBs

9 Understanding VOB Storage

This chapter describes the on-disk data structures that implement ClearCase *VOBs*.

9.1 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 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. In a dynamic view, the VOB-tag specifies a location at which the VOB is accessible as a file system of type MVFS. Chapter 25, *Understanding the ClearCase Registry* discusses this in detail.

9.2 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 invoked 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 host where the VOB storage directory physically resides. The **db** subdirectory must also be on that same host. (On UNIX hosts, storage pools can be remote.)

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 14, *Moving*

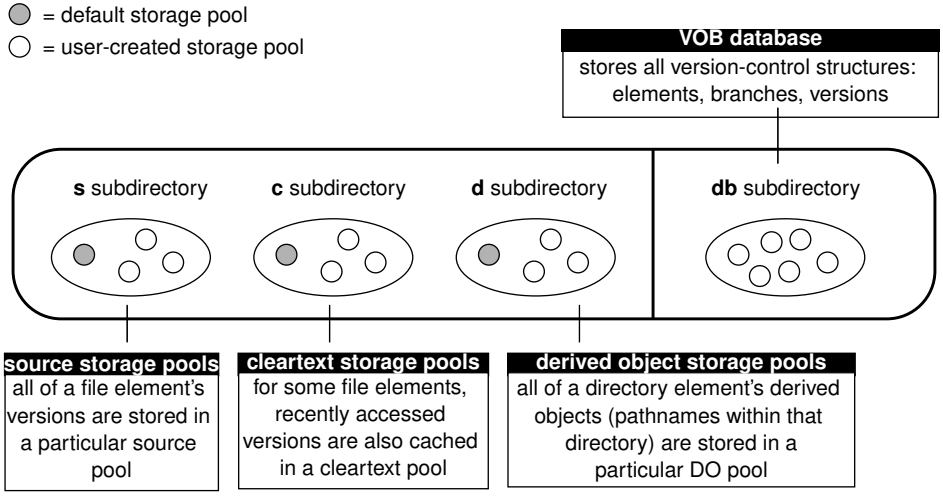
VOBs, for instructions. See Chapter 13, *Administering VOB Storage*, for steps you can take when a VOB database threatens to fill up its disk partition.

For the most part, ClearCase servers and the ClearCase scheduler manage routine VOB database maintenance automatically. See Chapter 13 for more on VOB maintenance.

9.3 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 3).

Figure 3 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 13, *Administering VOB Storage*.) A cache miss forces the type manager to reconstruct the version.

Derived Object Storage Pools

Each derived object storage pool holds a collection of derived object data containers. A derived object data container holds the file system data (usually binary data) of one DO, created by a ClearCase build tool (**clearmake**, for example, or **clearaudit).**

DO storage pools contain data containers only for the derived objects that have been promoted to the VOB, through ClearCase's *winkin* feature. Each directory element is assigned to a particular DO storage pool. The first time a DO created within that directory is winked in, its data container is copied to the corresponding DO storage pool. The data containers for unshared and nonshareable derived objects reside in view-private storage.

By default, derived object pools are periodically scrubbed to remove extraneous DOs, as described in Chapter 13.

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 a host has its own dedicated *vob_server* process.

9.4 Storage Pool Creation

When a new VOB is created, three subdirectories are created to hold *storage pools*, and a single default storage pool is created within each one:

c	Directory for all cleartext pools
c\cdft	Default cleartext pool
d	Directory for all derived object pools
d\ddft	Default derived object pool
s	Directory for all source pools
s\sdft	Default source pool

You can create as many additional storage pools as you need and adjust their contents as necessary using the ClearCase Administration Console or the **mkpool** and **chpool** commands.The **mkpool** command creates new storage pools within the VOB storage directory itself. For example, **mkpool –source srcpl2** creates a source pool as subdirectory **s\srcpl2** under the VOB storage directory. On UNIX hosts, you can optionally create *remote storage pools*.

Remote Storage Pools on UNIX

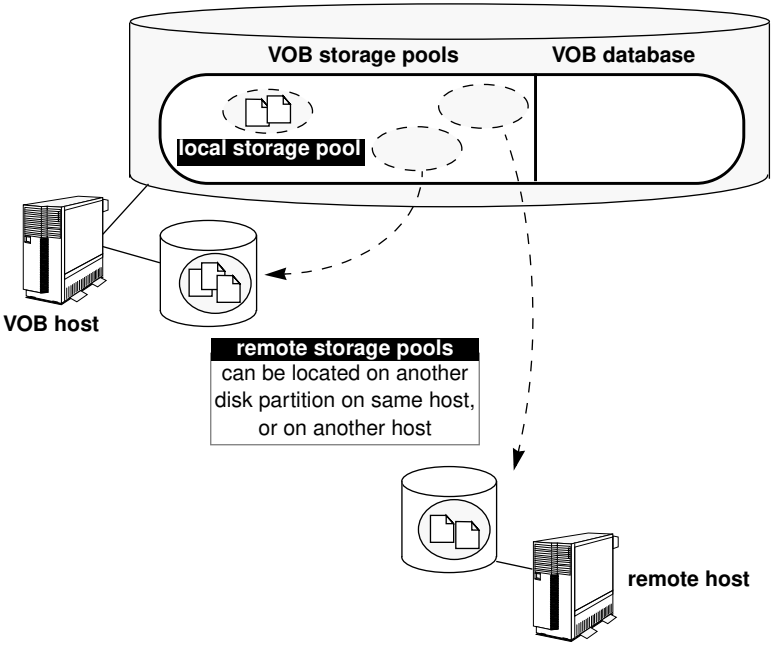
ClearCase UNIX users can use **mkpool –ln** to create a *remote storage pool*, leaving behind a standard UNIX symbolic link that points to the remote location:

```
# cleartool mkpool –source –ln /net/ccsvr04/ccase_pools/srcpl3 srcpl3
```

In this example, a storage pool directory is created at the remote location **/net/ccsvr04/ccase_pools/srcpl3**. Within the VOB storage directory, a symbolic link is created instead of a subdirectory; the text of the link is **/net/ccsvr04/ccase_pools/srcpl3**.

The remote location can be on another host—even a non-ClearCase host—or in another disk partition on the local host (Figure 4). Either way, this capability enables a VOB to circumvent the UNIX restriction that a directory tree must be wholly contained within a single disk partition. It also allows you to locate storage pools on high-capacity and/or high-speed file servers on which ClearCase is not installed.

Figure 4 Local and Remote VOB Storage Pools



This is a powerful feature, enabling a single *logical* entity to be distributed *physically*. But there are some complications:

- The important task of data backup is considerably more complicated for a VOB with remote storage pools than for a VOB contained within in a single disk partition. See *Backing Up a UNIX VOB with Remote Storage Pools* on page 176.
- The pathname you use for the remote location must be valid on all client hosts that will access the VOB. In particular, you cannot use the *network region* facility to handle network idiosyncrasies, such as hosts with multiple network interfaces.

9.5 Elements' Source Pool Assignments

ClearCase VOB-creation utilities create each new VOB with a single directory element, the *VOB root directory*. Users access this directory using the VOB's *VOB-tag* (the VOB's mount point in a dynamic view). This top-level directory is assigned to the three default storage pools. By default, all newly created elements inherit the pool assignments of their parent directories. Thus, all elements in a VOB use the default storage pools, unless you create new pools and reassign elements to them.

You can use the **chpool** command to change the source and/or cleartext pool associated with an element. Changing the source pool of a file moves all its data containers. Changing the source pool of a directory element changes the source pool to which new elements created within it are assigned. (See also *Creating Additional Storage Pools for UNIX VOBs* on page 209 and the **mkvob** reference page.)

9.6 Tools for Working with Storage Pools

You can use the ClearCase Administration Console on a Windows NT host to manage storage pools on Windows or UNIX hosts anywhere in your ClearCase network. Navigate to the Pools subnode of the VOB storage node for a VOB. The VOB storage node is itself a subnode of the host node for the host on which the VOB storage directory resides. Using the Pools node, you can create, delete, and rename a storage pool and change the parameters used for scrubbing a pool.

You can also use the command line. The following commands are basic tools for working on the command line with VOB storage pools. Each has its own reference page in the *ClearCase Reference Manual*, which provides complete details on its use.

cleartool Subcommands

mkpool
: Creates a new storage pool; with **–update**, it adjusts an existing pool's scrubbing parameters.

lspool
: Lists basic information about one or more storage pools. The **describe pool:***pool-name* command lists the same information.

rename pool:*pool-name*
Renames a storage pool.

rmpool
Deletes a storage pool.

chpool
Reassigns elements to a different pool.

checkvob
Finds and fixes inconsistencies between the VOB database and VOB storage pools.

space –vob
Reports disk space used by the VOB database and each storage pool.

dospace
Reports disk space used by shared derived objects.

Utility Commands

scrubber
Deletes unneeded data containers from derived object and cleartext pools. (See also *Scrubbing VOB Storage Pools* on page 207.)

view_scrubber
With **–p** option, transfers data containers from view-private storage to a VOB's derived object storage pool. (See also *Scrubbing View-Private Storage* on page 314.)

10 Setting Up VOBs

This chapter presents a procedure for setting up your network's ClearCase data repository, a set of globally accessible VOBs. These are the major steps:

- Select a host for a new VOB.
- Modify operating system resources, if necessary.
- Create one or more VOB storage locations if you want.
- Create the VOB (and, if necessary, adjust its registry and identity information, to ensure global accessibility and implement access controls).
- Coordinate the VOB with other existing VOBs.
- Populate the VOB with new or existing development data.

After VOBs are set up, they can be activated for use with dynamic views with the **cleartool mount** command (or from the VOB Admin Browser on UNIX or, on Windows, with the ClearCase Explorer or Windows Explorer). Two common user mistakes users make that can make a VOB invisible to their dynamic views are forgetting to mount the VOB and forgetting to work in a view. Users of snapshot views must load the view from a VOB before they can access VOB data. Chapter 19, *Setting Up Views*, discusses the procedure for setting up ClearCase views.

By default, UNIX hosts mount all *public* VOBs at ClearCase startup time. See the **mount** reference page in the *ClearCase Reference Manual* for details.

To arrange for a Windows host to mount a VOB each time you log on, do **one** of the following:

- Select the **Reconnect at Logon** check box in the **Mount** dialog box when you first mount the VOB.

- Use the **–persistent** option to **cleartool mount**.
- Add a **cleartool mount** command to a **.BAT** file in the Startup program group.

10.1 Selecting a VOB Host

A host on which one or more VOB storage directories reside is a *VOB host*. A typical network distributes its VOBs among several VOB hosts. Nearly any host supported by ClearCase can be a VOB host (the exceptions are Windows 95 and Windows 98). Selecting and configuring an appropriate VOB host are crucial to obtaining satisfactory ClearCase performance.

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

The computer chosen as a VOB host 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, you should add 7 MB of memory per VOB regardless of VOB database size, as well as 750 KB of memory for any **view_server** process that will run on the VOB host. 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/or to increase the number of concurrent users without degrading performance. For the best performance, configure the VOB host with enough main memory to hold the entire VOB database.
- **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 432.
- **Processing power.** A VOB host 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 VOB host.

- **Availability.** A VOB intended for shared access must be located in a disk partition that is accessible to all ClearCase client hosts. Wherever possible, select a host that can be accessed with the same host name by all ClearCase hosts. (A host with multiple network interfaces presents a different name through each interface.) If a VOB host has multiple names, you must create multiple *network regions*, to logically partition the network.

See also Chapter 32, *Improving VOB Host Performance*.

10.2 Planning for One or More VOBs

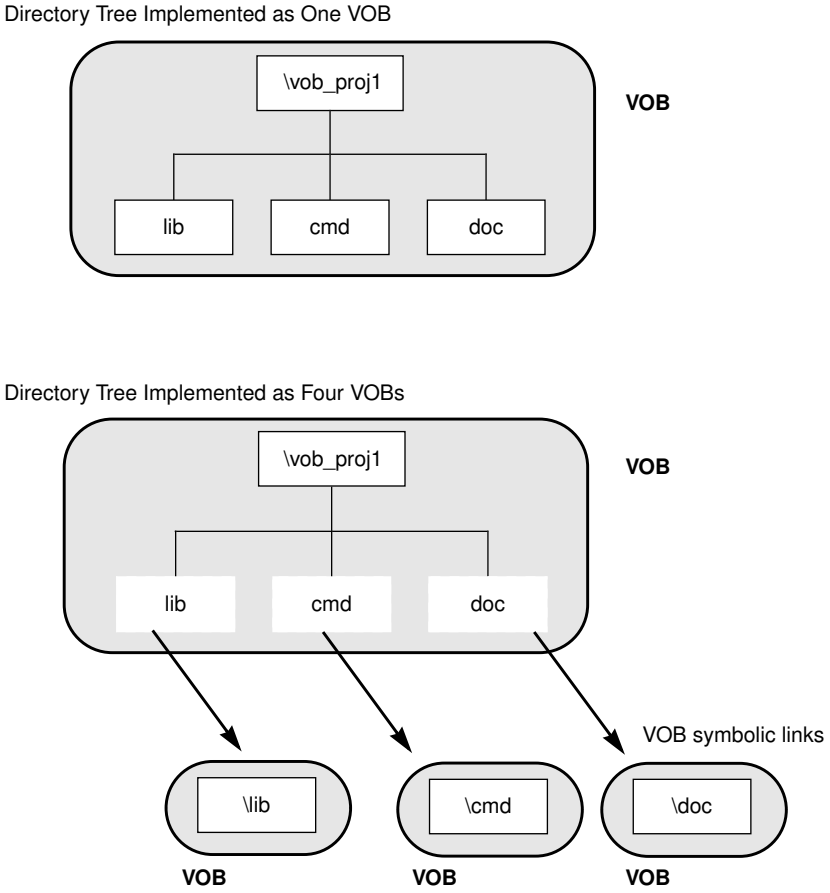
Your organization must decide how to allocate data to one or more VOBs on a VOB host. These are the principal trade-offs to consider:

- Splitting data into several smaller VOBs increases your flexibility. It is easy to move an entire VOB to another host. It is more difficult to split a VOB into several new VOBs and move one or more of them to another host.
- Typically, you have fewer performance bottlenecks when you use several smaller VOBs rather than one very large VOB.
- Having fewer VOBs facilitates data backup.
- Having fewer VOBs facilitates synchronizing label, branch, and other definitions across all the VOBs. It reduces reliance on the availability of admin VOBs and globally defined metadata types.

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 5).

Figure 5 Linking Multiple VOBs into a Single Directory Tree



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

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

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

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

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

Planning for Release VOBs

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

10.3 Modifying a UNIX VOB Host for ClearCase

Each VOB is managed by a battery of server processes, which run on the host where the VOB storage directory resides. 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 UNIX VOB host:

- **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 VOB host after ClearCase has been up and running for some time. For more information, see *Tune Block Buffer Caches on UNIX* on page 433.

Optional Software Packages

To ensure correct ClearCase operation, you may need to install one or more optional software packages available from your hardware vendor. Consult the installation instructions in the *ClearCase Product Family Installation Notes* for more information.

10.4 Creating VOB Storage Locations

ClearCase allows you to specify one or more *server storage locations* for each network region defined in your ClearCase registry. You designate whether a server storage location is to be used for VOB or view storage when you create it. VOB storage locations provide administrators with a way to designate specific hosts and disks that will be recommended to users or other administrators creating new VOBs. Although these storage locations are not the default choice for VOB-creation tools, creating and using them can simplify many VOB setup and administration tasks.VOB storage locations should be created on a disk partition that has plenty of room for VOB database growth and is accessible to all ClearCase client hosts in the region.

On UNIX hosts, the partition must be exported so that other UNIX clients can mount it. If you plan to make the VOB accessible to Windows clients using CCFS or TAS, the server will need to be configured for that purpose as described in Chapter 7.

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

10.5 Creating a VOB

Follow these steps to plan and create each new VOB:

1. **Log on to the VOB host.** Make sure the host meets the criteria laid out in *Selecting a VOB Host* in this chapter. 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. VOB tags exhibit syntactic differences on UNIX and Windows hosts because they must conform to the network naming conventions of each platform.

   - On UNIX hosts, each ClearCase client mounts the VOB as a file system of type MVFS, so the VOB-tag must be the full pathname of a mountable file system, for example:

     ```
     /vobs/flex
     ```

     This pathname usually includes a local mount-point (**/vobs** in the example above) and the name of a mountable file system (**/flex** in the example above). Unless there is a compelling reason to do otherwise, all clients should mount the VOB at the same local mount point.

   - On Windows hosts, each ClearCase client host uses the VOB-tag as though it were the name of a Windows directory. In a Windows network, a VOB-tag is the VOB's registered name and also its root directory. A Windows VOB-tag must have exactly one backslash (\), which must be the first character of the VOB-tag (for example, **\vob_project2)**. When activated with the **cleartool mount** command, a VOB-tag appears as a subdirectory under each view-tag visible on the *dynamic-views drive* (**M:** by default).

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 host named **pluto**, 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**.

By default, VOBs are created as private VOBs. Specify **-public** on the **mkvob** command line to make a public VOB. Public VOBs can be mounted one at a time using the **cleartool mount** command, or as a group using the command **cleartool mount –all**. If you specify **-public** on the **mkvob** command line, you are prompted to enter the ClearCase *registry password*. This password is normally established when ClearCase is installed at a site. If you need to create or change this password, see the **rgy_passwd** reference page.

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.

NOTE: Before any user can access a new VOB on a client host, the following conditions must be met:

- The user must have a working ClearCase view.
- To be accessed from a dynamic view, the VOB must be mounted.
- To be accessed from a snapshot view, appropriate elements of the VOB must be loaded into the snapshot view.

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 149.

Creating a VOB on a Remote Host

UNIX users can use standard UNIX remote host access methods (telnet, for example, and the X Window System) to access a remote host for purposes of creating a VOB.

If you are using Windows NT and want to create a view or VOB on a remote machine, ClearCase must have access to information stored in the remote machine's registry. Remote access to the Windows Registry can be restricted by setting security on the key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePipeServers\Winreg

By default, this key is not present on Windows NT workstations. This allows the underlying security on the individual keys to control access.

On Windows NT Server machines, the key is preset by default and may prevent workstations from reading the server's registry. When this is the case, if you are on a remote workstation and attempt to create views or VOBs on the server, the creation fails with the error `Invalid argument`.

To work around this problem, either remove the key or modify the security on it.

ClearCase Doctor, which runs as part of the installation, warns you if the registry key will prevent you from creating views or VOBs on this machine, and optionally can fix it.

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 not all prospective users of the VOB 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 125 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.

Ensuring Global Access to the VOB—Special Cases for UNIX

The output from the **mkvob** command (Step #3 on on page 125) includes a global (networkwide) pathname for the VOB storage directory. This pathname is derived heuristically; that is, it's an intelligent guess. Depending on the accuracy of the guess, you may have some more work to do before the VOB will be accessible to all ClearCase hosts.

Two of the most common reasons this guess can be wrong are unique to UNIX clients.

Guess Was Wrong, But Global Pathname Does Exist

There may be a global pathname that all ClearCase client hosts can use to access the VOB storage directory, but **mkvob**'s heuristically derived pathname is not the right one. These are the most common reasons:

- Use of a nonstandard automount program
- Use of a home-grown (perhaps manual) scheme for mounting file systems around the network

In this case, use the **mktag** command to correct **mkvob**'s guess. For example:

host and hpath information must be valid on the VOB host. gpath must be a valid pathname to the VOB on all hosts. pathname to the storage directory must be valid on the local host

```
cleartool mktag –replace –vob –public –tag /vobs/flex \
  –host ccsvr01
  –hpath /vobstore/flex.vbs
  –gpath /allvobs/flex.vbs
  /vobstore/flex.vbs
Vob tag registry password: <enter password>
   .
   .
```

Network Requires Multiple Global Pathnames

There may be no single global pathname for the VOB storage directory. The most common reason is that the VOB must be accessible to hosts with different operating systems (for example, UNIX and Windows) which have different conventions for global pathnames. In this case, you must partition your network into multiple *network regions*; each region must have a single global pathname to VOB storage. For background information, see Chapter 26, *Administering Regions*.

Another possible reason is that the VOB host is a UNIX computer that has multiple network interfaces, each with its own hostname. Because only one VOB-tag per region can be associated with a given host storage path, a VOB that has storage on a host with multiple network interfaces must have a tag in each region where the hostname is listed if clients in each region need access to the VOB.

If a host named pluto is in region **ccregion1**, then the **mkvob** command in Step #3 on on page 125 created the VOB-tag in that region. If **pluto** has a second network interface that uses the

hostname **pluto2**, you must make **pluto2** a member of a different region, and then create a VOB-tag in that region before clients in the region can access the VOB created in that step.

valid pathname –> to VOB on all hosts in network region ccregion2

```
cleartool mktag –vob –public -region ccregion2
–tag /vobs/flex
–host pluto2
–hpath /vobstore/flex.vbs
–gpath /net/pluto2/vobstore/flex.vbs
/vobstore/flex.vbs
Registry password: <enter password>
  .
  .
```

NOTE: Your **–gpath** value may need to take into account the use of a nonstandard automount program or other mounting idiosyncrasies within each network region.

This example used the same VOB-tag even though it was initially created in a different region. This is a practice that we encourage you to follow.

Enabling Setuid and Setgid Mounting of the Viewroot and VOB File Systems on UNIX Hosts

By default, the *viewroot* and VOB file systems are mounted with setUID and setGID disabled; this is the recommended configuration. However, if any hosts at your site must mount these file systems with setUID and setGID enabled (for example, if you run setUID tools from a VOB), you must configure your release area with the **site_prep** script to allow this. When ClearCase is installed on these hosts, the viewroot and VOB file systems are mounted accordingly. (Alternatively, you can run the **change_suid_mounts** script on an individual host to enable honoring of setUID programs in VOBs mounted on that host. However, this setting does not persist across installs, and you must stop and restart ClearCase on the host after running the script.) For more information, see the discussion of **site_prep** in *ClearCase Product Family Installation Notes*.

Creating Remote Storage Pools on UNIX Hosts

As described in the discussion of *Remote Storage Pools on UNIX* on page 115, you can take advantage of UNIX symbolic links to create remote storage pools on a UNIX VOB server host. Typically, you don't add remote storage pools to a VOB until a disk-space crisis occurs. But as you gain more experience with how your group uses ClearCase and how VOBs grow, you may

want to add remote storage pools when you first create a VOB. This approach can help to postpone the disk-space crisis, perhaps even avoid it altogether.

See *Creating Additional Storage Pools for UNIX VOBs* on page 209 for the procedure. See also the **vob** reference page.

10.6 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.

10.7 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 includes several utilities for exporting data from other configuration management systems. This section present several examples of how these 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. Importing in a dynamic view improves the performance of the import operation.

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 /proj/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. Importing in a dynamic view improves the performance of the import operation.

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_ssafe**, **clearexport_cvs**, and **clearimport** reference pages.

10.8 Converting a SourceSafe Configuration

This section uses a sample configuration to show how to use the **clearexport_ssafe** and **clearimport** utilities to migrate a set of source files from the Windows 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 and mounted 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 6 Sample SourceSafe Payroll Configuration

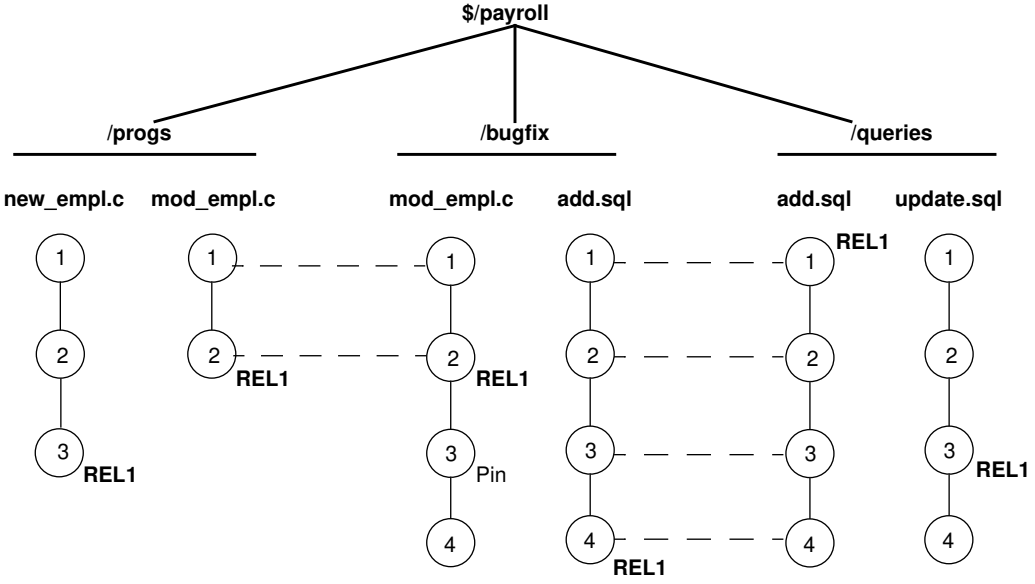


Figure 6 shows the configuration hierarchy in SourceSafe, which consists of these projects:

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

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

Shares

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

Branches

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

Labels

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

Pins

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

Setting Your Environment

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

Setting Environment Variables

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

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

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

set TMP=c:\temp

Setting Your SourceSafe Current Project

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

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

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

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

Running clearexport_ssafe

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

Using the Recursive Option

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

Example

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

ss cd

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

clearexport_ssafe –r –o c:\datafiles\paycvt

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

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

Running clearimport

Use any view. **clearimport** ignores the config spec. Importing in a dynamic view improves the performance of the import operation.

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

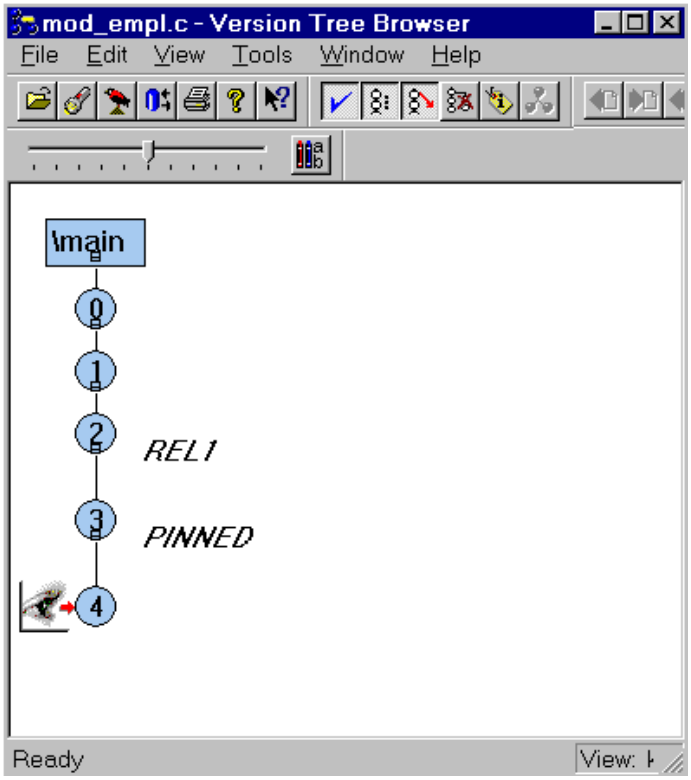
```
Validating label types.
Validating directories and symbolic links.
Validating elements.
Creating element ".\bugfix/add.sql".
     version "1"
     version "2"
     version "3"
     version "4"
Creating element ".\bugfix/mod_empl.c".
     version "1"
     version "2"
     version "3"
     version "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 7 shows the version tree for the **\bugfix\mod_empl.c** element.

Figure 7 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.

10.9 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 and MultiSite 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 primary purpose of feature levels is to manage VOBs that are replicated (using ClearCase MultiSite) across server machines that are not all running the same ClearCase release.

Every ClearCase release is associated with a feature level. Read *ClearCase and MultiSite Release Notes* for information on which releases correspond to which feature levels.

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 unreplicated VOB that is not already at the feature level corresponding to the release of ClearCase that is installed on the host.

When you use MultiSite, each VOB replica has a feature level, and the VOB family has a feature level. Replicas in the same family can have different feature levels. The family feature level is the feature level that is equal to or less than the lowest replica feature level found among members of the VOB family. Before you raise the feature level of a VOB family, you must raise the feature levels of all replicas in that family. For more information about feature levels and VOB replicas, see *ClearCase MultiSite Manual*.

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 and MultiSite Release Notes* for information about VOB database formats in the current release.

In general, all VOBs that reside on a VOB server host must use the same database format. To change the database formats for VOBs:

1. Back up all VOBs on the VOB server host. See Chapter 12, *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 Product Family 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*.

11 Using Administrative VOBs and Global Types

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

11.1 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.

11.2 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.

11.3 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 124.

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 149.
2. Create global types in it. See *Creating a Global Type* on page 155.

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.

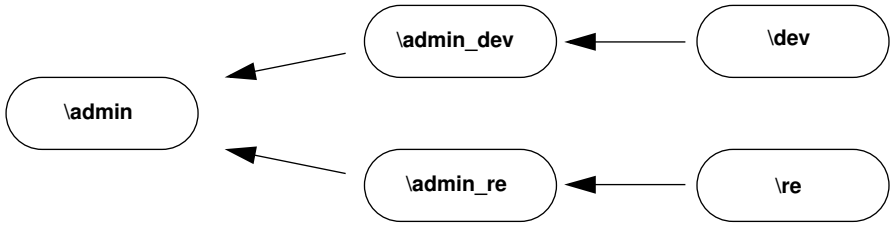
cleartool mkhlink –c "link to admin VOB" AdminVOB vob:\dev vob:\admin_dev

```
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 8 illustrates this example.

Figure 8 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.

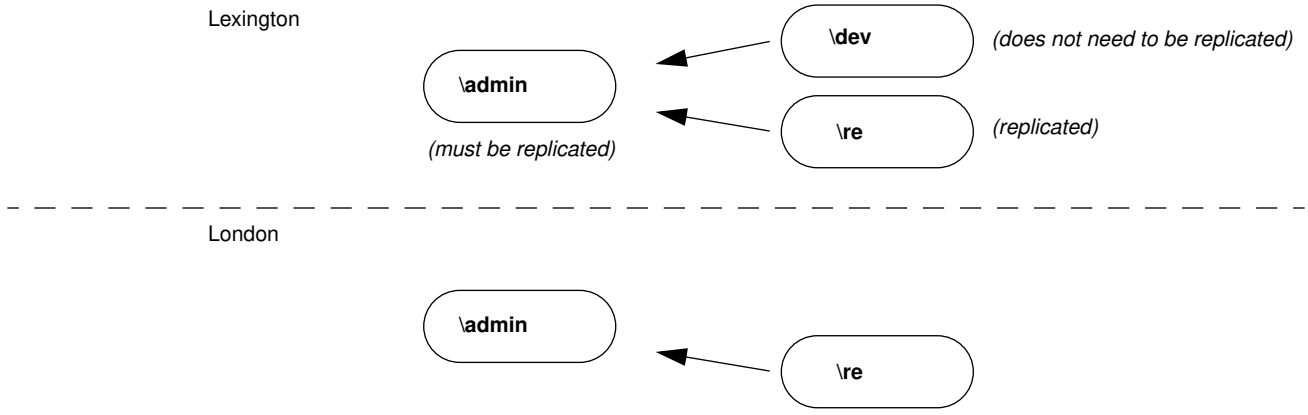
An administrative VOB does not have to be mounted to give client VOBs access to global type definitions. However, the administrative VOB must be registered and must have a VOB-tag.

Using Administrative VOBs with MultiSite

When you use administrative VOBs in a MultiSite environment, auto-make-type operations are logged as "make type" events at the client replica.

If a client VOB is replicated, the client's administrative VOB must be replicated. In the example in Figure 9, **\re** is a client of **\admin**, **\re** is replicated, so all sites that have a replica of **\re** must also have a replica of **\admin**.

Figure 9 Replicated Administrative VOBs.



If you replicate a VOB that is linked to an administrative VOB, the MultiSite **mkreplica –export** command prints a reminder that you must replicate all administrative VOBs in the hierarchy above the VOB you are replicating. The output lists the administrative VOBs. The command does not check whether these administrative VOBs are replicated, so you can ignore the message if you have already replicated them

Because local type objects in a client VOB are linked to global type objects in the administrative VOB, we recommend that you synchronize a client VOB and its administrative VOB at the same time. If you do not, users may have trouble accessing type objects.

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 completely 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.

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

Creating a Global Type

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

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

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

cleartool mklbtype –c "final label for REL6" –global REL6@\admin

```
Created label type "REL6".
```

You cannot create a global type if any client VOBs contain 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 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 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".
```

Auto-Make-Type of Shared Global Types

This section applies only if you use global types in replicated VOBs.

If a global type is shared, ClearCase can create a local copy of the type only if the type is mastered by the Administrative VOB replica at the current site. If the shared global type is not mastered at the current site, you can create instances of the type only if the client VOB replica contains a local copy of the type. This restriction applies even if your current replica masters the object to which you are attaching the instance. This mastership restriction prevents conflicting, simultaneous creation of a given type with a given name at multiple sites.

If the client VOB at your site does not contain a local copy of the type, you must create a local copy at the site that masters the type. Then, export an update packet from the client VOB at that site to the client VOB at your site and import the packet at your site.

For example, a release engineer at your site (London) tries to apply the **V2.1** label to a version in the **\dev** VOB. The command fails because the label type is shared, no local copy of the type exists, and the type is mastered at a different site.

cleartool mklabel –nc V2.1 \dev\file.txt

```
cleartool: Error: Type must be mastered in original replica "london" to use
copy type.
cleartool: Error: Unable to create label "V2.1" on "\dev\file.txt" version
"/main/3".
```

To create a local copy of the type in the replica at your site:

1. Determine the VOB-tag of the administrative VOB.

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

2. Determine which replica of the administrative VOB masters the type.

   cleartool describe –fmt "%n\t%[master]p\n" lbtype:V2.1@\admin
   ```
   V2.1     lex@\admin
   ```

3. At the site where the type is mastered, create a local copy of the type in the client VOB.

   **cleartool cptype –c "forcing creation of local copy" lbtype:V2.1@\admin **
   lbtype:V2.1@\dev
   ```
   Copied type "V2.1".
   ```

4. At the site where the type is mastered, export an update packet to the replica at your site.

   multitool syncreplica –export –fship london@\dev
   ```
   ...
   ```

5. At your site, import the update packet.

   multitool syncreplica –import –receive
   ```
   ...
   ```

After the packet is imported, the engineer can create the label:

cleartool mklabel –nc V2.1 \dev\file.txt
```
Created label "V2.1" on "\dev\file.txt" version "/main/3".
```

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".
```

The following command describes a global type in a replicated VOB. Note that because the master replica of the type is in a different VOB family than the replica in which you enter the command, the output includes the VOB-tag of the master replica in addition to the replica name.

cleartool describe –long lbtype:SHARED@\tests

```
label type "SHARED"
  created 03-Aug-99.12:29:01 by Pete Sharon (pds.user@argon)
  master replica: raleigh@\tests_admin
  instance mastership: shared
  owner: pds
  group: user
  scope: global
  constraint: one version per branch
  Hyperlinks:
    GlobalDefinition@43@\tests <- lbtype:SHARED@\tests
```

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 admin VOB, global types are also listed):

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

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

Listing History of a Global Type

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

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

cleartool lshistory –minor lbtype:REL6@\dev

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

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

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

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

Changing Protection of a Global Type

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

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

cleartool protect –chown jtg lbtype:LABEL1@\dev

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

cleartool describe –local lbtype:LABEL1@\re

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

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

Locking or Unlocking a Global Type

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

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

cleartool lock –c "freeze" lbtype:REL6@\dev
```
Locked label type "REL6".
```

Attempts to create instances of the label type fail:

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

If you enter a **lock** command in a client VOB that does not contain a local copy of the specified type, ClearCase 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 Mastership of a Global Type

Changing mastership of a global type does not change the mastership of its local copies. Likewise, changing the mastership of a local copy changes the mastership of the local copy only. Mastership of the global type and all other local copies is not changed.

For example, the global label type **V3.2** is mastered by the **london** replica in the VOB family **\admin**, and the local copy in the client VOB **\client** is mastered by the **london** replica in the VOB family **\client**:

cleartool describe –fmt "%n\n %[master]p\n %[type_scope]p\n" lbtype:V3.2@\admin
```
V3.2
  london@\admin
  global
```

**cleartool describe –local –fmt "%n\n %[master]p\n %[type_scope]p\n" **
lbtype:V3.2@\client
```
V3.2
  london@\client
  local copy
```

When the mastership of the global type is transferred to the **lex** replica, the mastership of the local copy remains the same:

multitool chmaster lex@\admin lbtype:V3.2@\admin
```
Changed mastership of label type "V3.2" to "lex@\admin"
```

cleartool describe –fmt "%n\n %[master]p\n %[type_scope]p\n" lbtype:V3.2@\admin
```
V3.2
  lex@\admin
  global
```

cleartool describe –local –fmt "%n\n %[master]p\n %[type_scope]p\n" ^ lbtype:V3.2@\client
```
V3.2
  london@\client
  local copy
```

If you enter a **chmaster** command in a client VOB that does not contain a local copy of the specified type, the command fails with the message `type not found`. For example:

multitool chmaster lex@\client lbtype:DOC_SOURCE@\client
```
multitool: Error: Label type not found: "DOC_SOURCE".
```

For more information on mastership, see the **chmaster** reference page and *ClearCase MultiSite Manual*.

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

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

   cleartool describe lbtype:IDENT@\admin

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

cleartool describe lbtype:IDENT@\dev

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

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

   cleartool mklbtype –replace –global –acquire IDENT@\admin

   ```
   Replaced definition of label type "IDENT".
   ```

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

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

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

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

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

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

Removing a Global Type

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

For example:

cleartool rmtype –nc lbtype:LABEL1@\dev

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

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

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

Notes on removing global types:

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

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

Cleaning Up Global Types

Use **checkvob –global** to check and fix global types that are in an inconsistent state. For more information, see Chapter 16, *Using checkvob*.

12 Backing Up and Restoring VOBs

The most important maintenance task for a ClearCase 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 229.

12.1 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 and Solaris for example), **tar** resets file access times, which can disrupt DO and cleartext storage pool scrubbing patterns. For example, the **scrubber** utility, by default,

scrubs DOs older than four days (96 hours). A nightly **tar** operation that backs up DO pools will keep DOs forever underaged, preventing the **scrubber** from doing its work.

On UNIX hosts, VOBs can be created with remote storage pools. If you are backing up a VOB with remote storage pools, it is important to back up these pools as well.

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.

12.2 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. (Remote storage pools complicate the picture. See *Backing Up a UNIX VOB with Remote Storage Pools* on page 176.)

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 461, 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 170.
   - Stop ClearCase on the VOB host, back up the VOB storage directory, unlock the VOB, and restart ClearCase.

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

NOTE: There are various techniques, outside the scope of this manual, to minimize VOB lock time while preserving backup integrity: copying the VOB image to a local disk before backup, using mirrored disk subsystems, running OS-level products creating local copies of a file system or disk partition, using ClearCase MultiSite, separating build VOBs from source VOBs, and so on.

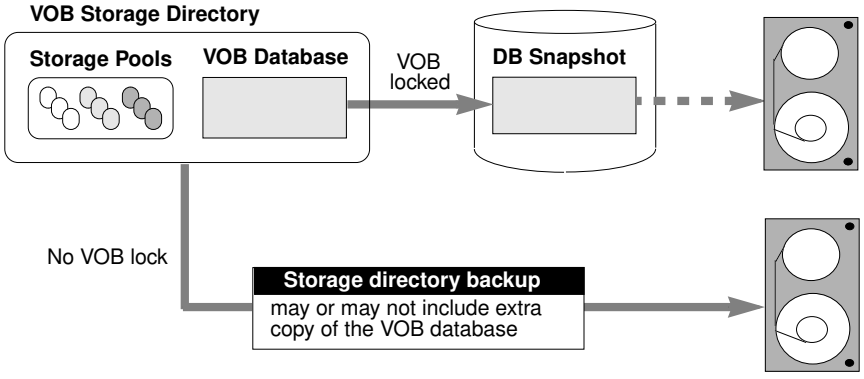
Choosing Between Standard and Semi-Live Backup

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. Note that ClearCase build programs are designed to cope with locked VOBs. They will "sleep" when they are unable to open an object in a locked VOB, and then retry as described on the **clearmake** reference page.As illustrated in Figure 10, 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.

  If your UNIX VOB servers have remote storage pools, you must back them up as well.

Figure 10 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, DO and 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 192.

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.

NOTE: The commands used in the following sections do not require a ClearCase view context or mounted VOBs.

Determining a VOB's Location

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

cleartool lsvob –long \vob_flex

```
Tag: \flex
  Global path: \\ccsvr01\vobstore\vob_flex.vbs
  .
  .
  Region: uno
 .
 .
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 170), you must lock the VOB against all users. Use the ClearCase Administration Console, the UNIX **clearvobadmin** GUI, 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.

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, 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, use the ClearCase program in Control Panel.

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 UNIX, you can use the VOB Admin Browser (**clearvobadmin**). Click **VOB**>**Lock** or **VOB**>**Unlock**. (The **–nusers** option has no analog in the VOB Admin Browser.)

On the command line, use **cleartool lock** and **unlock**:

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

<perform backup>

cleartool unlock vob:/vobs/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 5 to determine the relative importance of the various directories.

Table 5 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
Derived object storage pools	Important, but not required
Cleartext storage pools	Optional
Administrative directory	Important, but not required

DO Pool Backup

Backing up derived object storage pools is not required because, by definition, DOs can be rebuilt from sources. The importance of backing up these pools may change over time:

- In the early stages of a project, when the source base is changing rapidly, the useful life of most derived objects is very short. Omitting DO storage pools from a backup regimen at this stage should not cause a problem.
- When a project is relatively stable, a VOB's DO storage pools contain many often-reused objects. At this stage, a complete build of a software system may *wink in* virtually all DOs, rather than building them. Loss of a DO storage pool may increase the time required for such a complete system build by an order of magnitude.

Before choosing not to back up DO pools, make sure that you and your development staff can accept the time it may take to rebuild these DOs, which can be very long.

If you do not back up DO pools, include each pool's roots in the backup—the pool's root directory (**d\ddft**, for example) and **pool_id** file, but not its subdirectories. Doing so prevents pool root check failure at restore time (see also *Pool Root Check Failure* on page 267).

WARNING: A shared derived object has two parts: an object in the VOB database and a data container in a DO storage pool. Losing DO data containers (for example, by failing to back them up) throws the VOB's database out of sync with its DO storage pools. To resynchronize, you must delete all the empty DOs from the VOB database, using **cleartool rmdo** and/or **checkvob**. See also *VOB and View Resynchronization* on page 200.

Cleartext Pool Backup

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

If you do not back up cleartext pools, include each pool's roots in the backup—the pool's root directory (**c\cdft**, for example) and **pool_id** file, but not its subdirectories. Doing so prevents pool root check failure at restore time (see also *Database or Storage Pool Inconsistencies* on page 241) 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 VOB server host 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.

12.3 Backing Up a UNIX VOB with Remote Storage Pools

If a VOB has remote storage pools, its on-disk storage is probably not wholly contained within a single disk partition. It is quite likely that the storage is distributed among two or more hosts. Here is the procedure for this situation:

1. Lock the VOB.
2. Back up the VOB storage directory.
3. Back up some or all of the VOB's storage pools.

   Use the **lspool** command in any view to determine which storage pools are remote, and their actual locations:

   cleartool lspool –invob vob:/vobs/flex

   ```
   13-Jan.16:58   vobadm     pool "cdft"
     "Predefined pool used to store cleartext versions."
   26-Jan.22:02   vobadm     pool "cltxt01"
     "remote cleartext storage pool for 'flex' VOB"
   13-Jan.16:58   vobadm     pool "ddft"
     "Predefined pool used to store derived objects."
   13-Jan.16:58   vobadm     pool "sdft"
     "Predefined pool used to store versions."
   ```

   In this example, there is one remote pool, **cltxt01**. If you are not sure which storage pools are remote, enter a **lspool –long** command to list the pool storage global pathname of every pool, and examine these pathnames to determine which ones specify remote locations:

   cleartool lspool –long –invob vob:/vobs/flex

   ```
   pool "cltxt01"
        .
        .
     pool storage global pathname
      "/vobstore/flex.vbs/c/cltxt01"
        .
        .
   ```

4. Unlock the VOB.

12.4 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 12.2 and 12.3) 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 VOB host.** 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 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 /vobs/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 179
- *vob_restore: Restoration Scenarios* on page 187
- *vob_restore: Restoring with a Database Snapshot* on page 192

4. (If necessary) **Re-create additional VOB-tags.** When it re-registers the VOB, **vob_restore** creates or re-creates a VOB-tag for the current network region. If the VOB had tags in multiple regions, use the *vob-stg-dir***/../register_save**_*tagleaf* file to re-create the remaining tags. (**vob_restore** runs **cleartool lsvob –region * –long –storage** *registered-stg-location* and saves its output in the **register_save** file.)

   If your site registers the VOB in another registry (not in another region, but in a second registry, served by a second registry host), you must go to a host served by the second registry host and re-register the VOB manually.

5. **If the VOB is replicated, run the MultiSite restorereplica command.** It is critical to perform this processing while the VOB is still locked.

6. **While the VOB is still locked, run some consistency checks.** Activate the VOB on a single host with **cleartool mount** and confirm that it is intact by checking event history on various components, examining recently changed elements, and so on.

7. **Unlock the restored VOB**. **vob_restore** always leaves the VOB locked when it exits.

   ```
   # cleartool unlock vob:/vobs/flex
   Unlocked versioned object base "/vobstore/flex.vbs".
   ```

8. **Mount the restored VOB on all hosts.** The restored VOB is now ready to use. Have users remount the VOB on their workstations, using **cleartool mount**.

9. **If necessary, resynchronize the VOB and views.** See *VOB and View Resynchronization* on page 200.

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**.

*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 187.

```
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.

```
The vob must now be made unavailable. This script will unregister the vob
from this hosts's registry. If it is registered in any other registries
you must unregister it from those registries before continuing. Do not
unregister it from this host's registry. You may quit for now to perform
this operation. Press <RETURN> to continue only when this host's registry
is the only registry the replica remains registered in.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
```

Next, **vob_restore** saves registry parameters before removing the VOB-tag, unregistering the VOB storage directory, and shutting down ClearCase. If you have customized registry entries or cloned the VOB-tag for multiple network regions, you must restore these registry additions manually, using the saved registry file that **vob_restore** creates.

```
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

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
```

If you are restoring a UNIX VOB with remote storage pools, restore them now.

```
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. Failure to do so will irreversibly break a replicated VOB.

```
Do you want to have checkvob examine the vob for possible problems?
Valid responses are (yes,quit,help)
The default is yes: yes<RETURN>

Would you like to have checkvob run in 'check only' mode?
Valid responses are (yes,no,quit,help)
The default is yes: yes<RETURN>

Source pools?
Valid responses are (yes,no)
The default is yes: yes<RETURN>

Derived object pools?
Valid responses are (yes,no)
The default is yes: yes<RETURN>

Cleartext pools?
Valid responses are (yes,no)
The default is yes: yes<RETURN>

Checkvob expects to be run in a view context. If you are not currently in
a view, checkvob will accept a view tag argument. Supply one now if you
are not in a view.
view_tag: adm_view<RETURN>

Checkvob will now be run in 'check only' mode. This may take a while on
large vobs. Checkvob will generate logs in the directory
    checkvob_sum.03-Oct-96.14.21.06
It will emit rather verbose information to standard output as it runs. All
of this information is also saved in the log directory for later viewing.
You may be prompted for information as well. You may quit for now or press
<RETURN> to continue.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
```

At this point, **checkvob** takes control temporarily from **vob_restore**:

```
The session's log directory is 'checkvob_sum.03-Oct-96.14.21.06'.
====================================================================
Starting "source pool" processing at 03-Sep-96.14:21:59

... lots of checkvob output ...

The VOB's derived pools are healthy.

Poolkind transcript log:
checkvob_sum.03-Oct-96.14.21.06\poolkind_derived\transcript
====================================================================
```

checkvob's check-only pass is complete. Control has returned to **vob_restore**, which summarizes the results and prompts you run **checkvob** in fix mode to repair any problems:

```
1 source containers are either missing or corrupt
0 derived object containers are either missing or corrupt
0 source containers are misprotected
0 derived object containers are misprotected
Cleartext containers were not checked.
More details may be found in the log files in the above directory

Do you want to have checkvob run in repair mode?
Valid responses are (yes,no,quit)
The default is yes: yes<RETURN>

Source pools?
Valid responses are (yes,no)
The default is yes: yes<RETURN>

Checkvob will now be run in fix mode. This may take a while on
large vobs. Checkvob will generate logs in the directory
    checkvob_fix.03-Oct-96.14.22.02
It will also emit rather verbose information to standard output as it
runs. All of this information is also saved in the log directory for
later viewing. You may be prompted for information as well.
Do you want to proceed?
Valid responses are (yes,quit)
The default is yes: yes<RETURN>
```

```
While running checkvob the the vob will be locked for all but user
Administrator (or other account used to invoke vob_restore)
The vob will be open to possible modifications by anyone running as
this user. This MUST NOT be allowed to happen. If you need to make
arrangements to ensure that no one with this identity will attempt to
modify this vob before proceeding you may quit for now. Press <RETURN>
if you are satisfied that no such modifications will occur.
Valid responses are (quit,<RETURN> to continue)
There is no default response: <RETURN>
```

checkvob 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.
```

If you restore a VOB replica, **vob_restore** directs you to run **restorereplica** immediately. If you fail to do so, the replica will remain unsynchronized permanently.

VOB restoration is now complete. Go to Step #6 on page 178.

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.

In the previous sections, the examples 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

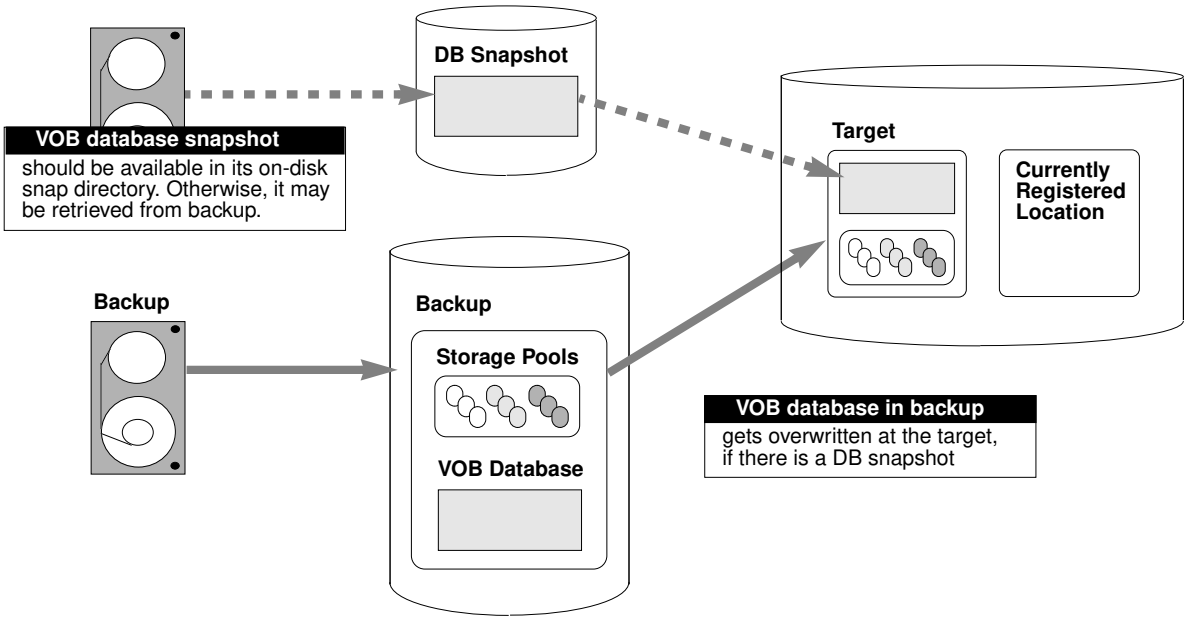
When you run **vob_restore**, it prompts for three critical pieces of information:

- 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. (It may not be registered.)

As shown in Figure 11, **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 11 VOB Restoration



vob_restore uses your input to describe the restore scenario. For example, Figure 12 shows the scenario from the sample restore session in the previous section.

Figure 12 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 local.
- All subdirectories should be local if possible.
- *Snapshot* can reside on 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 large, 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 190.
- 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.

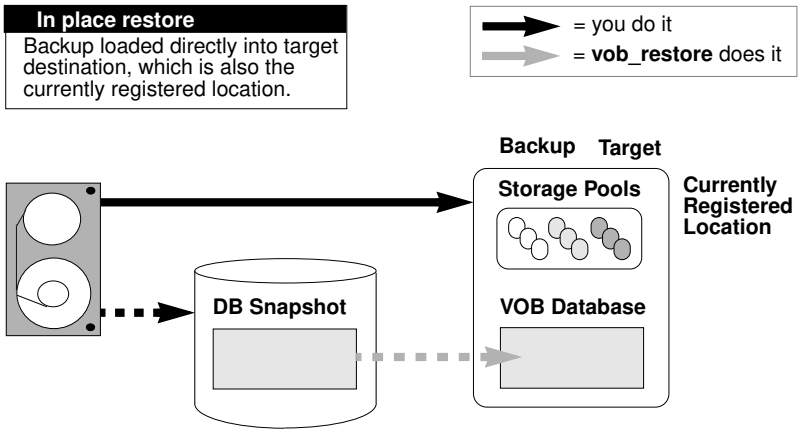
vob_restore: In Place

This scenario is illustrated in Figure 13

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

Advice. Use this scenario whenever practical.

Figure 13 vob_restore: In Place



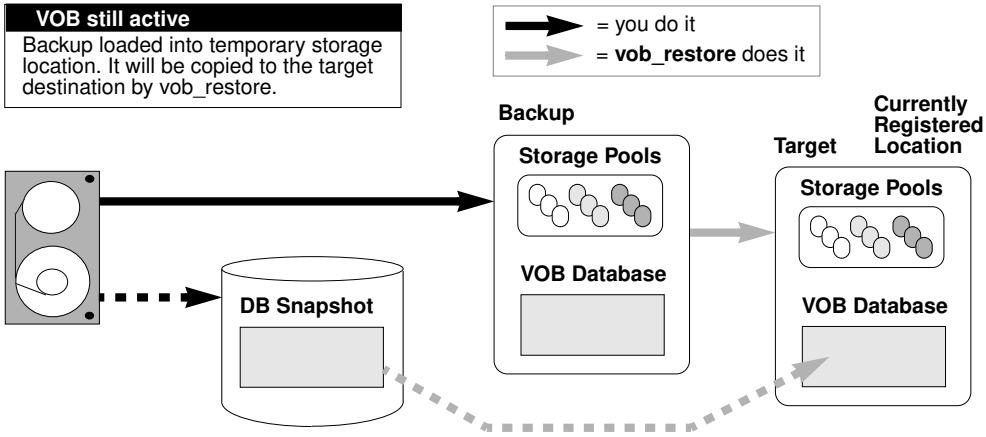
vob_restore: VOB Is Active

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 14.)

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 14 vob_restore: VOB Is Active



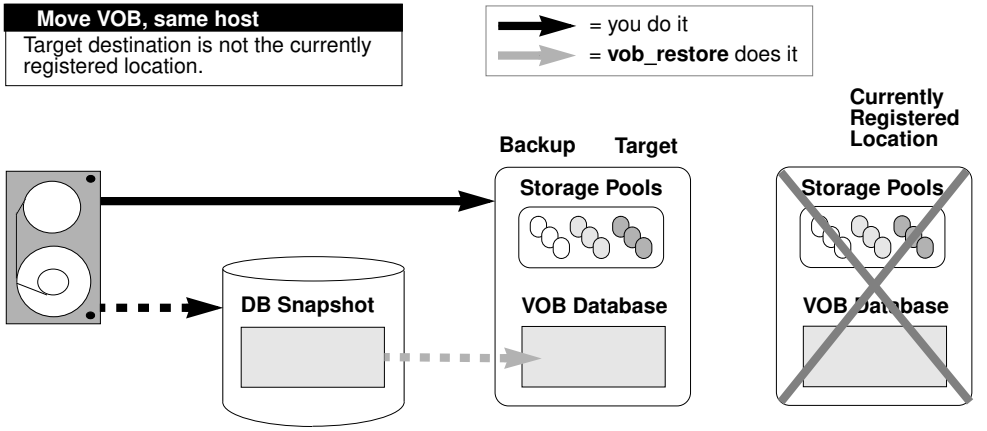
vob_restore: Move VOB on Same Host

In this scenario, the backup is restored to a different location on the same host. (See Figure 15.)

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

Advice. Load the backup directly into the target destination.

Figure 15 vob_restore: Move VOB on Same Host



vob_restore: Move VOB to New Host

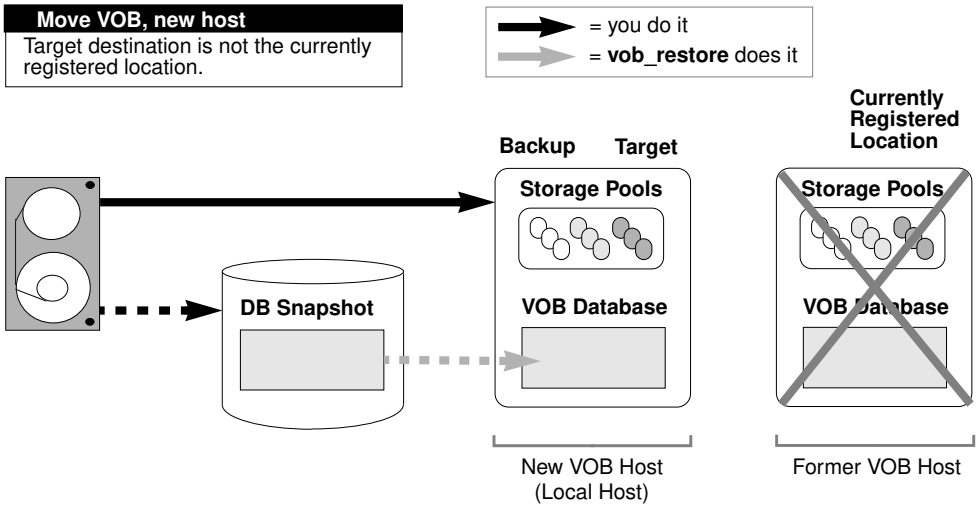
When moving a VOB as part of a **vob_restore** operation, the target host must have the same architecture (hardware and operating system). (See Figure 16.)

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

Advice. Load the backup directly into the target destination.

NOTE: You must run **vob_restore** on the target host.

Figure 16 vob_restore: Move VOB to New Host



vob_restore: Unregistered

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

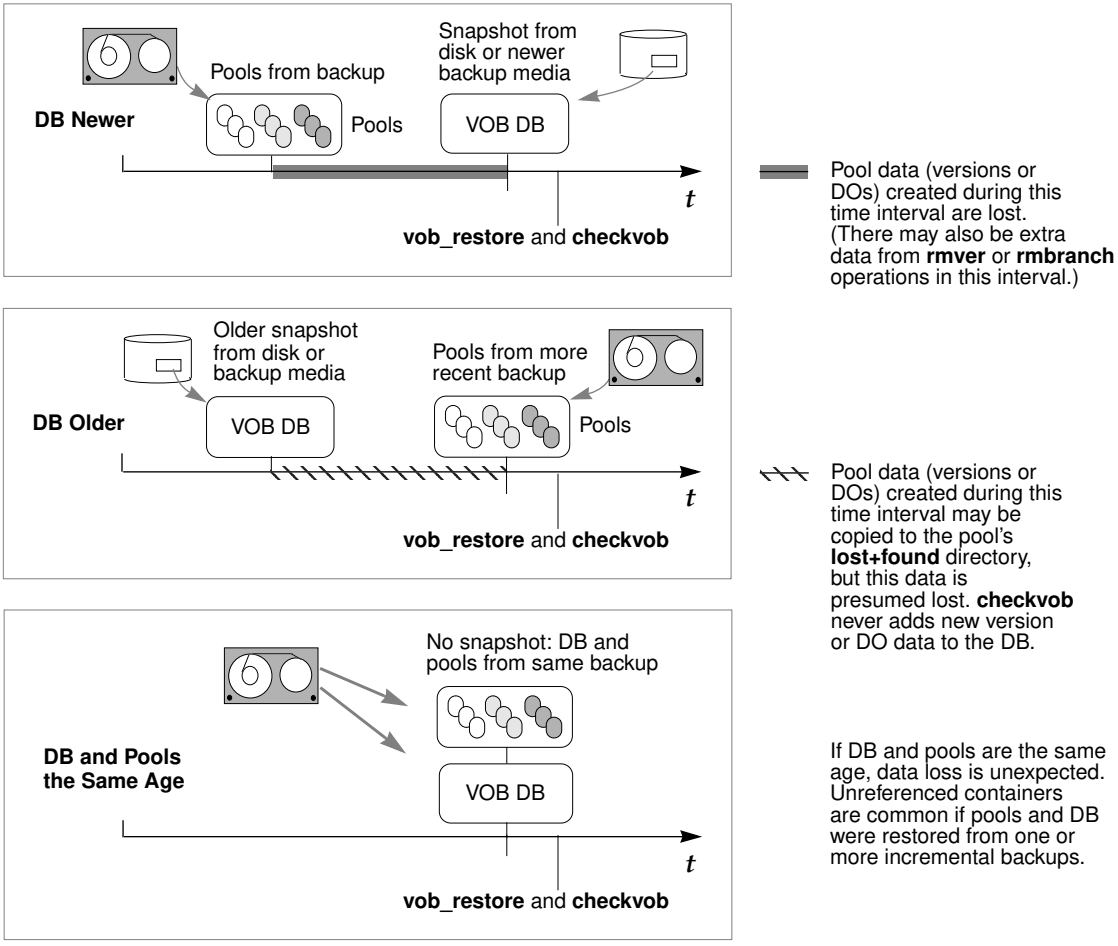
Advice. Load the backup directly into the target destination.

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 17 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 17 VOB Database or Storage Pool Skew Associated with VOB Snapshots



VOB snapshots have minimal impact on the your work at restore time. When prompted by **vob_restore**, you need only supply the correct snapshot directory. However, the **checkvob** portion of a **vob_restore** operation is critical. You must be prepared to respond intelligently to **checkvob** prompts, and to understand the implications of any errors reported in its output. Typically, **checkvob** finds and repairs numerous small problems, but it reports version and DO data loss for the interval between storage pool and VOB database reference times.

For more information, see the **checkvob** reference page and Chapter 16, *Using checkvob* in this manual.

12.5 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 168.

NOTE: You can restore a VOB to a new location, on the same host or on another host. In this case, you must re-register the VOB at its new location (Step #9).

1. **Unmount the VOB**. The VOB must be unmounted on each client host. For example:

   ```
   # cleartool umount /vobs/flex
   ```

2. **Log on to the VOB host.** 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.

3. **Check available disk space**. If you are restoring the VOB to a new location, make sure 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 #4 and Step #5 are critical to the integrity of your restored VOB.

4. **Shut down ClearCase on this host.** 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**

5. **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**

6. **Restart ClearCase on this host.** Starting ClearCase makes other VOBs on the host 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**

7. **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 36, *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.

   If the VOB is on UNIX and has remote storage pools, restore the backups of these pools at this point.

8. **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.

9. **If you restored the VOB to a new location, re-register the VOB**. You can use the ClearCase Registry node in the ClearCase Administration Console or **cleartool** commands to 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
   ```

10. **If the VOB is replicated, run the MultiSite restorereplica command.** It is critical that this processing is performed while the VOB is still locked.

11. **While the VOB is still locked, run some consistency checks.** Activate the VOB on a single host with **cleartool mount** 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 233, and so on.

12. **Unlock the restored VOB**.

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

13. **Mount the restored VOB on all hosts.** The restored copy of the VOB is now ready to use. Have users remount the VOB on their workstations, using **cleartool mount**.

14. **If necessary, resynchronize the VOB and views.** See *VOB and View Resynchronization* on page 200.

12.6 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 into 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. Unmount and unregister the VOB whose element has been deleted.
2. Restore the most recent backup of the VOB storage directory to a temporary location on a VOB server disk.
3. Register and mount the restored backup.
4. Create a temporary VOB to hold only this element, then mount it. 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. Unmount the backup VOB.
7. Remount the real VOB.
8. Use **clearexport_ccase** and **clearimport** to copy the element from the temporary VOB to the real 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. **Unmount the VOB whose element has been deleted**. This is required, because two copies of the same VOB must never be active at the same time.

   cleartool umount \proj

2. **Remove the VOB-tag and registry entries**. These entries prevent use of an old version of the same VOB. You can use the ClearCase Registry node in the ClearCase Administration Console or **cleartool** commands to unregister the VOB.

   cleartool rmtag –vob \proj
   cleartool unregister –vob \\sol\vobstore\proj.vbs

3. **(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.)

4. **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 194.

5. **Register and mount the backup VOB**. As in the preceding step, use a temporary mount point, not the original mount point. You can use the ClearCase Registry node in the ClearCase Administration Console or **cleartool** commands to register the VOB or you can use the cleartool command line.

   cleartool register –vob \\sol\users\tmp\proj.vbs
   cleartool mktag –vob –tag \oldproj \\sol\users\tmp\proj.vbs
   cleartool mount \oldproj

6. **Create a new, temporary VOB.** Because the real 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 unmount the backup, mount the real VOB, and copy the element from the temporary VOB to the real VOB. Because nobody else should access this VOB, do not make it *public*. 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.
     .
     .
   ```

7. **Mount the temporary VOB.**

   cleartool mount \tmpvob

8. **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:\> cleartool setview dfl *(set a dynamic view configured with the default config spec)*
(or use an appropriate snapshot view)

```
Assigned drive "Z:" to "M:\dfl"
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".
```

9. **Unmount and unregister the backup VOB**. This corresponds to the work done in Step #1 and Step #2. You can use the ClearCase Registry node in the ClearCase Administration Console or these **cleartool** commands to unregister the VOB.

   **cd **
   cleartool umount \oldproj
   cleartool rmtag –vob \oldproj
   cleartool unregister –vob \\sol\users\tmp\proj.vbs

10. **(UNIX server only) Terminate the backup VOB's server processes**. This is similar to Step #3 of this procedure. This time, search the process table for a **vob_server** and/or **vobrpc_server** invoked with **/usr/tmp/proj.vbs**.

11. **Re-register and remount the original VOB**. This time, make a public VOB-tag. You can use these **cleartool** commands to re-register the VOB.

    cleartool register –vob \\sol\vobstore\proj.vbs
    cleartool mktag –vob –public –tag \proj \\sol\vobstore\proj.vbs
    Vob tag registry password: *<enter password>*
    cleartool mount \proj

    NOTE: If you are registering a Unified Change Management *Process VOB*, you must supply the **–ucmproject** option to the **register** command.

12. **Copy the element from the temporary VOB to the original VOB.** Use the **clearexport_ccase** program as in Step #8 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".
```

13. **Clean Up.** Unregister and unmount 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 umount \tmpvob
```

```
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".
```

12.7 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 a variety of problems.

View-related problems

- View-private files can become stranded because the VOB directory element that they were cataloged in (or under) no longer exist in the VOB.
- 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.)
- DO promotions can fail because the VOB records DO information about unshared DOs that don't exist.
- Winked-in DOs become inaccessible when the VOB no longer contains the shared DOs. For example, this can occur if the VOB loses its DO pools and the shared DOs are removed by **checkvob** fix processing.

VOB-related problems

- DOs are prevented from being scrubbed because the VOB has recorded stale reference counts for views (references that the view currently doesn't have).

There are a several things you can do to correct the skew and identify and/or eliminate the problems. You can take action immediately following VOB or view restore, or you can wait until problems surface.

- When a view is restored, perform the resynchronization described in *Resynchronizing Views and VOBs* on the restored view.
- When a VOB is restored, resynchronize it on each view it references. To collect this view list, navigate to the Referenced Views subnode for the VOB storage node in the ClearCase Administration console or run the following command:

  cleartool describe –long vob:*restored-vob*

NOTE: The Referenced Views subnode and the **cleartool describe** command do not list views whose only interaction with the VOB has been to wink in DOs from the VOB. You must find and resynchronize any such views as well.

Resynchronizing Views and VOBs

To resynchronize views and VOBs:

1. (Dynamic views only) Run **cleartool recoverview –synchronize**. This command recovers stranded view-private files. If a view was restored, synchronize it with all of the VOBs. If a VOB was restored, you can restrict the recovery synchronization to the restored VOB. See the **recoverview** reference page for more information.

2. For a dynamic view, inspect the list of checked-out files in the Private Files subnode of the view storage node in the ClearCase Administration Console or run **cleartool lsprivate –co**. For a snapshot view, 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, this probably means that the user had once checked out the element but then resolved the checkout (**uncheckout** or **checkin**) at some point more recent than the restored VOB's backup time. If it was a checkin, the data has been lost.

   If the view was restored, any view-private modifications to the checked-out element have been lost.

   In either case, the user must now cancel the checkout (**uncheckout**) or re-create the version data and check it in.

3. For a dynamic view, inspect the list of files in the Private Files subnode of the view storage node in the ClearCase Administration Console or run **cleartool lsprivate –other –long**. For a snapshot view, run **cleartool ls –recurse –view_only**. Look at the output for any elements that are marked as "eclipsed."

   If the VOB was restored, the user had probably checked out the element at a time more recent than the restored VOB's backup time. If the view was restored, this probably means that the user had checked out the element but canceled the checkout later on, or had checked in the version data now present in the restored view.

   If the version had been checked in, the eclipsing view-private file can be removed and the user can check out a new version and continue working. If the version had not been checked in, the eclipsing view-private file may still contain the most recent changes. In this case, it must be moved aside (renamed, not removed) so that the a new version of the element can be checked out to hold these changes.

4. (Dynamic views only) View the list of derived objects in the Private Files subnode of the view storage node in the ClearCase Administration Console or run **cleartool lsprivate –do**.

Attempt to open each file. The view then detects winked-in DOs that are no longer valid and removes them.

NOTE: On UNIX systems, you can use the UNIX **xargs** utility to automatically open the file names returned by **lsprivate.** This command does the job for 50 DOs:

cleartool lsprivate –do | xargs –n50 file

5. (Dynamic views only) Remove all of the view's **.cmake.state** files. The view then queries the VOB regarding the state of DOs created by this view. Rebuilding in the view causes the view to detect view-private files that are no longer recognized as DOs and rebuild them.

After you fix DO promotion problems for a dynamic view, isolated problems may still occur (for example, when a view's DO container does not exist). To prevent this problem from reoccurring, remove the offending DO with **rmdo**. The following section shows a sample cleanup of DOs whose config records have been lost, causing errors at build time.

Reestablishing Consistency of a View's Derived Object State

This section applies to dynamic views in any situation where a VOB's database has been rolled back to a previous state.

After you restore a VOB from backup, its VOB database may be out of date with respect to certain derived objects. The old database does not store config records for any DOs that were created in subsequent ClearCase builds. As a result, errors occur during hierarchical builds that reuse those late-arriving DOs to construct higher-level targets. The following example is typical of the error messages generated when this problem occurs:

```
======== Rebuilding "libbld.a"========
building libbld.a
        rm –f libbld.a
        rm –f /vobs/atria/sun5/pvtlib/libbld.a
        /opt/SUNWspro/bin/cc –c –o ...
        ar cq libbld.a  bld.o bld_pp.o ...

Will store derived object "/vobs/proj/sun5/libbld_V.o"
Will store derived object "/vobs/proj/sun5/libbld.a"
clearmake: Error: INTERNAL ERROR detected and logged in
"/var/adm/atria/error_log".
```

The **error_log** file shows:

```
Sunday 12/19/93 15:55:02. host "scandium", pid 440, user "chase"
Internal Error detected in "../bldr_vob.c"line 114
clearmake/cm/bldr_vob:
Error: VOB "scandium:/vobs/proj"
missing config record for derived object (OID)
"0b5759d0.fb1811cc.a0af.08:00:69:02:2e:aa"
```

To reestablish the view's consistency with the VOB:

1. **Determine which DOs are causing the inconsistency.** The **cleartool ls** command annotates them with `[no config record]`:

   cleartool ls

   ```
   bldr_comm.ugh@@09-Dec.18:26.287028
   bldr_cr.msg.o [no config record]
   bldr_cr.o [no config record]
   bldr_cr.ugh [no config record]
   bldr_cr_cache.msg.c@@24-May.20:51.42929
     .
     .
     .
   ```

2. **Remove the DOs that have no config record.** Use the standard **rm(1)** command on UNIX:

   rm bldr_cr.msg.o bldr_cr.o bldr_cr.ugh

   Use the **del** command on Windows NT.

   del bldr_cr.msg.o bldr_cr.o bldr_cr.ugh

13 Administering VOB Storage

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

13.1 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 18 on page 206 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.

The ClearCase Administration Console displays information on disk space used by the VOBs on all ClearCase hosts on your network:

- The VOB storage node for a VOB (a subnode of the host node for the host where the VOB storage directory resides) shows current and historical disk space use for the VOB.
- The Derived Objects subnode of a VOB storage node shows disk space used by shared derived objects in the VOB. The display shows which views have references to these DOs.

You can also use **cleartool** subcommands to display information on disk space used in VOBs:

- The **space –vob** command shows current and historical disk space use for a VOB.

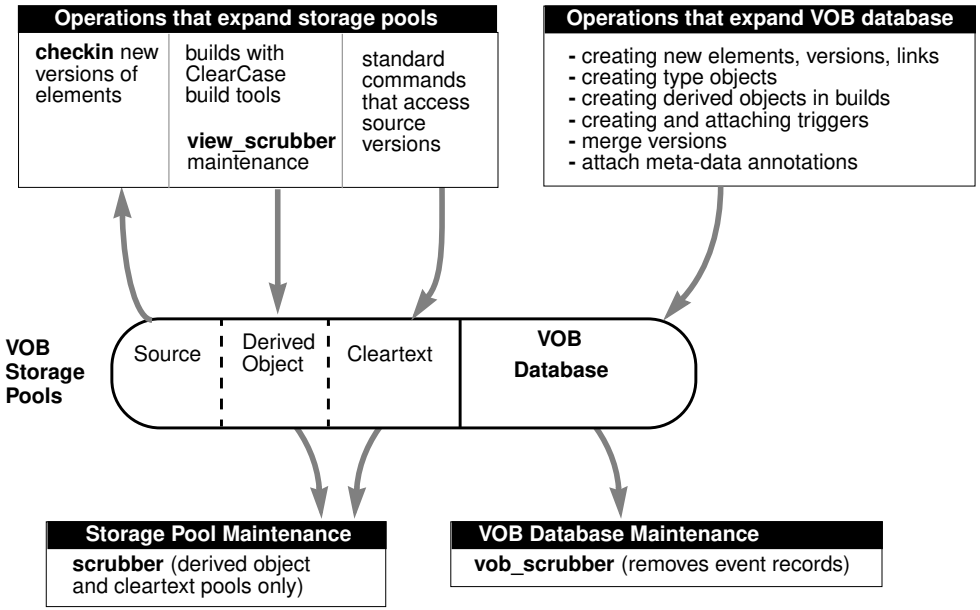
- The **dospace** command shows disk space used by shared derived objects in the VOB. The display also shows which views have references to these DOs.

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
- Weekly data gathering on disk space used by shared derived objects
- 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 28, *Managing Scheduled Jobs*.

Figure 18 Controlling VOB Growth



Scrubbing VOB Storage Pools

On any VOB host, the ClearCase scheduler runs a daily job that scrubs the storage pools of all VOBs whose storage directories reside on that host:

- 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 208.)
- Derived object pools are scrubbed to delete DO data containers that are no longer being used by any view (those whose *reference counts* are zero). Scrubbing also removes the corresponding derived objects from the VOB database.
- 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 derived object and cleartext storage pool has its own scrubbing parameters, which control how the **scrubber** processes that pool. To change the way VOB storage pools are scrubbed, you can change the scrubbing parameters for an individual pool using the ClearCase Administration Console or a **mkpool –update** command.

See *Adjusting Storage Pool Scrubbing* on page 213 for an example of modifying a pool's scrubbing parameters.

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 host has a scrubber configuration file, *ccase-home-dir***config\vob\vob_scrubber_params**, which controls **vob_scrubber** operation for all VOBs on that host. 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 and derived objects 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 and derived objects.

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.

13.2 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 380 for more on this topic.

13.3 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 196.)

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

13.4 Creating Additional Storage Pools for UNIX VOBs

If a VOB on a UNIX server is threatening to fill up its disk partition, you may want to rework the VOB's storage pools to take advantage of remote storage. (This feature requires UNIX symbolic links, so it's not available on Windows NT.) The VOB database (**db** subdirectory) must be physically located within the VOB storage directory, but any or all of its storage pools can be remote. If necessary, you can abandon the VOB's default storage pools and transfer all of the VOB's file-system data to other disk partitions and/or remote hosts.

The host on which you create a new storage pool need not have ClearCase installed. It can have any hardware/software architecture, but the pool must be NFS-accessible at the same pathname from all ClearCase hosts (for example, **/net/ccsvr02/ccase_pools/do_3**). See *Remote Storage Pools on UNIX* on page 115 for additional information.

NOTE: The ClearCase *network region* facility does not apply here. The pathname of a remote storage pool must be truly global.

When deciding which hosts to use for new storage pools, consider that each kind of storage pool has a different pattern of use:

- **Source pools.** These pools store the most precious data: the checked-in versions of file elements. Traffic involving these pools is relatively light, but data integrity is very important. (See *Caution on Remote Source Pools*.) The ideal location for such pools is a robust file server, with a large capacity and frequent, reliable data backups.
- **Cleartext pools.** These pools probably get the heaviest traffic (assuming that many of your file elements are stored in *delta* and/or compressed format). But the data in cleartext pools is expendable, because ClearCase can reconstruct it. The ideal location for such pools is a machine with a fast file system.

- **Derived object pools.** These pools can become quite large, depending on the number of active configurations (three new development projects, two old releases in maintenance, and so on). Anticipate the storage requirements in the new pool for each active configuration; make sure the disk partition can handle the total storage requirement.

Storage pools can be examined, created, moved, and manipulated in various ways from the ClearCase Administration Console. You can also use the command line. These are the principal commands for working with storage pools:

lspool
: Lists a VOB's existing storage pools.

describe
: Lists an element's storage pool assignments. You must use the extended naming symbol (@@) when specifying the element:

cleartool describe getcwd.c *(wrong)*
cleartool describe getcwd.c@@ *(right)*

mkpool
: Creates a new storage pool. Use the **–ln** option to create a remote storage pool.

chpool
: Reassigns a file or directory element to another storage pool. (A directory element itself is stored entirely within the VOB database; a directory's pool assignments control which pools are used by new file elements and derived objects created within the directory.)

Caution on Remote Source Pools

We recommend that you keep source storage pools local, within the VOB storage directory. This strategy optimizes data integrity: a single disk partition contains all of the VOB's essential data. It also simplifies backup/restore procedures. This concern typically overrides performance considerations, because losing a source pool means that developers must re-create the lost versions.

Example: Assigning All Files in a Directory to a New Pool

The following example shows how to create a new, remote source storage pool, and then reassign all the current and future elements in a particular directory to the new pool.

1. **Create the new storage pool**. Specify a global pathname for the remote pool that is valid for all hosts that will access the VOB.

   cd /vobs/bgr
   cleartool mkpool –source –ln /net/ccsvr02/ccase_pools/bgrsrc2 bgrsrc2
   ```
   Comments for "bgrsrc2":
   ```
   remote source storage pool
   .
   ```
   Created pool "bgrsrc2".
   ```

2. **Reassign existing file elements to the new pool.** This example reassigns all the file elements in a particular development subdirectory.

   cd libbgr
   cleartool find . –type f –exec 'cleartool chpool –force bgrsrc2 $CLEARCASE_PN'
   ```
   Changed pool for "./Makefile" to "bgrsrc2".
   Changed pool for "./errmsg.c" to "bgrsrc2".
   Changed pool for "./fork3.c" to "bgrsrc2".
   Changed pool for "./get.c" to "bgrsrc2".
   Changed pool for "./getcwd.c" to "bgrsrc2".
     .
     .
   Changed pool for "./stint.h" to "bgrsrc2".
   Changed pool for "./strut.c" to "bgrsrc2".
   ```

3. **Reassign the directory element to the new pool, too.** All newly created file (and directory) elements in this directory use the new pool, also.

   cleartool chpool bgrsrc2 .
   ```
   Changed pool for "." to "bgrsrc2".
   ```

Example: Moving an Existing Storage Pool to Another Disk

There is no single command for moving an existing storage pool. ClearCase routines perform standard UNIX operations to access storage pools and the data containers within them. Thus, you can move a storage pool in this way:

1. **Determine the location of the storage pool**. Use the **lspool** command:

cleartool lspool –long d_aux@/vobs/bgr

```
pool "d_aux"
 .
 .
pool storage global pathname
"/net/ccsvr01/vobstore/bgr.vbs/d/d_aux"
```

2. **Lock the VOB**. Any new shared DOs are not being placed in the storage pool while you are working on it:

   cleartool lock vob:/vobs/bgr

   ```
   Locked versioned object base "/net/ccsvr01/vobstore/bgr.vbs".
   ```

3. **Copy the contents of the storage pool**. The storage pool is a standard UNIX directory. You can copy its contents to a new location using **cp**, **rcp**, **tar**, or other commands. For example:

   rlogin ccsvr01
   mkdir –p /vobstore_2/DO_pools
   cp –r /vobstore/bgr.vbs/d/d_aux /vobstore_2/DO_pools

4. **Replace the old storage pool with a symbolic link**. Move the old storage pool aside; then create the link in its place.

   cd /vobstore/bgr.vbs/d
   mv d_aux d_aux.MOVED
   ln –s /net/ccsvr01/vobstore_2/DO_pools/d_aux d_aux

   Be sure the text of the symbolic link is a globally valid pathname to the new storage pool location.

5. **Unlock the VOB.**

   cleartool unlock vob:/vobs/bgr

   ```
   Unlocked versioned object base "/net/ccsvr01/vobstore/bgr.vbs".
   ```

6. **Remove the old storage pool**. When you have verified that the storage pool is working well in its new location, you can remove the old pool:

   rm –fr /vobstore/bgr.vbs/d/d_aux.MOVED

NOTE: If a VOB-tag exists for the VOB in a Windows region, you need to re-create that VOB-tag to account for the relocated storage pool. See *Windows Tags for UNIX VOBs with Symbolically Linked Storage* on page 100.

13.5 Adjusting Storage Pool Scrubbing

Typical motivations for adjusting a VOB host's default procedures for storage pool scrubbing include:

- **Not enough space.** The disk partitions in which VOB storage pools reside may fill up frequently. A more aggressive scrubbing strategy may be necessary.
- **Not enough time.** The **scrubber** utility may be taking too much time to complete, interfering with other overnight activities, such as nightly software builds. A less aggressive scrubbing strategy may be necessary.

You may need to experiment. For example, adjusting scrubbing to take place less frequently may cause disk-space problems that you had not previously experienced. Before making any scrubbing adjustments on a VOB host, be sure to analyze its **scrubber_log** file. The **scrubber** reference page explains how to read this file.

NOTE: If you use a Windows NT backup tool that changes the time stamps in VOB storage directories, the DO and cleartext pools in those directories may never be scrubbed. The **scrubber** command, by default, scrubs objects that have not been referenced for the last 96 hours. If such a backup tool runs every night, the access time on objects in the pools is reset every night and the objects are never scrubbed.

The following sections present some simple examples of adjusting the way VOB storage pools are scrubbed.

Scrubbing Derived Objects More Often

By default, **scrubber** allows data containers of unreferenced derived objects to remain in their storage pools for 4 days (96 hours). If DOs are filling up a VOB's disk partition, you can shorten this grace period. This command empties a VOB's default DO storage pool (**ddft**) of unneeded data containers every 24 hours:

cleartool mkpool –update –age 24 ddft@*vob-tag*

```
Updated pool "ddft".
```

Fine-Tuning Derived Object Scrubbing

Suppose that the adjustment in the grace period is not enough to keep the disk partition from filling up. You may decide to run the **scrubber** utility more often: during the work day as well as overnight. To minimize the impact on users during the work day, you can pinpoint the scrubbing—perhaps to the DOs created in a particular directory. This example shows the UNIX command line syntax for moving a DO pool and scrubbing it more often:

1. **Determine the directory's current DO storage pool assignment.** You will need to clean up this storage pool.

   cd /vobs/proj
   cleartool describe –long reorg@@
   ```
   directory element "reorg@@":
     .
     .
     ...  derived pool: ddft
   ```

   This directory uses the default DO pool.

2. **Assign the directory to a separate storage pool.** This assignment enables finer control of scrubbing, which can be invoked on a per-pool basis:

   cd /vobs/proj
   cleartool mkpool –derived new_do_pool
   ```
   Comments for "new_do_pool":
   ```
   pool for DOs created in /vobs/proj/reorg
   .
   ```
   Created pool "new_do_pool".
   ```
   cleartool chpool new_do_pool reorg
   ```
   Changed pool for "reorg" to "new_do_pool".
   ```

3. **Determine the location of the VOB storage directory**. You'll need the pathname of the VOB's storage directory for **scrubber**. Use **lsvob** to determine the pathname:

   cleartool lsvob /vobs/proj
   ```
   * /vobs/proj /net/ccsvr03/vobstore/proj.vbs
   ```

4. **Scrub the new storage pool thoroughly and often**. There are many ways to accomplish this. You can create a new task for the ClearCase scheduler that invokes the **scrubber** utility for your new pool:

   ```
   "$ATRIAHOME/etc/scrubber –e –p new_do_pool \
      /net/ccsvr03/vobstore/proj.vbs"
   ```

This script invokes the **scrubber** utility on the derived object storage pool **new_do_pool**. The **–e** option to **scrubber** empties the pool of all zero-referenced DOs.

You can then register your task in the scheduler's task database and create a new scheduled job to execute the task several times per day. For more information on tasks and jobs, see Chapter 28, *Managing Scheduled Jobs*.

5. **Clean up the old DO storage pool**. The **chpool** command in Step #2 does not move existing DO data containers; it only affects where a new DO's data container is stored. Accordingly, you should clean up the old storage pool:

   ccase-home-dir/**etc/scrubber –e –p ddft /net/ccsvr03/vobstore/proj.vbs**

Scrubbing Less Aggressively

If the ClearCase scheduler's scrubbing regimen takes too long (perhaps spilling over into the work day), you can make the starting time for the ClearCase default Daily VOB Pool Scrubbing job earlier. Alternatively, you can disable the Daily VOB Pool Scrubbing job and define your own job that changes the way that scrubber is invoked, so that it takes less time to run.

Here's how you can revise scrubbing to process DO pools only, leaving cleartext pools alone:

1. Define a *task* whose executable program invokes the scrubber as follows:

   ```
   ccase_home_dir/etc/scrubber -f -a -k do
   ```

2. Register the task in the scheduler's task database.

3. Define a new *job* in the scheduler that runs your task daily. Choose an appropriate starting time.

4. Disable the ClearCase default Daily VOB Pool Scrubbing Job in the scheduler. You can disable a job by setting its end date to a time in the past or by deleting the job.

5. Check all other jobs with sequential schedules. Change the schedule for any job that is defined to follow the ClearCase default Daily VOB Pool Scrubbing job to follow your new job instead. The ClearCase default Daily VOB Snapshots job follows Daily VOB Pool Scrubbing, so you must change this job to follow your new job.

For information about defining tasks and jobs for the ClearCase scheduler, see Chapter 28, *Managing Scheduled Jobs*.

14 Moving VOBs

VOBs cannot be copied from one location to another using an ordinary file copy utility. Special procedures must be followed to maintain the integrity of VOB data and to preserve permissions and ownership on VOB storage locations. This chapter presents procedures for moving VOBs. Three scenarios are covered:

- Moving a VOB from one disk partition to another on Windows NT or between two Windows NT hosts.
- Moving a VOB from one disk partition to another on UNIX or between one UNIX host and another UNIX host with the same binary format.
- Moving a VOB from one UNIX host to another UNIX host with a different binary data format.

In addition, it describes special considerations you must take when moving a replicated VOB.

CAUTION: Failure to follow the instructions in this chapter when you move a replicated VOB may result in replica divergence and data loss.

14.1 Restrictions on Moving a VOB

In general, when you move a VOB using the procedure presented here, the user and group information in the new location must match the user and group information in the old location.

NOTE: On Windows NT, when you move a VOB from one host to another, both hosts must be in the same Windows NT domain.

Do not use ClearCase MultiSite to create multiple replicas of a VOB in a single ClearCase region. Because the VOB UUID is identical for all replicas in a VOB family and is stored in many structures within a VOB, there is no way to make a replica unique. Creating and using multiple replicas of a VOB in a single region causes ClearCase to exhibit unpredictable behavior, may cause data loss, and is not supported by Rational Software.

You cannot use the procedures described in this chapter to move a VOB from a UNIX host to one running Windows NT. You may be able to use the following procedures:

- If you use ClearCase MultiSite, you can use MultiSite facilities to make a new replica of the existing VOB and then delete the old replica. For more information, see the *ClearCase MultiSite Manual*.
- You can use **clearexport_ccase** to copy information in the existing VOB onto a disk volume that both the UNIX and Windows NT host can access, and then use **clearimport** to import the exported data into the new VOB.

14.2 Special Considerations for Replicated VOBs

If you are moving a VOB replica to a new host, take the following steps first:

1. **Make sure the replica masters its own replica object.** For a replica named **portland**:

   a. Check which replica has mastership of the **portland** replica object:

   cleartool describe replica:portland@\libpub
   ...
   `master replica: west`

   b. If the replica object is not self-mastered, change its mastership. At the replica that masters the replica object:

   multitool chmaster replica:portland@\libpub replica:portland@\libpub

   This step is not mandatory, but it is generally recommended that all replicas master their own replica objects. (You can make this change at any time, as long as the replicas are synchronized.) For the purposes of moving a replica, this self-mastership prevents your team from having to diagnose and repair misdirected packet problems that may result after the move. If the replica object names the wrong host, packets are sent to that host. (If this happens, move misdirected packets to the correct host and then import them.)

2. **Change the replica's host name property.** Use **multitool chreplica** to update the replica's host name. You must run this command from the site that masters the replica, which is the current site in most cases:

   multitool chreplica –host *target-host* **replica:portland@\libpub**
   ```
   Updated replica information for "portland".
   ```

NOTE: All replicas that export to the replica to be moved must be updated after the move through synchronizations from the moved replica's site (or from the site that executed the **chreplica** command in Step #2, if it was not the current site).

14.3 Moving a VOB on Windows NT

This section explains how to move a VOB to another disk partition on the same Windows NT host or to a new Windows NT host.

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 mounted by client hosts at **\libpub**.
- The new location for the VOB storage directory is **c:\vobstore2\libpub.vbs**. The example includes these cases:
  - The new location is also on **sol**.
  - The new location is on another host, named **ccsvr04**.

The Move VOB Procedure for Windows NT

Perform this procedure as the VOB owner:

1. **Deactivate the VOB.** Issue this command on each host where the VOB is currently active:

cleartool umount \libpub

NOTE: It may not be practical to unmount the VOB from all hosts. In this case, the VOB lock to be applied in Step #4 ought to prevent unintended VOB access.

2. (If applicable) **Disable VOB snapshots on the current host.** 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 12, *Backing Up and Restoring VOBs*

4. **Lock the VOB**.

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

5. **Stop ClearCase**. On the host where the VOB storage directory resides, open Control Panel. In the ClearCase program, click the **Services Startup** tab and 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.

   - To the same host:

     C:\vobstore> *ccase-home-dir***\etc\utils\ccopy libpub.vbs \vobstore2\libpub.vbs**

   - To a different host:

     C:\> **cd \vobstore**
     C:\vobstore> **net use w: \\ccsrv04\vobstore2**
     C:\vobstore> *ccase-home-dir***\etc\utils\ccopy libpub.vbs w:\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 host where the VOB storage directory resides, open the ClearCase program; click **Startup ClearCase** on the **Services Startup** tab.

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, and you can use the VOB Objects subnode of the ClearCase Registry node to remove the VOB object. You can also use these commands:

   cleartool unregister –vob \\sol\vobstore\libpub.vbs
   cleartool rmtag –vob –all \libpub

10. **Register the VOB at its new location**. When you move a *public* VOB, you must enter the *registry password*. In the ClearCase Administration Console, you can use the VOB Tags node for the tag's regions to create VOB-tags, and you can use the VOB Objects subnode of the ClearCase Registry node to create a VOB object entry. You can also use these commands:

    cleartool register –vob \\ccsvr04\vobstore2\libpub.vbs
    cleartool mktag –vob –public –tag \libpub \\ccsvr04\vobstore2\libpub.vbs
    `Vob tag registry password: <enter password>`

    NOTE: If you are registering a Unified Change Management *Process VOB*, you must supply the **–ucmproject** option to the **register** command.

    If your network has several network regions, see *Ensuring Global Access to the VOB—Special Cases for UNIX* on page 128 for a discussion of adjustments and additional registry entries you may need to make.

11. **Reactivate the VOB**. On all client hosts:

    cleartool umount \libpub *(if not already done)*

    cleartool mount \libpub

12. **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.

13. (If applicable) **Enable VOB snapshots on the new host.** If you want to enable VOB database snapshots on the new VOB host, run the **vob_snapshot_setup** command and supply the appropriate parameters.

14.4 Moving a VOB on UNIX (Same Architecture)

This section explains how to move a VOB to another disk partition on a single UNIX host or between UNIX hosts whose binary formats for the files that implement the VOB database are the same. To move a VOB between UNIX hosts with different formats for VOB database files, see *Moving a VOB Between UNIX Hosts (Different Architectures)* on page 225.

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 mounted by client hosts at **/libpub**.
- The new location for the VOB storage directory is **/vobstore2/libpub.vbs**. The example includes these cases:
  - The new location is also on **sol**.
  - The new location is on another host, named **ccsvr04**.

The UNIX Move VOB Procedure

To move the VOB:

1. **Determine whether the VOB has any nonlocal storage pools.**

   cleartool lspool –long –invob /proj/libpub | egrep '(^pool|link)'

   ```
   pool "cdft"
   pool "ddft"
   pool "sdft"
   pool "s_2"
     pool storage link target pathname "/net/ccsvr04/ccase_pools/s_2"
   ```

   There is one remote pool, **s_2**, and its pathname must be valid on the new VOB host and on all client hosts.

2. **Verify the validity of pathnames to nonlocal pools**. Moving a VOB storage directory does not move any of its remote storage pools. You must make sure that the VOB's new host can access each remote storage pool using the same global pathname as the VOB's current host:

   - If you are moving the VOB to another location on the same host, the validity of remote storage pool global pathnames is assured.
   - If you are moving the VOB to a different host, log on to that host and verify that all the remote storage pool global pathnames are valid on that host.

3. **Deactivate the VOB**. Issue this command on each host where the VOB is currently active:

   cleartool umount /proj/libpub

   NOTE: It may not be practical to unmount the VOB from all hosts. In this case, the VOB lock to be applied in Step #4 ought to prevent unintended VOB access.

4. (If applicable) **Disable VOB snapshots on the current host.** If VOB database snapshots are enabled on the VOB, disable them with **vob_snapshot_setup rmparam /proj/libpub**.

5. **Back up the VOB storage directory**. Use the procedures described in Chapter 12, *Backing Up and Restoring VOBs*.

6. **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".

7. **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.

   - To the same host on a different disk partition:

     # **cd /vobstore**
     # **tar –cf – libpub.vbs | (cd /vobstore2 ; tar –xBpf –)**

   - To a different host:

     # **cd /vobstore**
     # **tar –cf – libpub.vbs | rsh ccsvr04 'cd /vobstore2 ; tar –xBpf –'**

NOTE: The **-B** option to the **tar** command may not be supported on all architectures. Also, the **rsh** command may have a different name, such as **remsh**, on some platforms. Refer to the *ClearCase and MultiSite Release Notes* for more information, or check 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.

8. **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

9. **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.

10. **Register the VOB at its new location**. When you move a *public* VOB, you must enter the *registry password*. In the ClearCase Administration Console, you can use the VOB Tags node for the tag's regions to create VOB-tags, and you can use the VOB Objects subnode of the ClearCase Registry node to create a VOB object entry. You can also use these commands:

    # cleartool register –vob /net/ccsvr04/vobstore2/libpub.vbs
    # cleartool mktag –vob –public –tag /proj/libpub
    /net/ccsvr04/vobstore2/libpub.vbs
    `Vob tag registry password: <enter password>`

    NOTE: If you are registering a Unified Change Management *Process VOB*, you must supply the **–ucmproject** option to the **register** command.

    If your network has several network regions, see *Ensuring Global Access to the VOB—Special Cases for UNIX* on page 128 for a discussion of adjustments and additional registry entries you may need to make.

11. **Reactivate the VOB**. On all client hosts:

    cleartool umount /proj/libpub *(if not already done)*

    cleartool mount /proj/libpub

12. **Unlock the VOB.**

```
# rlogin ccsvr04
Password: <enter root password>
# cleartool unlock vob:/proj/libpub
Unlocked versioned object base "/proj/libpub".
```

13. **Free any old or unneeded VOB storage**. If you did not reuse the existing VOB storage directory in Step #7, delete it to free the disk space it consumes.

14. (If applicable) **Enable VOB snapshots on the new host.** If you want to enable VOB database snapshots on the new VOB host, do so with **vob_snapshot_setup modparam**, supplying the appropriate parameters.

14.5 Moving a VOB Between UNIX Hosts (Different Architectures)

This section explains how to move a VOB between UNIX hosts with different binary formats for the files that implement the VOB database.

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

If you are moving a VOB replica, see *Special Considerations for Replicated VOBs* on page 218.

Moving a VOB between hosts with different architectures includes converting the binary-format files that implement the *VOB database*. 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 mounted by client hosts at **/proj/libpub**.
- The new location for the VOB storage directory is **/src/vobstore/libpub.vbs** on host **ccsvr04**, whose architecture differs from **sol**'s.

NOTE: All the restrictions on moving a VOB to a host with the same architecture apply to moving a VOB to a host with a different architecture.

To move the VOB, follow these steps:

1. **Determine whether the VOB has any nonlocal storage pools.**

cleartool lspool –long –invob /proj/libpub | egrep '(^pool|link)'

```
pool "cdft"
pool "ddft"
pool "sdft"
pool "s_2"
  pool storage link target pathname "/net/ccsvr04/ccase_pools/s_2"
```

There is one remote pool, **s_2**, and its pathname must be valid on the new VOB host and on all client hosts.

2. **Verify the validity of pathnames to nonlocal pools**. Moving a VOB storage directory does not move any of its remote storage pools. You must make sure that the VOB's new host can access each remote storage pool using the same global pathname as the VOB's current host:

   - If you are moving the VOB to another location on the same host, the pathnames are valid.
   - If you are moving the VOB to a different host, log on to that host and verify that all the remote storage pool global pathnames are valid on that host.

3. **Deactivate the VOB**. Issue this command on each host where the VOB is currently active:

   cleartool umount /proj/libpub

   NOTE: It may not be practical to unmount the VOB from all hosts. In this case, the VOB lock to be applied in Step #6 should be sufficient to prevent unintended VOB access.

4. (If applicable) **Disable VOB snapshots on the current host.** If VOB database snapshots are enabled on the VOB, disable them with **vob_snapshot_setup rmparam /proj/libpub**.

5. **Back up the VOB storage directory**. Use the procedures in Chapter 12, *Backing Up and Restoring VOBs*.

6. **Dump the VOB's database to ASCII dump files.** Do this as the *root* user on the host where the VOB storage directory resides. You don't need to lock the VOB beforehand; the **reformatvob** command does this automatically.

rlogin sol –l root
Password: *<enter root password>*
\# **cleartool reformatvob –dump /vobstore/libpub.vbs**
<warning message>
Reformat versioned object base "/vobstore/libpub.vbs"? [no] **yes**
Dumping database ...
.
.
Dumped versioned object base "/vobstore/libpub.vbs".
Done.
Checking for VOB tag registry entry...
VOB tag registry entry found for versioned object base
"/vobstore/libpub.vbs".

This command marks the VOB database as invalid. The VOB cannot be used for development work until it is processed by a **reformatvob –load** command.

7. **Copy the VOB storage directory**. First, make sure that the parent directory of the target location exists and is writable. Then, copy the entire VOB storage directory tree (but not remote storage pools) to the new location.

   <verify that '/src/vobstore' already exists on remote host 'ccsvr04'>

   \# **cd /vobstore**
   \# **tar –cf – libpub.vbs | rsh ccsvr04 'cd /src/vobstore ; tar –xBpf –'**

   NOTE: The **-B** option to the **tar** commands may not be supported on all architectures. Also, the **rsh** command may have a different name, such as **remsh**, on some platforms. Refer to the *ClearCase and MultiSite Release Notes* for more information or check the reference pages for your operating system.

8. **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**

9. **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.

10. **On the new VOB host, re-create the VOB database from the dump files.** Do this as the *root* user:

```
# rlogin ccsvr04
Password: <enter root password>
# cleartool reformatvob –load /src/vobstore/libpub.vbs
  <warning message>
Reformat versioned object base "/src/vobstore/libpub.vbs"? [no] yes
cleartool: Warning: Renamed old database directory to
 "/src/vobstore/libpub.vbs/db.01.27".
cleartool: Warning: Please remove this database backup
 when you are satisfied with the reformat.
Loading database...
Dumped schema version is nn
 ...
73 pass 2 actions performed.
Done.
Checking for VOB tag registry entry...
cleartool: Warning: VOB tag entry not found for versioned object base
"/src/vobstore/libpub.vbs".
cleartool: Warning: Use the mktag command to create a registry entry.
```

11. **Create a tag for the VOB at its new location.** The **reformatvob** command has already created an entry in the *VOB object* registry; you must create an entry in the *VOB-tag* registry.

    ```
    # cleartool mktag –vob –public –tag /proj/libpub /net/ccsvr04/src/vobstore/libpub.vbs
    Vob tag registry password: <enter password>
    ```

    If your network has several network regions, see *Ensuring Global Access to the VOB—Special Cases for UNIX* on page 128 for a discussion of adjustments and additional registry entries you may need to make.

12. **Reactivate the VOB**. On all client hosts:

    ```
    # cleartool mount /proj/libpub
    ```

13. **Delete the backup VOB database**. This backup was created by **reformatvob –load** in Step #10. It is a date-stamped subdirectory of the VOB storage directory. For example, it may be named **/src/vobstore/libpub.vbs/db.01.27** (01.27 means January 27).

    ```
    # rm –fr /src/vobstore/libpub.vbs/db.01.27
    ```

14. **Free any old or unneeded VOB storage**. If you did not reuse the existing VOB storage directory in Step #7, delete it to free the disk space it consumes.

15. (If applicable) **Enable VOB snapshots on the new host.** If you want to enable VOB database snapshots on the new VOB host, do so with **vob_snapshot_setup modparam**, supplying the appropriate parameters.

15 Removing VOBs

This chapter presents procedures for making a VOB inaccessible to clients temporarily and for removing a VOB altogether.

15.1 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, the UNIX **clearvobadmin** tool, or 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.

15.2 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 **/proj/libpub**. To make the VOB inaccessible:

1. **Have all clients unmount the VOB.** On each client, this command unmounts the VOB:

   cleartool umount /proj/libpub

NOTE: You can also unmount and unregister a VOB from the ClearCase context menu in Windows Explorer or, on UNIX, using the **clearvobadmin** utility

2. **Remove the VOB from the object registry.** Removal prevents anyone from reactivating the VOB. Use the ClearCase Administration Console's VOB Objects node (a subnode of the ClearCase Registry node) to remove the VOB object. You can also use this command:

   cleartool unregister –vob /net/sol/vobstore/libpub.vbs

3. **(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 from each region where you want to temporarily take the VOB out of service. If you do not remove the VOB-tag from the tags registry, attempts to mount the VOB produce the following error message at the client:

   cleartool mount /proj/libpub
   ```
   cleartool: Error: An error occurred mounting.
   Refer to the log file "/var/adm/atria/log/mntrpc_server_log"
   for more information on the warning or failure.
   ```

   If you remove the VOB-tag, the error message from **cleartool mount is** more informative:

   cleartool rmtag –all –vob /proj/libpub
   cleartool mount /proj/libpub
   ```
   cleartool: Error: /proj/libpub is not a registered vob tag.
   ```

   (The **–all** option ensures correctness in a network with multiple regions; the VOB-tag is removed from all the logically distinct tags registries.)

4. **Terminate the VOB's server processes.** If the VOB host 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 the VOB Objects subnode of the Registry node in ClearCase Administration Console to create a VOB object entry. Or use this command:

   cleartool register –vob /net/sol/vobstore/libpub.vbs

NOTE: If you are registering a Unified Change Management *Process VOB*, you must supply the **–ucmproject** option to 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 #3 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 /proj/libpub /net/sol/vobstore/libpub.vbs

   (Repeat, as necessary, for other network regions.)

3. **Have clients reactivate the VOB**. On each client host, this command mounts the VOB:

   cleartool mount /proj/libpub

15.3 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. Unmount the VOB on all hosts where it is active.
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 host where the VOB storage directory resides.

NOTE: If the VOB is replicated, you must follow the procedures for removing replicas as described in the *ClearCase MultiSite Manual*.

16 Using checkvob

This chapter explains how to use the **checkvob** utility to find and fix inconsistencies between a VOB database and its storage pools, to clean up broken cross-VOB hyperlinks, and to find and fix global type problems.

See also the **checkvob** reference page.

16.1 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 170.) We recommend that you run checkvob as part of your normal maintenance routine.

16.2 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.

16.3 Checking Global Types

In global types mode (**–global**), **checkvob** verifies that no VOB has more than one administrative VOB. If any VOB has more than one administrative VOB, **checkvob** lists it and stops. If the administrative VOB hierarchy is valid, **checkvob** lists this information:

- Global types with local copies whose names do not match
- Eclipsing types
- Eclipsing locks on local copies of global types
- Mismatched protections between global types and their local copies

If you specify a VOB selector, **checkvob** starts its search for global types in that VOB and checks all global types it finds in the administrative VOB hierarchy associated with that VOB. If you specify a type selector, **checkvob** checks only the specified type.

You can restrict the checks by specifying one or more of **–acquire**, **–protections**, **–lock**, or **–unlock**. See the **checkvob** reference page for descriptions of these options.

Fix Processing

With the **–fix** option, **checkvob** does the following:

- Changes the name of each local copy to match the name of its global type. **checkvob** queries for confirmation unless you specify **–force**.

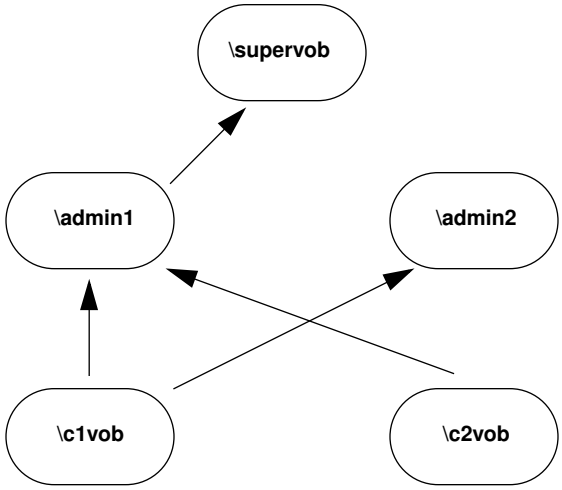
- Attempts to acquire eclipsing local copies and eclipsing ordinary types. **checkvob** queries for confirmation unless you specify **–force**. If a type being acquired is locked, the locks are discarded if there is a lock on the global type. If there is no lock on the global type, the lock information from the first acquired type is applied to the global type.
- Corrects eclipsing locks on local copies. **checkvob** queries for confirmation and whether to unlock or lock the copy unless you specify **–force** and either **–lock** or **–unlock**.
- Changes ownership of each local copy to match the ownership of its global type. **checkvob** queries for confirmation unless you specify **–force**.

Output Log for Global Type Checking

The output from **checkvob** is captured in a log file. By default, the file is named **checkvob.***date***.***time* and written to the current directory.

Example Check or Fix Scenario

In this scenario, the VOB hierarchy looks like this:



To check and fix the global types problems in the hierarchy:

1. Run **checkvob –global** in any VOB in the hierarchy.

   cleartool checkvob –global vob:\supervob

   ```
   The session's log file is "checkvob.03-Aug-99.17:15:15".
   Starting analysis of Admin VOB hierarchy.

   cleartool: Error: There are too many Admin VOBs for vob "\c1vob".

   Analysis of Admin VOB hierarchy complete.
   cleartool: Error: 4 VOBs analyzed, 1 VOBs had problems.
   cleartool: Error: Please fix the listed problems, then rerun this command.
   ```

2. List the **AdminVOB** hyperlinks for **\c1vob**.

   cleartool describe –long vob:\c1vob

   ```
   versioned object base "\c1vob"
   ...
     Hyperlinks:
       AdminVOB@2@\c1vob -> vob:\admin2
       AdminVOB@3@\c1vob -> vob:\admin1
   ```

3. Delete one of the **AdminVOB** hyperlinks. In this example, the hyperlink to **\admin2** is deleted.

   cleartool rmhlink AdminVOB@2@\c1vob

   ```
   Removed hyperlink "AdminVOB@2@\c1vob".
   ```

4. Run **checkvob –global** again to check for problems with the global types in the hierarchy.

cleartool checkvob –global vob:\supervob

```
The session's log file is "checkvob.03-Aug-99.17:23:25".
Starting analysis of Admin VOB hierarchy.

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

Starting "global type" processing.

Detection of eclipsing local copies is: ENABLED
Detection of protection mis-matches is: ENABLED
Detection of eclipsing local locks is: ENABLED
Correction of detected errors is: DISABLED
cleartool: Error: Definition of element type "global_el" in \supervob is
eclipsed by definition(s):
        element type "global_el" in \admin1
        element type "global_el" in \c2vob
cleartool: Error: Definition of branch type "global_br" in \supervob is
eclipsed by definition(s):
        branch type "global_br" in \admin1
        branch type "global_br" in \c2vob
cleartool: Error: Global branch type "global_br" in "\supervob" has local
copies with non-matching names:
        "global_br_mismatch" in "\c1vob"
cleartool: Error: Definition of attribute type "global_at" in \supervob is
eclipsed by definition(s):
        attribute type "global_at" in \admin1
        attribute type "global_at" in \c2vob
cleartool: Error: Definition of hyperlink type "global_hl" in \supervob is
eclipsed by definition(s):
        hyperlink type "global_hl" in \c2vob
cleartool: Error: Global hyperlink type "global_hl" in "\supervob" has
local copies with non-matching names:
        "global_hl_wrong" in "\admin1"
```

```
cleartool: Error: Definition of label type "global_lb" in \supervob is
eclipsed by definition(s):
        label type "global_lb" in \admin1
cleartool: Error: Definition of element type "global_el" in \admin1 is
eclipsed by definition(s):
        element type "global_el" in \c2vob
cleartool: Error: Definition of branch type "global_br" in \admin1 is
eclipsed by definition(s):
        branch type "global_br" in \c2vob
cleartool: Error: Definition of attribute type "global_at" in \admin1 is
eclipsed by definition(s):
        attribute type "global_at" in \c2vob
cleartool: Error: Global label type "global_lb" in "\admin1" has local
copies with non-matching names:
        "nt_global_lb" in "\c2vob"


Completed "global type" processing.
Processed 9 global types in 4 VOBs.
The following 9 global types may still have problems:
        element type "global_el" in "\supervob"
        branch type "global_br" in "\supervob"
        attribute type "global_at" in "\supervob"
        hyperlink type "global_hl" in "\supervob"
        label type "global_lb" in "\supervob"
        element type "global_el" in "\admin1"
        branch type "global_br" in "\admin1"
        attribute type "global_at" in "\admin1"
        label type "global_lb" in "\admin1"
```

5. Review the log file, and then run **checkvob –global –fix** to correct the problems.

cleartool checkvob –global –fix vob:\supervob

```
The session's log file is "checkvob.03-Aug-99.17:31:06".
Starting analysis of Admin VOB hierarchy.

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

Starting "global type" processing.

Detection of eclipsing local copies is: ENABLED
Detection of protection mis-matches is: ENABLED
Detection of eclipsing local locks is: ENABLED
Correction of detected errors is: ENABLED
Global element type "global_el" in "\admin1" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global element type "global_el" in "\admin1" has local
copies with inconsistent protections in VOBs:
        \c2vob
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global branch type "global_br" in "\admin1" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global branch type "global_br" in "\admin1" has local
copies with inconsistent protections in VOBs:
        \c2vob
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global attribute type "global_at" in "\admin1" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global attribute type "global_at" in "\admin1" has local
copies with inconsistent protections in VOBs:
        \c2vob
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
cleartool: Error: Global label type "global_lb" in "\admin1" has local
copies with non-matching names:
        "nt_global_lb" in "\c2vob"
```

```
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global element type "global_el" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
Global branch type "global_br" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global branch type "global_br" in "\supervob" has local
copies with non-matching names:
        "global_br_mismatch" in "\c1vob"
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global attribute type "global_at" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
Global hyperlink type "global_hl" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.
cleartool: Error: Global hyperlink type "global_hl" in "\supervob" has
local copies with non-matching names:
        "global_hl_wrong" in "\admin1"
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
cleartool: Error: Global hyperlink type "global_hl" in "\supervob" has
local copies with inconsistent protections in VOBs:
        \c2vob
Correct this problem?  [no] yes
Attempting to correct this problem ... corrected.
Global label type "global_lb" in "\supervob" is eclipsed by acquirable
types.
Correct this problem?  [no] yes
Attempting to acquire ... acquire completed successfully.

Completed "global type" processing.
Processed 9 global types in 4 VOBs.
```

6. Run **checkvob –global** again to confirm that there are no problems.

   cleartool checkvob -global vob:\supervob

```
The session's log file is "checkvob.03-Aug-99.17:36:49".
Starting analysis of Admin VOB hierarchy.


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

Starting "global type" processing.

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

Completed "global type" processing.
Processed 5 global types in 4 VOBs.
```

16.4 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* and *derived object pools*. (Cleartext pools are caches and are expendable.) The VOB database reflects all additions (checkins, branch creation, element creation, DO promotion, and so on) and deletions (removal of DOs, versions, branches, and elements) in the pools. Each data container is referenced from the database.

In general, inconsistencies between the database and storage pools occur because of damage to the VOB database, 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: **protectvob**, **mkpool**, **rename**, **mkelem**, **rmelem**, **checkin**, **checkout** with auto-make-branch, **mkbranch, rmbranch**, **rmver**, **chtype**, **chpool**, **protect**.

- Any operation that creates or deletes shared DOs can create database or DO pool inconsistencies: **clearmake**, **winkin**, **clearaudit**, **rmdo**, **scrubber**, **protectvob**, **mkpool, rename**, **chpool**, **protect**.

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**)
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 or DOs 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 or DO data is recorded in the database but missing from the storage pools, **checkvob** updates the VOB database, dereferencing this data with the equivalents of **rmver -data** (missing source containers) and **rmdo** (missing DO containers).

Updating the VOB Database

checkvob never updates the VOB database to reference newer pool data. If versions or DOs 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 or DO information to the VOB database, **checkvob** updates the VOB database in the following ways:

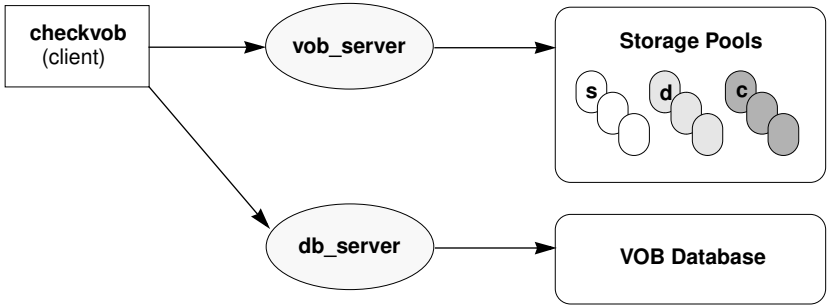
- It removes references to versions and DOs that are confirmed missing by **checkvob** and accepted as missing by the user (interactive **yes** or **–force –fix** option).
- After successfully reconstructing a missing container, in whole or part, **checkvob** adjusts the database to reference the newly 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 VOB host where it executes. As shown in Figure 19, the **vob_server**, which runs as the VOB owner on UNIX and as the **clearcase_albd** user 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 19 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.

Replicated VOB Considerations

checkvob is a per-replica operation. You run it to achieve local pool or database consistency. **checkvob** does not create oplog entries for its updates. (In fact, this is a requirement, because in a VOB replica restoration scenario, **checkvob** must be able to run before **restorereplica**.) To synchronize replicas after running **checkvob**, run the **restorereplica** command.

When fixing a data loss (missing container) problem, **checkvob** does not search other replicas for missing containers or version data. Similarly, after the current replica's database is updated with **rmver –data** to reflect missing version data, you cannot repopulate this database with version data from another replica, other than by hand. If you choose this approach (create new branches and versions, move labels, and so on), version selection based on config records is not affected; the old versions (now with no version data) are still selected.

Data loss (missing containers) at the current replica does not affect synchronization exports or imports. Data loss at the current replica can be propagated only with **mkreplica**. In this case, the new replica inherits the "lost data" state. For example, if data loss occurs on the replica that created the lost version, there are two synchronizing export scenarios:

- Subsequent export packet contains a dataless "create version" operation (as if **rmver –data** followed **checkin**).

  The system appends a comment at the end of the event record for a dataless sync-import "create version" event. The additional text says the a dataless checkin occurred, and it identifies the replica (OID) from which this dataless "create version" originated.

- A full data "create version" operation was already propagated to a sibling replica. The sibling replica retains the data.

Running checkvob

To accept as lost any pool data (versions or DOs) 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 252*).

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 20 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 20 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 21 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 21 checkvob Output Log: Condensed Transcript File

```
==========================================================================
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

==========================================================================
Checking the installed type managers...

==========================================================================
Checking "source pool" pool root storage areas...

==========================================================================
Starting "source pool" Referenced Container Checking at 11-Oct-98.18:16:52.
...
---------- Container problem report:
\vob_src\foo.c@@
    s\sdft\3d\38\0-77857609cb6711cfb178080009935288-2r (missing)
----------
... 1 containers checked; 1 broken (1 missing, 0 misprotected)

==========================================================================
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:
...
==========================================================================
Starting "source pool" Unreferenced Container Checking at 11-Oct-98.18:16:54.
...
Check Processing Summary:
...
==========================================================================
Starting "source pool" Unreferenced Container Fixing at 11-Oct-98.18:16:55.
...
Fix Processing Summary:
...

"source pool" Processing Summary:
...

The VOB's source pools are healthy.

==========================================================================
```

Header
- VOB ID
- Pool ID
- Problem processing
- Output log dir

Check type managers

Check pool roots

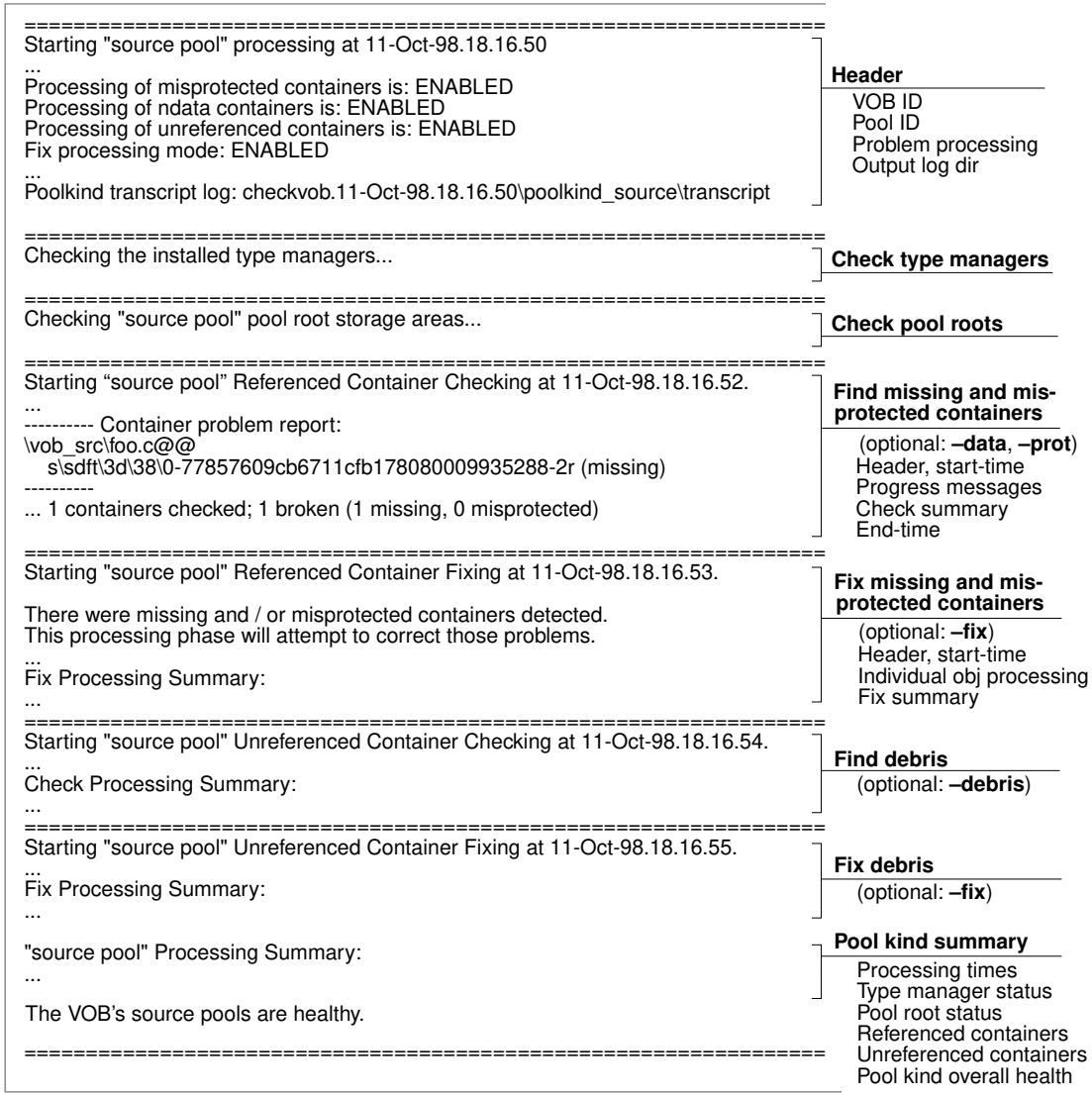
Find missing and misprotected containers
- (optional: **–data**, **–prot**)
- Header, start-time
- Progress messages
- Check summary
- End-time

Fix missing and misprotected containers
- (optional: **–fix**)
- Header, start-time
- Individual obj processing
- Fix summary

Find debris
- (optional: **–debris**)

Fix debris
- (optional: **–fix**)

Pool kind summary
- Processing times
- Type manager status
- Pool root status
- Referenced containers
- Unreferenced containers
- Pool kind overall health



Overview of checkvob Processing

The following sections describe **checkvob** actions.

Individual File Element or DO 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.

DO processing:

1. Check for **–data** and **–protection** problems.

2. If fix mode (**–fix**), fix protection and missing container problems:

   a. Fix missing protection problems.

   b. For each missing container, remove the DO with the equivalent of **rmdo**.

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 or DO 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-99.00:02:02:00**, 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-99.00:02:02:00
  ```

  If **checkvob** cannot find an expected container with data that was created before **6-Oct-99.00:02:02:00**, 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** (and by **mkreplica –import**, for replicated VOBs) when you upgrade a VOB server host to a new ClearCase release. If this part of the **reformatvob** or **mkreplica** 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 **checkvob**-enabling type managers are installed. 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 (source or DO pools) 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, DO, or Cleartext Pool: Bad Pool Roots

Description

Misnamed or misidentified pools**.**

Cause

mkpool, **rmpool**, or **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, DO, 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 or DO Pool: Misprotected Container on Windows NT

Description

FAT file system: incorrect RO attribute.
NTFS file system: incorrect RO attrib 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 prevent 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 (for a source pool) or a missing derived object (for a DO pool). 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 and DO 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 252* 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 get 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. For example:

   - A **chtype binary_delta *.gif** operation (from element type **file**) been lost to the (older) database. **checkvob** uses the newer pool's (unreferenced) delta containers to reconstruct the missing whole copy containers expected by the (older) database.

   - A **chpool** operation is not reflected by older, rolled-back storage pools. **checkvob** may have to find and clone hundreds, or thousands, of unreferenced data containers.

Source Pool: Unreferenced Container (Debris)

Description

Source pool includes a data container that is not referenced by the VOB database. Such containers are tentatively classified as debris, but must pass several tests before being moved to the applicable pool's **lost+found** directory.

Causes

- (Database newer) Database references newer, post-checkin container name (which is *missing*), but pool stores older, unreferenced container.

- (Database newer) Database records **rmver** or **rmbranch** events, but older pool stores container with branches or versions still in place.

- (Pool newer) Pool has container with versions from one or more checkin operations not recorded in the older database.

- (Pool or database reference times differ) A **chpool** operation on a file element in the interval between pool and database reference times leaves the database pointing at the wrong pool. The containers, being at an unexpected location, are unreferenced.
- Restoring a pool from incremental backups.
- Unexpected events.

Fix Processing

checkvob usually moves an unreferenced container to the applicable pool's **lost+found** subdirectory (*vob-storage-dir***s****sdft****lost+found**, by default). It leaves in place unreferenced containers that fit into these categories:

- **May be needed. checkvob** found, but did not fix, a *missing* container problem, and the pathname of the current *unreferenced* container suggests that it may be able to contribute to reconstruction of the missing container on a subsequent **checkvob** run. If you specify [**–force**] **–fix –debris**, without **–data**, **checkvob** performs **–data** check (not fix) processing to identify debris that may be needed.
- **Underage.** The container is less than one hour old. **checkvob** skips underage containers to avoid removing a newly created container before the VOB database has been updated to reference it. Typically, the time between new container creation and database reference update is less than one second, but **checkvob** takes a conservative approach because of the critical nature of source containers.
- **Scheduled for deletion.** The VOB is configured for deferred source container deletion (see **vob_snapshot_setup** and **vob_server**), and the container is already marked for deletion.

NOTE: When the pool is newer than the database and includes more recent versions not recorded in the (older) database, **checkvob** does not salvage versions from the unreferenced containers and update the database. It uses the unreferenced containers to return the pool to the state expected by the database, and it moves the unreferenced containers (with the latest version data) to *pool-dir***lost+found**. These versions should be presumed lost. Contact Rational Technical Support for more information.

Source Pool: Corrupted Container

Description

File truncated and similar conditions

Cause

Unexpected events

Fix Processing

None. **checkvob** does not find or fix corrupted data containers. A container with the right pathname is considered healthy.

DO Pool: Missing Container

Description

VOB database references a DO pool data container that does not exist at the expected location.

Causes

- (Database newer) Database references recently promoted DO, but the older pool does not include its container. See also the information on derived objects in *Building Software with ClearCase*.
- (Pool newer) Older database references a DO that has been scrubbed from the newer pool. See also the **scrubber** reference page.
- Unexpected events.

Fix Processing

With the equivalent of **rmdo**, adjust the database to dereference the DO container.

DO Pool: Unreferenced Container (Debris)

Description

DO pool includes a data container that is not referenced by the VOB database. Such containers are classified as debris and moved to the pool's **lost+found** directory. (The **scrubber** does not remove unreferenced DO data containers, only those with zero reference counts.)

Causes

- (Database newer) Older pool includes DO that was subsequently scrubbed.
- (Pool newer) DO was promoted to the DO pool, but the older database is not aware of it.
- Unexpected events.

Fix Processing

Container moved to pool's **lost+found** directory.

DO Pool: Corrupted Container

Description

Blocks of NUL characters and similar conditions

Causes

- Unexpected events

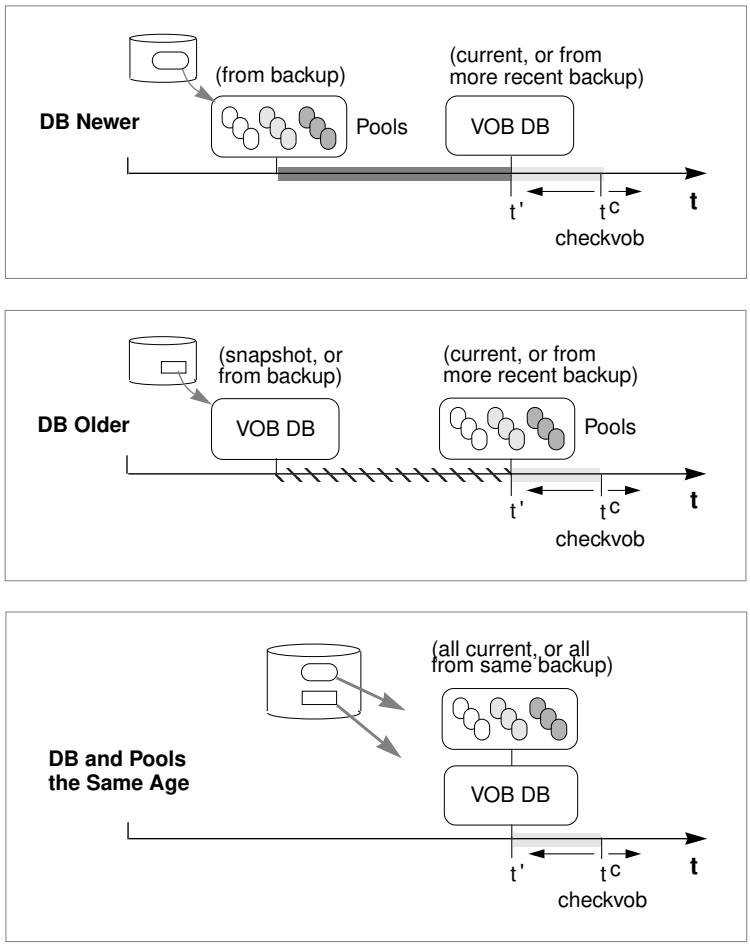
Fix Processing

None. **checkvob** does not find or fix corrupted data containers. A container with the right identify information at the right location is considered healthy for the purposes of **checkvob**.

16.5 Sample Check and Fix Scenarios

The following sample scenarios (see Figure 22) illustrate some general guidelines for using **checkvob**.

Figure 22 Common Scenarios in VOB Database or Storage Pool Synchronization



Somewhere in this time interval is t^c — the point at which you run **checkvob**. If $t^c \neq t'$, VOB activity in the Interval is lost

Pool data created during this time interval is lost. (There may also be "extra" data from rmver or rmbranch operations in this interval.)

Pool data created during this time interval may get copied to the pool's **lost+found** directory, but this data should be presumed lost. checkvob never adds new version or DO data to the DB.

If DB and pools are the same age, data loss is unexpected. Unreferenced containers are common if pools and DB were restored from one or more incremental backups.

Scenario 1: VOB Database Newer Than Storage Pools

A disk crash destroyed remote storage pools, which have been retrieved from backup. The VOB database is current, and the VOB is locked against any further write operations.

The VOB database records development activity that is not substantiated in the storage pools. Under these circumstances, **checkvob** ought to find these anomalies:

- File element or source pool problems: 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.

- DO or DO pool problems: missing data containers. The database references DOs that were promoted to VOB storage containers, but these containers are not found in the (older) pool. **checkvob** deletes these DOs from the database with the equivalent of **rmdo**.

- DO or DO pool problems: unreferenced data containers. DOs were scrubbed from VOB storage, but the (older) pool still includes their containers.

Running checkvob

To accept as lost any pool data (versions or DOs) 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 **vob_snapshot**. Under these circumstances, **checkvob** ought to find these anomalies:

- File element or source pool problems: 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 gap, recovering all versions is not possible (and, presumably, not desirable).

- DO or DO pool problems: missing data containers. The **scrubber** has deleted DOs from the (newer) pool, but the (older) database still references them.

- DO or DO pool problems: unreferenced data containers. DOs have been promoted from view-private to VOB storage, but the (older) database does not know about them.

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.

16.6 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.

16.7 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.

- A promoted DO is reported missing.

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.

16.8 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.
- A scrubbed DO is reported missing.

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.

16.9 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**.

16.10 Pool Root Check Failure

Log: *ccase-home-dir***doc\examples\checkvob_sample_logs\checkvob.pool_roots**

In this case, we created a new source pool, **sdft2**, and restored an older pool set that was missing **sdft2**. As a result, **checkvob** reports `pool roots are not healthy`.

Pool check failure can happen in response to various kinds of pool-related events—especially events that occur in any interval between VOB database and storage pool reference times.

Use the following procedures to diagnose and fix pool root problems.

Fixing Pool Roots: Getting Started

Try to determine the failure scenario. Use **lshistory** to see what has happened recently in the VOB:

- Run **lshistory** on any problem pools (newly created pools or renamed pools, for example).

  cleartool lshistory –l pool:sdft@vob:*vob-tag*

- Run **lshistory** on the VOB itself (to find pools that have been removed, for example).

  cleartool lshistory vob:*vob-tag*

Fixing Pool Roots: The Most Common Problems

The following sections explain how to fix common pool root inconsistencies.

For the following sections, the term *skew* refers to inconsistency in what the VOB database expects and what actually appears in the storage directory.

Pool Skew Caused by Addition of New Pool

Problem: An older storage directory doesn't include the newest pool (which is known to the newer VOB database). To fix the skew:

1. In the VOB storage directory, create the pool root with **mkdir** (or**,** for a remote pool on a UNIX host, using **ln -s**.):
2. Add a **pool_id** file as described in *How to Re-Create a Pool's pool_id* on page 269.

   If the pool is a remote pool on a UNIX host, it is probably intact and needs only a new symbolic link, not a **pool_id** file.

Pool Skew Caused by Pool Deletion

Problem: An older database looks for a pool that has been removed from the more recent storage directory.

To fix the skew, see *Pool Skew Caused by Addition of New Pool* on page 268.

Pool Skew Caused by Renamed Pool

To fix the skew, rename the pool root in the storage directory so that it matches the database. The **pool_id** info is already correct. Only the pool's name has changed, not its identity or OID.

A More Complex Pool Skew Scenario

Problem: You removed pools **sdft1** and **sdft2** and redistributed their containers to new pools **sdft3** and **sdft4**, and then renamed **sdft3** to **sdft2**. As a result:

- Some new pools were created.
- Some pools were deleted.
- A new pool was renamed to reuse the name of a deleted pool.

The database is older than the storage directory. Therefore, from the database's perspective:

- **sdft2** is has the wrong **pool_id** info. (The pool is a different object.)
- **sdft4** is extra.

To fix the storage directory to match what is expected by the database:

- Create a **sdft2** pool that has the correct identity.
- Eliminate **sdft3** without losing its containers.

To fix the skew:

1. Edit the storage directory's **sdft2\pool_id** information to set the pool's OID to that which the database associates with **sdft2**.

2. Merge the extra pools into the existing pools and remove the extra pools. It is critical that the merged-in containers retain the same tree structure. For example, **sdft4\0\1***leaf* becomes **sdft2\0\1***leaf*

   To merge **sdft4** into **sdft2**:

   a. Log on as the privileged user.

   b. Go to the **s/sdft** pool directory:

   c. Make sure that the **pool_id** file for **sdft2** is not overwritten:

   d. Copy the pool directory trees, making sure to preserve container ownership and access control information:

   NOTE: If you fail to preserve protection data, **checkvob** detects and fixes the errors in the moved containers.

   e. Delete the extra pool:

How to Re-Create a Pool's pool_id

The **pool_id** file must have a single line with the following format:

`poolkind=`*poolkind* `pool_oid=`*pool-oid* `replica_uuid=`*replica-uuid* `vob_oid=`*vob-oid*

- *poolkind*. **s**, **d**, or **c** for source, derived, cleartext respectively.
- *pool-oid*. The pool's OID, as determined from a command like this one:

  cleartool describe –fmt '%On\n' pool:sdft2

- *replica-uuid*. The VOB's replica UUID (from the VOB storage directory's **replica_uuid** file or from **lsvob –long** (`VOB replica UUID` field))
- *vob_oid*. The VOB's family OID (from the VOB storage directory's **vob_oid** file or from **lsvob –long** (`VOB family UUID` field))

17 Splitting VOBs with relocate

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, or provide better load balancing across more VOB servers.

Relocating elements is a complex operation, because it must preserve all of the data and metadata associated with each relocated element and also ensure that all clients can access the relocated elements along with their metadata.

This chapter first describes how **relocate** works. Then, it examines the before, during, and after phases of 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.

17.1 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
- The config records associated with checked-in derived objects (DOs)

- 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.

- Deletes nonversioned DOs and config records for DOs contained in relocated directories. (See the **rmdo** reference page.)

- Strands view-private files and DOs contained in relocated directories (**recoverview –sync** *view-tag* recovers these files to *view-stg-dir***\.s\lost+found**).

- 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 view-private files and non-versioned DOs stored in relocated directories.
- 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.

NOTE: Because nonversioned DOs do not move with relocated directories (they are instead removed, as if by **rmdo**), any views that exist to help re-create past releases must rebuild any nonversioned DOs that were stored in relocated directories.

17.2 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.

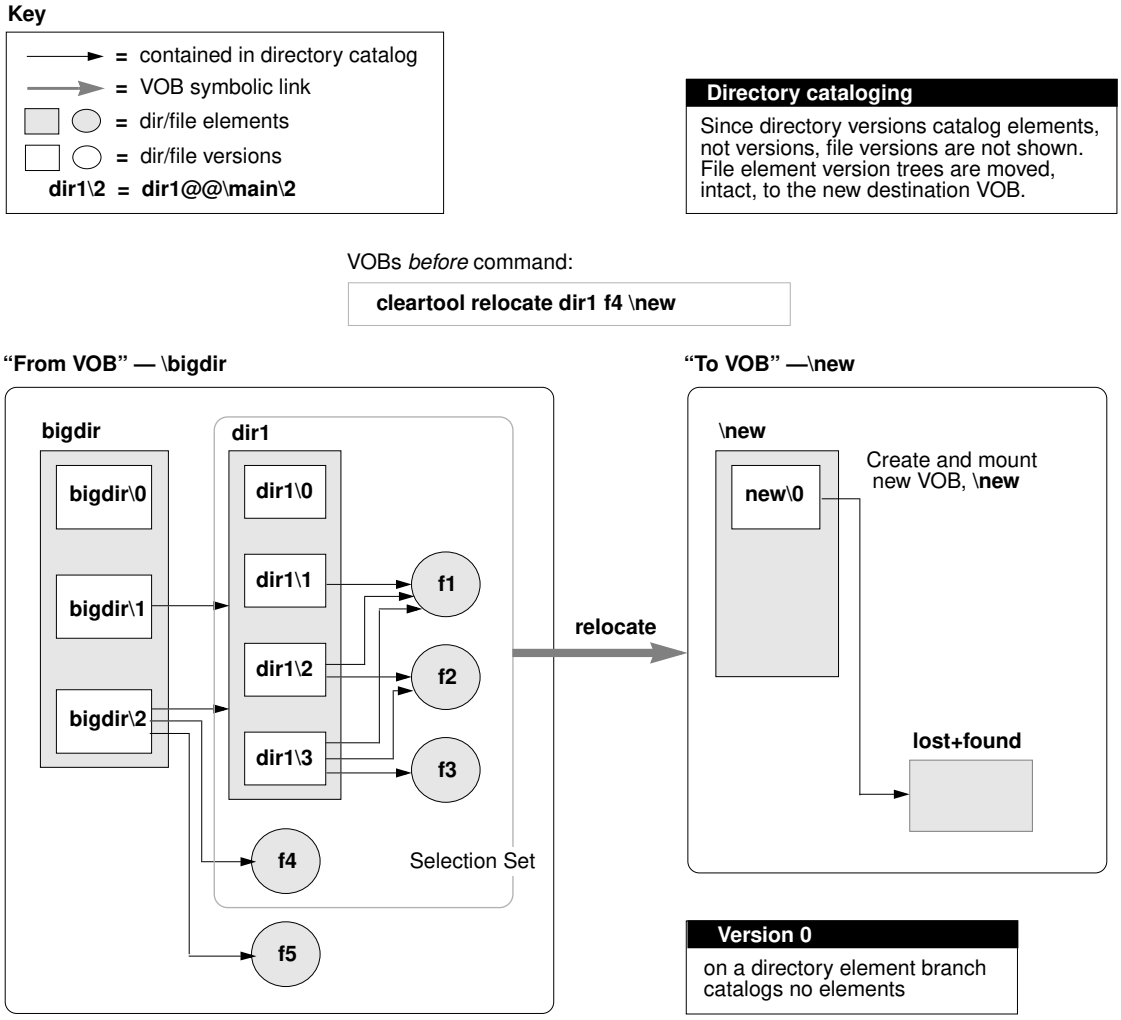
NOTE: All figures use the following shorthand notation for directory versions:

dir1\1 = dir1@@\main\1

Figure 23 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
```

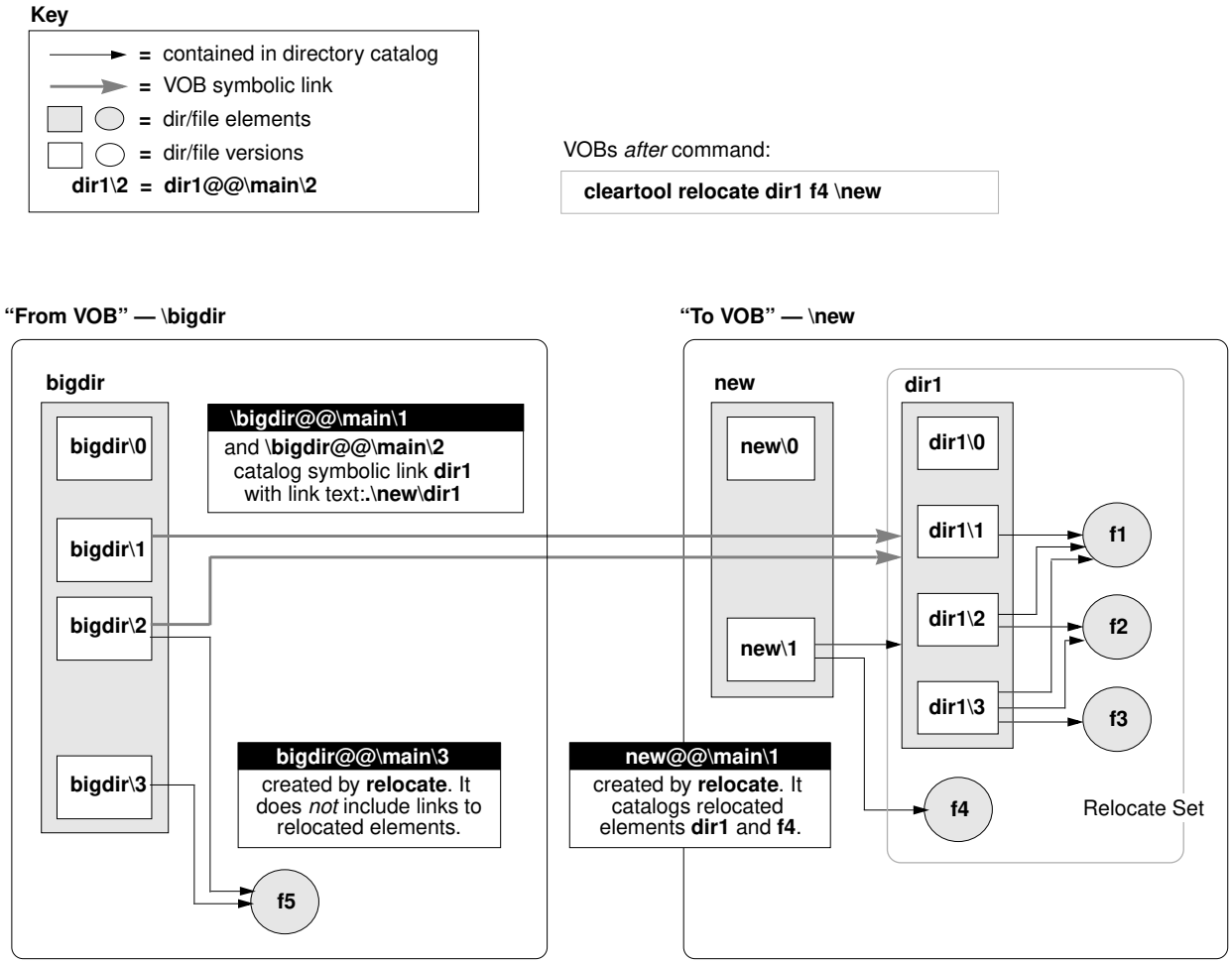
Figure 23 Elements Cataloged by Directory Versions Before Relocate Operation



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

Figure 24 shows the VOBs after **relocate** runs.

Figure 24 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 24) 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.

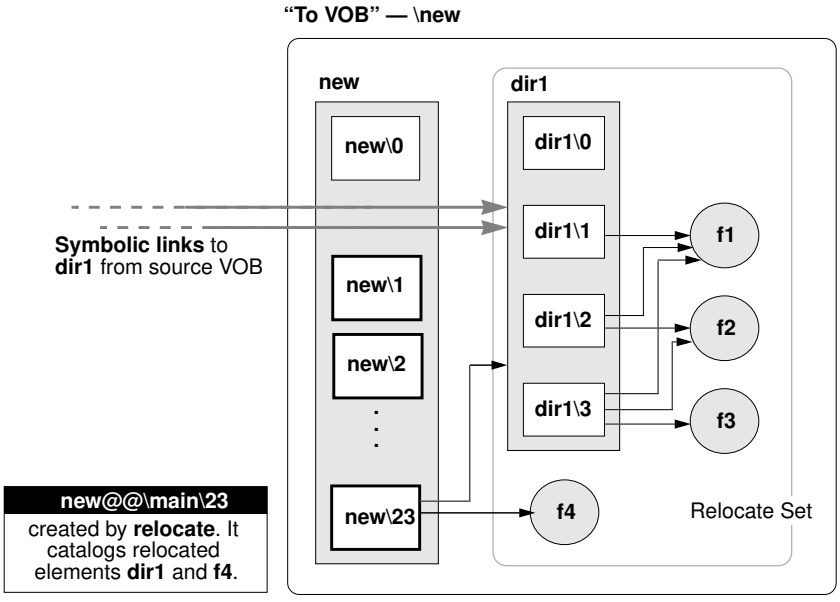
relocate checks in a new version of each relocated element's containing directory (**bigdir@@\main\3** in Figure 24). The new directory version created by **relocate** does not catalog

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 284 for some reasons to avoid relying too heavily on VOB symbolic links. See *Updating Directory Versions Manually* on page 287 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 24 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 25 shows the destination VOB from Figure 24, expanded to include multiple versions of the target directory.

Figure 25 Destination VOB That Includes Multiple Versions

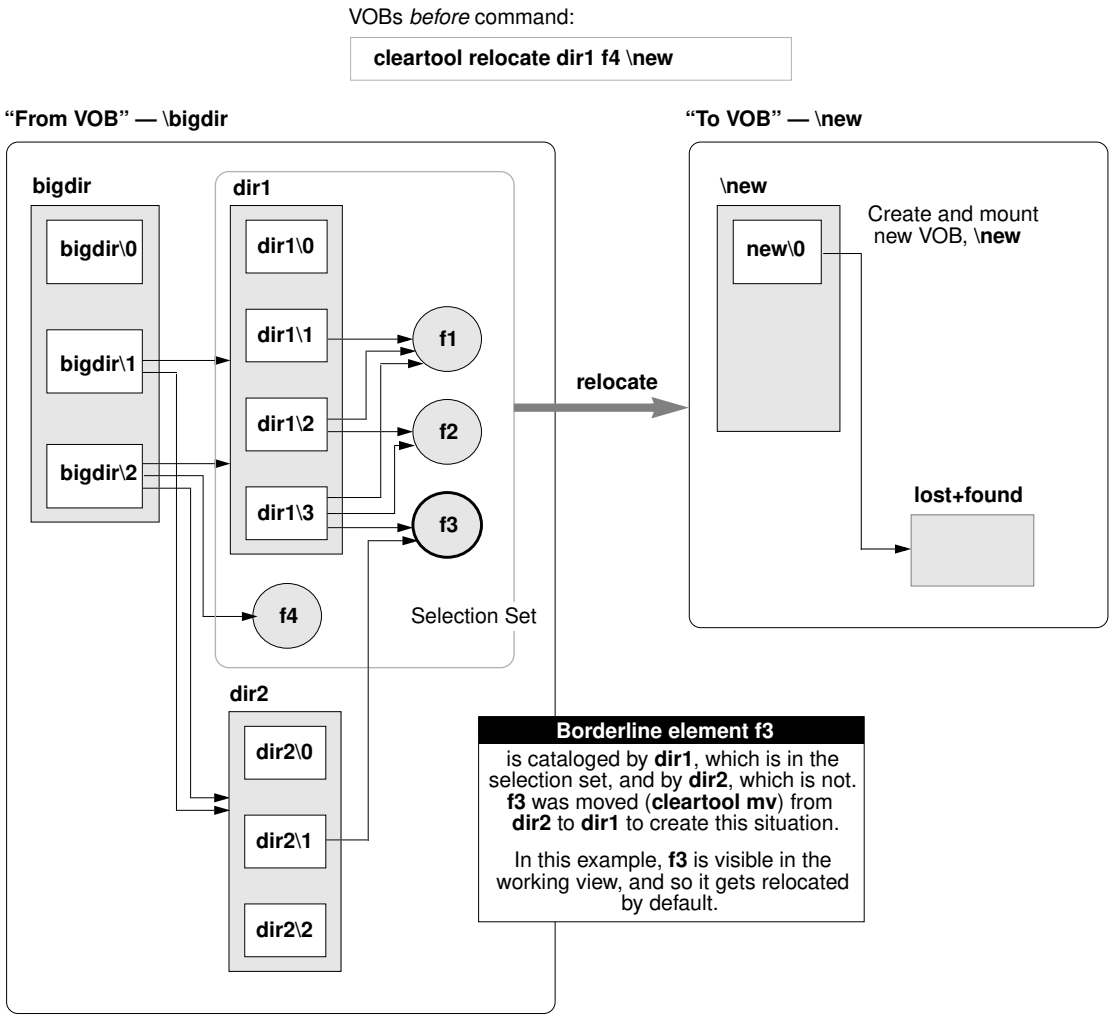


See *Updating Directory Versions Manually* on page 287 for information on adding symbolic links to relocated elements where circumstances warrant it.

Relocating Borderline Elements

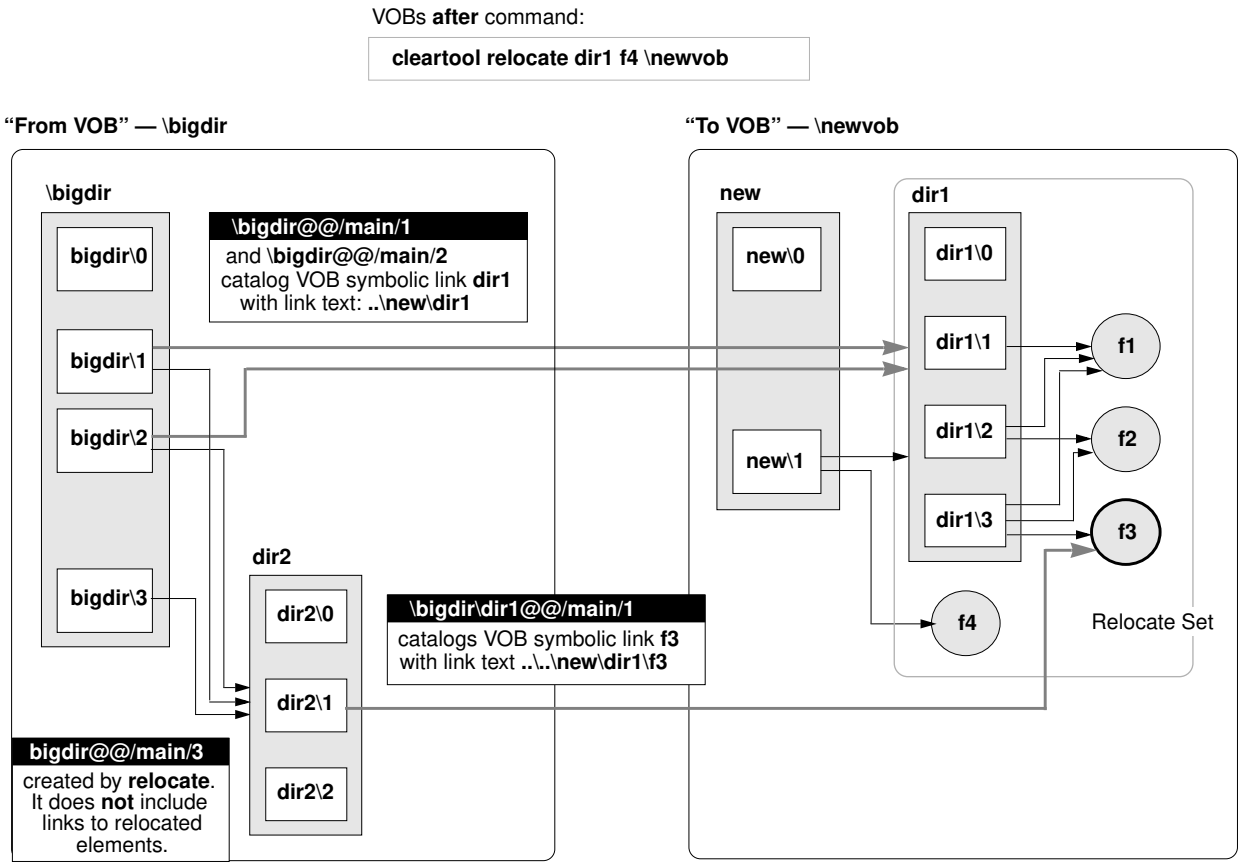
Figure 26 illustrates the relocate operation shown in Figure 23 and Figure 24, but this time the source VOB includes a *borderline element* (**f3**). This element is cataloged both in a version of a directory in the selection set and in some version of a directory outside the selection set (in a directory that stays behind). (See key in Figure 23 on page 274.)

Figure 26 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 27.

Figure 27 Source and Destination VOBs with Borderline Element Relocated

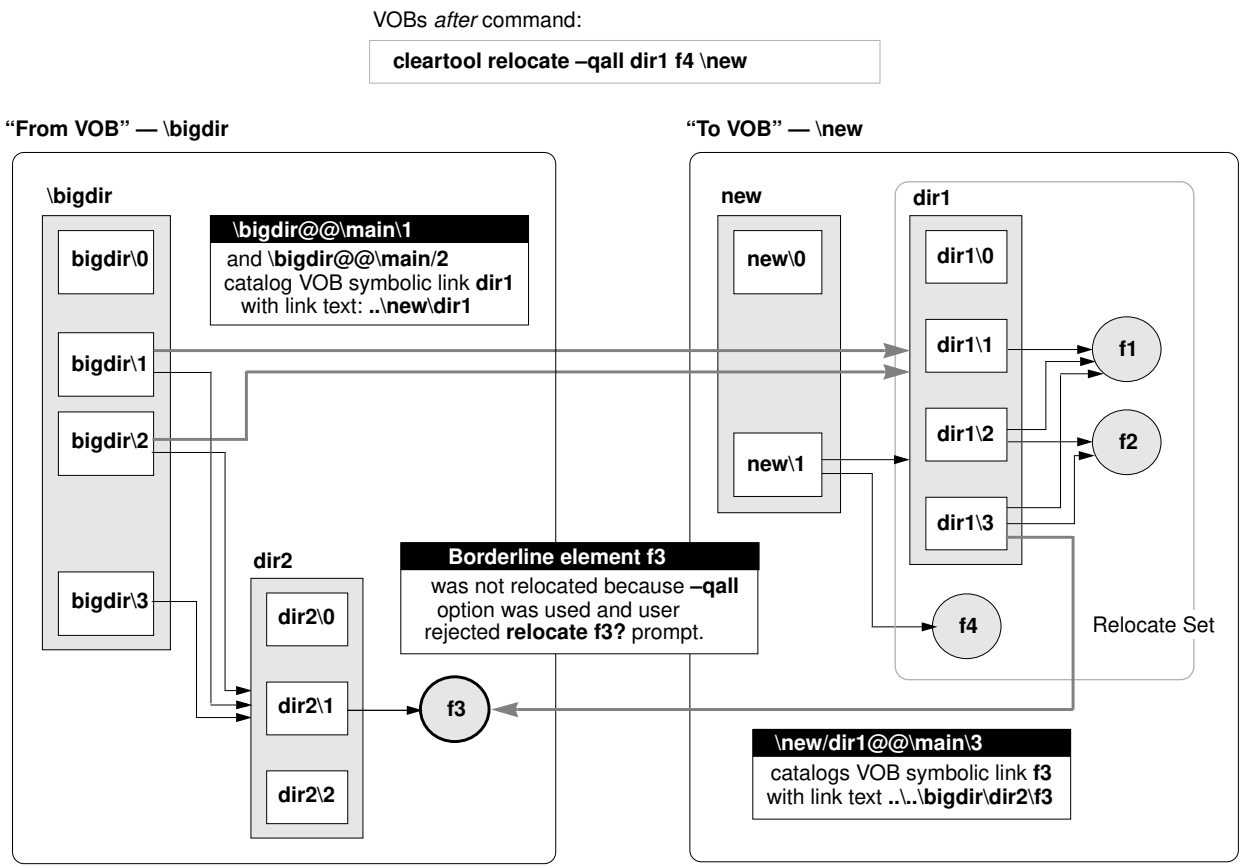


Element **f3** is relocated, and the directory version in the source VOB that referenced it (**dir2@@\main\1**) is updated with a VOB symbolic link to its new location.

Figure 28 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 28 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.

17.3 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.

Inform users that relocated elements will cause rebuilding of DOs that include them as dependencies.

2. **Have users move view-private files out of directories that are to be relocated.** If view-private files are not moved from relocated directories, they become stranded.

   These files can be recovered to *view-stg-dir***.s\lost+found** with **recoverview –sync** *view-tag*. However, it is easier to move them to new locations before running **relocate**.

   Checked-out files are also view-private and must be resolved before running **relocate**. The command aborts if any files are checked out from a directory it is trying to relocate.

3. **Coordinate with administrators of all other VOB replicas, if any**. Affected elements must be inactive at all replica sites. See the section on replicated VOBs in the **relocate** reference page.

4. **Check for views with checkouts or DOs in the VOB.** Use the VOBs node in ClearCase Administration Console. This node has DOs and Referenced Views subnodes that list all views with checkouts and/or DOs in the VOB and allow you to manage these objects from the console window. You can also use the **cleartool lsdo** and **lscheckout** commands to list DOs and checkouts on a per-view basis.

5. **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.

6. **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 283.

7. **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]
```

17.4 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.

17.5 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.

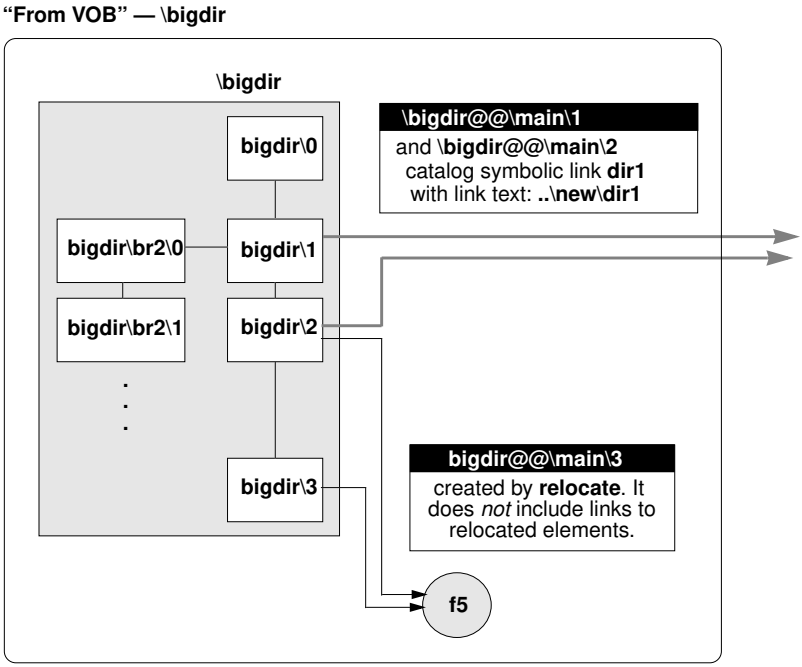
- **Check important and historical views to see whether symbolic links work.** Some kinds of config specs are likely to have trouble:

  - 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 29 shows the source VOB from Figure 24, 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 23 on page 274.)

Figure 29 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 284 to make relocated elements visible again.

- **Recover any stranded view-private files.** Run (or have users run) **recoverview –sync** on each view used to access the source VOB. Any view-private files and DO data files are moved to the view storage directory's **lost+found** directory, where they appear under the expected names (**f2**, **test.c**, and so on). For example:

  cleartool recoverview –sync alh_main

  NOTE: **cleartool lsprivate** lists stranded objects by their object IDs (OIDs).

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. Also, you cannot use these commands in a replicated VOB.

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 25 on page 276, **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**.

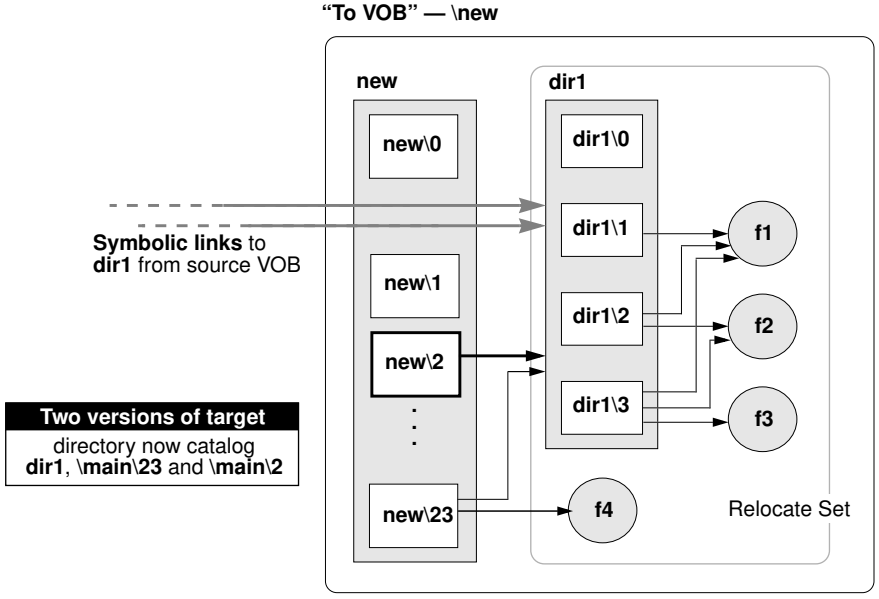
In a case like this, you can manually update previous versions of **relocate**'s destination directory to catalog relocated elements. For example, to add **dir1** to directory version **\new@@\main\2**:

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 30 shows the effect of the **cleartool ln –nco** command. (See the key in Figure 23 on page 274.)

Figure 30 Destination VOB After Modifying Old Version of Destination Directory



Modifying Newest Version of Source Directory to See Relocated Elements

In Figure 24 on page 275 and Figure 27 on page 279, 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** (see Figure 24) 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

18 Understanding View Storage

This chapter describes the contents of ClearCase views. See also the **view** reference page for additional information about views.

18.1 ClearCase Views

A ClearCase development environment can include any number of *views*. A typical view is private to a single user, or perhaps to 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. ClearCase has two kinds of views: *dynamic views* and *snapshot views*.

18.2 Dynamic Views

A dynamic view provides transparent access to versions of elements. Each time you access an element, the **view_server** evaluates the view's *config spec* and selects a version of the element.

Each view implements a *virtual work area*, which presents users with an extended file system that appears to be an ordinary file system hierarchy. The work area is implemented by the ClearCase *multiversion file system (MVFS)*. This work area includes these items:

- Selected versions of elements (actually stored in VOB storage pools)

- Files that are being modified (checked-out file elements, stored in the view's private storage area)
- Directories that are being modified (checked-out directory elements, maintained in the VOB database)
- Derived objects built by users working in this view (stored in the view's private storage area); configuration records that correspond to these derived objects
- Derived objects originally built in another view and then winked in to this dynamic view (stored in VOB storage pools)
- *View-private objects*: miscellaneous files, directories, and links that appear only in this view (stored in the view's private storage area).

Each view is implemented as an ordinary tree whose top-level directory is the *view storage directory*. These are the main components of the view storage directory:

- **View database.** The **db** subdirectory contains the binary files managed by an embedded DBMS. The database tracks the correspondence between VOB objects and view-private objects. For example, a checkout of a file element creates a checked-out-version object in the VOB database and a corresponding data file in the view's data storage area. The view database records the relationship between these two objects.
- **Private storage area.** The **.s** subdirectory is the top level of a directory tree. In a dynamic view, it contains all view-private objects: checked-out versions of file elements, unshared and nonshareable derived objects, text-editor backup files, and so on. Each view-private file, which appears to be located in some directory within some VOB, is actually stored in a *data container* in the view's private storage area.
- **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.
- **Identity directory.** On UNIX hosts, the **.identity** subdirectory contains files that establish the view's owner, its principal group, and its group list.

View Database

Each view has its own database, implemented as a set of files in the **db** subdirectory of the view storage directory. On-disk overhead for the database is quite small, usually less than 1 MB.

A ClearCase server program, the **view_server**, starts when the view is activated. The **view_server** enables ClearCase client programs and other programs to use the view to access both VOB data and view-private data. The **view_server** process runs on the host where the view storage directory resides.

CAUTION: Moving a view requires several steps. You cannot simply move the view database directory (**db**) to another host. See *Moving a View* on page 319.

View's Private Storage Area

A view's *private storage area* is a subtree in the view storage directory. It provides disk storage for view-private files (including checked-out versions of file elements) and for derived objects actually built in that view. **clearmake (**and, on Windows NT, **omake)** also cache configuration records of recently built derived objects in this area, to speed *configuration lookup* (build avoidance).

Typically, nonshareable and unshared derived objects consume the most disk space in a view's private storage area. When a derived object is created, both its data container file and its configuration record are stored in the view. The first time the derived object is *winked in* to another view or promoted to the VOB, the view interacts with the VOB as follows:

- The configuration record is moved to the appropriate VOB database or databases. If the build script creates derived objects in several VOBs, each VOB database gets a copy of the same configuration record.
- The data container is copied (not moved) to a VOB storage pool. If the winkin was done by a **clearmake** or **omake** build, the original data container remains in view storage, to avoid interference with user processes that are currently accessing the data container. If the winkin was done with the **winkin** or **view_scrubber –p** command, the data container in the view is removed after it is promoted to the VOB storage pool.

  From time to time, you (or the view owner) may find it worthwhile to remove the redundant storage containers from views with the **view_scrubber** utility. (See Chapter 21, *Administering View Storage*.)

Remote View Storage on UNIX

On a UNIX host, a view's private storage area can be located remotely from the view storage directory and accessed through a standard UNIX-level symbolic link. This arrangement

resembles the remote VOB storage pool facility, discussed in *Remote Storage Pools on UNIX* on page 115, but the facility for views is less elaborate.

- A VOB can have any number of storage pools, any of which can be remote.
- A view has a single private storage area: the directory tree with *view-storage-dir*/**.s** as its root. By default, the **mkview** command creates a view storage directory with **.s** as an actual subdirectory; the **mkview –ln** command creates a view with **.s** as a symbolic link to another location.

The restriction on remote VOB storage pools also applies to views: a remote private storage area must be NFS-accessible at the same pathname from all ClearCase hosts (for example, **/net/cccvr02/view_storage/drp**).

If a view storage directory threatens to fill up its disk partition, you can move its **.s** directory to a larger partition. See Chapter 21, *Administering View Storage* for details.

18.3 Snapshot Views

A snapshot view contains copies of versions of elements, along with view-private objects. The view has a config spec, but the view never selects versions automatically. To make a version available, you must copy, or *load*, the version into the view. To reevaluate the config spec and possibly copy another version into the view, you *update* the view.

Copies of versions reside in the snapshot view directory, which constitutes the work area for a user of the view. Unlike the virtual work area of a dynamic view, which is implemented by the MVFS, the work area of a snapshot view is a native file-system directory tree. Copies of versions are standard file-system directories and files. In addition to copies of versions, including checked-out versions, this directory can contain view-private file-system objects that are outside ClearCase control. The root directory of the snapshot view also contains some files that ClearCase maintains to identify the directory as part of a view and to do record-keeping.

Like a dynamic view, a snapshot view has a view storage directory. The name of this directory is **.view.stg** on UNIX hosts and **view.stg** on Windows hosts. This directory may or may not be a subdirectory of the snapshot view directory. These are the main components of the view storage directory:

- **View database.** The **db** subdirectory contains the binary files managed by the embedded DBMS. 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.

Unlike a dynamic view, a snapshot view does not use a private storage area in the view storage directory to store view-private files, checked-out versions, and derived objects. All view-private files and copies of checked-out versions reside in the snapshot view directory. Snapshot views cannot contain derived objects. You can run **clearmake** and **omake** in a snapshot view, but no build avoidance takes place.

A snapshot view has an associated **view_server** process, which runs on the host where the view storage directory resides. Some ClearCase hosts can contain snapshot views but cannot run **view_server** processes. Snapshot view directories can reside on these hosts, but the view storage directories for these snapshot views must reside on other hosts that can run **view_server** processes.

For administrative purposes, it is important to know whether the view storage directory of a snapshot view resides within or outside the snapshot view directory. For example, when you back up or move a snapshot view, you must take account of both the snapshot view directory and the view storage directory. For more information, see Chapter 20, *Backing Up and Restoring Views* and Chapter 22, *Moving Views*.

19 Setting Up Views

This chapter discusses setting up views for individual users, views to be shared by groups of users, and views through which ClearCase data is made accessible to non-ClearCase hosts.

19.1 Setting Up an Individual User's View

In a typical ClearCase development environment, most views are created by individual developers on their own workstations for their own use. If a user's workstation has local storage, it makes sense for the user's views to reside in that storage. Alternatively, you can place the storage for some or all views on a file server host. In either case, view storage must be backed up regularly. There may be important view-private files, including checked-out files, in the view that do not exist in VOB storage.

In deciding where to place views, keep in mind these architectural constraints:

- Each view has an associated server process, its **view_server**, which executes on the host where the view's storage directory is created.
- ClearCase must be installed on the host where a view storage directory is created and the **view_server** process runs.
- If a host is to keep several views (and their several **view_server** processes) active concurrently, it must be configured with more main memory.

View Storage Requirements

Each ClearCase view is associated with a *view storage directory*, a directory tree that holds a database, along with a private storage area. In a dynamic view, the private storage area contains view-private files, checked-out versions of elements, and unshared derived objects. In a snapshot view, the private storage area does not contain data; copies of versions and files that are not under ClearCase control reside in the snapshot view directory tree, not in the view storage directory.

View Database

The view database is a set of ordinary files, located in subdirectory **.db** of the view storage directory.

View's Private Storage Area

A view's private storage area is implemented as a directory tree named **.s** in the view storage directory. On UNIX computers, **.s** is an actual subdirectory, so that all data stored in the view occupies a single disk partition.

When deciding where to create view storage for a dynamic view, consider that nonshareable and unshared derived objects typically make the greatest storage demand on the view. To obtain a useful estimate of the maximum disk space required for a view, calculate the total size of all the binaries, libraries, and executables for the largest software system to be built in that view. If several ports (or other variants) of a software system will be built in the same view, it must be able to accommodate the several kinds of binaries.

For a snapshot view, the snapshot view directory must be located on a partition with enough space to accommodate all copies of versions loaded into the view as well as all file-system objects in the view that are not under ClearCase control. The view storage directory may or may not reside within the snapshot view directory. The view storage directory must reside on a ClearCase host that is configured to run **view_server** processes. You may use one or more such ClearCase hosts as central locations for snapshot view storage directories.

19.2 Setting Up a Shared View

Views can be shared by multiple users. For example, a project may designate a shared view in which all of its software components are built in preparation for a release. The entire application may be built each night in such a view.

An ideal location for a shared view is a dedicated host that is configured similarly to a client workstation. If no dedicated host is available, distribute shared views around the network on the least-used (or most richly configured) client workstations. Avoid placing too many views on any single machine; avoid placing shared views on VOB hosts unless you do so for the specific purpose of supporting non-ClearCase access.

Here is a simple procedure for setting up a shared view:

1. **Determine who will use the view**. In particular, determine whether all of the view's prospective users belong to the same group.

2. **(If necessary) Change your group.** If all of the view's prospective users belong to the same group, make sure that you are logged on as a member of that group. Make sure that you have all necessary groups in your group list. Use the UNIX **id** command, or on Windows NT, *ccase-home-dir***etc****utils****creds**, to verify group membership.

3. **On UNIX, set your umask appropriately.** A UNIX view's accessibility is determined by the **umask**(1) of its creator. If the view's users are all members of the same group, temporarily set your umask to allow writing by group members:

   umask 2

   Otherwise, you must set your umask to allow any user write access:

   umask 0

4. **Choose a location for view storage directory.** Use the discussion in *View Storage Requirements* on page 300 to decide where to locate the view storage directory. If a server storage location for views has been created in your registry region, it will be used as the default location for view storage

5. **Choose a view-tag**. Choose a name that indicates the nature of the work to be performed in the view. For example, you may select **integ_r1.3** as the tag for a view to be used to produce release 1.3 of your application.

6. **Create the view storage directory.** Use the View Creation Wizard, or run the **mkview** command:

cleartool mkview –tag integ_r1.3 /net/ccsvr05/viewstore/integr13.vws

```
Created view.
Host-local path:      ccsvr05:/viewstore/integr13.vws
Global path:          /net/ccsvr05/viewstore/integr13.vws
It has the following rights:
User : vobadm   : rwx
Group: dvt      : rwx
Other:          : r-x
```

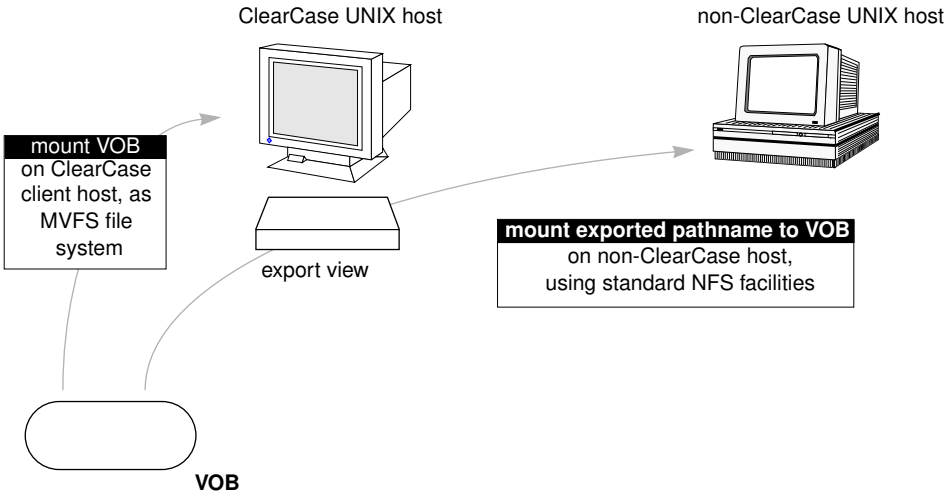
(See *View's Private Storage Area* on page 300 for a command that creates a view with a remote private storage area.)

7. **If you used mkview, verify your work.** Examine the **mkview** command's output to verify that the access permissions are in accordance with your decisions in Step #1–Step #3. In addition, examine the host-local path and global path. You may need to make adjustments similar to those discussed in *Ensuring Global Access to the VOB—Special Cases for UNIX* on page 128.

19.3 Setting Up an Export View for Non-ClearCase Access

A VOB on a UNIX computer can be made available to UNIX hosts on which ClearCase cannot be installed (for example, a machine whose architecture ClearCase does not support). This *non-ClearCase access* feature involves setting up a dynamic *export view*, through which the VOB is seen on the non-ClearCase host (Figure 31).

Figure 31 Export View for Non-ClearCase Access



NOTE: Export views are to be used only for non-ClearCase access to UNIX VOBs. To make a view accessible on a remote host, use the **startview** or **setview** command on that host. An export view can be mounted on a ClearCase host, but not on the viewroot directory **/view**.

The general procedure for exporting a view/VOB combination is as follows:

1. A ClearCase client host that is configured to use the MVFS activates (mounts) the VOB. The VOB must have been marked for export using the **–ncaexported** option to **mkvob** or **mktag**.
2. The host starts an export view, through which the VOB is accessed by non-ClearCase hosts. The view must have been marked for export using the **–ncaexported** option to **mkview** or **mktag**.
3. The host uses a ClearCase-specific exports file to export a view-extended pathname to the *VOB mount point*—for example, **/view/exp_vu/vobs/proj**.

   NOTE: Any symbolic link in this VOB mount point must be a relative symbolic link. Any absolute target of a symbolic link that includes the VOB mount point must be changed to relative for the export to work.
4. One or more non-ClearCase hosts in the network perform an NFS mount of the exported pathname.

See the **exports_ccase** reference page for the procedure specific to your operating system. It describes the simplest (and recommended) setup, in which the VOB and the export view are

located on the same host. The following sections discuss this issue in greater detail, including advice on how to proceed if you don't want to locate the export view on the VOB server host.

NOTE: If you modify an export view's config spec, make sure that all users who may currently have the view mounted for non-ClearCase access unmount and then remount the view. Remounting the view ensures access to the correct set of files as specified in the updated config spec.

Exporting Multiple VOBs

If you choose to put each VOB and its export view on the same host, it is likely that developers working on a non-ClearCase host will access several export views at the same time. For example, a project may involve three VOBs located on three different hosts. Because the three VOB/view pairs are located on different hosts, three export views are involved. On the non-ClearCase host, the NFS mount entries may be these:

```
saturn:/view/beta/vobs/proj /vobs/proj nfs rw,hard 0 0
neptune:/view/exp_vu/vobs/proj_aux /vobs/proj_aux nfs rw,hard 0 0
pluto:/view/archive_vu/vstore/tools /vobs/tools nfs rw,hard 0 0
```

The three VOBs can be accessed on the non-ClearCase host as subdirectories of **/vobs**. But developers must keep in mind that three views are involved, for such operations as checkouts. Developers need not be concerned with multiple-view issues when building software on the non-ClearCase host.

Multihop Export Configurations

Accessing data from a non-ClearCase host can involve three hosts:

- The host on which the VOB storage directory resides
- The host on which the view storage directory of the export view resides
- The non-ClearCase host

This configuration requires MVFS-level communication, which is slower than NFS communication, between the two ClearCase hosts. Creating a multihop configuration introduces the possibility of access cycles, in which two of the hosts depend on each other for network-related services, or such a dependency is created through third-party hosts. Such

situations result in timeouts (if VOBs are soft-mounted) or deadlocks (if VOBs are hard-mounted).

A sure way to avoid access cycles is to avoid multihop configurations.

- Locate the storage directory of the export view on the same host as the storage directory for the VOB.
- Make sure that neither the VOB nor the view has remote data storage. That is, the VOB cannot have any remote storage pools, and the view's private storage area (**.s** directory tree) must be an actual subdirectory, not a symbolic link to another host.

Restricting Exports to Particular Hosts

In a multihop situation, we recommend using an **–access** option in each entry in the ClearCase exports file, **/etc/exports.mvfs**. This option restricts the export to specified non-ClearCase hosts and/or netgroups. This restriction greatly reduces the likelihood of creating access cycles. For example:

```
/view/exp_vu/usr/src/proj –access=galileo:newton:bohr:pcgroup
```

When combining **–access** with other options, be sure to specify options as a comma-separated list that begins with a single hyphen.

NOTE: Be sure to read the reference pages for your operating system's **nfs** options.

20 Backing Up and Restoring Views

This chapter describes procedures for backing up and restoring a view. For information on backing up a VOB, see Chapter 12, *Backing Up and Restoring VOBs*. In particular, when restoring a view, see *VOB and View Resynchronization* on page 200.

Usually, backing up VOBs is more important than backing up views. Views are expendable and often short-lived work areas, not data repositories. Encourage users to check in files regularly so that views do not contain irreplaceable data.

20.1 Backing Up a View

Backing up views is similar to backing up VOBs, but simpler:

- Users can create views under their home directories. In this case, whatever backup regimen you have in place to back up users' home directories picks up their view storage directories as well.
- Views do not have multiple storage pools. A dynamic view has a single private storage area, its **.s** subdirectory. If the view is on a UNIX host, this directory can be located on a remote computer. Otherwise it must be located on the host where the **view_server** runs. A snapshot view's files and directories reside in the snapshot view directory tree.
- It never makes sense to attempt a partial backup of a view storage directory (or, for a snapshot view, a view directory tree); all the data is important, because it's the only copy of one or more users' current work.

If you are backing up a snapshot view, you must back up 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 back up both.

Use the following procedure 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`:

   cleartool lsview –long akp_vu

   ```
   Tag: akp_vu
       Global path: /net/neptune/home/akp/views/akp.vws
    ...
       Region: dvt
    ...
   View on host: neptune
   View server access path: /home/akp/views/akp.vws
    ...
   ```

2. **Ensure integrity and 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
   ```

   This command does not prevent changes to the view's config spec. To keep the config spec from being changed during backup, rename the view storage directory before backing it up, and then stop the **view_server** with **cleartool endview –server** *view-tag*.

3. **On Windows NT, stop ClearCase on the view host.** Common Windows NT backup utilities do not back up files that are open for write. Because the view database files are typically held open for write while ClearCase is running, your backup operation skips these files, unless you stop ClearCase before performing the backup. Unless your backup software is known to capture files open for write access, *you must stop ClearCase on the view host before performing a backup.* Use the ClearCase program in Control Panel to stop ClearCase on the view host. If you are backing up a snapshot view, ensure that no users are working with files in the view's directory tree.

4. **On UNIX, determine whether the view has remote storage.** You can use a standard **ls(1)** command:

cd /home/akp/views/akp.vws
ls –ld .s

```
...  .s -> /net/ccsvr04/viewstore/akp.stg
```

A symbolic link indicates remote private storage area.

5. **Enter the backup commands.** For a snapshot view, back up the entire view directory tree. For a dynamic view or for a snapshot view whose view storage directory is outside the view's directory tree, back up the entire view storage directory tree. Use the local address or the networkwide address listed by **lsview** in Step #1.

   If you are backing up a snapshot view and its 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.

   NOTE: When you back up and later restore a snapshot view, it is important that the utility you use for the backup maintains the original modification times and ownership of all files and directories in the view. If it does not, loaded files become hijacked.

   If the view's private storage area is remote, you need to perform a second backup (typically, using a different backup tape):

6. If you made the view read-only in Step #2, make it writable.

   cleartool chview –readwrite akp_vu

   ```
   Properties: readwrite
   ```

7. If you renamed the view storage directory in Step #2, rename it back to its registered location.

20.2 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 307.

If you are restoring a snapshot 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: You can restore a view to a new location, on the same host or on another host. If you are restoring the view storage directory to a new location, you must re-register the view at its new location (Step #7). If you are restoring a snapshot view's directory tree to a new location, but the

location of the view storage directory (outside the view directory tree) does not change you do not need to re-register the view.

1. **Log on to the view host.** Log on to the host where the view storage directory (or, for a snapshot view, the directory tree) resides.

2. **Check disk space availability**. Make sure that there is enough free space in the view's disk partition to load the backup copy. If necessary, delete the view storage directory (or, for a snapshot view, the directory tree), or use other means to make enough space available.

3. **Stop the view server.** Use **endview –server** to stop the **view_server** process:

   cleartool endview –server akp_vu

4. **Move the original view aside.** For a snapshot view, rename or delete the view directory tree. For a dynamic view or for a snapshot view whose view storage directory is outside the view's directory tree, rename or delete the view storage directory:

   If you are restoring a snapshot view and its view storage directory is outside the view's directory tree, rename or delete both the view directory tree and the view storage directory.

5. **Make sure that the restored view will have the correct ownership.** Depending on your platform and your backup tool, you may have to observe one or more precautions now to ensure that the view is restored with the correct owner and group identities.

   On UNIX platforms, each view 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 view backup:

   - If you used **tar(1)** to back up the view, use the **–p** option when restoring the view. 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 view, no special options are required in the **cpio** command that restores the backup data.

   On Windows NT, if your backup tool does not maintain ACLs correctly, you may need to fix them after the restore with **fix_prot**. See Chapter 36, *Repairing VOB and View Storage Directory ACLs on Windows NT*, for details.

6. **Restore the backup.** For a snapshot view, restore the view directory tree from the backup medium. For a dynamic view or for a snapshot view whose view storage directory is outside the view's directory tree, restore the view storage directory from the backup medium.

If you are restoring a snapshot view and its view storage directory is outside the view's directory tree, be sure to restore the backups of both the view directory tree and the view storage directory.

NOTE: When you restore a snapshot view, the utility you use to restore 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.

7. **Re-register the view.** This is necessary only if you restored the view storage directory to a new location. In the ClearCase Administration Console, you can use the View Tags node for the tag's regions to change the properties of view-tags, and you can use the View Objects subnode of the ClearCase Registry node to change the properties of a view object entry. You can also use **cleartool** commands. For example, if you restored the view storage directory to new location **/usr2/akp.vws**:

   cleartool unregister –view /usr2/akp.vws
   cleartool register –view /usr2/akp.vws
   cleartool mktag –view –replace –tag akp_vu /usr2/akp.vws

8. **Run cleartool recoverview.** Use the following command to update the view database:

   cleartool recoverview –tag akp_vu

The view is now ready for use.

Administering View Storage

21

This chapter discusses procedures for managing disk storage used by views.

21.1 View Storage Maintenance

Like any other isolated work area, a view's private storage area tends to accumulate some unneeded files: temporary files, text-editor backup files, excerpts from mail messages and source files, and so on. Encourage users to clean up their private views periodically; for shared views, the cleanup task may fall to you. See *Cleaning Up a View Manually* on page 315.

Users can remove derived objects from their views using standard tools (**rm** on UNIX, **del** on Windows, or the `make clean` targets in *makefiles*.

The ClearCase Administration Console displays information on disk space used in views:

- The Derived Objects subnode of a VOB storage node shows disk space used by shared derived objects in the VOB. The display shows which views have references to these DOs.
- The view storage node for a view (a subnode of the host node for the host where the view storage directory resides) shows current and historical disk space used for the view.
- The Private Files subnode of the view storage node for a dynamic view lists view-private objects, including files and derived objects, in the view.

Several **cleartool** subcommands also display information on disk space used in views:

- The **dospace** command shows disk space used by shared derived objects in a VOB. The display shows which views have references to these DOs.
- The **space –view** command shows current and historical disk space used for a view.

The ClearCase scheduler runs several jobs that gather data on view disk space use:

- Daily data gathering on view disk space use
- Weekly data gathering on disk space used by shared derived objects
- Daily and weekly execution of jobs that you can customize to run your own programs

For more information on the ClearCase scheduler, see Chapter 28, *Managing Scheduled Jobs*.

From an administrator's standpoint, limiting the growth of a dynamic view storage directory typically involves one issue: removing redundant derived object (DO) data containers. When a DO is first built by **clearmake**, **omake**, or **clearaudit**, its data container is placed in the private storage area of the user's view. The first time a DO is *winked in* during a **clearmake** or **omake** build, the data container is copied to a VOB's derived object storage pool. (Moving it may disrupt user processes that are currently accessing the DO.) This leaves a redundant copy of the data container in view-private storage. (When you wink in a derived object with the **winkin** or **view_scrubber –p** command, the data container in the view is removed after it is promoted to the VOB storage pool.)

Typically, you need not do anything about these redundant copies:

- In a view that is frequently used for builds, old (and potentially redundant) DO data containers are replaced by newer ones by the execution of build scripts.
- There can be at most one redundant copy of each DO in a view. (Contrast this with the situation for VOBs: if the **scrubber** utility never runs, the VOB accumulates many DOs that are no longer used.)

Unless disk storage is extremely scarce, you may conclude that it is not worth the effort to clean up redundant data containers in view-private storage. Accordingly, ClearCase does not include any automated procedures for removing them.

Scrubbing View-Private Storage

If you decide that redundant DO data containers must be removed from a view's private storage area, use the **view_scrubber** utility. You can also use this utility to migrate the data containers of unshared or nonshareable DOs to VOB storage.

The following example shows how some DOs can be built and then transferred immediately to VOB storage:

clearmake hello
<build messages>
*ccase-home-dir***bin\view_scrubber –p hello hello.o util.o**

```
Promoted derived object "hello"
Scrubbed view-resident data container for "hello"
Promoted derived object "hello.o"
Scrubbed view-resident data container for "hello.o"
Promoted derived object "util.o"
Scrubbed view-resident data container for "util.o"
```

See the **view_scrubber** reference page for more information.

21.2 Cleaning Up a View Manually

Use the following procedure to remove unwanted files from a view's private storage area. (You may want to adapt this procedure to your own organization's needs, and then present it to all ClearCase users.)

This procedure applies only to dynamic views. For a snapshot view, you can use the **cleartool ls –recurse –view_only** command to inspect files in the view.

Suppose the view's view-tag is **r2integ**.

1. **Establish the view context.** On UNIX, use the setview command

   cleartool setview r2integ

   On Windows NT, use the **net use** command:

   net use * \\view\r2integ

   In Windows Explorer, use **Tools**>**Map Network Drive**.

2. **Take inventory of the view's private files with lsprivate**. The Private Files subnode of the view storage node in the ClearCase Administration Console or the **lsprivate** command lists view-private files using the pathnames at which they appear in VOBs.

```
cleartool lsprivate >C:\tmp\r2integ.lsp
type C:\tmp\r2integ.lsp
\proj\lib\pick.o
\proj\lib\spar.o
\proj\lib\get.c  [checkedout]
\proj\lib\get.c~
\proj\lib\querytty.c  [checkedout]
\proj\lib\querytty.c~
\proj\lib\strut.c  [checkedout]
  .
  .
  .
```

Be sure to place the output in a scratch inventory file, as in this example. Don't worry if some `not available - deleted perhaps?` error messages appear. Such messages are also captured in the scratch file.

3. **Extract the names of unneeded files**. Use a text editor or any text filtering tool to extract from the scratch file the names of files that can safely be deleted. Write this list to another file—for example, **c:\tmp\r2integ.deleteme**. Exclude from the this list any checked-out files. Such files are annotated with `[checkedout]` in the **lsprivate** output, as shown in Step #2.

4. **Double-check the list**. Make sure it contains only files to be deleted.

5. **Delete the view-private files**.

   NOTE: The following steps are appropriate only if `not available - deleted perhaps?` error messages appeared in Step #2.

6. **Decide which stranded files to delete**. The error messages, and corresponding lines with `VOB-` or `DIR-` in the inventory file, describe *stranded view-private files*. Such files belong to VOBs or directories that are not currently accessible—and, in some cases, may never become accessible again. Consult the **lsprivate** reference page to learn more about stranded files, and to decide which files to delete. In general, you don't select individual files, but entire directories or entire VOBs, all of whose view-privates files are to be deleted.

7. **Collect the appropriate UUIDs**. Determine the UUID of each VOB directory and each VOB whose files are to be deleted. For example, the following lines from **lsprivate** output describes a stranded file named **hello.c.ann**:

   ```
   <VOB-beeb313c.0e8e11cd.ad8e.08:00:69:06:af:65>^
        <DIR-375b5ca0.0e9511cd.ae20.08:00:69:06:af:65>\hello.c.ann
   ```

   (The line is split here so that you can read it easily; it is not split in **lsprivate**'s output.) The VOB from which the file is stranded has this UUID:

```
beeb313c.0e8e11cd.ad8e.08:00:69:06:af:65
```

The VOB directory in which the stranded file was created has this UUID:

```
375b5ca0.0e9511cd.ae20.08:00:69:06:af:65.
```

8. **Move stranded files to the view's lost+found directory**. To remove a set of stranded files, first transfer them to the view's **lost+found** directory, using the **recoverview** command. For example, this command transfers all stranded view-private files created in the same directory as **hello.c.ann:**

   cleartool recoverview –dir 375b5ca0.0e9511cd.ae20.08:00:69:06:af:65 –tag r2integ
   ```
   Moved file ccsvr03:\vus\integ\.s\lost+found\57FBB6DF.0418.util.c.ann
   Moved file ccsvr03:\vus\integ\.s\lost+found\2203B56D.00C2.hello.c.ann
   ```

   In this example, **recoverview** transfers two files, **util.c.ann** and **hello.c.ann**, to the **lost+found** directory.

9. **Delete the files from the lost+found directory**. You can now use a standard file removal command to delete the stranded files:

   cd \vus\integ\.s\lost+found
   ```
   c:\vus\integ\.s\lost+found> del 57FBB6DF.0418.util.c.ann
   c:\vus\integ\.s\lost+found> del 2203B56D.00C2.hello.c.ann
   ```

22 Moving Views

This chapter presents procedures for moving views.

22.1 Moving a View

WARNING: When moving a VOB or view storage directory, make sure that the copy or backup software you use does not change file and directory ownership and access control information.

This section presents a procedure for moving a view to another location, either on the same host or on another host with the same architecture. For a snapshot view, you can move the entire view; if the view storage directory is outside the view directory tree, you can move either the view directory tree, the view storage directory, or both. (To move a view to a host of a different architecture, see *Moving a View to a UNIX Host with a Different Architecture* on page 324.) For clarity, this section uses an example:

- The current location of the view storage directory to be moved is **/users/sue/viewstore/sue.vws**, on a host named **earth**.
- The new location for the view storage directory is **/public/sue.vws**. We consider these cases:
  - The new location is also on **earth**.
  - The new location is on another host, named **ccsvr04**.

Moving a View on UNIX

To move the view:

1. **Go to the view's host**. Log on to the view host, **earth**, as the view's owner:

   rlogin earth –l sue

2. **Determine whether the view has a nonlocal private storage area.**

   ls –ld /users/sue/viewstore/sue.vws/.s

   ```
   ...    .s -> /public/view_aux/sue
   ```

   The symbolic link indicates that the private storage area is remote.

3. **Deactivate the view.** Use **cleartool endview –server** to deactivate the view.

4. **(If necessary) Validate the private storage area's global pathname**. This step is required only if the view's private storage area is remote, and you are moving the view to another host. You must verify that the view's new host can access the private storage area using the same global pathname as the view's current host.

   rlogin ccsvr04
   ls /public/view_aux/sue
   .
   . *(this command should succeed)*
   .
   exit

   If the intended destination host cannot access the view's private storage area in this way, select and validate another host.

5. **Back up the view storage directory**. Use the procedure in *Backing Up a View* on page 307.

6. **Copy the view**. First, make sure that the desired parent directory of the target location exists and is writable. Then, if you are moving a snapshot view directory tree, copy the entire directory tree to the new location. If you are moving a dynamic view, or if you are moving the view storage directory for a snapshot view and the view storage directory is outside the view directory tree, copy the entire view storage directory tree (but not a remote private storage area) to the new location.

   - To the same host (dynamic view or snapshot view storage directory outside the view's directory tree):

<verify that /public already exists>

cd /users/sue/viewstore
tar –cf – sue.vws | (cd /public ; tar –xBpf –)

➢ To a different host (dynamic view or snapshot view storage directory outside the view's directory tree):

<verify that '/public' already exists on remote host 'ccsvr04'>

cd /users/sue/viewstore
tar –cf – sue.vws | rsh ccsvr04 'cd /public; 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 *ClearCase and MultiSite Release Notes* for more information, or check the reference pages for your operating system.

If you are moving both the view directory tree and the view storage directory for a snapshot view whose view storage directory is outside the view's directory tree, copy both the view directory tree and the view storage directory to the new locations.

NOTE: If you are moving a snapshot view, it is important that the utility you use to copy the view maintain the original modification times and ownership of all files and directories in the view. Otherwise, loaded files become hijacked.

7. **Ensure that the old view cannot be reactivated**. Remove it from the ClearCase storage registries.

   cleartool rmtag –view –all sue
   cleartool unregister –view /users/sue/viewstore/sue.vws

   This step is not necessary for a snapshot view if the view storage directory has not moved.

   This prevents reactivation by client hosts.

8. **Register the view at its new location** (without starting the view_server).

   cleartool register –view /public/sue.vws
   cleartool mktag –nstart –view –tag sue /public/sue.vws

   This step is not necessary for a snapshot view if the view storage directory has not moved.

If your network has several network regions, you need to make additional registry entries. This procedure is essentially similar to the one in *Ensuring Global Access to the VOB—Special Cases for UNIX* on page 128.

9. **Reactivate the view.** For a dynamic view, start the view:

   cleartool startview sue

   For a snapshot view:

   - If the new snapshot view directory is on a local drive, access any file in the view.
   - If the new snapshot view directory is on another computer, right-click the root directory of the view in Windows Explorer.

10. **Update corresponding VOB databases.** The view's old location is still recorded in the databases of all VOBs that the view accessed with **checkout** and/or **clearmake**. For each such VOB, update the view-location information by checking out one of the VOB's elements in that view. (You can cancel the checkout (**uncheckout**) immediately, if you want.)

11. **Delete the old view storage directory.** If you did not overwrite the existing view storage directory, delete the old one. Be sure to first verify that the view can be accessed at its new location.

    Moving a view does not modify the **.view** file in the view storage directory. The information in this file always describes the view's first incarnation.

Moving a View on Windows NT

To move the view:

1. **Go to the view's host**. Log on to the view host, **earth**, as the view's owner:.
2. **Back up the view storage directory**. Use the procedure in *Backing Up a View* on page 307.
3. **Verify that the view server on the view's host is not running.** If necessary, stop the **view_server** with **cleartool endview –server** *view-tag*.
4. **Copy the view**. First, make sure that the desired parent directory of the target location exists and is writable. Then, if you are moving a snapshot view directory tree, copy the entire directory tree to the new location. If you are moving a dynamic view, or if you are moving

the view storage directory for a snapshot view and the view storage directory is outside the view directory tree, copy the entire view storage directory tree to the new location.

- To the same host (dynamic view or snapshot view storage directory outside the view's directory tree):

  <verify that 'c:\public' already exists>

  ```
  c:\> cd \users\sue\vwstore
  c:\users\sue\vwstore> ccase-home-dir\etc\utils\ccopy sue.vws \public\sue.vws
  ```

- To a different host (dynamic view or snapshot view storage directory outside the view's directory tree):

  <verify that 'c:\public' already exists on remote host 'ccsvr04'; for simplicity, you might share that directory and, on earth, assign a drive (here, w:) to it>

  ```
  c:\> cd \users\sue\vwstore
  c:\users\sue\vwstore> net use w: \\earth\public
  c:\users\sue\vwstore> ccase-home-dir\etc\utils\ccopy sue.vws w:\sue.vws
  ```

If you are moving both the view directory tree and the view storage directory for a snapshot view whose view storage directory is outside the view's directory tree, copy both the view directory tree and the view storage directory to the new locations.

NOTE: If you are moving a snapshot view, it is important that the utility you use to copy the view maintain the original modification times and ownership of all files and directories in the view. Otherwise, loaded files become hijacked. We recommend that you use **scopy** (a Windows NT Resource Kit command) rather than **ccopy** to move a snapshot view.

5. **Ensure that the old view cannot be reactivated**. Remove it from the ClearCase storage registries. In the ClearCase Administration Console, you can use the View Tags node for the tag's regions to remove view-tags, and you can use the View Objects subnode of the ClearCase Registry node to remove a view object entry. You can also use the following commands:

   ```
   cleartool rmtag –view –all sue
   cleartool unregister –view \\earth\users\sue\vwstore\sue.vws
   ```

   This step is not necessary for a snapshot view if the view storage directory has not moved.

6. **Register the view at its new location (without starting the view_server).** In the ClearCase Administration Console, you can use the View Tags node for the tag's regions to create view-tags, and you can use the View Objects subnode of the ClearCase Registry node to create a view object entry. You can also use the following commands:

cleartool register –view \\earth\public\sue.vws
cleartool mktag –nstart –view –tag sue \\earth\public\sue.vws

This step is not necessary for a snapshot view if the view storage directory has not moved.

If your network has several network regions, you need to make additional registry entries. This procedure is essentially similar to the one in *Ensuring Global Access to the VOB—Special Cases for UNIX* on page 128.

7. **Reactivate the view.** For a dynamic view, run the **net use** command or, in Windows Explorer, use **Tools**>**Map Network Drive.**

   For a snapshot view:

   - If the new snapshot view directory is on a local drive, access any file in the view.
   - If the new snapshot view directory is on another computer, right-click the root directory of the view in Windows Explorer.

8. **Update corresponding VOB databases.** The view's old location is still recorded in the databases of all VOBs that the view accessed with **checkout** and/or **clearmake**. For each such VOB, update the view-location information by checking out one of the VOB's elements in that view. (You can cancel the checkout (**uncheckout**) immediately, if you want.)

9. **Delete the old view storage directory.** If you did not overwrite the existing view storage directory, delete the old one. Be sure to first verify that the view can be accessed at its new location.

   Moving a view does not modify the **.view** file in the view storage directory. The information in this file always describes the view's first incarnation.

22.2 Moving a View to a UNIX Host with a Different Architecture

This section documents the procedure for moving a view between UNIX hosts with different binary formats for the files that implement the view database. For the procedure to move a view on the same host or between hosts with the same format for view database files, see *Moving a View* on page 319.

WARNING: When moving a VOB or view storage directory, make sure your copy/backup software preserves ownership and access control information unchanged. For example, the **cp** command does not preserve file attributes (permissions, modify times).

Moving a view to a UNIX host with a different architecture includes converting the binary-format files that implement the *view database*. For clarity, we use an example:

- The current location of the view storage directory to be moved is **/users/sue/viewstore/sue.vws**, on a host named **earth**.
- The new location for the view storage directory is **/public/sue.vws** on host **ccsvr04**, whose architecture differs from **earth**'s.

To move the view:

1. **Go to the view's host.** Log on to the view host, **earth**, as the view's owner:

   rlogin earth –l sue

2. **Determine whether the view has a nonlocal private storage area.**

   ls –ld /users/sue/viewstore/sue.vws/.s

   ```
   ...   .s -> /public/view_aux/sue
   ```

   The symbolic link indicates a remote storage pool.

3. **Deactivate the view.** Use **cleartool endview –server** to deactivate the view.

4. **(If necessary) Validate the private storage area's global pathname.** This step is required only if the view's private storage area is remote, and you are moving the view to another host. You must verify that the view's new host can access the private storage area using the same global pathname as the view's current host.

   rlogin ccsvr04
   ls /public/view_aux/sue
   .
   . *(this command should succeed)*
   .
   exit

   If the intended destination host cannot access the view's private storage area in this way, select and validate another host.

5. **Back up the view storage directory**. Use the procedure in *Backing Up a View* on page 307.

6. **Dump the view's database to ASCII dump files.**

   cleartool reformatview –dump /users/sue/viewstore/sue.vws

This command creates files **view_db.dump_file** and **view_db.state** in the view storage directory. It also renames the view database subdirectory to **db.dumped**.

7. **Copy the view storage directory**. First, make sure that the desired parent directory of the target location exists and is writable. Then, if you are moving a snapshot view directory tree, copy the entire directory tree to the new location. If you are moving a dynamic view, or if you are moving the view storage directory for a snapshot view and the view storage directory is outside the view directory tree, copy the entire view storage directory tree (but not a remote private storage area) to the new location.

   <verify that '/public' already exists on remote host 'ccsvr04'>

   cd /users/sue/viewstore
   tar –cf – sue.vws | rsh ccsvr04 'cd /public; tar –xBpf –'

   NOTE: The **-B** option to the **tar** command may not be supported on all architectures. Refer to the *ClearCase and MultiSite Release Notes* for more information, or check the reference pages for your operating system.

   If you are moving both the view directory tree and the view storage directory for a snapshot view, and if the view storage directory is outside the view's directory tree, copy both the view directory tree and the view storage directory to the new locations.

   NOTE: If you are moving a snapshot view, it is important that the utility you use to copy the view maintains the original modification times of all files and directories in the view. If it does not, loaded files become hijacked.

8. **Ensure that the old view cannot be reactivated.** Remove it from the ClearCase storage registries:

   cleartool rmtag –view –all sue
   cleartool unregister –view /users/sue/viewstore/sue.vws

   Removal prevents reactivation by ClearCase client hosts.

9. **Register the view at its new location.**

   cleartool register –view /public/sue.vws
   cleartool mktag –view –nstart –tag sue /public/sue.vws

   If your network has several network regions, you need to make additional view-tag entries. This procedure is essentially similar to the one in *Ensuring Global Access to the VOB—Special Cases for UNIX* on page 128.

NOTE: The **mktag** command fails if you omit the **–nstart** option.

10. **Re-create the view database from the dump files.**

    cleartool reformatview –load –tag sue

11. **Reactivate the view.**

    cleartool startview sue

12. **On the new view host, delete the backup view database**. This backup, named **db.dumped**, was created on **ccsvr04** by **reformatview –load** in Step #6.

    rm –fr /public/sue.vws/db.dumped

13. **Delete the old view storage directory**. If you did not overwrite the existing view storage directory, delete the old one. Be sure to first verify that the view can be accessed at its new location.

NOTE: Moving a view does not modify the **.view** file in the view storage directory. The information in this file always describes the view's first incarnation.

22.3 Moving a Dynamic View's Private Storage Area on UNIX

Use the following procedure on a UNIX host to move a dynamic view's private storage area to another location. For clarity, we use an example involving view storage directory **/users/sue/viewstore/sue.vws**, on host **earth**. The procedure works both when the private storage area is local (**.s** is an actual subdirectory of the view storage directory) and when it is remote (**.s** is a UNIX-level symbolic link to a remote location).

1. **Log on to the view's host as the view's owner.**

   rlogin earth –l sue

2. **Deactivate the view.** Use **cleartool endview –server** to deactivate the view.

3. **Go to the private storage area.** The private storage area is a standard UNIX directory tree, with **.s** as its root.

   cd /users/sue/viewstore/sue.vws/.s

4. **Copy the entire directory tree.** You can copy the directory tree to a new location using **cp**, **rcp**, **tar**, or other commands. For example:

   mkdir –p /public/view_aux/sue.priv
   cp –r * /public/view_aux/sue.priv

   Be sure to select a new location that is globally accessible.

5. **Replace the old .s directory with a symbolic link**. It doesn't matter whether the existing **.s** is an actual subdirectory or a symbolic link. Move it aside and create a (new) symbolic link in its place.

   cd ..
   mv .s .s.MOVED
   ln –s /public/view_aux/sue.priv .s

6. **Reactivate the view**. Use **startview** or **setview**.

7. **Remove the private storage area.** When you have verified that the private storage pool is working well in its new location, you can remove the old one:

   - If old private storage area **.s.MOVED** is an actual directory:

     rm –fr /users/sue/viewstore/sue.vws/.s.MOVED

   - If old private storage area **.s.MOVED** is a UNIX symbolic link:

     cd /users/sue/viewstore/sue.vws/.s.MOVED
     rm –fr *
     cd ..
     rmdir .s.MOVED

23 Removing Views

This chapter presents procedures for rendering a view inaccessible.

23.1 Taking a View Out of Service

The procedure for taking a view out of service temporarily is essentially equivalent to the VOB procedure described in *Taking a VOB Out of Service* on page 229. The primary difference is in the way you terminate the supporting process. For a view, you use **cleartool endview** to stop the single view-related server process, the **view_server**, which runs on the host where the view storage directory resides (the *view host*):

cleartool endview –server alh_main

Restoring the View to Service

The procedure for restoring a view to service is similar to that described in *Restoring the VOB to Service* on page 230.

23.2 Permanent Removal of a View

To permanently remove a view, use the ClearCase Administration console:

1. Navigate to the view storage node for the view. This is a subnode of the host node for the host where the view storage directory resides.

2. Click **Action**>**All Tasks**>**Remove View**.

You can also use the **rmview** command to remove a view permanently. These commands remove all relevant entries from the ClearCase storage registry. The procedure for removing a view also removes references to the view from VOBs.

If you have tried to remove a view by removing its storage directory, you may need to remove references to the view (checkouts and derived objects) from some VOBs. Use the following procedure:

1. Use the following command, and note the view's UUID from the list of views referenced by a VOB:

   cleartool describe –long vob:*vob_tag*

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

2. If the view-tag still exists, remove the tag from the registry. In the ClearCase Administration Console, you can use the View Tags node for the tag's regions to remove view-tags. You can also use the following commands:

   cleartool endview –server *view_tag*
   cleartool rmtag –view *view_tag*

3. Unregister the view. In the ClearCase Administration Console, you can use the View Object node to remove a view object. You can also execute the following command, using the view's UUID from Step #1:

   cleartool unregister –view –uuid *uuid*

4. For each VOB that holds references to the view, remove the view-related records from the VOB. In the ClearCase Administration Console, you can use the Referenced Views subnode of a VOB storage node to remove a view's records from a VOB. You can also execute the following command, using the view's UUID from Step #1:

   cleartool rmview –all –uuid *uuid*

After you have removed references to the view from all VOBs, remove the view storage directory if any of it remains.

Administering ClearCase Licenses

Administering ClearCase Licenses 24

This chapter provides an introduction to the ClearCase floating license scheme.

24.1 Floating License Architecture

ClearCase implements an active-user floating license scheme. To use ClearCase, a user must obtain a license, which grants the privilege to use ClearCase commands and data on any number of hosts 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. If it gets one, the user can keep it for an extended period. Entering any ClearCase command renews the license. If the user doesn't enter a ClearCase command for a substantial period—by default, 60 minutes—another user can take the license.

One or more hosts in the local area network are designated as ClearCase license server hosts. Each license server host maintains a list of license keys. UNIX hosts maintain this list in a file named **/var/adm/atria/license.db**. Windows NT hosts store keys in the **LicenseKeys** value in the Windows Registry key **HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion,** which contains one or more license entries. Each license entry defines a specified number of licenses, allowing that number of ClearCase users to be active at the same time. The **license.db** reference page includes a description of the license database file format.

When a user first attempts to use ClearCase software on any host in the network, a license-verification check is made:

- ClearCase client software looks for the name of the license server host. On UNIX, this name is stored in the file **/var/adm/atria/config/license_host**. On Windows, it is stored as the

LicenseHost value in the Windows Registry key **HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion**.

- ClearCase communicates with the license server process on the license server host, to verify the user's right to use ClearCase. (The license server process is actually **albd_server**, performing these duties in addition to its other tasks.)
- The license server process determines the user's rights and returns an appropriate message.
- Depending on the message the license server sends, the command either proceeds or is aborted.

Subsequently, similar license-verification checks are performed periodically.

License Priorities

Each user can (but need not) be assigned a *license priority* in the license database file. Each user specified in a `-user` line gets a priority number: the first user gets priority 1 (highest priority), the second user gets priority 2, and so on. All users who are not specified in any `-user` line share the lowest priority.

License Expiration

Each license entry can have an expiration date. (The expiration time is at 00:00 hours on that date.) During the 72-hour period before the expiration date, attempts to use a license from that license entry succeed, but a warning message appears. After the expiration time, attempts to use those licenses fail.

License Report Utility

The **clearlicense** utility produces a report on the licenses defined in the license database file and on current user activity. You can also use this utility to force a user to release a license, freeing it for use by another user.

24.2 Setting Up a License Server

The task of setting up a host to be a license server host is part of the overall ClearCase installation process. Initial license server setup is described in the *ClearCase Product Family Installation Notes*.

Adding New Licenses to an Existing License Server Host

Some organizations purchase an initial set of ClearCase licenses and purchase additional sets of licenses later. Each time a new set of licenses is purchased, you will receive a text line containing the new license authorization code from Rational.

1. **Determine the host name of the license server host.** If you do not know the name of the license server host, you can find out from any ClearCase client host. On a UNIX client host, examine the contents of the **license_host** file.

   cat /var/adm/atria/config/license_host
   ```
   fermi
   ```

   On a Windows host, run Control Panel and open the ClearCase program. Click the **Licensing** tab to display the name of the license server host.

2. **Log on to the license server host.** Log on as *root* on a UNIX host or as a local administrator on a Windows NT host**.**

3. **Add the licenses**. On UNIX, you can use any text editor to append the license strings to the end of the **license_db** file. Be sure to enter this line exactly as it appears on the fax from Rational.

   On a Windows host, run Control Panel and open the ClearCase program. Click the **Licensing** tab and select the **The local system can act as a license server** check box. Enter the license key exactly as it appears on the fax from Rational into the **License Keys** box. Click **Apply** or **OK**.

Setting Up Additional License Server Hosts

When you purchase a new set of ClearCase licenses, you can place them on another host, rather than adding them to the existing license server host. Having two or more license server hosts provides a redundant source of licenses.

NOTE: You must make this decision before you send the *ClearCase Product Family License Registration Form* to Rational; you must include the machine ID of the intended license server host on this form.

To set up a new license server host:

1. **Determine the machine ID of the new license server host.** You can use the **clearlicense** command:

   ```
   clearlicense –hostid
   XXXYYYZZZ                    (architecture-dependent machine-ID string)
   ```

   (If you need to determine the machine ID of a host on which ClearCase is not yet installed, follow the instructions on the License Registration Form to determine the machine ID.)

   On a Windows host, run Control Panel and open the ClearCase program. Click the **Licensing** tab to display the licensing hostid.

2. **Fill out the** *ClearCase Product Family License Registration Form*. A copy of this form appears at the back of the *ClearCase Product Family Installation Notes*. Fax the form to Rational and wait for a return fax that lists your license authorization code.

3. **Log on to the new license server host.** Log on root on a UNIX host or a local administrator on a Windows NT host**.**

4. **Add licenses to the new host's license database**. On UNIX, you can use **cat** to append the license strings to the end of the **license_db** file:

   ```
   # cat >> /var/adm/atria/license.db
   –license ClearCase sun *.19 19960419 xxxxx.yyyyy.zzzzz
   <CTRL+D>
   ```

   Or you can use any text editor. Be sure to enter this line exactly as it appears on the fax from Rational.

   On Windows NT, run Control Panel and open the ClearCase program. Click the **Licensing** tab and select the **The local system can act as a license server** check box. Enter the license key exactly as it appears on the fax from Rational into the **License Keys** box. Click **Apply** or **OK**.

5. Stop and restart ClearCase on the license server host.

6. Reconfigure ClearCase client hosts to use the new license server host. On UNIX hosts, edit **/var/adm/atria/config/license_host** to contain the name of the new license server host. On Windows NT, run Control Panel and open the **ClearCase** program. Click the **Licensing** tab

and enter the new license server's name in the **Hostname of the license server** box. Click **OK**.

You can also use the ClearCase Administration Console to change the license server used by any ClearCase host in your network:

- Go to the host node for the host whose license server you want to change. This can be the My Host node or a subnode of the ClearCase Network node in ClearCase Administration Console, or the top-level node in ClearCase Host Administration.
- Click **Action**>**Properties**. This command opens a dialog box in which you can change the host's license server. To change the license server for a remote host, the host must be configured to allow remote administration, and you must be a member of the *ClearCase group*.

24.3 Moving Licenses to Another Host

This section presents a procedure for moving a set of licenses to another host. Note the following:

- If a *license database* contains more than one set of licenses (that is, more than one `license` line), you can move some sets and not move others.
- This procedure is not required if you merely change the hostname of a license server host. It is only required if the license hostid (as reported by **clearlicense –hostid**) changes for any reason. Common reasons include moving the license server host function to a new computer or replacing the network interface card of an existing host.

To move licenses to another host:

1. **Complete the** *Request to MOVE ClearCase Product Family Licenses* **form.** A copy of this form appears at the back of the *ClearCase Product Family Installation Notes*. Fax the form to Rational and wait for a return fax that lists your replacement license authorization codes.
2. **Move the licenses:** If you are moving licenses to a host that is already a license server host, follow the procedure in *Adding New Licenses to an Existing License Server Host*. If you are moving licenses to a host that is already a license server host, follow the procedure in *Setting Up Additional License Server Hosts*.

24.4 Renaming a License Server Host

Renaming a license server host does not invalidate its license authorization code; the code incorporates a hardware-level machine identifier, not its OS-level hostname. After renaming a host, switch the license server host assignments of some or all ClearCase hosts, as in Step #6 in the section *Moving Licenses to Another Host* on page 337.

Administering the ClearCase Registry

25 Understanding the ClearCase Registry

This chapter describes the mechanisms by which ClearCase VOBs and views are made available throughout the local area network. ClearCase maintains a central registry of names for VOBs and views and of globally valid paths to VOB and view storage directories. The ClearCase registry also maintains information about some global default values. These include default locations for VOB and view storage, and site-wide default parameters for some ClearCase operations.

You can use the ClearCase Registry node of the ClearCase Administration Console to administer the registry server. You must be a member of the *ClearCase group* to modify data in the registry using the ClearCase Administration Console.

25.1 Storage Directories and Access Paths

Each ClearCase VOB and view has a physical location and a global name:

- **Physical location.** Each *VOB storage directory* and *view storage directory* is actually a directory tree, located on a ClearCase host in a regular file system. For daily work, developers need not know the actual locations of these storage directories.
- **Global name**. Each VOB and view also has a tag, a name that is associated with a global path to the VOB or view storage directory. In their daily work, developers use *VOB-tags* and *view-tags* to access the data structures.

Distributed VOBs and Views on UNIX

On UNIX computers, ClearCase allows you to distribute the data storage for a given VOB or view to more than one host. You can create any number of additional VOB *storage pools* that are remote from the VOB storage directory (**mkpool** command); similarly, you can place a view's private storage area on a remote host (**mkview –ln** command).

In both cases, remote data storage is implemented at the UNIX level. As far as ClearCase servers are concerned, the data is located within the VOB or view storage directory; standard UNIX symbolic links implement the reference to remote storage.

NOTE: Remote data storage is outside the scope of this chapter; see Chapter 9, *Understanding VOB Storage* for more information. The ClearCase storage registries discussed here are not used to resolve the symbolic links that implement distributed data storage.

25.2 Storage Registries

All VOB storage directories are registered in a database that constitutes the *VOB registry*; all view storage directories are registered in a database that constitutes the *view registry*. These storage registries record physical locations—host names and pathnames on those hosts. They also record the logical *access paths* that clients and servers use to access VOB and view data. These databases reside on a host on the network, called the ClearCase registry host, that is accessed by all ClearCase client and server applications.

A storage registry has two parts: an *object registry* and a *tag registry*. The following sections provide an overview of these components; for details, see the **registry_ccase** reference page.

25.3 Object Registries

The *VOB object* and *view object* registries record the location of each VOB and view storage directory using a *host-local pathname*. That is, the pathname is valid on the host where the storage directory resides. These pathnames are used by the ClearCase server processes (**view_server**, **vob_server**, and so on), which run on that host.

An entry is placed in the appropriate object registry when a VOB or view is first created (**mkvob**, **mkview**). The entry is updated whenever the VOB or view is reformatted (**reformatvob**,

reformatview). You can also update or remove the entry manually (**register, unregister**) when you move a VOB or view or rename a server host.

Object registry entries are used mostly by ClearCase server processes.

25.4 Tag Registries

For most purposes (including virtually all daily development activities), VOBs and views are not referenced by their physical storage locations. Instead, they are referenced by their VOB-tags and view-tags.

Tag Registries on UNIX

- The *view-tag* of an *active* dynamic view appears as a subdirectory entry in a host's *viewroot* directory, **/view**. For example, a dynamic view with tag **oldwork** appears in the host's file system as directory **/view/oldwork**. To access ClearCase data, developers must use a view, either implicitly (by setting a dynamic view) or explicitly (by using a view-extended pathname for a dynamic view or by loading element versions into a snapshot view).
- In a dynamic view, the *VOB-tag* of a VOB is its mount point as a file system of type MVFS. Developers access all ClearCase data in dynamic views at pathnames below VOB mount points. In a snapshot view, a VOB-tag appears as a subdirectory of the snapshot view root directory.

Tag Registries on Windows NT

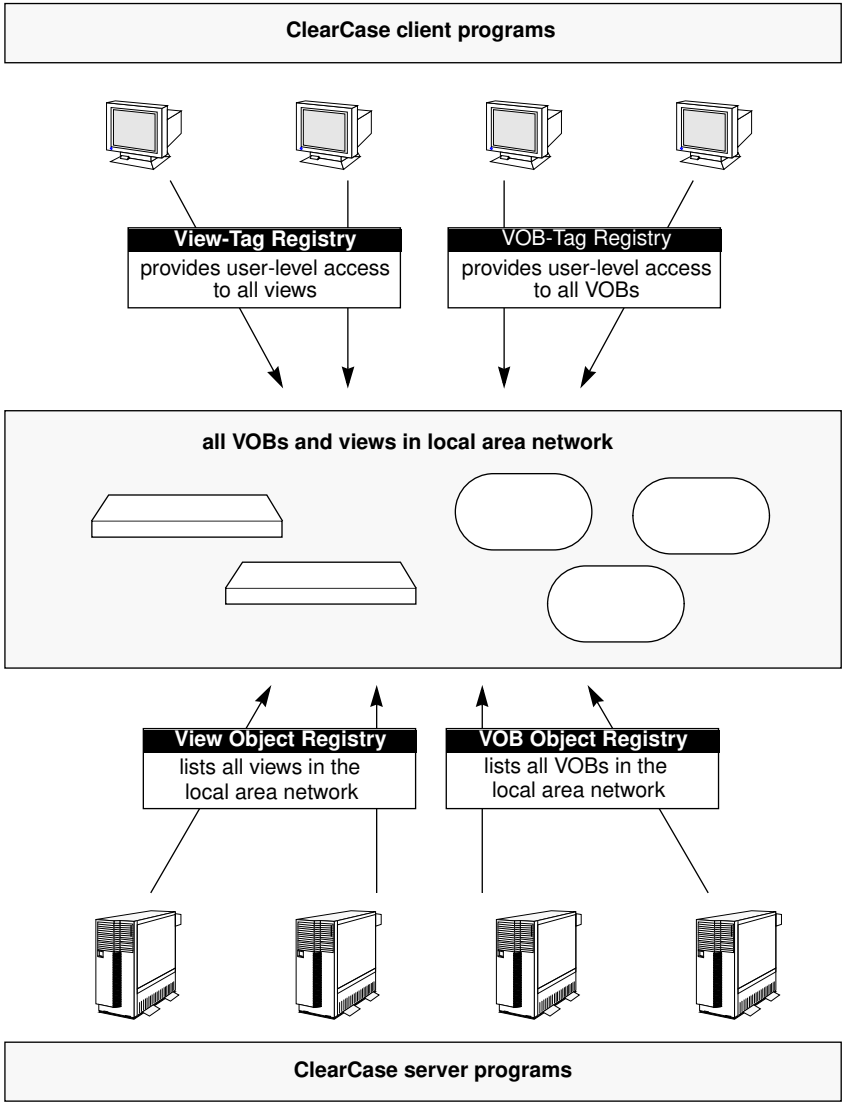
- The *view-tag* of an *active* dynamic view appears in the *dynamic-views root directory* (**\\view** by default) or the *dynamic-views drive* (**M:** by default). For example, a view with tag **oldwork** appears in the file system as directory **\\view\oldwork** or **M:\oldwork**.
- The *VOB-tag* of a VOB is its registered name and its logical root directory. A VOB-tag has a single component and begins with a backslash (\). For example, **\myvob** and **\vob_project2** are legal VOB-tags. When a VOB is mounted, its VOB-tag appears as a subdirectory under each dynamic view's view-tag visible on the **M:** drive. In a snapshot view, the VOB-tag of any VOB configured in the view's load rules appears as a subdirectory

of the snapshot view root directory. Developers access all ClearCase data at pathnames below VOB-tags.

Thus, any reference to a ClearCase file-system object involves both a VOB-tag and a view-tag. ClearCase uses the networkwide *VOB-tag* and *view-tag* registries to resolve these logical locations to physical storage locations. Each tag registry entry includes a *global pathname* to the storage area—a pathname that is valid on all ClearCase client hosts. Figure 32 illustrates how tag registries and object registries are used to access the network's set of data storage areas.

In some networks, it is not possible to devise global pathnames to all ClearCase storage areas that are valid from every host at a site. The ClearCase *network region* facility handles such situations; see Chapter 26, *Administering Regions*. Figure 32 illustrates a network that has a single network region.

Figure 32 ClearCase Object and Tag Registries (Single Network Region)



25.5 Networkwide Accessibility of VOBs and Views

Networkwide storage registries make all VOBs and views visible to all users. You can use the All VOBs and All Views nodes of the ClearCase Administration Console or the **lsvob** and **lsview** commands to list them all.

Typically, VOBs and views have different patterns of use:

- Most users require access to most (or all) VOBs.
- Most users require access to only a few views.

Accordingly, there are different schemes for *activating* VOBs and dynamic views on each client host.

- On UNIX, the set of *public* VOBs is activated (mounted) by the ClearCase startup script on a client host. On Windows NT, the system administrator typically sets up ClearCase client hosts to mount the set of *public* VOBs at user logon time.
- On both UNIX and Windows, users activate their views as needed with explicit commands.

Public and Private VOBs

To provide control over how a VOB is activated, each VOB-tag is designated as *public* or *private* when it is created.

- On UNIX computers, all public VOBs are mounted as a group when ClearCase starts. To defeat this behavior, use the **noauto** mount option at VOB creation time (see **mkvob –options**).
- On Windows computers, all public VOBs are mounted as a group when you issue the following command:

  cleartool mount –all

  Commonly, you automate this command's execution for ClearCase users in either of two ways:

  - Adding the command to the startup script for ClearCase users.
  - Placing the command in a batch file for use in each user's **Startup** program group.

This technique is particularly useful because, in its role as a *network provider*, the MVFS deactivates all VOBs and views on the local host at user logon time. That is, each time a user logs on, the **M:** drive is empty until VOBs and views are reactivated. However, Unified Change Management can remount VOBs automatically when you log on, and you can arrange to remount a VOB when you log on by selecting the **Reconnect at Logon** check box in the **Mount** dialog box when you first mount the VOB.

A password facility controls public VOB creation. When creating a public VOB or VOB-tag (with the ClearCase Administration Console or with **mkvob** or **mktag –vob**), you must enter a password that is usually established when ClearCase is installed. (For information on how to create or change the VOB registry password, see the **rgy_passwd** reference page.)

NOTE: Any user can mount any VOB, public or private. The private designation means only that a VOB will not be mounted by a **cleartool mount -all** command, but must be mounted explicitly, by name.

25.6 Managing VOB and View Registry Entries

ClearCase creates VOB-tag and VOB object registry entries when you create a VOB. It creates view-tag and view object registry entries when you create a view. You can use the ClearCase Registry node of the ClearCase Administration Console or various **cleartool** subcommands to inspect, create, and repair registry entries.

Viewing VOB and View Registry Entries

Use the VOB Objects and View Objects subnodes of the ClearCase Registry node in the ClearCase Administration Console to inspect VOB and view object entries:

1. Open the ClearCase Administration Console.
2. Navigate to the VOB Objects or View Objects subnode of the ClearCase Registry node.
3. Select a VOB object or view object in the details pane.
4. Click **Action**>**Properties**.

Use the VOB Tags and View Tags subnodes of a region in the ClearCase Registry node in the ClearCase Administration Console to inspect VOB-tag and view-tag entries:

1. Open the ClearCase Administration Console.
2. Navigate to the VOB Tags or View Tags subnode of a region in the ClearCase Registry node.
3. Select a VOB-tag or view-tag in the details pane.
4. Click **Action**>**Properties**.

You can also use **lsvob** or **lsview** with the **–long** option to report key registry information. If you cannot see a VOB with **lsvob** or a view with **lsview**, then no tag for the view is registered in the current network region (or for the region specified in the command). For a VOB or view with a registered tag, the output from **lsvob –long** and **lsview –long** includes this information:

- The VOB-tag or view-tag
- The host name for the host on which the storage directory resides
- The host-local access pathname (from the **vob_object** or **view_object** file), which the relevant VOB or view server uses to access the storage directory
- The global pathname (from the **vob_tag** or **view_tag** file), which is used to resolve references to the newly created VOB

For example:

cleartool lsvob –long

```
  .
  .
Tag: /vobs/src
  Global path: /net/venus/vobstore/src.vbs
  Server host: venus
  .
  .
  Vob server access path: /vobstore/src.vbs
  .
  .
```

cleartool lsview –long

```
  .
  .
Tag: main
  Global path: /net/venus/viewstore/main.vws
  Server host: venus
  .
  .
  View server access path: /viewstore/main.vws
  .
  .
```

If the global and server access paths are identical or if they don't look right to you, then you may anticipate problem reports regarding access to the VOB or view.

Creating VOB and View Registry Entries

The **mkvob** and **mkview** commands and the GUI tools that create VOBs and views create both the object and tag registry entries necessary for ClearCase client access. (There is one exception: if you create a VOB with a public tag, but the tag password fails, the VOB is created without the tag. You must create the tag in a separate operation with **mktag**.) **mkvob** and **mkview** output includes this information:

- The host-local access pathname (from the **vob_object** or **view_object** file)
- The global pathname (from the **vob_tag** or **view_tag** file)

Sample **mkvob** output:

```
  .
  .
Host-local path: venus:/vobstore/src.vbs
Global path:     /net/venus/vobstore/src.vbs
  .
  .
```

Sample **mkview** output:

```
  .
  .
Host-local path: venus:/viewstore/main.vws
Global path:     /net/venus/viewstore/main.vws
  .
  .
```

Creating VOB-Tags and View-Tags

Use the ClearCase Administration Console or **mktag** to add or replace tag entries for existing VOBs and views. These are two common uses:

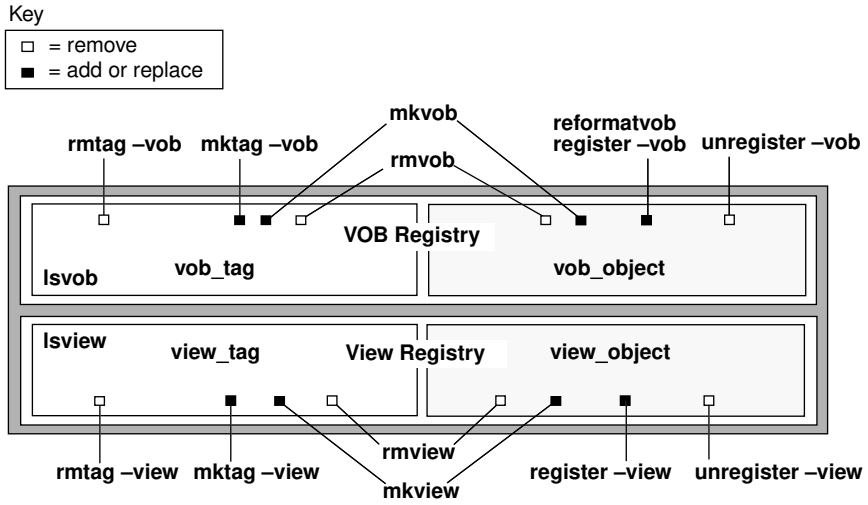
- Creating additional VOB-tags and view-tags to support multiple network regions
- Converting a private VOB to a public VOB

To change a tag's name, or to change its assigned network region, use the ClearCase Administration Console to remove the tag and create a new one, or use **rmtag** and then **mktag**, not **mktag –replace**. See the **mktag** and **rmtag** reference pages for details on tag creation and removal.

To change the assigned network region for a tag or to copy a tag in another region, you can drag the tag from one region to another using the Regions subnode of the ClearCase Registry node in the ClearCase Administration Console. If you change a tag's region, be sure that the global path is valid in the new region.

Figure 33 shows the **cleartool** commands that affect registry files. See also the **registry_ccase** reference page.

Figure 33 cleartool Commands and the ClearCase Storage Registry



25.7 Creating Server Storage Locations

The ClearCase registry includes information about *server storage locations* for VOBs and views. A VOB server storage location is a shared directory on a ClearCase server that appears in a list of recommended VOB storage locations when a user creates a VOB. A view server storage location is a shared directory on a ClearCase server that appears in a list of recommended view storage locations when a user creates a view.

Each server storage location is valid for a particular region. It is a global path that must be valid for all ClearCase hosts in the region.

You can create an initial set of server storage locations after you install ClearCase on a server.

- On a Windows NT computer, if you specify during installation that locations on that server are to be available to ClearCase clients for creating VOB and view storage directories, the ClearCase Server Storage Configuration Wizard runs when you first log on after the installation restart. You can use the wizard to establish the initial a set of storage locations for VOBs and views.
- On a UNIX or Windows NT computer, you can use the **mkstgloc** command to create server storage locations for VOB and view storage.

You can also use the ClearCase Administration Console to add, change, or remove server storage locations for a given registry region:

1. Start the ClearCase Administration Console.
2. Navigate to the Storage Locations subnode of the ClearCase Registry node.
3. To create a new storage path, click **Action**>**New**>**Server Storage Location**.
4. To change the properties of an existing storage location, select one and click **Action**>**Properties**.
5. To remove an existing storage location, select one from the list and click **Action**>**Remove Server Storage Location.**

25.8 Registering Site-Wide Properties

The ClearCase registry maintains a set of properties for ClearCase. ClearCase uses the value for a site-wide property when you perform an operation that uses that property and you don't specify the property's value. For example, when you create a view and do not specify one of the shareable DOs options, ClearCase uses the site-wide value.

You can use the ClearCase Administration Console to view or edit site-wide properties on the ClearCase registry server for a region:

1. Start the ClearCase Administration Console.

2. Navigate to the ClearCase Registry node.

3. Click **Action**>**Properties**.

You can also use the **cleartool lssite** command to display the set of site-wide properties and their values. Use the **cleartool setsite** command to set new values for site-wide properties. For information about the available properties and possible values for them, see the **setsite** reference page in *ClearCase Reference Manual*.

25.9 Registry Guidelines

Following are some general tips, warnings, and guidelines regarding ClearCase storage registry administration:

- You cannot access a VOB or view (even to remove it) unless it has both a tag registry entry and an object registry entry. Use the ClearCase Administration console or **lsvob** or **lsview** to see whether the tag is missing (no listing at all) or the object entry is missing (no storage paths appear in the **–long** output).

  If you cannot access a VOB or view, first make sure it has a tag. If it does not appear in the output of an **lsvob** or **lsview** command, it has no tag. Create one with the ClearCase Administration console or **mktag**.

  If it has a tag, list the tag with the ClearCase Administration console or **lsvob –long** or **lsview –long**. If the output includes incorrect pathnames (identical local and global pathnames usually mean trouble), you can try to re-register the object (with **mktag** and/or **register**) to supply correct pathnames. If everything looks correct and you still cannot access the VOB, contact Rational Technical Support.

- When using UNIX symbolic links, make sure the links can be resolved by all hosts that need to access the VOB. For example:

  /usr/vobs -> /usr/vobstore *<--- wrong*

  When this link is resolved on a client host, the full pathname **/usr/vobstore** references the **/usr** directory on the client host.

  /usr/vobs -> ../vobstore *<--- right*

  When this link is resolved on a client host, the relative pathname **../vobstore** references a location on the VOB host.

Similarly, beware of full pathnames in VOB symbolic links that point to locations in other VOBs.

- On UNIX VOB and view hosts, watch out for links to unexported disk partitions. If you cannot access a VOB and suspect a faulty registry pathname, follow the registered global pathname to the storage host and **pwd** to see where you are. Make sure a link does not point to a location in an unexported partition. For example:

  /usr/vobs -> ../vobstore

  This link causes failures if the location **/usr/vobstore** is in another partition that is not exported.

- Don't rename a tag; remove it and create a new one.

- Don't use non-ClearCase commands to delete a storage directory; this doesn't clean up the registry entries.

- Don't delete tags before deleting the storage directory; let **rmvob** or **rmview** delete tags for you.

- When creating a VOB or view, creating a VOB-tag or view-tag, registering or unregistering a storage directory, or reformatting a VOB—especially when using the **–host –hpath –gpath** options together—execute the command on the host where the VOB or view storage directory resides. This enables ClearCase to validate some pathnames, which it cannot do when the command is executed from a remote host.

- Use the **rgy_check** utility to diagnose problems or for periodic cleanup of obsolete or stranded registry entries.

Multiple Registries

We do not recommend configuring multiple registry hosts to serve clients in the same network region. A network configuration that includes a separate registry server for each region can be used if organizational or other concerns require it. However, maintaining multiple registry servers can significantly increase the workload of a ClearCase administrator. Registry access does not usually create significant server load, and any benefits that may be achieved through load-balancing should be weighed carefully against the costs of maintaining registry data on several hosts. even for an active community of ClearCase users to spread the registry server access load across multiple server hosts.

If you decide to maintain multiple registry hosts, keep the following points in mind:

- If you want clients in separate registries to share one or more VOBs or views, create all shared VOBs and views in the same region before registering them in additional registries.
- When registering a VOB or view in an additional registry, run the **mktag** and **register** commands. Also, to register a VOB or view in a region served by registry host **luna**, you must run **mktag** and **register** from a host assigned to registry host **luna**.
- The following administration tools expect a single registry and cannot be used effectively in a multiple registry installation: Region Synchronizer, **rgy_check**, and **rgy_switchover**.
- Although you can create regions of the same name in different registries, this is likely to cause confusion and is not recommended.

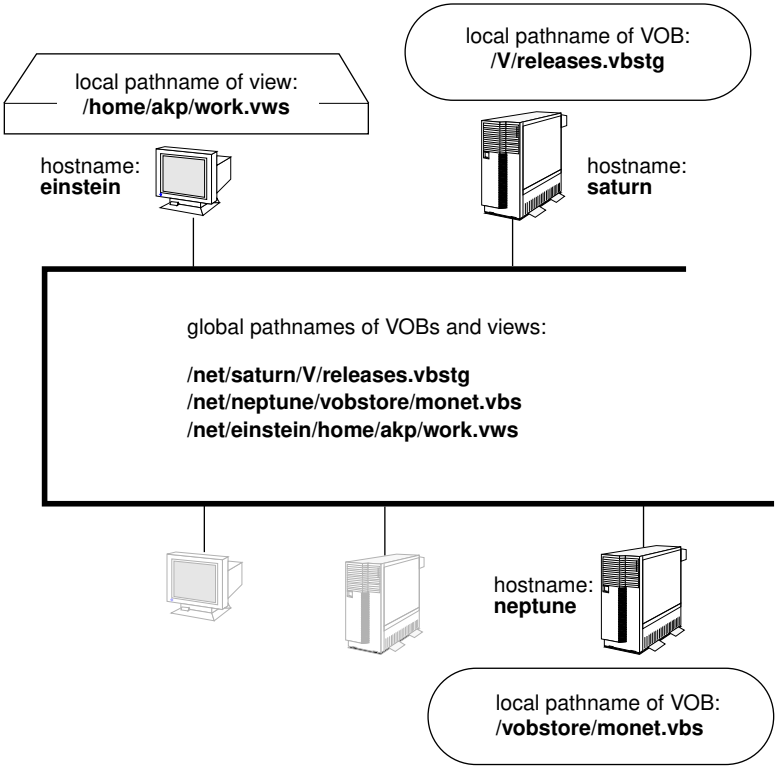
26 Administering Regions

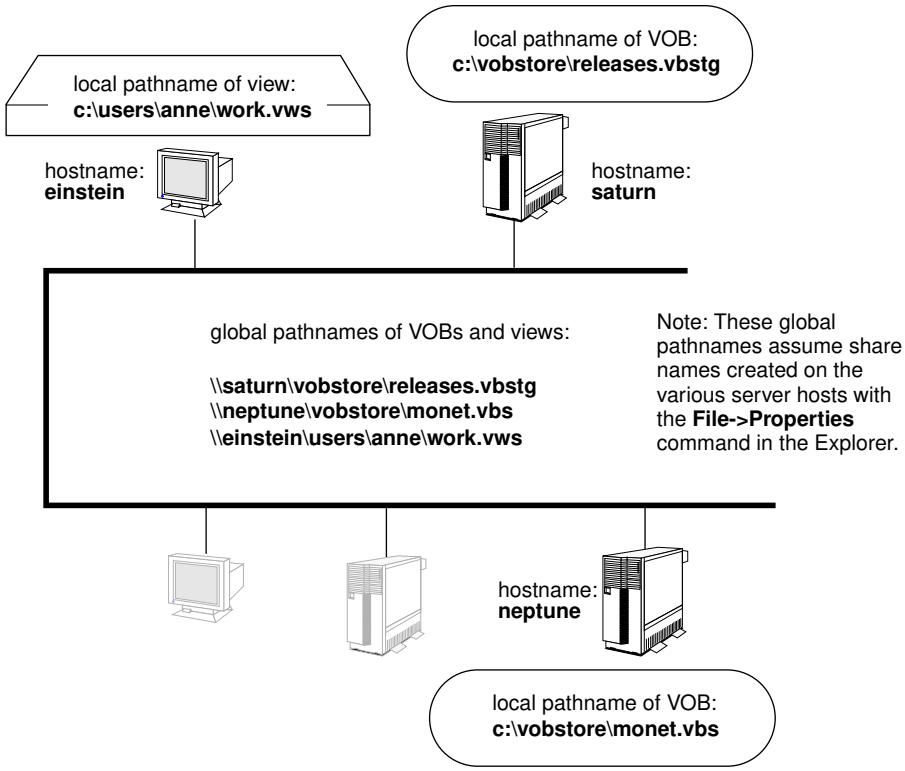
This chapter discusses administration of *network regions*. A region is a conceptual subset of a network in which hosts refer to shared resources using the same global pathnames. You can have a single region if all hosts in your network can use the same pathname to access each VOB and view storage directory. In some networks, not all hosts can use the same pathnames to refer to shared resources. Most commonly, this is because of differences among operating systems in conventions for forming global pathnames. In these cases, you can define separate network regions for hosts with different naming conventions. Regions are maintained by the ClearCase registry.

26.1 Network Regions

Ideally, your network's VOB and view storage directories are accessible at the same pathnames throughout the network. Figure 34 shows a simple network in which global naming has been achieved.

Figure 34 Network with Global Naming





Uniform global naming may not be achievable if your network supports both Windows and UNIX hosts, hosts with multiple network interfaces, or UNIX computers with multiple aliases.

ClearCase servers require consistent pathnames to shared storage areas. If you cannot achieve global consistency, you must partition your network into a set of *network regions*, each of which is a consistent naming domain. Each region has the following characteristics:

- Each ClearCase host must belong to a single network region.
- All hosts in a given network region must be able to access ClearCase physical data storage (that is, all VOB storage directories and the storage directories of shared views) using the same full pathnames.
- Developers access VOBs and views through their VOB-tags and view-tags. All hosts in a given network region use the same tags.

- Each VOB must have a tag in the region of the host on which the VOB storage directory resides. Each view must have a tag in the region of the host on which the view storage directory resides. A VOB or view can have tags in other regions as well.

For example, a VOB and a view may be accessed in different network regions as follows:

Region: **nt_dev**
VOB storage: **\\neptune\public\vega_project.vbs**
VOB-tag: **\vega**
View storage: **\\saturn\shared_views\int_43.vws**
View-tag: **int_43**

Region: **unix_dev**
VOB storage: **/net/neptune/public/vega_project.vbs**
VOB-tag: **/vobs/vega**
View storage: **/net/saturn/shared_views/int_43.vws**
View-tag: **int_43**

Use the Regions subnode of the ClearCase Registry node in the ClearCase Administration Console to create, view, and remove regions. You can also use the **mkregion**, **lsregion**, and **rmregion** commands to create, view, and remove regions.

Registries in a Multiple-Region Network

Conceptually, each network region has its own *view-tag registry* and *VOB-tag registry*. Each VOB can have at most one tag in a region. In a typical network with *n* regions, each VOB or view storage directory has *n* tag entries.

A VOB or view need not have a tag in every region; it is inaccessible for development work on hosts in any region for which there is no tag. You can use network regions as access domains instead of naming domains. (However, a VOB must have a tag in the region to which the VOB storage directory's host belongs, and a view must have a tag in the region to which the view storage directory's host belongs.)

If possible, keep the tag itself constant across the regions. For example:

Region	VOB-tag	Pathname to Storage Area in Region
nt1	**\project**	**\\neptune\public\vega_project.vbs**
unix1	**/project**	**/net/neptune/public/vega_project.vbs**

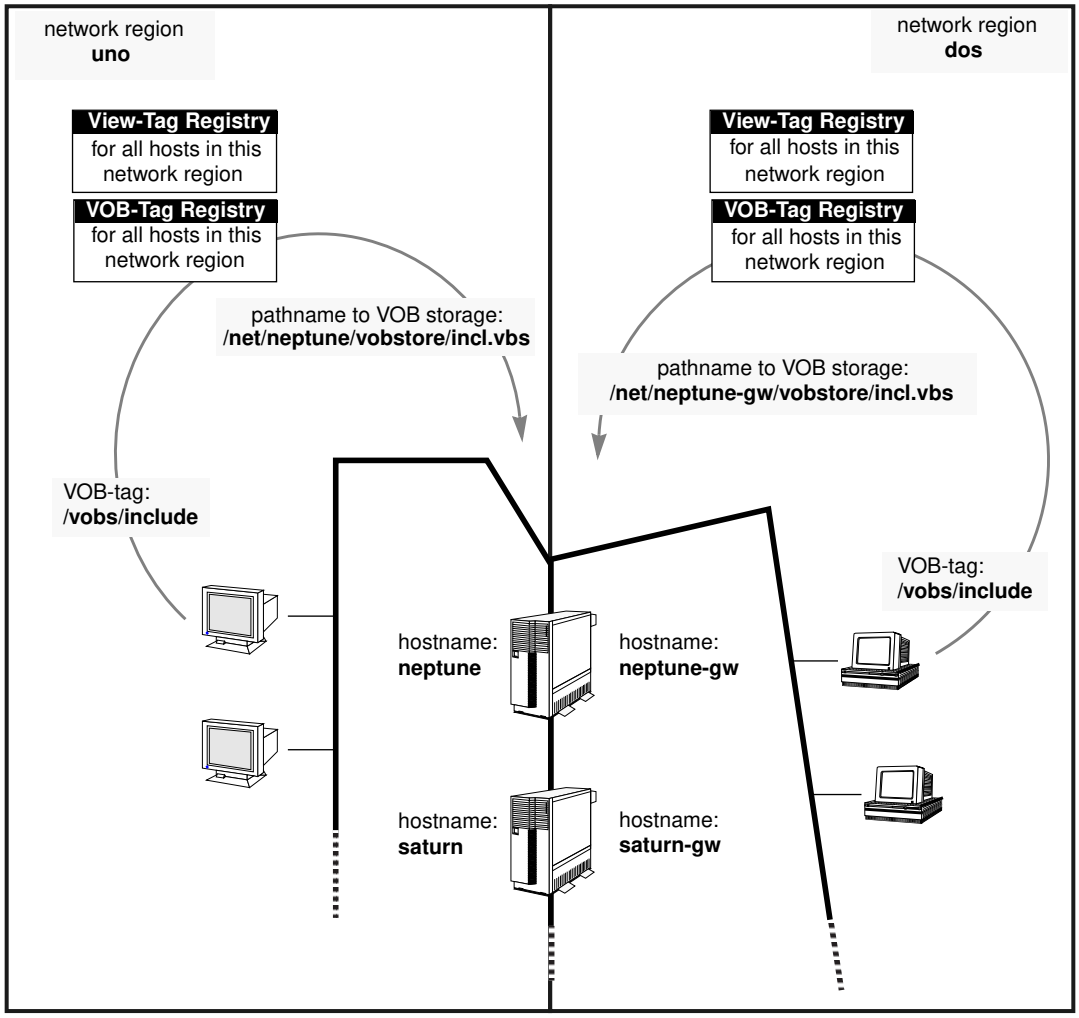
NOTE: In a mixed Windows and UNIX environment, VOB-tags that differ only in their initial directory separator character—backslash (\) or slash (/)— are equivalent.

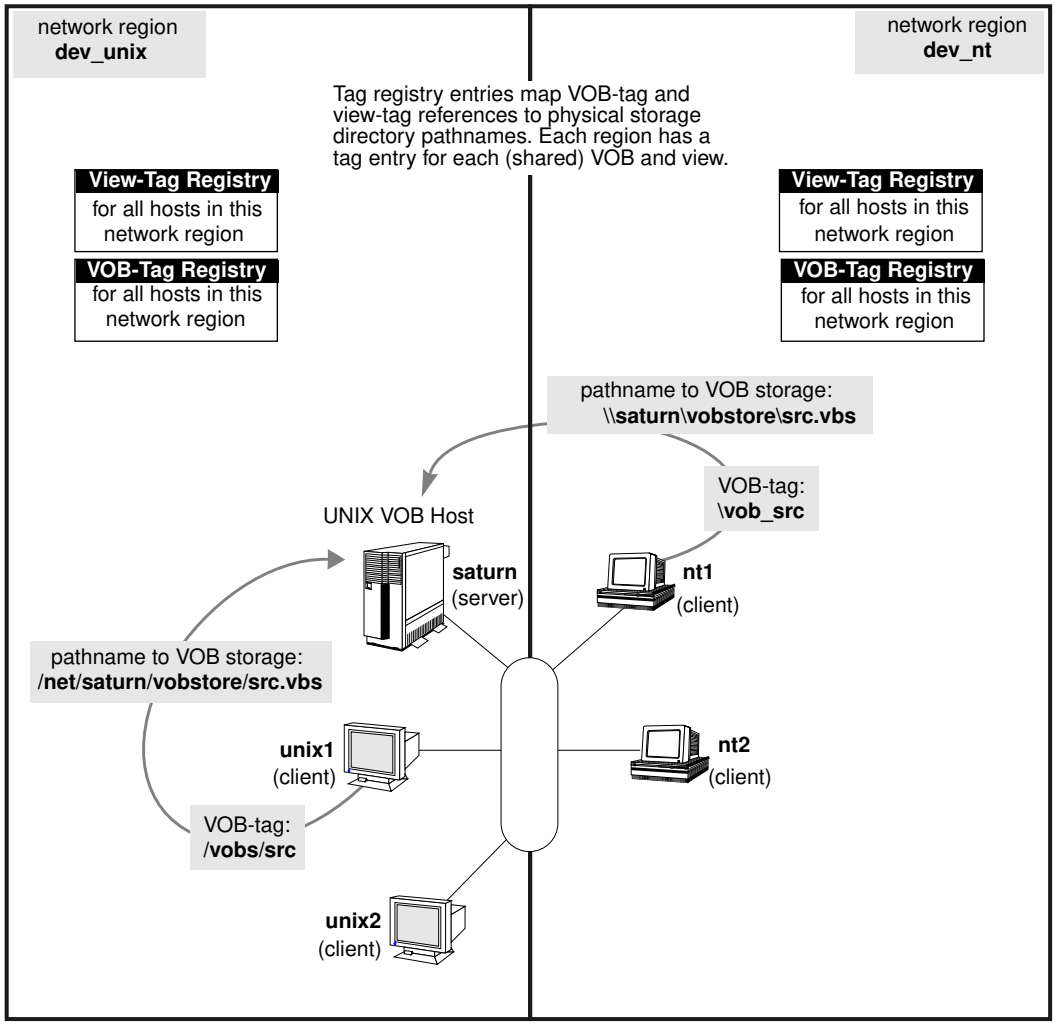
This set of tags provides a standard way for users to refer to a VOB, even though network file system idiosyncrasies may dictate platform-specific names for the VOB's physical storage location.

Tag Registry Implementation

Figure 35 illustrates a simple two-region network, each with its own logical set of tag registries. All hosts in a network region use the same VOB-tags and view-tags, and access ClearCase data storage areas using the same pathnames, provided by registry lookups.

Figure 35 Network Regions and Their Tag Registries





Establishing Network Regions

The site preparation program (described in the *ClearCase Product Family Installation Notes*) prompts you to specify the name of a network region. This name becomes the default region, which can be accepted or overridden during ClearCase installation on individual hosts. To list a host's network region, use the host node of the ClearCase Administration Console, the ClearCase program in Control Panel, or the **cleartool hostinfo –long** command.

26.2 Adding a Network Region

Ensuring Global Access to the VOB—Special Cases for UNIX on page 128 outlines the circumstances that may require you to adjust the registry entry generated for a VOB or view. That section presents some sample **mktag** commands that fix the storage registry by explicitly specifying, for a VOB or view, the host-local and global access paths. This section addresses the situation in which a single, global pathname to a VOB or view storage directory does not exist for all network hosts that must access it, so you must partition your network into at least two *network regions*.

NOTE: The most common multiple region scenario involves a mixed network environment of Windows and UNIX hosts. If you are working in, or planning, such an environment, read the remainder of this section to acquaint yourself with the procedure for adding a network region. Then use procedures described in Chapter 8, *Configuring VOB and View Access in Mixed Environments* to perform the actual work of registering VOBs and views.

When to Create Additional Regions

There are several aspects of a computer network that may require you to create one or more additional registry regions:

- **Multiple host architectures.** Network naming conventions differ between UNIX and Windows. If you have to support ClearCase on both UNIX and Windows hosts, you will need at least two registry regions. For example, a VOB that is accessed as **/net/neptune/vobstore/incl.vbs** on a UNIX host may be accessed as **\\neptune\vobstore\incl.vbs** on a Windows host.

- **Multiple network interfaces.** A UNIX or Windows NT VOB host or view host may have two or more interfaces to the network, each corresponding to a different host name. A VOB with storage on such a host will have one global storage pathname for each network interface (and hostname) the host supports. For example, for a host with two interfaces and two names (neptune and neptune-gw), the global storage path to a VOB with local storage at **/public/project.vbs** is different for each hostname:

  ```
  /net/neptune/public/project.vbs
  /net/neptune-gw/public/project.vbs
  ```

- **UNIX hosts with multiple aliases.** The standard UNIX facilities for assigning names to hosts—file **/etc/hosts** or NIS map **hosts**—allow each host to have any number of alternate names, or aliases. This is a possible hosts entry:

```
195.34.208.17 betelgeuse bg                    (alias)
```

If shared storage resides on this host, ClearCase clients may be able to access the storage using either a **/net/betelgeuse/**... pathname or a **/net/bg/**... pathname.

There is another partitioning scenario with a different focus: you want to prevent, not promote, VOB and view sharing. You can create a separate region—or even a separate registry, served by a second registry host—to administer a cluster of hosts for which some or all VOBs and views visible in another region are not registered and, hence, not visible.

No matter how many regions you have, each ClearCase host can belong to only one region.

Multiple Regions Vs. Multiple Registries

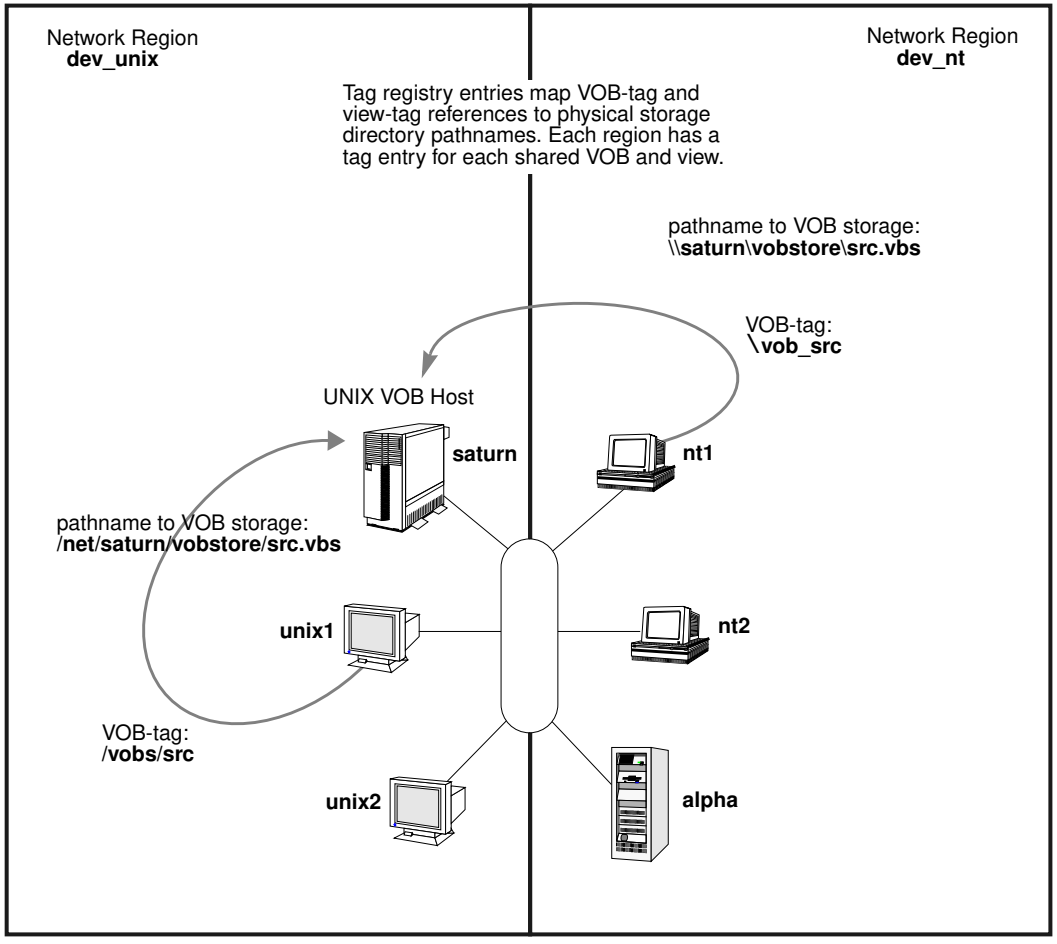
Instead of, or in addition to, creating multiple network regions in a single ClearCase registry, you can construct a network with multiple registry hosts, each serving separate clusters of ClearCase hosts that may or may not share VOBs and views. This approach complicates the administration process; it is not recommended unless specific circumstances require it.

The procedures here are intended primarily to support single registry installations and offer only basic guidance for multiple registry (as opposed to multiple region) sites. Some general guidelines are available in the section *Multiple Registries* on page 353.

A Sample Network Partition

Figure 36 illustrates a mixed-architecture (Windows and UNIX) network. There are two network regions, one for Windows hosts and one for UNIX hosts. Two regions are required because UNIX servers store VOBs and views that must be accessed by *all* network hosts. In this example, hosts in both regions use the same view-tags, but not the same VOB-tags. Furthermore, each region's hosts access VOB and view storage using different pathnames. Therefore, separate tag registry entries are required.

Figure 36 Sample Network with Two Regions



Procedure for Adding a Network Region

Use this procedure to create one or more additional regions after your first region has been created. This procedure assumes that:

- ClearCase has been installed on each host.
- VOBs and views have been created on the region's VOB and view storage hosts.

To Create a New Region

Use the ClearCase Administration Console to create a new region (in this example, the region is named **dev_nt**):

1. Start the ClearCase Administration Console.
2. Navigate to the Regions subnode of the ClearCase Registry node.
3. Click **Action**>**New**>**Region Tag**. This command opens a dialog box in which you create the new region.

You can also use the **mkregion** command to create region **dev_nt:**

cleartool mkregion –tag dev_nt –tcomment "Windows NT ClearCase hosts"

To Move a Host into a New Network Region

A host gets its default region and registry server assignment at ClearCase installation time. To move the host into a different region served by the same registry server:

- On a UNIX host, edit the file */var/adm/atria/rgy/rgy_region.conf* to contain the name of the new region. You must be logged in as root to change this file.
- On a Windows host use the Registry tab of the ClearCase program in Control Panel to specify the new region.
- Stop ClearCase on the host
- Restart ClearCase on the host

To move the host into a different region served by a different registry server:

- On a UNIX host, edit the file **/var/adm/atria/rgy/rgy_region.conf** to contain the name of the new region and edit the file **/var/adm/atria/rgy/rgy_hosts.conf** to contain the name of the new registry host and optional backup registry host. You must be logged in as root to change these files.
- On a Windows host, run Control Panel and click the Registry tab of the ClearCase program to specify the new region and a new Registry server host.
- Stop ClearCase on the host.
- Restart ClearCase on the host.

You can verify that the new region assignment is in effect by running the **hostinfo** command to display the current registry host and network region.

```
cleartool hostinfo –long
```

NOTE: If a host has been configured to allow remote administration, you can also use the host node of the ClearCase Administration Console to change the host's region, though you cannot use the ClearCase Administration Console to stop or start ClearCase on a host.

To Create VOB-tags and View-tags for a New Network Region

A Note About Creating Tags. In this scenario (registering UNIX VOBs and views for a Windows region in the same ClearCase registry), the Region Synchronizer offers an alternative to the **mktag** commands presented below. See Chapter 8, *Configuring VOB and View Access in Mixed Environments*, for details. You can also copy tags from one region to another using the ClearCase Registry node of the ClearCase Administration Console. You must be careful to change the global paths of the new tags you create in this way to ensure that they are valid in the new tags' regions.

VOB-tags and view-tags are created for the new region using **mktag** commands of the form:

cleartool mktag –vob –tag *vob-tag* **–region** *new-network-region*
–host *hostname*
–hpath *host-local-pathname*
–gpath *global-pathname*
storage-dir

cleartool mktag –view –tag *view-tag* **–region** *new-network-region*
–host *hostname*
–hpath *host-local-pathname*
–gpath *global-pathname*
storage-dir

For example, host **nt1** in Figure 36 accesses a UNIX VOB through VOB-tag **\vob_src**, which was created for the **dev_nt** region with the following command:

```
cleartool mktag –vob –tag \vob_src –region dev_nt ^
  –host saturn ^
  –hpath /vobstore/src.vbs ^
  –gpath \\saturn\vobstore\src.vbs ^
   \\saturn\vobstore\src.vbs
```

From this point on, each new shared VOB or view created on the network requires a tag entry for each region. That is, if a new VOB or view is created anywhere on the network, and it must be visible to hosts in both the **dev_unix** and **dev_nt** regions, you must use the ClearCase

Administration Console or an additional **mktag** command to create a new tag in the second region. For example:

- After creating VOB **/vobs/lib** on **saturn** in the **dev_unix** region:

  ```
  cleartool mktag –vob –tag \lib ^          (Create, for all Windows NT hosts
      –region dev_nt ^                          in the  dev_nt region, a VOB-tag
      –host saturn ^                            registry entry for the VOB
      –hpath /vobstore/lib.vbs ^                whose storage directory resides
      –gpath \\saturn\vobstore\lib2.vbs ^       at \\saturn\vobstore\lib.vbs)
      \\saturn\vobstore\lib.vbs
  ```

  The **mktag** command can be executed on a UNIX host. But the final argument, used by **mktag** itself to find the storage directory, is different on UNIX—**/vobstore/lib.vbs**, for example, rather than **\\saturn\vobstore\lib.vbs**.

- After creating view **view4** on host **nt2** in the **dev_nt** region:

  Do nothing. UNIX hosts cannot use Windows views, so there is no need to create a new tag for the **dev_unix** region.

  NOTE: There must be separate tag entries for each region, but actual tag *names* (**\src** or **\vob_src**, **myview** or **r2.1_view**, for example) should, if possible, be the same for all regions. The ideal is consistent, transparent VOB access from any network region. However, this ideal may be impractical in some environments, because Windows *VOB-tags* are limited to a single pathname component.

If the New Region Is Served by a Separate Registry Host

If the new region belongs to a separate ClearCase registry (that is, the region is served by a separate registry host):

- You cannot use the Region Synchronizer to create tags for the new region.

- In addition to **mktag**, you must also use the ClearCase Administration Console or the **register** command to add entries to the second registry's VOB or view object file (**vob_object** or **view_object**). For example:

  cleartool register –view \\saturn\viewstore\view1.vws

Guidelines for Multiple Network Regions

- If possible, use the same VOB-tags in all network regions. Each VOB or view that must be visible in multiple network regions must have a separate tag *entry* in each region. Making the *name* the same for all regions provides consistent, transparent access from any network region.
- Use consistent naming conventions for directories that hold UNIX VOB mount points and for directories that store VOB or view storage directories.
- Don't use remote VOB storage pools for a UNIX VOB that must be accessed by hosts in more than one network region. If you use a remote storage pool for a shared VOB, the pool must be accessible from all hosts, in all regions, using the same global pathname.
- You can isolate an arbitrary group of hosts either with a separate network region or by setting up a separate registry host with its own registry files and assigning the desired hosts to the new registry. It is more common (and easier to administer) to use a network region for this purpose.
- A VOB's first tag (the one created by the VOB Creation Wizard or **mkvob**) must be in the home region—the region in which the VOB host resides. The same restriction applies to a view's first tag. You can then create additional tags for other regions, but make sure that a home region tag always exists.

26.3 Removing a Network Region

At some point, you may want to dissolve a network region and reincorporate its hosts into one or more existing regions. Move each host to its new region as described in *Adding a Network Region* on page 362. Then, use the ClearCase Administration Console to delete the region:

1. Start the ClearCase Administration Console.
2. Navigate to the subnode for the region you want to remove under the ClearCase Registry node.
3. Click **Action**>**All Tasks**>**Remove Region Tag**.

You can also delete the region with **rmregion**:

cleartool rmregion –tag dev_nt –rmall

27 Moving, Renaming, and Backing Up the ClearCase Registry

This chapter describes procedures for changing ClearCase registry server hosts and for backing up the registry.

27.1 Moving the ClearCase Registry: Registry Switchover

From time to time the registry server host may become unavailable; you then need to designate a new registry server. At that time, you may or may not have a backup host prepared to become the new registry server. This section uses two examples of these conditions:

- You have a functioning backup registry host, on which **rgy_backup** has been running at regular intervals.
- No designated backup registry host is available.

In both cases, the **rgy_switchover** utility does the key work.

Moving the Registry to an Active Backup Registry Host

In this scenario, primary registry host **rgy1** fails, and the ClearCase administrator makes backup registry host **rgy2** the new primary server. (**rgy_backup** has been executing successfully on **rgy2**.) While **rgy1** is down and **rgy2** is the primary registry server, **rgy3** becomes the backup registry server. Later, **rgy1** becomes available again, and the administrator reverts to **rgy1** as the primary registry server.

Use this procedure to switch the registry server. You must have write permission to the directory *ccase_var_dir*\rgy to run **rgy_backup**.

1. **Run rgy_switchover.** Make **rgy2** the primary registry server. Make **rgy3** the new backup registry server:

   rgy_switchover –backup rgy3 rgy1 rgy2

2. **Handle unreachable clients.** Record the names of any client hosts for which the switchover fails. You will reset these hosts by hand (or have their owners do so) when they become available. Later, when a failed client becomes available, take the following steps:

   a. Log on to the client.

   b. Stop ClearCase.

   c. Configure the client to use the new registry host. On UNIX clients, set the first line of its **rgy_hosts.conf** file to **rgy2**. On Windows clients, use the **Registry** tab of the ClearCase program in Control Panel. Click **Use registry server on host**. In the **Use registry server on host** box, enter **rgy2**. On the **Registry** tab, enter **rgy3** in the **Backup registry host** box. Click **OK** or **Apply**.

   d. Restart ClearCase.

   ClearCase client-only hosts and Attache clients. ClearCase hosts with client-only installations (no ClearCase server processes) and ClearCase Attache clients cannot be reconfigured automatically and always appear on the returned list of unreachable clients.

3. Host **rgy1** becomes available again.

4. **Deactivate rgy1. rgy1** is still configured as a registry host. Reconfigure it to recognize **rgy2** as the registry host. Then make **rgy1** a backup registry host, so that it takes a registry snapshot in preparation for returning to its role as primary registry server:

   a. Log on to **rgy1**.

   b. Stop ClearCase

   c. Reconfigure **rgy1** as a backup registry server. On UNIX, edit **rgy_svr.conf** to remove the string **master**, then edit **rgy_hosts.conf** to change its first line to **rgy2** and its second line to **rgy1**. On Windows NT, run Control Panel and click the Registry tab of the ClearCase program. Click **Use registry server on host**. Enter **rgy2** in the **Use registry server on host** box. On the **Registry** tab, enter **rgy1** in the **Backup registry host** box. Click **OK** or **Apply**.

   d. Restart ClearCase.

5. **Copy the active registry files to rgy1.** Run **rgy_backup** manually on **rgy1**, forcing it to take a snapshot of the active registry files on **rgy2**, in preparation for returning **rgy1** to service as the primary registry server. On **rgy1**, run **rgy_backup**

6. **Run rgy_switchover.** Make **rgy1** the primary registry server host. Return **rgy2** to its former role as the backup registry host with the following command:

   rgy_switchover –backup rgy2 rgy2 rgy1

7. **Handle unreachable clients.** Any **rgy_switchover** operation is likely to require this step. See Step #2 on page 370 for details.

Moving the Registry to a Host Not Configured for Registry Snapshots

In this scenario, either your site has failed to configure or maintain a backup registry host, or the backup registry host has failed along with the primary registry host. You want to move the ClearCase registry from host **rgy1** to host **rgy2**. When you do so, the primary registry host, **rgy1**, may or may not be available. The procedures presented here handle both cases.

No Backup Host: Primary Registry Host Is Available

You must have write permission to the directory *ccase_var_dir*\rgy to run **rgy_backup**.

1. **Configure rgy2 as a backup registry host.** Follow the procedure in *Changing Backup Registry Host Without rgy_switchover* on page 373.

2. **Copy the registry files to rgy2.** Run **rgy_backup** on **rgy2**:

   rgy_backup

3. Follow the switchover procedure in Step #1 on page 370.

No Backup Host: Primary Registry Host is Down

Primary host **rgy1** is unavailable; prospective registry host **rgy2** is not configured as a backup registry host. You must restore the **rgy** directory backup from **rgy1** to the **rgy** directory tree on **rgy2** and reconfigure all hosts, as described in the **registry_ccase** reference page.

The following procedure is a potentially time-saving alternative, in which you restore and re-create registry-related files on the target registry host, **rgy2**, making it possible to run **rgy_switchover** as if **rgy2** were a real backup registry host.

1. **Restore registry files to target registry host, rgy2, from backup**. Retrieve the ClearCase **rgy** directory and **client_list.db** file from backup and load its files into *ccase-var-dir***/rgy/backup.** When you are finished, the **backup** subdirectory includes all the registry data files listed in the **registry_ccase** reference page.

   Look for the client_list.db file. A registry host stores its **client_list.db** file alongside (not in) the **rgy** directory in *ccase-var-dir*. If you do not have this file, **rgy_switchover** cannot reconfigure clients for you. You must do it by hand by editing the **rgy_hosts.conf** file on UNIX clients or, on WIndows, running Control Panel and using the **Registry** tab of the ClearCase program.

2. **Create a backup_list file.** Create a text file named **backup_list** and put it into the **backup** directory. The first line of **backup_list** must be the name of the primary registry server, in this case **rgy1**. The remaining lines of **backup_list** must be the names of all the registry data files listed in the **registry_ccase** reference page, one file name per line. List only the name of each file, not the path to the file.

3. Follow the switchover procedure in Step #1 on page 370.

27.2 Renaming the Registry Server Host

ClearCase client hosts cache the name of the registry server host. To rename the network's registry server host:

1. **Shut down ClearCase on each client host.**

2. **Switch registry server host assignments.** Do this on all ClearCase hosts, as described in Step #2 on page 370.

3. **Rename the registry server host.** Make the change using the vendor-supplied procedure or tool.

4. **Restart ClearCase on all hosts.**

27.3 Changing the Backup Registry Host

You can change the backup host as part of a **rgy_switchover** operation or without changing the primary registry host.

Changing Backup Registry Host Using rgy_switchover

You are promoting the current backup registry host to primary host, as described in *Moving the ClearCase Registry: Registry Switchover* on page 369. In this case, the **–backup** argument to **rgy_switchover** specifies the new backup registry host. In addition, **rgy_switchover** updates all reachable clients automatically to recognize the new configuration.

Changing Backup Registry Host Without rgy_switchover

You are changing the backup registry host, without changing the primary registry host with **rgy_switchover**. The following procedure makes host **rgy3** the backup registry host while leaving **rgy1** the primary registry host:

1. **Log on as the privileged user on each ClearCase host.**
2. **Stop ClearCase.**
3. **Reconfigure the host** to use rgy3 as the backup registry host. On UNIX, open **/var/adm/atria/rgy/rgy_hosts.conf** and change its second line to **rgy3**. On Windows, use the **Registry** tab of the ClearCase program in Control Panel. Enter **rgy3** in the **Backup registry host** box. Click **OK** or **Apply**.
4. **Restart ClearCase**.

27.4 Renaming a VOB or View Host

You must make registry-level adjustments when you rename a host that holds the physical storage for one or more VOBs or views. From the registry perspective, the rename-host procedure is quite similar to the move-storage-directory procedures described in Chapter 14, *Moving VOBs* and Chapter 22, *Moving Views*.

1. **Make sure that the storage directories are not being used.** If the host is home to one or more VOBs, make sure that all clients unmount those VOBs (**cleartool umount**). If the host is home to one or more views, terminate all processes using the views.
2. **Shut down ClearCase processing on the host.**

3. **Rename the host.** Make the change using the vendor-supplied procedure or tool. In most cases, this involves a restart of the operating system, which also restarts ClearCase processing. In any event, make sure that ClearCase processing is restarted on the host.

   NOTE: If the host is a license or registry server host, you must reconfigure clients.

4. **Adjust the registry entries for each storage directory.** Use **register –replace** to update object registry entries, use **mktag –replace** to update tag entries, and use **mkstgloc** to update server storage locations.

5. **Update VOB databases corresponding to the host's views.** If any views have their storage directories on the host, you must update the databases of all VOBs that the views have accessed with **checkout** and/or **clearmake**. (The views' old locations are still recorded in the VOBs.) For each view/VOB combination, check out one of the VOB's elements in the view. (You can cancel the checkout (**uncheckout**) immediately, if you want.)

Administering Scheduled Jobs

28 Managing Scheduled Jobs

The ClearCase job scheduling service runs programs periodically. Using the scheduler, you can define jobs to be run one or more times at specified intervals or in specified sequences. The scheduler also provides the following features:

- E-mail notifications of job-related events.
- Remote administration 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 any host where the **albd_server** is installed.

28.1 Tasks and Jobs

The scheduler manages ClearCase-related 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 381 and *Managing Jobs* on page 383.

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 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 381.

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 383.

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 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**. A task has the following execution environment:

- The task runs with the identity of the **albd_server** (*root* on UNIX and usually the **clearcase_albd** account 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 **albd_server** identity (*root* or **clearcase_albd**). In addition, on Windows NT, **ATRIAHOME** is set to *ccase-home-dir*.

28.2 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 and derived object storage pools of all local VOBs, using **scrubber**.
- Copy the VOB database for all local VOBs that are configured for snapshots, using **vob_snapshot**.
- Copy the ClearCase registry from the *primary registry server host* (when run on a *backup registry server host*), using **rgy_backup**.
- Run user-defined daily operations in **ccase_local_day.[sh | bat]**.
- Generate and cache data on disk space used by all local views, using **space**.
- Generate and cache data on disk space used by all local VOBs, using **space**.

Weekly jobs:

- Scrub some ClearCase logs (see the **errorlogs_ccase** reference page).
- Scrub the databases of all local VOBs, using **vob_scrubber**.
- Run user-defined weekly operations in **ccase_local_wk.[sh | bat].**
- Generate and cache data on disk space used by derived objects in all local VOBs, using **dospace**.

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 13, *Administering VOB Storage*. For more information on managing view storage, see Chapter 21, *Administering View Storage*.

The default schedule also includes jobs to automate the synchronization of MultiSite replicas. These jobs are designed to run daily but are disabled by default, whether or not MultiSite is installed. For more information on these jobs and how to enable them for use with MultiSite, see *ClearCase MultiSite Manual*.

28.3 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 379). You can place the program anywhere in the 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 383. 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 386.

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 386. 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.

28.4 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 Host snap-in on the console has a Scheduled Jobs node from which you can manage jobs on any Windows NT or UNIX host accessible to the console. If the schedule ACL supports write access (see *Managing the Scheduler Access Control List* on page 388), you can also use the **cleartool schedule** command on the host where the job scheduler is running.

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 388.

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 381.
- 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 local host.
- The schedule on which the job runs. See *Specifying a Job's Schedule* on page 384.
- Job-related events that trigger e-mail notifications and recipients for the notifications. See *Specifying Job Notifications* on page 385.
- 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 host on which you want the new job to run.
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 379.

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 host on which you want to specify a job's schedule.

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:

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 each host where the scheduler runs. 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 the 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 host on which you want to view job properties.
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:

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 host on which you want to edit a job.
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 starts a text editor containing 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 host on which you want to run a job.
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 379.

Deleting a Job

To delete a scheduled job, use the ClearCase Administration Console:

1. Navigate to the Scheduled Jobs node for the host on which you want to delete a job.

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*

28.5 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 hosts running Windows NT and an NIS domain for hosts 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 6 shows the access types and their implications for access to the schedule and access to the ACL itself.

Table 6 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 host where the scheduler is running, 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 host on which you want to view or edit the scheduler's ACL.

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

29 Configuring a Web Server for the ClearCase Web Interface

This chapter explains how to configure a ClearCase Web server and to enable use of the ClearCase Web interface.

29.1 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 be installed on a system running a Web server. Supported Web servers are Apache, Microsoft Internet Information Server (IIS), and Netscape. If a ClearCase view server is run on the same computer (not required but preferable), the server needs enough free disk space to load ClearCase (see

ClearCase and MultiSite Release Notes), plus approximately 0.5 to 1.0 MB for each user who will use the ClearCase Web interface through the server. Because a large site may have many systems operating as multiple Web servers, first choose a server on which to install ClearCase.

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:_____

- ClearCase requires only a snapshot view client configuration (no MVFS is needed) on the Web server. For best performance, allow view servers to run on the Web server. If you choose to tun the view server on another host, the view must use a storage location registered in the ClearCase registry.

- If view servers are allowed to run on the Web server, the Web interface creates snapshot view directories for client users on the server.

  Although the snapshot view directories 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 snapshot 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.

29.2 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 host 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 host 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. Therefore, sites can enable Windows NT Challenge/Response authentication under either of the following conditions:

   - All VOBs accessed through the Web interface reside on the Web server.
   - All VOBs reside on UNIX servers, and CCFS access is enabled.

   Note that NFS access to UNIX VOBs is not supported, because there is no way to establish the credentials of the Web client user with the NFS server. In other words, if UNIX-based VOBs are to be accessed through a Windows NT Web server, CCFS must be used.

   Basic authentication must be enabled, so it can be used when needed. Enable Windows NT Challenge/Response authentication only when there are Internet Explorer users on client computers logged on to a Windows NT domain, and one of the above two conditions hold.

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.

30 Configuring Integrations with Microsoft Web Authoring Tools

This chapter explains how to configure an IIS Web server and enable use of the ClearCase integrations with Microsoft Web authoring tools.

30.1 Overview of the Integration

On Windows platforms, ClearCase 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 (client or server, server recommended, MVFS not required). 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, click **Start>Programs>Rational ClearCase Administration>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.

30.2 Server Setup Procedure

The supported server platforms and required software versions for the Web server are listed in Table 7.

Table 7 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)

Table 7 Supported Platforms for Web Servers

Operating System Platform	Web Server Software	FrontPage Server Extensions / Office Server Extensions
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 37 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 406.) 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 37.

Figure 37 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

Install ClearCase with the following installation configuration:

- Local servers only (**view_server**, **VOB_server**, and **albd_server**)
- No MVFS

The Web Authoring Integration Configuration Wizard installed with ClearCase requires configured storage locations to create a VOB. We recommend that you create the default storage locations on the server as a part of ClearCase installation. The storage locations configured by the ClearCase installation are in the form \\<*hostname*>**ccstg_**<*disk*>.

If you have selected the **Server** option for the ClearCase installation, **SvrStor.exe** runs on reboot after installing ClearCase to enable you to create storage locations. Alternatively, if you have selected the **Client** option for the ClearCase installation, you can create storage locations by running the **SvrStor.exe,** located in *ccase-home-dir***etc**. You can also use **cleartool mkstgloc** command with the **–view** or **–vob** options.

Step 4: Run the Web Authoring Integration Configuration Wizard

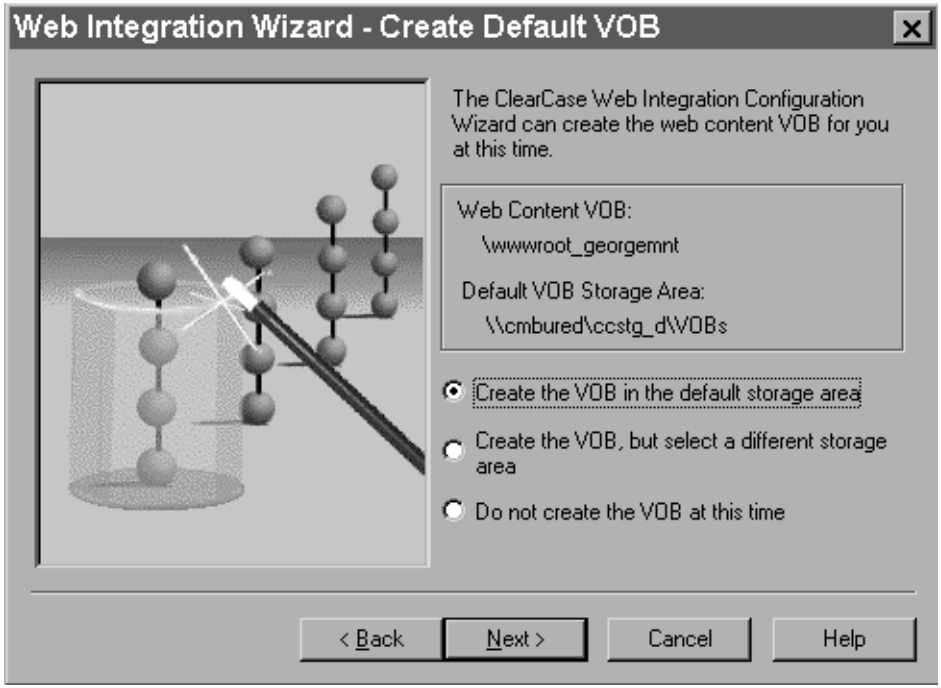
Run the Web Authoring Integration Configuration Wizard and follow the instructions. To start the wizard, click **Start**>**Programs**>**Rational ClearCase Administration**>**Integrations** >**Web Authoring Integration Configuration.**

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 presents a **Replace SourceSafe** panel. To use the integration, you must click **Yes**. Doing so does not disable any part of SourceSafe except for the SourceSafe COM API.

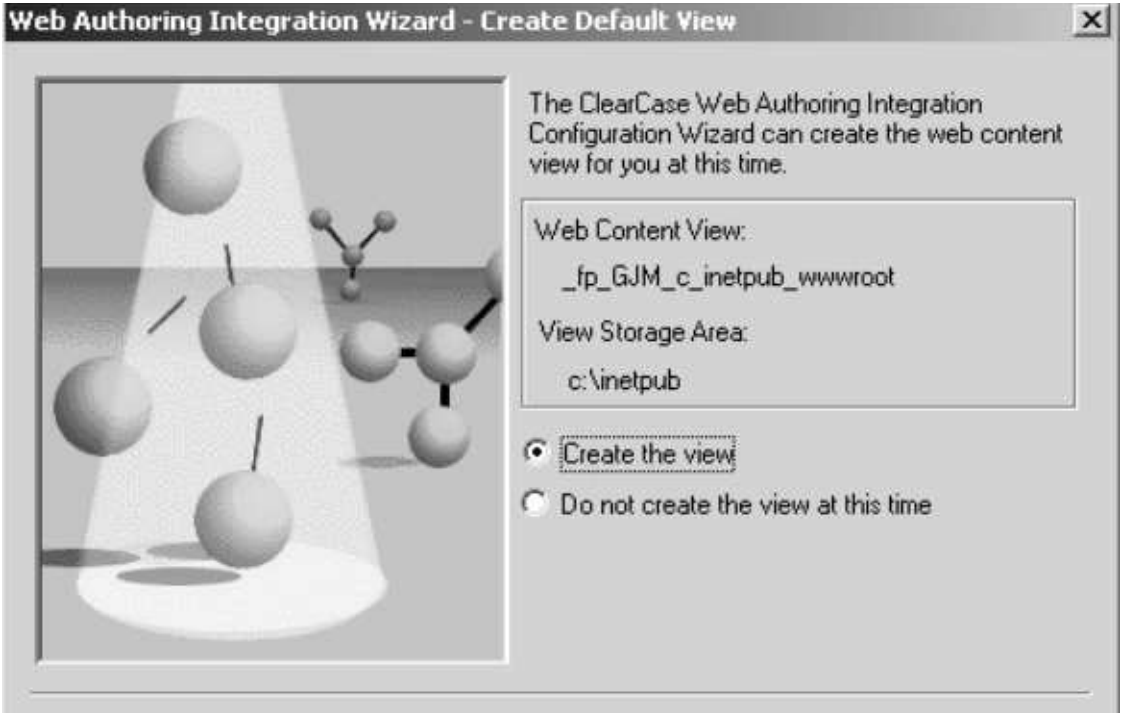
After the Web Authoring Integration Configuration Wizard checks the versions of IIS and server extensions on the Web server, you are prompted to create a content area VOB. You can accept the default VOB storage location, as in Figure 38, or use the wizard to select another VOB storage location.

Figure 38 Setting Up the VOB Storage for the Integration



The wizard then prompts you to create a content area view, as in Figure 39.

Figure 39 Setting Up the View Storage for the Integration



If you have not prepared storage locations for creation of the content VOB, you must create them and then rerun the wizard. We recommend that you put the VOB on the Web server, which increases 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 snapshot views that have a different hijacked-file detection mechanism than do standard snapshot views. This is necessary because FrontPage and InterDev rewrite files frequently. Attempts to use snapshot views created in any other way will fail. The integration does not support UCM-enabled snapshot 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.

30.3 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 on your client computer.
2. Run the Web Authoring Integration Configuration Wizard. Click **Start**>**Programs**>**Rational ClearCase Administration**>**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 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 40); in Visual InterDev, they are locks (see Figure 41).

Figure 40 FrontPage Source Control Icons

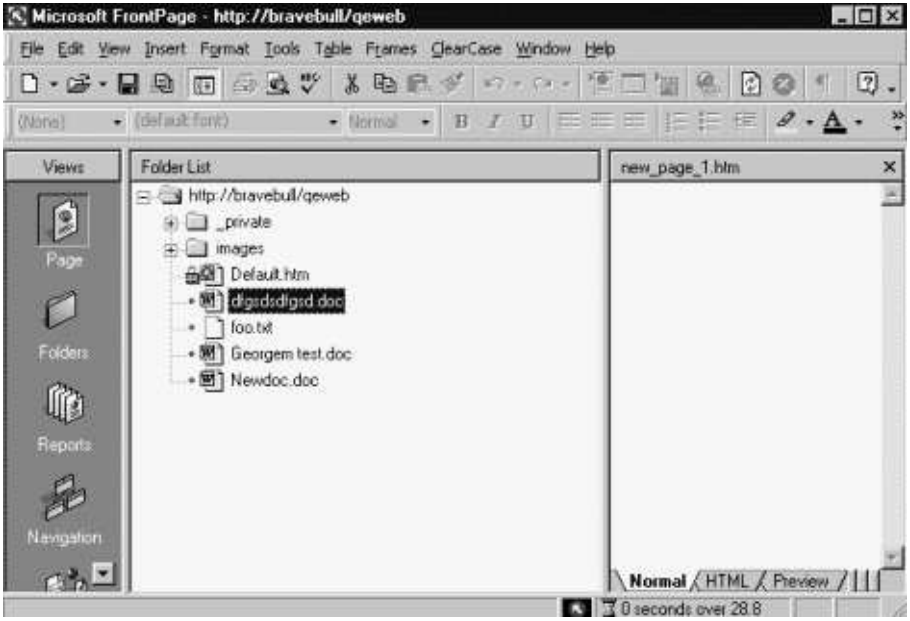
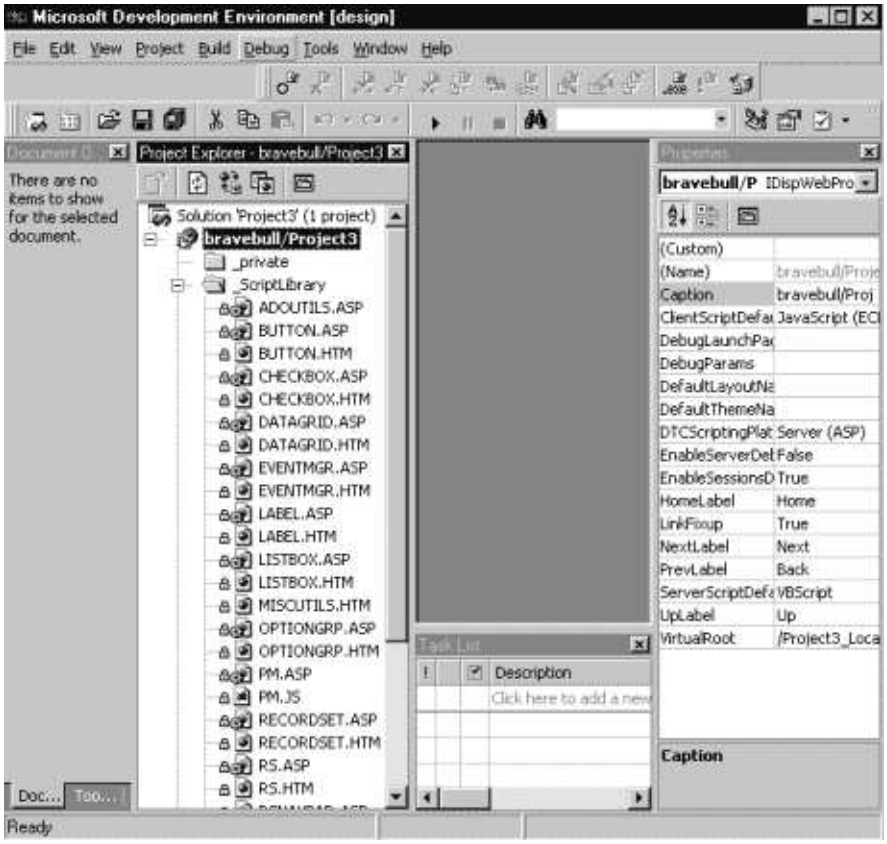


Figure 41 Visual InterDev Source Control Icons



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 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 (minimum client, view servers recommended).
3. Run Web Authoring Integration Configuration Wizard. Click **Start**>**Programs**>**Rational ClearCase Administration**>**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.

30.4 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.

30.5 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.

30.6 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.

30.7 Accessing Help Information for the Integration

When ClearCase 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 on their local systems. That is, if the elements are under ClearCase control, but ClearCase 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.

31 Using Dynamic Views to Develop and Deliver Web Content

This chapter presents a model for using dynamic views to manage Web content. It also explains how to configure IIS, Netscape, or Apache Web servers to support this model.

31.1 Overview of Using Dynamic Views on a Web Server

The difficulty in keeping Web content current and accurate increases as change cycles become shorter and more frequent. Using dynamic views as the basis for managing Web sites, you can update Web content without risk of the redundancy and delay common to copy-based update models. The model presented here uses three dynamic views to support the work flow for Web site development: one to create prototypes, one to test changes, and one to deliver completed work. This configuration allows your Web team to test changes thoroughly in the most current Web context without affecting the performance or availability of the external site.

Using dynamic views to develop and deliver Web content offers several advantages:

- You can keep Web content under ClearCase version control, and ClearCase supports a seamless interface with HTML editing tools.
- You can isolate testing, changing, and staging of Web content from the external Web site.
- You can restrict the group of people who can make changes to Web content.
- You can identify baseline states for the Web content. For example, after work that supports a new product release is complete, you can capture the state of the data by labeling the version of all elements included in this new baseline. If the Web site must return to a

previous version of the content, it is easy to roll back by using ClearCase views configured to select an earlier baseline.

- If some Web developers are not working at the same site as the external Web server, you can use ClearCase MultiSite to distribute and manage development.

31.2 Example Scenario

The Web team consists of a content development team in Massachusetts and a publishing team in California. The Web server for the external Web site is in California.

The Web team follows this publishing cycle:

1. Developers change the source files.
2. Development team leaders approve the changes.
3. Approved changes are tested in Massachusetts.
4. Changes are sent to California.
5. The publishing team runs a final test on the new content.
6. New content is published.

The team leaders at both sites established these policies:

- All source files must be tested for correctness of links, style, and colors, and the presence of ALT tags for images before developers can submit them for approval.
- Only the team leaders can approve changes for publishing.

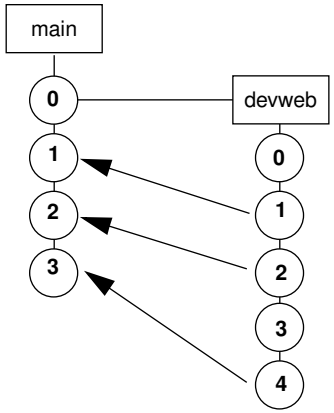
The following sections describe how each team uses ClearCase to control the publishing cycle and enforce policies.

VOB and Branch Configuration

All source files (.HTML, .PDF, .GIF, and .DOC) are stored in ClearCase VOBs. The Web team uses branches to separate ongoing development work from the work approved for publication. The

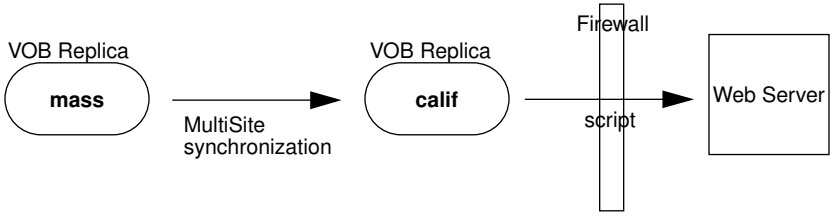
development team uses the **devweb** branch. Approved changes are merged to the **main** branch, and the Web site is published from the latest versions on the **main** branch. Figure 42 shows a version tree, including merges, for an element.

Figure 42 Development and Publishing Branches



Because the development team works at a different site, the VOB that stores Web content is replicated. The developers work in the **mass** VOB replica, and the publishing team publishes files from the **calif** replica. Because no development is done in California, synchronization of the replicas is unidirectional (changes are sent from **mass** to **calif**). The Web server is located outside the firewall, so the publishing team uses custom scripts to copy the content from the VOB to the external Web server.

Figure 43 VOB and Web Server Configuration



If development were also done in California, each site would have a site-specific branch (for example, **devweb_ma** and **devweb_ca**), and changes would be merged to the **main** branch at one site. Synchronization of the replica would occur in both directions, and depending on the speed of development, synchronization might be as frequent as four times an hour.

View Configuration for Tasks

By displaying the Web content in different views, the webmaster can use versioning and branching to control when and where content changes. Three dynamic views support the publishing cycle.

Developing New Content

The **web_devel** view is used to design site enhancements. Developers prototype and test new content designs in this view. New development is done on a new branch, to isolate changes from the other views. This is the config spec for this view:

```
element * CHECKEDOUT
element * /main/devweb/LATEST
element * /main/LATEST -mkbranch devweb
```

When developers are satisfied with new work, they check in their changes to the **devweb** branch. If a file does not meet the publication submission standards (see *Testing Source Files Before Checkin* on page 421), the checkin fails, and the developer must fix the problems and attempt the checkin again.

Merging and Testing Approved Changes

Development team leaders use the **web_merge** view to merge changes from the **devweb** branch to the **main** branch. Before checking in the merged changes, the team leaders test them. This is the config spec for the **web_merge** view:

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

The team leaders use this command to find changes that need to be merged:

cleartool findmerge . –all –fversion \main\devweb\LATEST –query –merge

Viewing and Testing Content

The **web_public** view is used for internal intranet access and provides a read-only view of the latest version of the content. The config spec for this view contains one rule:

```
element * /main/LATEST -nocheckout
```

New baselines for the Web site are tested extensively in this view. Thus, problems are identified and fixed before updated files are transferred to the external staging area.

Accessing Content from a Web Browser

To access files from a Web browser, the Web team uses different server port numbers to load each dynamic view through different URLs. For example, after the server process is started at ports 80, 5990, and 6990, each view is accessible through these URLs:

View	URL
web_public	**http://mass**
web_merge	**http://mass:5990**
web_devel	**http://mass:6990**

For information on configuring multiple dynamic views on an Apache, Netscape, or IIS Web servers, refer to *Configuring the Web Server* on page 423.

Implementing Policies

The following sections describe how the Web team uses ClearCase to enforce policies.

Testing Source Files Before Checkin

When a developer tries to check in a file on the **devweb** branch, a trigger fires and runs a script that performs the following tasks:

- It writes an entry to a report log to track the state of checkins and checkouts at the Massachusetts site. The Massachusetts Web administrator uses the report log to audit all development work.
- It runs a link-check script to check any hypertext or image references in the file. Relative paths are used in URL addresses so that all hyperlinks work regardless of which view is being accessed.
- It checks for ALT tags for all images and consistent page background colors.
- It checks for stylistic consistency, including font face and font size.

If the script finds any problems, the checkin fails and the developer receives an e-mail message that lists the problems. If the script does not find any problems, the checkin succeeds.

Restricting the Users Who Can Approve Changes

Only the team leaders are allowed to check in changes on the **main** branch. To enforce this policy, the **main** branch type is locked for all users except the team leaders. For example:

cleartool lock –nusers emma,greg,maia brtype:main@\web

This lock prevents all users except **emma**, **greg**, and **maia** from checking out versions from the **main** branch of any element in the VOB.

Labeling Approved Sources

The team uses date-based labels to indicate which files have been published on the external Web site. The publishing team applies the label before copying the latest files on the **main** branch to the Web server.

cleartool mklbtype –c "June 1, 2000 update" WEB_UPDATE_2000_06_01@\web

cleartool mklabel –recurse –c "June 1, 2000 update" WEB_UPDATE_2000_06_01 \web

Synchronizing the Massachusetts and California VOB Replicas

The team uses the MultiSite synchronization scripts to synchronize the replicas:

1. In Massachusetts, the **sync_export_list** script runs every night to send changes to the California replica. The team uses the ClearCase scheduler run the script nightly.
2. When a synchronization update packet is received at the California replica, the **sync_receive** receipt handler runs to import the packet. The **sync_receive** script is listed as the receipt handler in the **shipping.conf** file on UNIX or **MultiSite Control Panel** on Windows.

For more information about synchronization, see *ClearCase MultiSite Manual*.

Copying Files to the Web Server

Because the Web server is outside the firewall, the publishing team uses custom scripts to copy files from the VOB replica to the Web server. The script uses a view that selects the latest versions on the **main** branch.

Rolling Back to Previously Published Versions

If any problems occur with published material, the publishing team can use a ClearCase view to select a previous baseline and copy those files to the Web server. For example, the team uses a view with the following config spec to roll back to the files published on May 15, 2000:

```
element * WEB_UPDATE_2000_05_15
```

31.3 Configuring the Web Server

Table 8 lists the supported platforms for Web servers.

Table 8 Supported Web Server Platforms

Web Server Software	Supported Operating System
IIS 4.0	Windows NT 4.0
IIS 5.0	Windows NT 2000

Table 8 Supported Web Server Platforms

Web Server Software	Supported Operating System
Apache	Windows NT 4.0 Windows NT 5.0 Windows 98 Windows 95 Sun HP AIX IRIX LINUX
Netscape Enterprise Server	Windows NT 4.0 (SP5) Windows NT 5.0 Sun HP Linux

When configuring a Web server using a Windows platform, ensure that the dynamic views are started and the required VOBs are mounted at boot time. The process that starts the views and mounts the VOBs should be completed before the Web server process starts. Refer to the *Windows NT 4.0 Resource Kit* for information on using the **srvany.exe** function to start views and mount VOBs at boot time.

The steps for making the contents of a dynamic view accessible through a URL are different for each Web server.

NOTE: When you configure ClearCase on your Web server, do not locate the ClearCase license or registry server processes on the Web server because this introduces a single point of failure into your network architecture.

Configuring the Apache Web Server

Using the configuration file, you can load each dynamic view as a different instance of the Apache Web server process using different server port numbers.

To Configure an Apache Web Server on Windows

1. Install ClearCase Release 4.0 or later on the Apache Web server. Install ClearCase as a server only if VOBs will be local to Apache machine; you will also need to enable MVFS to use dynamic views.

2. Create the dynamic views to support the different stages of content development.

3. Edit *<server-root>***\conf\httpd.conf** on the Windows Web server to add a new virtual host for each dynamic view by specifying a different port number for each view using the **Listen** directive.

```
Listen 9990
NameVirtualHost 172.21.170.146:9990
<VirtualHost 172.21.170.146:9990>
ServerAdmin admin@rational.com
DocumentRoot "//view/web_public/vobs/web"
DirectoryIndex index.html
ServerName plankroad.atria.com
ErrorLog logs/error_status.log
TransferLog logs/access_status.log
</VirtualHost>
```

   The **DocumentRoot** directive specifies the dynamic view using the following syntax:

   //view/*<view-tag>***/***<VOB location>*

   Additionally, you can define the default document file name with the **DirectoryIndex** directive to enable all users to access a common file when requesting the URL of the Web site root directory.

To Configure an Apache Web Server on UNIX

1. Install ClearCase Release 4.0 or later on the Apache Web server. Install ClearCase as a server only if VOBs will be local to Apache machine; you will also need to enable MVFS to use dynamic views.

2. Create the dynamic views to support the different stages of content development.

3. Create and edit **/etc/init.d/httpd** on the UNIX Web server to add a new server instance for each dynamic view by specifying a different port number for each view using the **Command** directive where **N** would be 1, 2, 3 up to the number of server instances required.

```
COMMANDN="-c "\Port 7990\"\
-c \"User http\"\
-c \"Group http\"\
-c \"ErrorLog /xweb/apache_m/logs/error_log\"\
-c \"Options Indexes\"\
-c \"CustomLog /xweb/apache_m/logs/access_log common\"\
-c \"PidFile /xweb/apache_m/logs/httpd.pid\"\
-c \"ScoreBoardFile /xweb/apache_m/logs/httpd.scoreboard\"
```

4. For each of the command blocks you have created in Step 3, you must include a command to start the Web server. Within the **/etc/init.d/httpd** script, add the following **cleartool** command:

cleartool setview –exec "*<path-to-Web-server-executable> $COMMANDN" <dynamic-view-name>*

5. Create a symbolic link for **/etc/init.d/httpd** from the appropriate **/etc/rc.n** directory (where **n=1,2,3** and so on). For example, suppose that ClearCase is started up in **/etc/rc2.d/ S77atria**, then you can create a symbolic link to **httpd** with the following command:

   ln -s /etc/init.d/httpd S99httpd

Configuring the Microsoft Internet Information Server

You must log in as Administrator to use IIS administration tools to create a new Web site for each of the dynamic views at different port numbers.

To configure an IIS 4 or IIS 5 Web server for the dynamic views:

1. Install ClearCase Release 4.0 or later on the IIS Web server. Install ClearCase as a server only if VOBs will be local to IIS machine; you will also need to enable MVFS to use dynamic views.

2. Create the dynamic views to support the different stages of content development.

3. Start IIS service manager. Click **Start**>**Programs**>**Administrative Tools**>**Internet Services Manager**.

4. Select the server machine name in the left pane. To start the **Web Site Creation Wizard**, right-click to select **New**>**Web Site** from the shortcut menu.

5. In the **Web Site Creation Wizard**, do the following:

a. Type a description for the site.

b. Specify a TCP port number for the site.

c. Enter the path to the Web site's home directory. This is a dynamic view that you select from the **M:** drive by clicking **Browse**.

d. Select **Allow anonymous access to this Web site**.

e. Specify a user name and password to access the Web site.

f. Specify the Web site access permissions.

6. Select the root directory for the new Web site, right-click to display the shortcut menu, and click **Properties**.

7. On the **Documents** tab, use the **Add** and **Remove** buttons to configure the default document for the Web site.

8. Use a Web browser to access the default document at you new Web site by accessing **http://**<*Web_server_name*>**:**<*TCP port_number*>.

Configuring the Netscape Enterprise Web Server

Using the Netscape Enterprise Server Administration tools, you can load each dynamic view as a new server instance, using a different port address, of the Netscape Enterprise Server.

To configure a Netscape Enterprise Web server on Windows or UNIX:

1. Install ClearCase Release 4.0 or later on the Netscape Enterprise Web server machine. Install ClearCase as a server only if VOBs will be local to Netscape machine; you will also need to enable MVFS to use dynamic views.

2. Create the dynamic views as necessary to support the different stages of content development.

3. On the Netscape Server General Administration page, click **Create New Netscape Enterprise Server**.

4. Create a new virtual Web service by editing the Netscape Server Installation screen and assigning a unique server port number.

- If you are creating a new virtual Web service on Windows, specify the dynamic view in the **Document Root** box using the following syntax:

  //view/*<view-tag>***/***<VOB location>*

- If you are creating a new virtual Web service on UNIX, specify the dynamic view in the **Document Root** box using the following syntax:

  /view/*<view-tag>***/***<VOB location>*

5. Click the **Content Management** tab and then click **Document Preferences**. Type the name of the default document in the **Index Filenames** box.

Tuning for Performance

32 Improving VOB Host Performance

This chapter presents techniques for improving ClearCase performance on VOB hosts.

Your organization's VOBs constitute a central data repository. Good VOB host performance ensures that the centralized resource does not become a bottleneck.

The work of a VOB host 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 VOB host 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 VOB host, see Chapter 10, *Selecting a VOB Host*.

32.1 Minimize Process Overhead

The most effective measures for ensuring good performance from VOB hosts are also the easiest to implement:

- **Keep non-ClearCase processes off the VOB host.** Don't use the VOB host as a server host for another application (for example, a DBMS or Web Server) 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 VOB host.** Do not use the VOB server host for other ClearCase tasks such as building or other CM activities (diff/merge, **cleartool find**, and so on). The VOB server host should not also be used as a developer's desktop system.

- **Keep view_server processes off the VOB host.** This recommendation may be harder to implement; many organizations create shared views on the same hosts as VOBs. If possible, minimize this double use of VOB hosts.

  EXCEPTION: For reliable non-ClearCase access (avoiding multihop network access paths), place the VOB and the view through which it is exported on the same host. For more information, see *Setting Up an Export View for Non-ClearCase Access* on page 302.

32.2 Maximize Disk Performance

Follow these recommendations to obtain the best I/O performance on VOB hosts:

- 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.

32.3 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 VOB host 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
- Very slow **omake** or **clearmake** builds
- ClearCase clients experience RPC time-out errors

There is additional information about the relationship of main memory to VOB size in *Selecting a VOB Host* on page 120.

32.4 Tune Block Buffer Caches on UNIX

Most of the UNIX operating systems that 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 VOB host's block buffer cache average roughly 50% of the total size of all VOB database directories (*vob-stg-dir*/**db**) on the host. On most UNIX operating systems, the cache size increases when you add more main memory to the host. For information on architecture-dependent block buffer cache operation, see the *ClearCase and MultiSite Release Notes*.

If there is a substantial amount of non-ClearCase activity or ClearCase client activity on the host, 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.

Some UNIX variants provide special tools for monitoring buffer cache performance. See the *ClearCase and MultiSite Release Notes*.

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.

33 Improving View Host Performance

To speed its performance, the **view_server** process associated with a view maintains a cache. The default size of this cache is 500 KB for Windows NT and 32-bit UNIX platforms; it is 1 MB for some 64-bit UNIX platforms. You can configure a larger cache size to boost performance. A larger cache is particularly useful for views in which very large software systems are built by **omake** or **clearmake**.

This chapter includes information on interpreting view cache information and on configuring a view based on that interpretation. Basically, the more memory, the better your ClearCase performance, but you can also improve performance by increasing the size of the view server cache.

33.1 Obtaining View Cache Information

To examine a view's cache information, use **cleartool getcache**. The view server prints information about the view cache sizes and hit rates. You can use this information in several ways:

- To determine whether to increase the view cache
- To check the results after changing the view cache
- To analyze other view server processes

Example:

cleartool getcache –view jo_main

```
Lookup cache:    29% full,   1121 entries ( 56.8K),  15832 requests,  75% hits
Readdir cache:    4% full,     24 entries ( 36.5K),   4159 requests,  83% hits
Fstat cache:     31% full,    281 entries (105.1K),  55164 requests, 100% hits
Object cache:    26% full,   1281 entries (176.6K),  40626 requests,  72% hits
Total memory used for view caches: 375.0Kbytes
The current view server cache limits are:
Lookup cache:             201312 bytes
Readdir cache:            838860 bytes
Fstat cache:              352296 bytes
Object cache:             704592 bytes
Total cache size limit:  2097152 bytes
```

The view cache includes these types of subcaches:

- **Lookup**. Stores data used to accelerate the mapping of names to objects in the view.
- **Read Directory.** Stores data used to accelerate read-directory operations (for example, **ls** or **dir**) by the view.
- **File Stats.** Stores data on file attributes used by the view.
- **Object.** Stores data pertaining to objects used by the view.

To find the size of the total view cache, sum the sizes of these components.

Analyzing the Output

getcache provides this information:

- Cache type
- Percentage of the view cache being used
- Number of entries in cache
- Number of requests made
- Percentage of cache hit rates
- Amount of view cache memory being used

Here are some suggestions for analyzing this information:

- Check the percentage of view cache being used. This value is very important.
  - If the hit rate is 90% or less and the view cache is 100% full, the view cache may be too small. The combination of a hit rate greater than 90% and view cache that is 100% full indicates that the cache size is about right.
  - If you are at less than 50% on every portion, your cache is probably too large.
- Check this value more than once to be sure you are not seeing a transitory value, such as for a newly started or restarted view server.
- If you decide to increase the view cache size for a view, use the procedure described in *Reconfiguring a View*.

33.2 Reconfiguring a View

Use **setcache** to reconfigure a **view_server**'s cache to a new size. The ratio of memory allotted to each subcache is fixed. When specifying a cache size, keep the following guidelines in mind:

- The value cannot be smaller than 50 KB for 32-bit platforms or 100 KB for 64-bit platforms.
- Do not specify a value larger than the amount of physical memory on the server host that you want to dedicate to this view.
- Values greater than approximately 4 MB do not help much in most cases.
- Verify your changes by checking the hit rates and utilization percentages periodically to see whether they have improved.

To specify a new cache size for a single view, use the **setcache –view** command:

cleartool setcache –view –cachesize *nnn view-tag*

Note that you must be the privileged user or the view owner to use this command. See the **setcache** reference page for details.

Improving Client Host Performance 34

This chapter presents techniques for improving ClearCase performance on the client host.

Performance of a ClearCase client host can be adjusted at the client program level, at the *view_server*, and/or at the MVFS level.

34.1 Increasing System Resources

ClearCase client software makes demands on host memory, CPU, and disk subsystem performance that are comparable to the demands made by similar workstation applications. Any computer configured to support such applications will deliver good performance of the majority of ClearCase client tasks.

At a minimum, ClearCase requires 64 MB memory and 35 MB hard disk space on a client host.

Because ClearCase is a distributed application, it also requires good performance from the client host's network interface and the network to which it is connected. Poor network performance will have a negative impact on the performance of even a well-configured ClearCase client host.

Additional memory may be appropriate for some client hosts. For example, a client host that is expected to do additional work, such as hosting distributed builds or shared views, will need additional memory and perhaps a larger, higher-performance disk subsystem. And any client that must run other workstation applications while it is also running ClearCase may need to have its resources adjusted to adequately support simultaneous use of both.

34.2 Creating Remote Storage Pools for UNIX VOBs

By default, all of a VOB's file-system data is stored in the storage pools created by **mkvob**. These pools are located within the VOB storage directory. If a VOB host becomes I/O-bound, it may be due to high storage pool traffic, caused by either too many users or too many files.

If your VOB is hosted on a UNIX server, you can use UNIX symbolic links to add remote storage pools that supplement (or replace) the default pools. This strategy effectively enables a VOB to outgrow its storage directory's disk partition. Remote pools may be located on other disks on the same ClearCase host, on other ClearCase hosts, or on any UNIX host, whether or not it has ClearCase installed, that is accessible through NFS.

In some situations, remote storage pools can improve performance:

- If a particular view is being used heavily (a UCM integration view, for example), build performance may improve if the cleartext and derived object storage pools involved in the builds are located on the same host as the view storage directory.
- Faster access to any storage pool may be achieved if it is located on a server host with a very fast file system.

Caution on Remote Source Pools

Keep source pools local, that is, within the VOB storage directory. This strategy optimizes data integrity. When a single disk partition contains all of the VOB's essential data, it simplifies backup/restore procedures. This concern typically overrides performance considerations, because losing a source pool means that developers must re-create the lost versions.

If source pool access produces a significant processor or I/O bottleneck, you may temporarily move some elements into source pools on different hosts.

See *Creating Additional Storage Pools for UNIX VOBs* on page 209 for a procedure.

34.3 Examining and Adjusting MVFS Cache Size

Not all client hosts have the MVFS installed. Those that do may be able to benefit from MVFS cache tuning. Use the **getcache** command to print information about your MVFS cache. For example:

cleartool getcache –mvfs

```
Mnodes: (active/max)     1791/8192 (21.863%)
Mnode freelist:          1701/1800 (94.500%)
Cltxt freelist:          737/1800 (40.944%)

DNC:     Files:          848/1600 (53.000%)
        Directories:    185/400 (46.250%)
        ENOENT:         827/1600 (51.688%)

RPC handles:             4/10 (40.000%)
```

NOTE: The statistics presented above are in the form *current-number-on-list/list-capacity* and (*percentage-of-list-capacity*).

```
Attribute cache miss summary (for tuning suggestions, see the
documentation for administering ClearCase):
Attribute cache total misses:              49609        (100.00%)
Close-to-open (view pvt) misses:           18964        ( 38.23%)
Generation (parallel build) misses:         1179        (  2.38%)
Cache timeout misses:                        234        (  0.47%)
Cache fill (new file) misses:                  0        (  0.00%)
Event time (vob/view mod) misses:          29232        ( 58.92%)
```

You may want to change ClearCase MVFS cache sizes to improve performance, if your host performs builds that involve a large (greater than 400) number of files.

Remember, the size of physical memory is the key factor, but caching changes may help.

To resize the MVFS caches, do one of the following:

- Use **cleartool setcache –mvfs** to test different MVFS cache sizes. This method allows you to evaluate results before making a permanent change. See *Real-Time Updating of MVFS Cache Sizes* on page 444 for details.
- On a UNIX host, set the **mvfs_largeinit** option. This raises a number of default values and can usually be done with a single edit and recompile of MVFS. It is the preferred method for a permanent change. This option is set to a value of 1 if available main memory exceeds 24 MB. For details, see *Adjusting the MVFS Memory Initialization Factor* on page 445.

- On a Windows NT host, increase the scaling factor on the **MVFS Performance** tab in the ClearCase program in Control Panel. This raises a number of default values and is the preferred method for a permanent change. See the online help for the **MVFS Performance** tab as well as *Adjusting the MVFS Memory Initialization Factor* on page 445 for more details.

- Specifically override the values for some or all of the specific caching components. (On UNIX systems, you must recompile the MVFS after changing any of these values.) This method provides somewhat finer control, but the suggested values are generally close to those used by **mvfs_largeinit**. On Windows NT computers, this value is represented in the ClearCase Control Panel as the **scaling factor** on the MVFS Performance tab. See *Setting Individual Caching Parameters on UNIX* on page 447.

The larger caches add about 500 KB to the size of kernel (200 KB nonpageable, 300 KB pageable) memory for each increment of the **mvfs_largeinit** value or scaling factor. Enlarging the MVFS caches reduces the amount of memory available to applications but provides better performance when the working set of objects in a build or command exceeds the default cache allocations. Increase the value gradually and check the cache utilization with **cleartool getcache –mvfs**. If you change your MVFS configuration, also consider reconfiguring each view that is used to access ClearCase data on that host, increasing its **view_server** cache size. See *Reconfiguring a View* on page 437.

Table 9 describes each tunable caching parameter with recommendations for setting its size. Also shown is the relationship among the fields in the **getcache** output, the **setcache** options, and the names of the caching parameters.

Table 9 MVFS Cache Information (Part 1 of 3)

getcache Output Field Name	**Description**	**Make Adjustments with:**	
		setcache Option	**Caching Parameter (UNIX)** **Control Panel Value (Windows NT)**
`Mnodes`	An *mnode* is a data structure used in the MVFS to keep track of a file or directory. This list grows dynamically.	(N/A)	**mvfs_mnmax** on UNIX, (N/A on Windows NT)
`Mnode freelist`	Mnodes on the free list were used (open or had statistics fetched from them) recently but are currently idle. By keeping more mnodes on a free list, the time needed to reopen the associated file is reduced.	**–vobfree**	**mvfs_vobfreemax (UNIX)** **Control Panel value (Windows NT)**

Table 9 MVFS Cache Information (Part 2 of 3)

getcache Output Field Name	Description	Make Adjustments with:	
		setcache Option	Caching Parameter (UNIX) Control Panel Value (Windows NT)
`Cltxt freelist`	Mnodes on the free list may also contain pointers to underlying storage pool file objects. The cleartext free list is the collection of these file objects. Keeping these object pointers cached in the mnode speeds the reopening of an MVFS-resident file, but consumes additional memory resources on the client machine. (On a Windows NT computer, these resources include open file descriptors, for example, on a LAN Manager connection if the storage is on a remote computer.)	**–cvpfree**	**mvfs_cvpfreemax** Mnodes to keep for cleartext free list (Do not change this value except to fix error #2009 in the event log: `server could not expand a table because it reached the maximum size`.)

Table 9 MVFS Cache Information (Part 3 of 3)

getcache Output Field Name	Description	Make Adjustments with:	
		setcache Option	Caching Parameter (UNIX) Control Panel Value (Windows NT)
DNC Files	These caches perform name translations to files, directories, or nonexistent names. If a file name is looked up but not in the cache, the MVFS must perform a remote procedure call (RPC) to the **view_server** to translate the name to a file or directory (or receive an error that the name is not found). When a name is found in the cache, the MVFS saves time by avoiding the RPC. The capacity of each of these categories can be adjusted independently.	**–regdnc**	**mvfs_dncregmax** Entries for files
DNC Directories		**–dirdnc**	**mvfs_dncdirmax** Entries for directories
DNC ENOENT		**–noentdnc**	**mvfs_dncnoentmax** Entries for non-existent names
RPC handles	One RPC handle is used for each RPC in progress from the MVFS to a **view_server**. If the cache is 100% full and you perform large builds with many simultaneously active processes, you may want to increase this cache size to reduce the time taken to perform an RPC. After an RPC handle is added to this cache, it stays on the list until ClearCase is stopped on that host. If no RPC handles are available in the cache, a new one is created on demand. If the cache is not full, the new RPC handle is returned to the cache when the RPC completes. Otherwise, it is destroyed when the RPC completes.	**–rpchandles**	**mvfs_client_cache_size** RPC handles

Real-Time Updating of MVFS Cache Sizes

To perform real-time updating of the MVFS cache sizes, use **cleartool setcache –mvfs**. Changes made using this command change the cache sizes without requiring you to shut down and

restart your computer. Real-time updating is useful for testing different configurations until you find the one that is best for you.

- On UNIX computers, these changes are not persistent until you edit the values for each significant caching component, recompile the MVFS (described in subsequent platform-specific sections), and reboot the computer.
- On Windows NT computers, these changes are not persistent until you update the values on the **MVFS Performance** tab of the **ClearCase** Control Panel, and then shut down and restart your computer.

The entries in the sample output show the corresponding **setcache** options that can be used to make adjustments to these caches.

You probably also need to use the **mvfsstat** and **mvfstime** commands to determine the effectiveness of your cache before manipulating its size. A full cache is not necessarily ineffective if its hit rate is high enough. We recommend maintaining a hit rate of 85% or better.

NOTE: **setcache** can sometimes fail with a "Device busy" error. Such errors are usually transient and indicate that some other process is also trying to use the MVFS.

Example:

cleartool getcache –mvfs

```
                                                    (setcache adjustment option)
Mnodes: (active/max)     1791/8192 (21.863%)        (grows automatically)
Mnode freelist:          1701/1800 (94.500%)        (adjust with –vobfree option)
Cltxt freelist:          737/1800 (40.944%)         (adjust with –cvpfree option)

DNC:     Files:          848/1600 (53.000%)         (adjust with –regdnc option)
        Directories:    185/400 (46.250%)           (adjust with –dirdnc option)
        ENOENT:         827/1600 (51.688%)          (adjust with –noentdnc option)

RPC handles:             4/10 (40.000%)             (adjust with –rpchandles option)
.
.
.
```

Adjusting the MVFS Memory Initialization Factor

You can specify how much memory the MVFS uses for various caches by changing the MVFS scaling factor.

On UNIX platforms, this scaling factor is the value of **mvfs_largeinit**. By default, it is set to 1 if the amount of available main memory exceeds 24 MB. The way in which **mvfs_largeinit** is set is platform specific. See the *ClearCase and MultiSite Release Notes*. For greater control over your cache sizes, you can set the individual caching parameters as described in *Setting Individual Caching Parameters on UNIX* on page 447.

On Windows NT platforms, the scaling factor is set to 1 by default. To change the scaling factor, start the **ClearCase** Control Panel. On the **MVFS Performance** tab, set the scaling factor value in the **Scaling factor to initialize MVFS with more memory for better performance** box. Click **OK**. Changes take effect after you shut down and restart your computer. For finer control over the individual cache sizes, you can change some of the individual values. See *Setting Individual Cache Sizes on Windows NT* on page 448.

Changing the value of **mvfs_largeinit** or the Scaling Factor scales all of the MVFS cache sizes proportionally as shown in Table 10.

Table 10 Cache Sizes Corresponding to mvfs_largeinit or Scaling Factor Settings

MVFS Cache Name	**Cache Size for Specified Value of mvfs_largeinit (UNIX) or Scaling Factor (Windows NT)**			
	0	**1**	**2**	***N* > 2**
DNC File Cache	800	1600	2400	800(*N*+1)
DNC Directory Cache	200	400	600	200(*N*+1)
DNC ENOENT Cache	800	1600	2400	800(*N*+1)
Cleartext File Cache (UNIX)	900	1800	2700	900(*N*+1)
Cleartext File Cache (NT)	900	1800	1800	1800
RPC Handle Cache	5	10	15	5(*N*+1)
Mnode Freelist Cache	900	1800	2700	900(*N*+1)

Setting Individual Caching Parameters on UNIX

Platform-Specific instructions for changing caching parameters on all supported UNIX systems are provided in the *ClearCase and MultiSite Release Notes*.

The following list identifies all significant caching components and provides the default values as loaded at system startup. You can bypass this level of detail by choosing the **mvfs_largeinit** option as described in *Adjusting the MVFS Memory Initialization Factor* on page 445.

- Cache for the systemwide maximum number of MVFS-internal identifiers (*mnodes*) is set with the **mvfs_mnmax** parameter at up to 4,096 MVFS objects. This value must not be changed. The cache grows dynamically if more is required.
- Cache for the maximum number of unused objects to cache (at 400 bytes/object) is set with the **mvfs_vobfreemax** parameter with a default value of 900.
- Cache for the maximum number of unused cleartext files to cache differs by system. The value is set by **mvfs_cvpfreemax**. Individual system notes provide detailed information.
- Caches for MVFS objects are set with these parameters:
  - The number of regular file names to cache (at 100 bytes/entry) is set by **mvfs_dncregmax**. The default value is 0, which means that the size of the cache is determined by the value of **mvfs_largeinit**.
  - The number of directory names to cache (at 100 bytes/entry) is set by **mvfs_dncdirmax**. The default value is 0, which means that the size of the cache is determined by the value of **mvfs_largeinit**.
  - The number of names that produce name-not-found returns (at 100 bytes/entry) is set by **mvfs_dncnoentmax**. The default value is 0, which means that the size of the cache is determined by the value of **mvfs_largeinit**.

For the individual cache sizes associated with particular values of **mvfs_largeinit**, see *Adjusting the MVFS Memory Initialization Factor* on page 445.

After making these changes, you must recompile and/or reload MVFS. You may also have to reboot the computer.

Setting Individual Cache Sizes on Windows NT

Use the **ClearCase** Control Panel to modify individual cache sizes. On the **MVFS Performance** tab, select the **Override** check box for a particular value. Enter a new value for the parameter. When you have finished modifying fields, click **OK**.

Each mnode in the mnode freelist cache uses about 400 bytes. Each entry in the DNC files, directories, and ENOENT caches uses about 100 bytes.

After changing cache sizes, you must shut down and restart your computer to make your changes take effect.

Minimizing Attribute Cache Misses

The sections that follow this sample output describe each cache-miss category and the ways to reduce the miss count.

cleartool getcache –mvfs

```
.
.
.
Attribute cache miss summary (for tuning suggestions, see the
documentation for administering ClearCase):
Attribute cache total misses:              49609        (100.00%)
Close-to-open (view pvt) misses:           18964        ( 38.23%)
Generation (parallel build) misses:         1179        (  2.38%)
Cache timeout misses:                        234        (  0.47%)
Cache fill (new file) misses:                  0        (  0.00%)
Event time (vob/view mod) misses:          29232        ( 58.92%)
```

Attribute Cache Total Misses

Total misses is the sum of all the misses entries. This total is reduced by reducing the other categories of misses entries described below.

Close-to-Open Misses

Close-to-open misses occur when an open view-private file is reopened by another process. The MVFS rechecks with the **view_server** when a view-private file is reopened to refresh its cached attributes in case the file was modified by another MVFS client.

This behavior can be disabled on a per-computer (not a per-view) basis. If you are absolutely sure that no other client on the network is updating view-private files in any views you are using from an MVFS client, you may want to disable this behavior to reduce these cache misses.

Use **mvfscache** to change this setting.

Generation Misses on UNIX

Generation misses occur during a distributed parallel build. (Distributed and parallel builds are not supported on Windows NT.) If you are doing parallel builds, you cannot avoid these misses.

Cache Timeout Misses

Cache timeout misses occur when a file's attributes have not been verified recently. The MVFS periodically rechecks a file's attributes to ensure that it does not cache stale copies of the attributes.

These misses cannot be completely eliminated. However, the time-out period for the attributes may be adjusted. The cache timeout varies, depending on how recently a file or directory was modified; the more recently the file was modified, the shorter the time-out period. The value for this initial time-out period is constrained to lie within the minimum and maximum attribute-cache lifetimes as specified at VOB **mount** time through the **acregmin**/**acregmax** (for regular files) and **acdirmin**/**acdirmax** (for directories) parameters. These are the default values:

- **acregmin**: 3 seconds
- **acregmax**: 60 seconds
- **acdirmin**: 30 seconds
- **acdirmax**: 60 seconds

If you consider changing these values, be aware that a longer timeout means that this client may not notice other clients' updates to the VOB or view for longer periods.

To change these values, specify a mount option for the VOB in one of the following ways:

- At mount time. For example:

  cleartool mount –options acregmin=30 *vob-tag*

  This is a privileged operation. You must be the privileged user to do this.

- In the registry (either at VOB creation with **mkvob** or later using **mktag**). For example:

  cleartool mktag –vob –tag *vob-tag* **–replace –options** *mount-options* . . .

Cache Fill Misses

Cache fill misses occur when a file's attributes are not in the cache. If the percentage of these is high (above 20%), your cache may be too small for your working set. Consider increasing the number of mnodes on the free list by changing the **mvfs_vobfreemax** parameter as described in *Setting Individual Caching Parameters on UNIX* on page 447 and *Setting Individual Cache Sizes on Windows NT* on page 448.

Event Time Misses

Event time misses occur when a cached name-to-object translation requires revalidation of the attributes on the resulting object (for example, the name cache has "foo" mapped to a particular file object, but the attribute cache on that object needs to be refreshed because of a timeout or a parallel-build-induced generation miss).

These misses cannot be completely eliminated, but as with *Cache Timeout Misses* they can be reduced by adjusting minimum/maximum cache lifetimes.

Troubleshooting

35 Determining a Data Container's Location

This chapter demonstrates how to determine the actual storage locations of *MVFS files*—those accessed through VOB directories in dynamic views. Both standard operating system utilities and the ClearCase **mvfsstorage** utility are used. These tools can help diagnose problems in accessing ClearCase files.

35.1 Scenario

This chapter focuses on three files within VOB directory **\monet\src**, as seen through view **allison_vu**:

- Element **cmd.c** has element type **text_file**, and is currently checked out.
- Element **monet.icon** has element type *file*, and is not currently checked out.
- File **ralph_msg** is a view-private file, created by saving an electronic mail message to disk.

NOTE: On a UNIX computer, the VOB-tag would probably include a standard VOB mount point. For example, if this mount-point was **/vobs**, then the directory would be **/vobs/monet/src**.

35.2 Determining the ClearCase Status of Files

The **describe** command verifies that the three files are as described:

cleartool describe cmd.c monet.icon ralph_msg

```
version "cmd.c@@\main\CHECKEDOUT" from \main\6 (reserved)
  checked out 03-Feb-93.20:40:30 by (allison.mon@phobos)
  by view: allison_vu ("phobos:d:\users\people\arb\vw_store\arb.vws")
  element type: text_file

version "monet.icon@@\main\1"
  created 03-Feb-93.20:17:04 by (allison.mon@phobos)
  element type: file

View private file "ralph_msg"
  Modified: Wednesday 02/03/93 21:39:49
```

35.3 Determining the Full UNIX Pathnames of Files

The standard **ls**(1) and **pwd**(1) commands show the full pathnames of the files on UNIX:

ls –l 'pwd'/cmd.c

```
-rw-rw-r--   1 allison  mon     211 Feb  2 12:03 /proj/monet/src/cmd.c
```

ls –l 'pwd'/monet.icon

```
-r--r--r--   1 allison  mon     266 Feb  3 20:17 /proj/monet/src/monet.icon
```

ls –l 'pwd'/ralph_msg

```
-rw-rw-r--   1 allison  mon     852 Feb  3 20:40 /proj/monet/src/ralph_msg
```

35.4 Where Is the VOB?

The **describe** command shows you the path to the VOB root over the network and also relative to the host on which the VOB storage resides:

cleartool describe vob:\monet\src

```
versioned object base "\monet"
  created 01-Feb-93.17:35:03 by (vobadm.vobadm@sol)
  VOB storage host:pathname "sol:c:\vbstore\monet.vbs"
  VOB storage global pathname "\\sol_vbstore\monet.vbs"
  VOB ownership:
    owner vobadm
    group vobadm
```

You can also get this information from the All VOBs node of the ClearCase Administration Console, which lists all VOB-tags registered in the local host's region on the local host's registry server. The **Taskpad** and **Detail** views show the host and global path of the VOB storage directory along with other information about each VOB. From this listing you can navigate to the VOB storage node for the VOB in the ClearCase Administration Console, where you can manage VOB storage.

35.5 Where Is the View?

The **pwv** (print working view) and **lsview** (list view-tag) commands show the location of the view storage area:

cleartool pwv

```
Working directory view: allison_vu
Set view: allison_vu
```

cleartool lsview allison_vu

```
* allison_vu \\phobos_users\people\arb\vw_store\arb.vws
```

NOTE: On UNIX computers, you may need to use the **mount** command to identify the host on which the view storage area resides. (Which host contains directory **/net/phobos**?).

The All Views node of the ClearCase Administration Console lists all view-tags registered in the local host's region on the local host's registry server. The **Taskpad** and **Detail** views show the host and global path of the view storage directory along with other information about each view. From this listing you can navigate to the view storage node for the view in the ClearCase Administration Console, where you can manage view storage.

35.6 Where Are the Individual Files?

The *data containers* for all MVFS files are logically stored within a VOB or view storage area, as shown in Table 11.

Table 11 Storage Locations of MVFS Files

Kind of File	Storage Location
Version (checked-in)	VOB *source storage pool* (**s** subdirectory of the VOB storage directory) and perhaps VOB *cleartext* storage pool (**c** subdirectory of the VOB storage directory)
Checked-out version	View-private data storage (**.s** subdirectory of the view storage directory)
Unshared or nonshareable derived object	View-private data storage (**.s** subdirectory of the view storage directory)
Shared derived object	VOB *derived object storage pool* (**d** subdirectory of the VOB storage directory)
View-private file	View-private data storage (**.s** subdirectory of the view storage directory)

The following sections show how the **mvfsstorage** utility indicates the exact storage location of an MVFS file.

NOTE: On UNIX computers, this utility is located in directory *ccase-home-dir***/etc**. If this directory is not in your search path (and it usually isn't), run **mvfsstorage** using its full pathname.

Locating a Checked-Out Version

mvfsstorage shows the location in view-private data storage of the checked-out version of *text_file* element **cmd.c**:

mvfsstorage cmd.c

```
\\phobos\vw_store\arb.vws\.s\00050\8000000B.00B0.cmd.c
```

Locating a Checked-In Version's Cleartext Container

For a checked-in version of an element that uses a single data container to store all its versions, **mvfsstorage** shows the location of the cleartext data container into which the type manager extracts the version:

mvfsstorage cmd.c@@\main\1
```
\\sol\vobstore\monet.vbs\c\cdft\28\32\8a1a9a50010e11cca2ca080069021822
```

mvfsstorage cmd.c@@\main\2
```
\\sol\vobstore\monet.vbs\c\cdft\3a\33\8e4a9a54010e11cca2ca080069021822
```

Locating a Checked-In Version's Source Container

For a checked-in version of an element that uses a separate data container for each version, **mvfsstorage** shows the location of the data container in the source pool:

mvfsstorage monet.icon
```
\\sol\vobstore\monet.vbs\s\sdft\26\4\474fa2f4021e11cca42f0800690605d8
```

ClearCase does not maintain cleartext versions of elements that use separate data containers for each version. For these kinds of elements, the cleartext of a version is stored in its source pool data container and programs access the data container in the source pool directly.

Locating a View-Private File

Like a checked-out version, a view-private file is located in a view's private data storage:

mvfsstorage ralph_msg
```
\\sol\view_store\arb.vws\.s\00050\8000000C.00BD.ralph_msg
```

Issues with Nonlocal UNIX Storage

On UNIX computers, VOB storage pools can be located outside the VOB storage directory itself; likewise, a view's private storage area can be located outside the view storage directory.

If **mvfsstorage** indicates that a data container is in a nondefault VOB storage pool, use the **lspool** command to determine the location of the pool. The default pools are **sdft** (default source pool), **cdft** (default cleartext pool), and **ddft** (default derived object pool). For example:

*ccase-home-dir***/etc/mvfsstorage hello.h**
```
/vobstore/monet.vbs/c/clrtxt.1/36/f/6b6ed22b08da11cca0ef0800690605d8
```

cleartool lspool –l clrtxt.1
```
pool "clrtxt.1"
 08-Feb-93.10:25:46 by (vobadm.vobadm@starfield)
  "nonlocal cleartext storage for monet VOB"
  kind: cleartext pool
  pool storage remote host:path "sol:/netwide/public/ccase_pools/clrtxt.1"
  pool storage local pathname "/vobstore/monet.vbs/c/clrtxt.1"
  maximum size: 0  reclaim size: 0  age: 96
```

clrtxt.1 is a nondefault cleartext pool.

Use UNIX **ls** to determine whether a view's private storage area (subdirectory **.s**) is nonlocal:

ls –ld ~jones/view_store/temp.vws/.s
```
lrwxrwxr-x   1 jones    dvt        34 Feb 17 17:06
/usr/people/jones/view_store/temp.vws/.s -> /public/jones_temp
```

Links and Directories on UNIX

On UNIX computers, mvfsstorage deals with VPOB and file-system link and directory objects as follows:

- For a link, **mvfsstorage** indicates the storage location of the object to which the link points. This applies to all links: view-private hard links and symbolic links, VOB hard links, and VOB symbolic links.
- A view-private directory does not have a data container; nor does a directory element. In both cases, **mvfsstorage** echoes the directory pathname.

36 Repairing VOB and View Storage Directory ACLs on Windows NT

We recommend that you use NTFS-formatted disks for holding VOB and view storage directories. 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.

36.1 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_adm:(CI)(special access:)             (VOB owner)
        STANDARD_RIGHTS_ALL
        DELETE
        READ_CONTROL
        WRITE_DAC
        WRITE_OWNER
        SYNCHRONIZE
        STANDARD_RIGHTS_REQUIRED
        FILE_GENERIC_READ
        FILE_GENERIC_WRITE
        FILE_GENERIC_EXECUTE
        FILE_READ_DATA
        FILE_WRITE_DATA
        FILE_APPEND_DATA
        FILE_READ_EA
        FILE_WRITE_EA
        FILE_EXECUTE
        FILE_READ_ATTRIBUTES
        FILE_WRITE_ATTRIBUTES

NT_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_adm:(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
```

36.2 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 465.

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 465.

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

36.3 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.

36.4 Fixing Protection Problems

The following sections describe how to fix the protection problems described in *Causes of Protection Problems* on page 461.

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*

37 Preventing Accidental Deletion of Data by crontab Entries

This chapter describes changes made during ClearCase installation to ensure that **crontab**(1) scripts do not accidentally delete ClearCase data. In addition, we describe situations in which you may need to take measures to prevent such accidents from occurring.

37.1 Preventing Recursive Traversal of the Root Directory

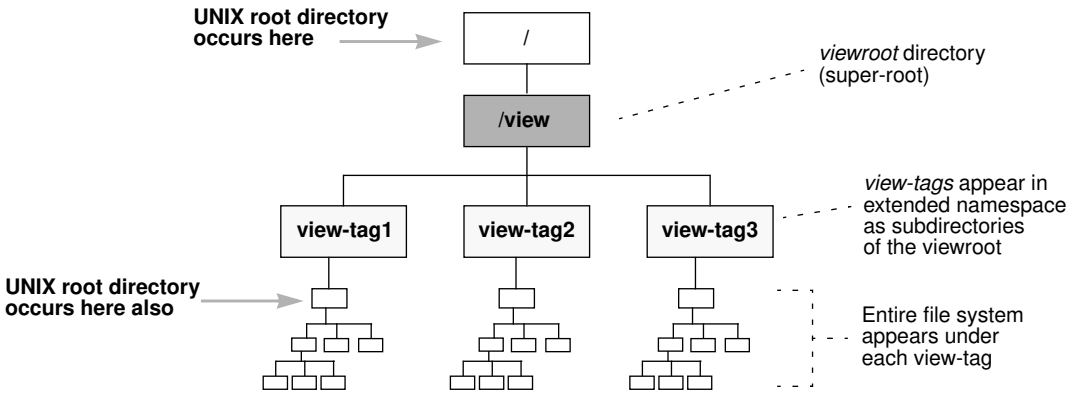
When the ClearCase MVFS is active on a client host, a **/view** directory, the *viewroot*, is created. This directory functions as a mount point for the MVFS namespace (see Figure 44). The **/view** directory causes the root directory to contain itself recursively, which makes certain recursive pathnames valid. For example:

```
/view/alpha/view/beta/view/alpha/view/gamma/usr/src/lib
```

As a result, commands that traverse the entire directory tree, starting at the UNIX root directory (/), loop infinitely. In particular, many UNIX systems configure the *root* user to have a daily **crontab**(1) script that performs a cleanup on the file-system tree. If the script uses a find / command, it runs the risk of looping infinitely.

NOTE: This situation applies equally to commands and programs executed by any user, either interactively or through a script.

Figure 44 Directory as a Super-Root



crontab Modification During ClearCase Installation

The ClearCase installation script, **install_release**, analyzes the **crontab** file of the *root* user on a client host. It modifies entries in this file to prevent the recursive traversal problem (or displays a message warning that it cannot perform the modification).

After installation, verify the correctness of the **install_release** changes. In addition, modify the **crontab** entries of other users, according to the instructions in *Modifying a crontab Entry*.

Modifying a crontab Entry

Use the following procedure to analyze and, if necessary, modify all of a host's **crontab** entries.

1. **Analyze the crontab entries.** Determine which entries will encounter the recursion problem:

```
% su
Password: <enter root password>
# grep "find /"  /usr/spool/cron/crontabs/*
/usr/spool/cron/crontabs/root:15 3 * * * find / –name .nfs\*
     –mtime +7 –exec rm –f {} \; –o –fstype nfs –prune
```

In the example above, long lines are broken for readability. In actual **crontab** files, each entry must be contained on a single physical text line.

2. **Revise crontab entries.** You must modify each **crontab** file in which an offending entry was found. For example:

# **su –** *username*	*(switch user identity)*
# **crontab –l > /tmp/C**	*(create temporary file with that user's crontab entries)*
# **vi /tmp/C**	*(modify those entries)*
# **crontab < /tmp/C**	*(configure the modified entries)*

During the edit session, change the command containing **find /** by adding the host-appropriate options. Detailed information on platform-specific options is in the online ClearCase platform-specific guide. It is also a good idea to add comments to the **crontab** files warning others not to add entries that cause recursive traversal problems.

Index

A

access control
See also permissions
about 23
algorithm used 29
ClearCase permissions 6
executables, running in MVFS 61
groups and group lists 4
in mixed environment 72
locking VOBs 229
locks 34
protection mode scheme 26
protection problems, fixing (procedure) 465
storage directories 41
to VOBs for nongroup members 127
views and contents 35
VOBs and contents 30

access to ClearCase
consistent user data 6
network region as access domain 358
non-ClearCase hosts 15
pathname heuristic 128
resolving registry problems 352
role of registry 341

ACLs
for storage directories 459
managing for scheduler 388

Administration Console 13
storage pools 117

administrative VOBs
checking and fixing global types 234
creating (procedure) 148
MultiSite implications 152
removing 155
restrictions 151
unavailable 152

albd_server process
how started 9

aliases, and global naming 362

Apache server, configuring 396

***attache-home-dir* directory** xxxiii

automount program
ClearCase use 20
non-ClearCase access 18

automounting
NFS clients 83

B

backing up views 307

backing up VOBs
about 167
checkvob run on incremental backup 267
finding pathname 172
incremental backups 175
locking the VOB 172
partial backups 173
remote storage pools (procedure) 176
semi-live vs. standard procedure 170
storage pools 170
summary of procedure 169
tools for 4
vob_snapshot and 170

backup tools, recommendations 167

block buffer cache
how used 433
statistics 433

build scripts
auditing system calls in 58

building software
on non-ClearCase host 17
views on non-ClearCase hosts 304
when VOBs are locked 170

C

caches
cleartext storage pools 114
MVFS, techniques for resizing 441
NFS clients, invalid 18

case-sensitivity
disabling case conversion on NFS clients 80
in mixed environment 73
MVFS and 59

ccase-home-dir **directory** xxxiii
CCFS (ClearCase File Server) 78
checkvob utility
broken hyperlinks 234
check/fix scenarios 262
detecting file protection problems 462
diagnostic uses of 245
force fix mode 252
log files 246
requirements for type manager 244
running on replicas 245
synchronizing database with storage pool 192
using -force -fix options 245
when to use 233
checkvob utility, sample runs
about 265
database newer than pools 265
database older than pools 266
incremental backup/restore 267
pool root check failure 267
ClearCase group
defining 50
setting up for NFS clients 84
clearexport_rcs command 132
clearexport_ssafe command
conversion example 138
clearlicense utility 334
cleartext pools
about 114
backing up 175
name of default 458
scrubbing 207
storage location of data containers 457
usage patterns and choice of location 209
client hosts 3
client/server processing 8
config specs
modifying for export view 304
configuration lookup
and view-private storage 295
on non-ClearCase host 17
conventions, typographical xxxiii
crontab scripts
deletion of ClearCase data by 467
modifying entries in 468

D

DACLs, about 459
data containers
DO 114
DO, removing 314
examining with checkvob 233
finding storage location 453
fixing inconsistencies in 243
missing 257
types of 113
data loss, VOB restored from backup 171
db_server process 9, 112
directories
finding location of 458
relocated, how cataloged in source VOB 275
relocated, how cataloged in target VOB 276
removing, effect on file elements 196
symbolic links to relocated 287
viewroot, view-tag and 343
VOB root 117
directory trees, overcoming disk partition restrictions 116
disk buffer cache, how used 432
disk space
conserving, and versions 209
conserving, and VOB storage pools 209
consumed by view storage, displaying 313
estimating for view storage 300
required for VOB host 120
semi-live backup 171
DO pools
about 114
backing up 174
corrupted containers 262
debris 261
incorrect permissions 256
missing containers 261
name of default 458
scrubbing 207
unreferenced containers 261
usage patterns and location 210
documentation
online help description xxxiv
domains
about 43, 70
access to network regions 358
configuring 43
creating (procedure) 46
moving VOBs in 217
security requirements 44
server process user and ClearCase group 50
setting up user accounts 47
user accounts across multiple 52
user accounts and groups 47
DOs (derived objects)
access control scheme 40
checkvob processing 251
data containers 114
data containers, removing 314
impact on view storage 295
restoring consistency of VOB database (procedure) 203
scrubbing, adjusting frequency 213

scrubbing, adjusting scope 214
sharing on non-ClearCase host 17
transferring to VOB storage (example) 315
unversioned, in relocated directories 272

E

elements
access control scheme 31
checkvob processing 251
directory, removing 196
moving to another VOB 271
moving to another VOB (illustrations) 272
relocating borderline 277
relocating, preliminary procedures 280
removing from VOBs 208
restoring removed 196
error logs 10
error logs, for servers 10
event records, scrubbing 207
export views
about 302
multihop configurations 304
restricting hosts for 305

F

FAT file system
converting to NTFS, security information 462
security of storage directories 459
feature levels 144
file descriptor table
modifying for VOB host 123
file elements
determining status of 453
editing permissions, and ClearCase protections 462
location of source data containers 113
source pool assignments 117
fix_prot utility 463
floating license scheme 333

G

global pathnames
how derived 128
remote storage pools 209
global types
about 147
changing mastership 162
changing protection 161
changing scope 164
checking and fixing 234
copying 163
creating 155
describing 158
how they work 147
listing 160
listing history 160
locking and unlocking 161
removing 166
renaming 164
group IDs, consistency on ClearCase hosts 6
group lists
in mixed environment 71
role in access control 4
groups
adjusting VOB identity information 127

H

Host Administration 13
storage pools 117
hosts
See also license server hosts; registry server hosts; VOB hosts
client 3
consistency of user IDs 6
non-ClearCase 3
release, renaming 22
remote administration 14
renaming (procedure) 373
server 3
verifying domain assignment 46
Windows NT, assigning to new network regions 365
-hosts map
use of and workarounds for 21
hyperlinks, checking 234

I

.identity subdirectory
contents of 112
IIS, configuring 397
importing data to VOBs
about 131
from PVCS (example) 133
from SourceSafe (example) 135
init command, ClearCase startup script 11
install_release script
crontab modification by 468

installing ClearCase, changing release area location 22
IP addresses, managing with DHCP 15

L

license server hosts
about 333
additional 335
in mixed environment 69
renaming 338
setting up 335
licenses
adding information to database (procedure) 335
expiration 334
licensing scheme 333
moving to another host 337
priorities 334
reports on user activity 334
verification check 333
links
absolute pathnames in 352
finding location of 458
unexported disk partitions 353
locking VOBs
locking objects in 34
procedure 173
techniques to reduce duration 169, 172
log files, checkvob 246
logon name, for NFS clients 82
lost+found directory
files in deleted directory 196
lsacl utility 464

M

mastership of global type, changing 162
memory
client host minimum 439
required for VOB host 120
Microsoft Internet Information Server, configuring 397
mount point of VOB, finding 454
moving VOBs
danger when moving database 112
different architecture (procedure) 225, 324
moving elements to other VOBs 271
replicas 218
restrictions 217
same architecture (procedure) 222
MultiSite
changing global type mastership 162
checkvob operations on replicas 245
interop-enabled mode 107
moving replicas 218
shared global types 157
MVFS (multiversion file system)
about 57
case-sensitivity 59
finding file storage location 453
limitations of 58
NFS configuration setup 79
performance 61
MVFS cache
adjustment techniques 441
cache-miss categories 448
changing size in real time 444
setting individual parameters 447
setting individual sizes 448
mvfs_larginit
effect on MVFS cache size 445
mvfsstorage utility 456

N

Netscape Enterprise Server, configuring 398
network
ClearCase components in 1
record of interfaces 20
Windows NT domains 43
network regions
about 7, 355
adding 362
administering 355
assigning Windows computers to 98
creating for mixed environment 98
establishing 361
example 363
multiple registry hosts as alternative 363
multiple, administration guidelines 368
multiple, registries in 358
registry hosts for, separate 367
remote storage pools 209
removing 368
NFS clients
case-sensitive file names 76
configuring for MFVS 79
disabling DOS file-sharing (procedure) 82
invalid cache problems 18
VOBs with linked storage pools 100
NFS file locking 19
non-ClearCase hosts
about 3
access to VOBs 15
creating VOB storage pools on 209
export views 302
NTFS file system
converting from FAT, security information 462

security of storage directories 459

O

object registries 342
online help, accessing xxxiv

P

pathnames
absolute, in links 352
determining full 454
recursive traversal 467
pathnames and symbolic links 60
performance
client hosts 439
disk I/O on VOB servers 432
MVFS file lookup 61
view_server cache size 435
VOB hosts, improving 431
VOB memory and storage required 120
permissions
See also access control
editing, and ClearCase protections 462
for executables in MVFS 61
how controlled 6
incorrect, on pools 256
pools
See storage pools
primary group
overriding (procedure) 24, 49
setting for user account (procedure) 48
verifying (procedure) 49
principal group
role in access control 4
private VOBs
activating and deactivating 346
process table
modifying for VOB host 123
processes
extraneous, on VOB host 431
properties of site, registering 351
protection mode 26
public VOBs
activating and deactivating 346
creating 126
PVCS, data conversion example 133

R

Region Synchronizer 99
registry
about 341
creating entries 349
default pathnames for storage directories 350
listing entries 347
moving 369
moving to active backup host (procedure) 369
site properties 351
registry server hosts
about 6
backup 3
backup, moving registry to 369
changing backup (procedure) 373
changing backup to primary 373
in mixed environment 69
moving registry to another (procedure) 371
multiple vs. network regions 363
renaming 372
separate, for new network region 367
spreading access load 353
VOB-tag password file location 125
release area, changing location 22
release hosts
about 2
renaming 22
relocate utility
about 271
sample operations (illustrations) 272
replicas
checkvob operations 245
moving 218
shared global types 157
restoring VOBs from backup
checkvob run from incremental backup 267
costs of semi-live backup 171
database snapshot 192
procedure 177
rules and guidelines 189
sample session 179
scenarios 187
synchronizing views and VOBs 200
without vob_restore (procedure) 194

S

scheduler
about 377
creating jobs 383

creating tasks 381
default job schedule 380
deleting jobs 387
deleting tasks 382
editing job properties 386
editing task definition 382
job notification mechanisms 385
managing jobs 383
managing tasks 381
running jobs 387
schedule types 384
viewing job properties 386
scrubbing
about 4
impact on cache hits 114
logical vs. physical 208
storage pools 207
view-private storage 314
VOB databases 207
Security Descriptor
effect of file-system conversion 462
semi-live backup
costs and benefits 171
how it works 170
server hosts 3
server process user
defining 50
setting up on NFS clients 84
server processes 9
servers, error logs 10
setGID and setUID
enabled for mounting 130
shutdown script, how invoked 11
site_prep program
establishing network regions 361
SMB server
See TAS SMB server
source pools
about 113
checkvob sample run 265
corrupted containers 260
debris 259
file element assignments 117
incorrect permissions 256
missing containers 257
name of default 458
recommended location 210
scrubbing 207
unreferenced containers 259
usage patterns and choice of location 209
SourceSafe, data conversion example 135
splitting VOBs
about 271
causes of common failures 282
cleanup 283
cleanup guidelines 285
element removal errors 283
handling unrelated errors 282
preliminary procedures 280
sample operations (illustrations) 272
startup script, how invoked 11
storage directories
ACLs, form of (example) 459
default pathnames in registry 350
global access to shared 357
native file-system permissions 41
retaining protection information of copied 461
security on NTFS vs. FAT 459
storage pools
See also cleartext pools; DO pools; source pools
about 113
adjusting scrubbing procedures 213
backing up 170
checkvob processing 252
cleartext, backing up 175
commands for 117
creating remote 130
creating remote (example) 210
default directories 115
distributed, implementation of 342
finding and fixing problems in 242
fixing root 267
host criteria for remote 209
inconsistent with VOB database 241
locating files in remote 457
misnamed 255
moving (example) 211
names of default 458
remote, about 209
remote, and client host 440
remote, backing up 176
remote, when not to use 368
scrubbing 207
setup mode in checkvob 253
symbolically linked, and NFS clients 100
synchronizing with VOB database 192
tools for 210
view and VOB, compared 295
storage registries
about 342
administration guidelines 352
guidelines for multiple 353
maintenance required 4
symbolic links
absolute pathnames in 60
access to remote storage 342
for moved elements 272
limitations 284

T

tag registries
- about 343
- creating and removing entries 349
- implementation in multiple-region network 359
- in multiple-region networks 358

TAS SMB server
- about 87
- configuring ClearCase to support 94
- configuring for ClearCase 90
- initial setup 89
- starting file service 93
- testing configuration 94
- upgrading from previous version 87

technical support xxxiv

text modes
- configuring for mixed environment 103

trust lists 43

type managers
- about 113
- checkvob requirements 244

type objects
- changing 163
- coordinating for multiple VOBs 131
- global 147
- locking 35

typographical conventions xxxiii

U

umask setting
- for NFS client 81
- for shared view 301

UNIX/Windows interoperation
- access control 72
- capabilities and constraints 66
- case-sensitive names 73
- ClearCase configurations 65
- file access mechanisms 77
- license and registry servers 69
- managing user accounts 69
- preparing UNIX hosts 98
- preparing UNIX VOBs and views 97
- when useful 65

user accounts
- in multiple domains 52
- managing in mixed environment 69

utility commands
- for storage pools 118

V

versions
- finding location of checked out 456
- finding storage location 455
- removing from VOB 209

view database
- about 5
- how implemented 294
- skew after VOB restoration 200

view hosts, preparing for mixed environment 98

view registry 342

view storage
- disk space, estimating required 300

view storage directories
- about 5
- creating 301
- requirements 300
- .view file, impact of moved view 322, 324

view_scrubber utility
- example 314

view_server process
- about 9
- cache size and performance 435
- location requirements 299
- performance of VOB host 432
- setting cache size 437

view-private files
- access control scheme 38
- finding storage location 455

view-private storage
- about 295
- builds on non-ClearCase hosts 17
- cleanup (procedure) 315
- dynamic view, moving (procedure) 327
- finding location of checked-out versions in 456
- location 300
- maintenance 313
- remote location 295

viewroot directory
- mounting 130
- recursive traversal of 467
- view-tag and 343

views
- about 5, 293
- access control scheme 35
- activating dynamic 346
- analyzing cache information 435
- backing up 307
- configuring text modes for mixed environment 103
- finding storage location 455
- limitations of symbolic links 284
- location, architectural constraints 299
- moving (procedure) 319
- moving to host with same architecture (procedure) 319

removing permanently 329
removing VOB references to 330
restoring consistency of DO state (procedure) 203
restoring from backup (procedure) 309
restoring to service 329
setting up shared 301
setting up single 299
taking out of service 329
umask setting 301
view-tags
about 341
characteristics of 343
creating for new network region 366
first in network region 368
importing in mixed environment 99
re-creating incorrect 99
renaming vs. removing 353
shared view 301
VOB database
about 112
contents of 111
danger when moving 112
inconsistent with storage pools 241
location of 112
proportion of cache size 433
scope of checkvob updating 243
scrubbing, logical vs. physical 208
scrubbing 207
skew after VOB restoration 200
synchronizing with storage pools 192
updating schema 144
VOB file system, mounting 130
VOB hosts
cache size to VOB database 433
criteria for selecting 120
export views on 304
improving performance 431
kernel resource adjustments 123
modifying for ClearCase access 123
moving VOBs in domains 217
non-ClearCase access 302
preparing for mixed environment 98
remote 127
VOB registry 342
VOB root directory
about 117
caching and storage 60
VOB storage
maintenance trade-offs 205
root directory as 60
vob_restore
about 177
database snapshot 192
restoration scenarios 187
sample session 179
vob_server process 9, 115
vobrpc_server process 9
VOBs
about 4
access control scheme 30
access, strategies for ensuring 128
activating 346
administration tasks 4
configuring text modes for mixed environment 103
coordinating type objects 131
creating (procedure) 124
creating on remote host 127
finding mount point 454
identity information adjustments 127
importing data 131
locking objects in 34
locking 229
public and private, activating 346
public, creating 126
reformatting 144
removing elements from 208
removing permanently 231
removing temporarily (procedure) 229
removing view references 330
restoring to service (procedure) 230
setting up 119
size and distribution trade-offs 121
structure of 111
upgrading to new release 143
VOB-tags
about 341
characteristics of 343
creating for new network region 366
creating 125
first in network region 368
importing in mixed environment 99
multiple-region network 368
password file location 125
re-creating incorrect 99
renaming vs. removing 353
repairing 101

W

Web servers
configuration checklist 393
configuration procedures 396
Windows NT hosts, assigning to new network region 365
winkin, impact on view-private storage 295