Rational® ClearCase®
Rational® ClearCase® LT

Administrator's Guide

VERSION: 2003.06.00 AND LATER

PART NUMBER: 800-026162-000

UNIX/WINDOWS EDITION

Rational®
the software development company

support@rational.com
http://www.rational.com

Legal Notices

Part Number: 800-026162-000
Version Number: 2003.06.00 and later

Third Party Notices, Code, Licenses, and Acknowledgements

Contents

Tables

Figures

Preface

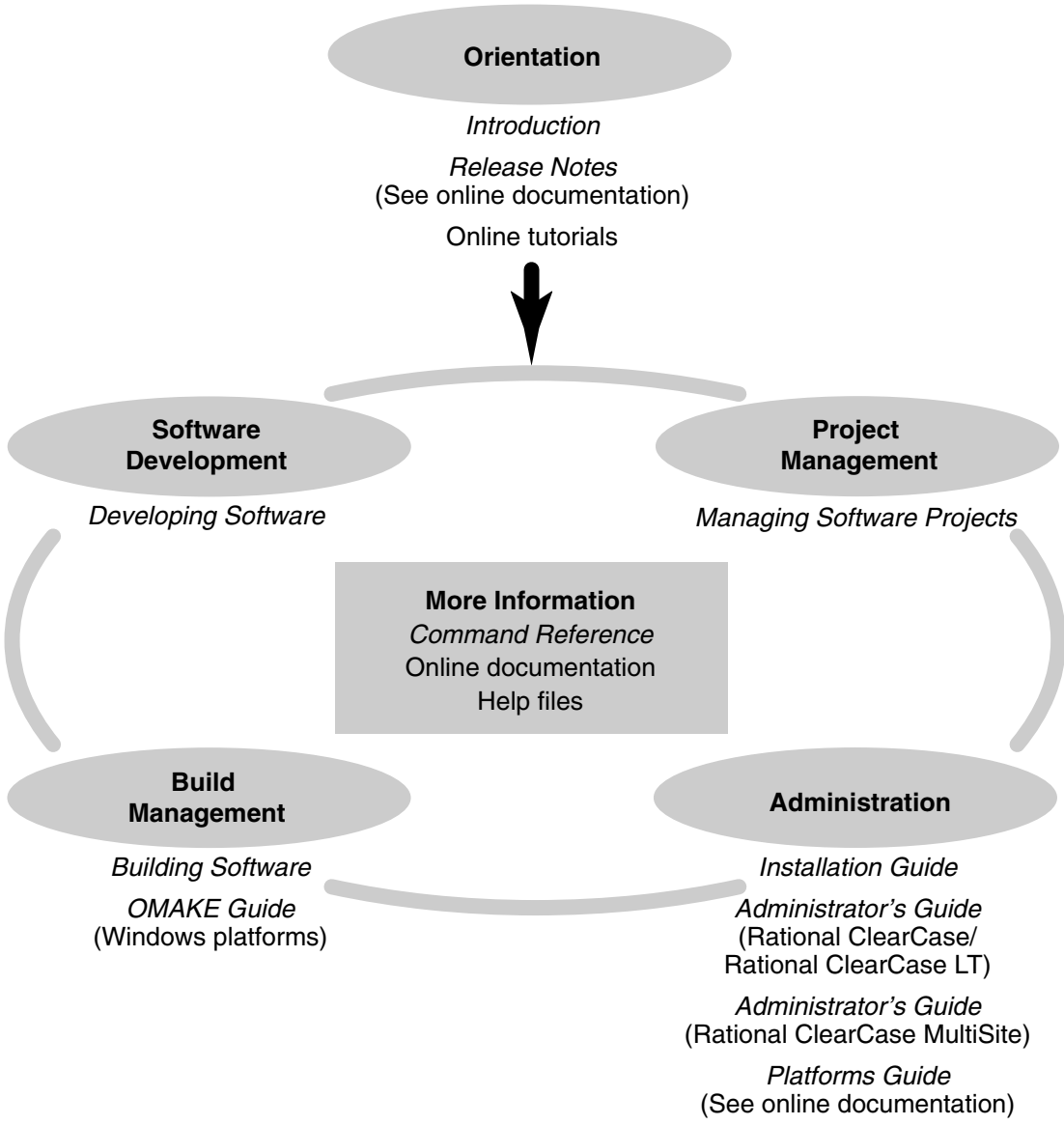
About This Manual

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

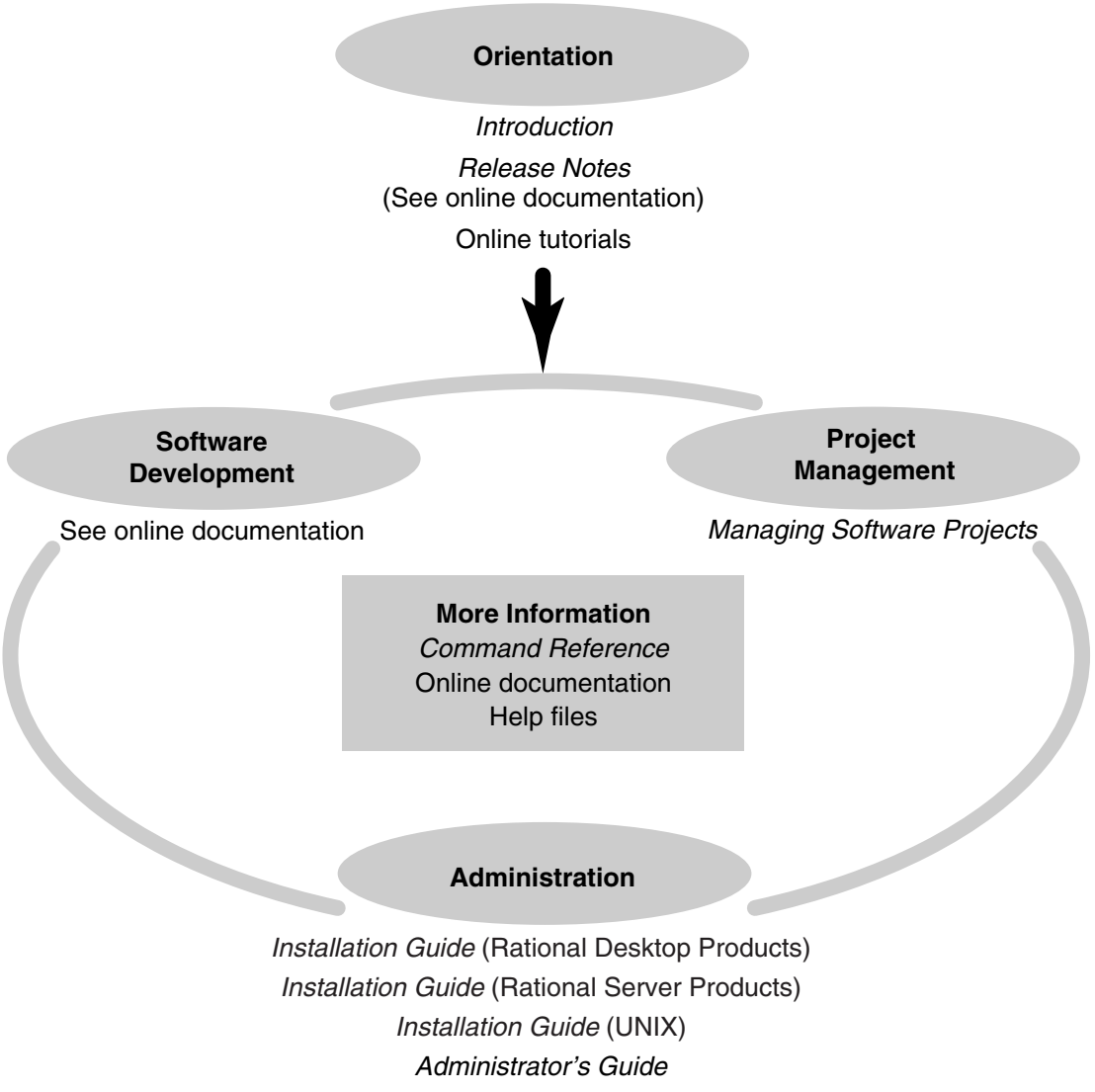
- Managing client and server hosts.
- Administering licenses.
- Administering the ClearCase registry, a central directory of shared information.
- Managing user and group accounts.
- Creating and managing VOBs (versioned object bases, or data repositories).
- Creating and managing views (user workspaces).
- Backing up and restoring critical data.
- Importing data from other configuration management systems.
- Periodic maintenance and troubleshooting.
- Tuning client and server hosts for better performance.
- Configuring optional third-party file access products.
- Configuring the Rational Web Platform and the ClearCase Web interface.

In this manual, the references to Windows apply to all Microsoft Windows platforms on which ClearCase or ClearCase LT are supported. References to UNIX apply to all UNIX and Linux platforms on which ClearCase or ClearCase LT are supported. See the *Installation Guide* for the most up-to-date list of supported UNIX and Windows operating systems.

ClearCase Documentation Roadmap



ClearCase LT Documentation Roadmap



ClearCase Integrations with Other Rational Products

Integration	Description	Where it is documented
Base ClearCase-ClearQuest	Associates change requests with versions of ClearCase elements.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*
Base ClearCase-Apex	Allows Apex developers to store files in ClearCase.	*Installing Rational Apex* (UNIX)
Base ClearCase-ClearDDTS	Associates change requests with versions of ClearCase elements.	*ClearCase ClearDDTS Integration*
Base ClearCase-PurifyPlus	Allows developers to invoke ClearCase from PurifyPlus.	PurifyPlus Help
Base ClearCase-RequisitePro	Archives RequisitePro projects in ClearCase.	*RequisitePro User's Guide* RequisitePro Help
Base ClearCase-Rose	Stores Rose models in ClearCase.	Rose Help
Base ClearCase-Rose RealTime	Stores Rose RealTime models in ClearCase.	*Rose RealTime Toolset Guide* *Rose RealTime Guide to Team Development*
Base ClearCase-SoDA	Collects information from ClearCase and presents it in various report formats.	*Using Rational SoDA for Word* *Using Rational SoDA for Frame* SoDA Help
Base ClearCase-XDE	Stores XDE models in ClearCase	XDE Help
UCM-ClearQuest	Links UCM activities to ClearQuest records.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*
UCM-PurifyPlus	Allows developers to invoke ClearCase from PurifyPlus.	PurifyPlus Help

Integration	Description	Where it is documented
UCM-RequisitePro	Allows RequisitePro administrators to create baselines of RequisitePro projects in UCM, and to create RequisitePro projects from baselines.	*RequisitePro User's Guide* RequisitePro Help *Using UCM with Rational Suite*
UCM-Rose	Stores Rose models in ClearCase.	Rose Help *Using UCM with Rational Suite*
UCM-Rose RealTime	Associates activities with revisions.	*Rose RealTime Toolset Guide* *Rose RealTime Guide to Team Development*
UCM-SoDA	Collects information from ClearCase and presents it in various report formats.	*Using Rational SoDA for Word* *Using Rational SoDA for Frame* SoDA Help
UCM-TestManager	Stores test assets in ClearCase.	*Rational TestManager User's Guide* TestManager Help *Using UCM with Rational Suite*
UCM-XDE	Stores XDE models in ClearCase	XDE Help
UCM-XDE Tester	Stores XDE Tester Datastores in ClearCase	XDE Tester Help

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 /opt/rational/clearcase on UNIX and C:\Program Files\Rational\ClearCase on Windows.
- *cquest-home-dir* represents the directory into which Rational ClearQuest has been installed. By default, this directory is /opt/rational/clearquest on UNIX and C:\Program Files\Rational\ClearQuest on Windows.
- **Bold** is used for names the user can enter; for example, command names and branch names.
- A sans-serif font is used for file names, directory names, and file extensions.

- **A sans-serif bold font** is used for GUI elements; for example, menu names and names of check boxes.
- *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 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, you can use "**...**" within a pathname as a wildcard, similar to "*" or "?". For more information, see the **wildcards_ccase** reference page.
- If a command or option name has a short form, a "medial dot" (·) character indicates the shortest legal abbreviation. For example:

  lsc·heckout

Online Documentation

The ClearCase Product Family (CPF) includes online documentation, as follows:

Help System: Use the **Help** menu, the **Help** button, or the F1 key. To display the contents of the online documentation set, do one of the following:

- On UNIX, type **cleartool man contents**
- On Windows, click **Start > Programs > Rational Software > Rational ClearCase > Help**
- On either platform, to display contents for Rational ClearCase MultiSite, type **multitool man contents**
- Use the **Help** button in a dialog box to display information about that dialog box or press F1.

Reference Pages: Use the **cleartool man** and **multitool man** commands. For more information, see the **man** reference page.

Command Syntax: Use the **–help** command option or the **cleartool help** command.

Tutorial: Provides a step-by-step tour of important features of the product. To start the tutorial, do one of the following:

- On UNIX, type **cleartool man tutorial**
- On Windows, click **Start > Programs > Rational Software > Rational ClearCase > ClearCase Tutorial**

PDF Manuals: Navigate to:

- On UNIX, *ccase-home-dir*/doc/books
- On Windows, *ccase-home-dir*\doc\books

Customer Support

If you have any problems with the software or documentation, please contact Rational Customer Support by telephone, fax, or electronic mail as described below. For information regarding support hours, languages spoken, or other support information, click the **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**

ClearCase Network Planning and Administration

1

The Rational ClearCase Product Family provides software configuration management solutions for teams of all sizes and geographic distributions. This manual is intended for experienced UNIX or Windows administrators who also have administrative responsibility for a ClearCase community (users, computers, shared data, and the network that connects them). Many of the procedures in this manual assume familiarity with the scripting and command-line conventions, file system access controls, networking protocols, and system administration tools for UNIX and Windows computers.

Overview of Administrative Responsibilities

This chapter presents a system administrator's view of a ClearCase community. It provides an overview of administrative responsibilities and serves as a roadmap to other chapters in this manual.

A ClearCase administrator has four principal concerns:

- Host and network administration, including installing and configuring server hosts, ensuring efficient client/server connectivity, managing and distributing server load, and tuning clients and servers for best performance.
- Data administration, including repository backup and recovery, data import/export, storage management, and periodic maintenance.
- User administration, including user and group account requirements, resolution of user access issues, and responsibility for data security.
- License administration, including license server administration, license key administration, and establishment of license use policies.

In addition, administrators of a ClearCase community where users access a common set of resources from both UNIX and Windows computers need to be aware of issues that may arise when users of these different operating systems share access to artifacts under ClearCase control.

The ClearCase Product Family

The ClearCase product family includes three members:

- Rational ClearCase, an enterprise-scale software configuration management solution that provides version control, workspace management, process configurability, and build management
- Rational ClearCase MultiSite, an optional add-on to Rational ClearCase, that provides data replication services which enable parallel development by geographically distributed teams
- Rational ClearCase LT, an entry-level version control tool that is designed for small project workgroups

This manual is intended for administrators who are responsible for communities using ClearCase, ClearCase LT, or both. In this manual, references to ClearCase apply to both ClearCase and ClearCase LT; references to ClearCase LT are product-specific.

Note: Rational ClearCase MultiSite administration is covered in its own manual, although several procedures in this manual include special considerations that apply when MultiSite is in use.

Differences Between ClearCase and ClearCase LT

This manual explains differences between ClearCase and ClearCase LT as the applicable topics are introduced. ClearCase LT provides a subset of the features found in ClearCase:

- A ClearCase LT community can have only one server host: the ClearCase LT Server. This host runs all ClearCase services and hosts all ClearCase data repositories created and used by the community.
- The ClearCase LT registry does not support registry regions, registry backup/switchover, or other features needed in a multi-server environment.
- ClearCase LT does not support dynamic views or any of the features (build management, for example) that depend on dynamic views.

Most of the material in this manual applies to both products. Any material that is not applicable to ClearCase LT is clearly identified.

Hosts and Host Administration

ClearCase is a distributed application with a client/server architecture. ClearCase LT enforces a strict client/server separation and limits a community to a single server

host. ClearCase supports multiple servers and provides greater flexibility in client and server configuration. In both ClearCase and ClearCase LT, many operations involve programs and data on more than one host. This section describes the types of hosts you may have in your ClearCase community and provides information on how each type of host is used.

Networkwide Release Hosts

In both ClearCase and ClearCase LT, one computer in the network acts as the networkwide release host. A directory on this host, known as a networkwide release area, stores an entire product release which has been extracted from distribution media or downloaded from Rational. When necessary, patches and service releases can be applied to update a release area with the latest enhancements and defect fixes. ClearCase can then be reinstalled from the release area to deploy these updates to individual hosts. Certain installation options allow ClearCase hosts running UNIX to access many ClearCase executables and libraries through symbolic links to the release area instead of installing them locally. ClearCase hosts running Windows do not access the networkwide release host after installation is complete. The networkwide release host does not need to run ClearCase

ClearCase LT Hosts

A ClearCase LT network includes only two types of hosts:

- **ClearCase LT client**. Each ClearCase LT user works at a client host, running ClearCase LT command line programs and graphical user interfaces, as well as other software (for example, development tools, a Rational Suite, and operating system utilities).
- **ClearCase LT server**. Every ClearCase LT client requires the services of a ClearCase LT server host. A ClearCase LT client host can access only one ClearCase LT server. A ClearCase LT server host can support multiple data repositories and can serve many ClearCase LT clients.

Note: For ClearCase LT server configuration guidelines, see *VOB Server Host Configuration Guidelines* on page 93.

ClearCase Hosts

Every host in a ClearCase network plays one or more of the following roles.

- **Client host.** Each ClearCase user works at a client host, running ClearCase command line programs and graphical user interfaces, as well as other software (for example, development tools, a Rational Suite, and operating system utilities). A ClearCase client installation can include the multiversion file system (MVFS),

which extends the host's native operating system to provide file system support for dynamic views.

- **Server host**. Any host on which a ClearCase data repository can be created is a server host. Some hosts may be dedicated servers on which client software rarely runs (and may not be installed). Other server hosts may double as clients.
- **License server host.** Every ClearCase community must have a license server host, which is normally designated during site preparation. The ClearCase license server supplies licenses that allow ClearCase commands to run. Each client or server host in the community is initially associated with a license server during ClearCase installation. Chapter 2 contains more information about the ClearCase license server and ClearCase license administration.
- **Registry server host.** Every ClearCase community must have a registry server host, which is normally designated during site preparation. The ClearCase registry provides a way for members of the community to locate shared resources, such as data repositories. Each client or server host in the community is initially associated with a registry server during ClearCase installation. Chapter 3 contains more information about the ClearCase registry and the registry server host.
- **Rational Web Platform server host.** At least one host in the network acts as the Rational Web Platform (RWP) server host. RWP provides application support for the ClearCase Web interface. Appendix C provides information on configuring RWP to support non-standard port assignments, proxy access, secure sockets, and other features that you may want to use at your site.
- **Network-attached storage (NAS) devices.** Certain NAS devices are certified for use with ClearCase. These devices can be used to store any ClearCase programs or data. NAS devices do not run ClearCase; they provide storage that ClearCase hosts access over a local area network. For information about configuring access to a certified NAS device, see *ClearCase and Network-Attached Storage Devices* on page 15. For a list of certified NAS devices, see the online *Platforms Guide*.
- **Non-ClearCase UNIX hosts.** UNIX hosts that cannot run ClearCase can still access ClearCase data, with limitations, through standard UNIX network file system facilities. For more information, see Appendix D.

Licensing and License Administration

Most ClearCase applications and commands (**cleartool** commands, for example, or GUIs like the ClearCase Explorer) require a license to run. ClearCase and ClearCase LT have different licensing mechanisms with different administrative requirements. ClearCase LT uses Rational Common Licensing, which is based on the FLEXlm license

management tool from GLOBEtrotter Software, Inc. ClearCase uses a proprietary licensing scheme. Users typically configure licensing on their own computers when they install ClearCase. An administrator usually needs to set up license server hosts, administer license keys and establish license use policies.

For more information about licensing and license administration, see Chapter 2.

Data Storage and Data Administration

ClearCase data is stored in several types of repositories:

- Information that enables clients to access repositories without needing to know the details of where they are hosted is maintained in the ClearCase registry.
- Versioned artifacts are kept in versioned object bases (VOBs), which are implemented using an embedded database management system.
- Artifact workspaces called views are supported by view databases, which are implemented using the same embedded database management system.

Repository maintenance is a critical aspect of ClearCase administration.

Note: ClearCase can be integrated with Rational ClearQuest, a change request management application that creates its own change request database. If your ClearCase community also uses ClearQuest, we recommend that your data administration plan include the ClearCase repositories.

The ClearCase Registry

The ClearCase registry provides a central repository of information about VOB and view locations, default values for various host ClearCase settings, and other information shared by a ClearCase community. Every ClearCase host is assigned to (is a client of) a single registry server host. In a ClearCase LT community, the registry server host is always the ClearCase LT server. In a ClearCase community, the registry server host can be any computer on which ClearCase has been installed.

A ClearCase community is defined in part by use of a common registry. If the registry server host becomes unavailable or registry data becomes corrupted, users cannot access artifacts under ClearCase control. For the most part, registry data is created, modified, and removed automatically, but there are some cases where administrative intervention may be required.

See Chapter 3 for more information about the ClearCase registry.

VOBs

A ClearCase community's permanent repository of software artifacts consists of one or (usually) more versioned object bases (VOBs), which can be located on any VOB server host. (In a ClearCase LT community, only the ClearCase LT Server can be a VOB server host.) Each VOB occupies a VOB storage directory, which holds an embedded database and related files.

VOB administration responsibilities include the following:

- **Creating VOBs.** Any user can create a VOB, but the ClearCase administrator is often responsible for creating and managing VOBs that hold artifacts that are important to all members of the community. For more information about creating and managing VOBs, see Chapter 6.
- **Importing data.** ClearCase includes a variety of tools that can import data from other configuration management systems. It is often the administrator's job to import this data into VOBs and organize it for the use of the community. For more information about importing data into a VOB, see Chapter 13.
- **Backup and recovery.** VOBs are critical data repositories with special backup and recovery requirements. Timely implementation of a reliable backup and restore strategy for VOBs is one of the ClearCase administrator's most important jobs. VOB backup and restore procedures are described in Chapter 10 and Chapter 11.
- **Managing network access.** Network access information for all VOBs is kept in the ClearCase registry, which is described in Chapter 3. In a typical network, registry maintenance is minimal; the commands that create VOBs also update the registry automatically. You do not need to change this information unless you move a VOB, change a VOB server host name, or enable VOB access in mixed networks of UNIX and Windows computers (see Chapter 4).
- **Managing user access.** Each VOB has an owner, a primary group, an optional supplemental group list, and a protection mode. Together, they control access to VOB data. Understanding and managing VOB access controls is an important task for the ClearCase administrator. For more information, see Chapter 5.
- **Periodic maintenance.** VOB administrators must 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. It also includes a job scheduling service and tools that you can run periodically to monitor and manage VOB data storage and to check VOB data integrity. Chapter 12 describes periodic maintenance tasks in detail.

Views

All users in a ClearCase community require access to one or more views to do their work. Views provide several important functions:

- Access to VOB data
- Workspaces where users can modify VOB data
- Short-Term storage for other data created during the development process

A typical view contains a combination of versioned artifacts (versions of VOB elements) and unversioned artifacts (view-private files that do not exist in any VOB).

There are three types of views:

- Snapshot views, which contain copies of versions of specified VOB elements, along with view-private objects. Snapshot views must be updated manually to access new versions that have been checked into the VOB from other views.
- Web views, which are similar to snapshot views but are accessed from the ClearCase Web interface.
- Dynamic views (not supported by ClearCase LT), which provide transparent access to versions of elements in the VOB and to view-private objects. A dynamic view can access any version of an element selected by the view's config spec as soon as the version is checked in.

Unlike VOBs, which are long-lived repositories, typically created and maintained by an administrator, views tend to be shorter lived and are usually created by individual developers. View administration is simpler than VOB administration, involving little more than occasional backups, periodic maintenance, and attention to access control issues. Chapter 9 and the **mkview** reference page provide more information about views and view administration.

User and Group Administration

ClearCase controls access to VOBs and views by means of users and groups. When a ClearCase application attempts an operation in a VOB or view, the application's credentials—which include the name and primary group of the user who started the application—are evaluated to determine whether the operation is permitted.

The identity with which a user logs on to the operating system of a ClearCase host establishes the user's ClearCase credentials. User names, group names, and each user's group memberships must be consistent across all ClearCase hosts in a community. This consistency is usually achieved by means of a networkwide user and group account database such as a Windows domain or the UNIX Network Information System (NIS). In environments where users access a common set of VOBs and views from UNIX and

Windows hosts, this consistency must extend to both platform types (user and group names used by ClearCase users, as well as each user's primary group assignment, must be the same on UNIX and Windows).

For more information on user and group names in mixed environments, see *Common User and Group Names* on page 55. For details about how user credentials control access to VOBs and views, see Chapter 5.

Note: In addition to the privileges normally accorded superusers such as **root** on UNIX or a member of the Administrators group on Windows, certain users and groups are recognized by ClearCase as having privileged status, which grants them unrestricted access to VOB data and other protected objects. For more information, see *Privileged Users and Groups* on page 70.

ClearCase Server Processes

ClearCase is a distributed application. Many of its operations involve several host computers and several server processes on each host. This section introduces the ClearCase server processes.

For the most part, these servers do not need to be explicitly configured, started, stopped, or managed. The information in this section is intended to help you understand status and error messages that may be generated by these servers, to access server logs, and to get a better idea of the role each server process plays in a ClearCase network.

Two server processes— **albd_server** and **admin_server**—run on every host that has been configured to support local VOBs and views, regardless of whether any VOBs or views have been created on the host. Other server processes are started as necessary to manage any VOBs and views that reside on the host.

Note: ClearCase LT clients cannot be configured with support for local VOBs or views. No ClearCase server processes run on a ClearCase LT client host or on a ClearCase host that has been configured with no support for local VOBs and views.

albd_server

The **albd_server** handles a variety of tasks on hosts configured to support local VOBs and views:

- Starting and stopping other ClearCase services as needed.
- Setting up network communications between ClearCase clients and servers.
- Managing execution of tasks run by the ClearCase scheduler.

- Responding to requests for registry information on a ClearCase registry server host.
- Responding to requests for licenses on a ClearCase license server host. (ClearCase LT does not use the **albd_server** in this way.)
- On UNIX hosts, responding to load-balancing queries from a remote **clearmake** process. (This feature is not supported on ClearCase LT.)

When you start ClearCase, the **albd_server** starts first, and runs with one of the following user identities:

- **root** on UNIX
- The built-in **LocalSystem** identity on ClearCase LT on Windows
- A specially created privileged user account on ClearCase on Windows

When you stop ClearCase, the **albd_server** stops all ClearCase services on the host and then exits.

Note: On a ClearCase LT client or a ClearCase client configured without support for local VOBs and views, the **albd_server** is not installed, and there are no other ClearCase services to stop or start.

When a client program wants to access a service (a VOB or view server, for example) on a ClearCase server host, it uses a remote procedure call (RPC) to send a request to the **albd_server** process on that host. The **albd_server** starts the requested service if it is not already started, then issues a response telling the client the service's port number. Thereafter, the client communicates directly with the specific service, without involving the **albd_server**.

ClearCase services started by the **albd_server** generally run with the identity of the resource (VOB or view) owner on UNIX and with the identity of the **albd_server** on Windows.

Port Assignment

The **albd_server** listens for RPCs on a well-known port (port 371) that has been reserved for it by the Internet Assigned Numbers Authority. ClearCase installation verifies that no other service uses this port for UDP or TCP communications. If a conflict is detected, we recommend that you modify the conflicting service to use another port. If you cannot reconfigure or remove the conflicting service, you must configure the **albd_server** to use a different UDP port.

- On UNIX, edit the local host's services database and/or the NIS services map.

- On Windows, create the appropriate entry in the file%SystemRoot%\System32\drivers\etc\services. (If there is no entry for the **albd_server** in this file, it will use port 371.)

Note: All **albd_server** processes in a community must use the same port number. If you change the **albd_server** port assignment on any ClearCase host, you must change it on every ClearCase host.

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

admin_server

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

- Retrieving server log files for display by the **getlog** command and the ClearCase Administration Console.
- Retrieving and changing the local host's ClearCase properties when requested by the ClearCase Administration Console.
- Moving registry files and reconfiguring clients if the host is a backup registry server.

credmap_server

The credentials mapping server **credmap_server** runs on any ClearCase host that is configured to support local VOBs and views. This server handles credentials mapping in environments where users access a common set of VOBs and views from both UNIX and Windows computers. For more information, see *Credentials Mapping* on page 56.

view_server

A **view_server** is a long-lived process that manages activity in a particular view. A **view_server** is started by the host's **albd_server** process whenever a client requests access to a view. A **view_server** remains active until it is terminated by a **cleartool endview –server** command, a system shutdown, or an operating system command that terminates the **view_server** process.

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

vob_server

For each VOB, a long-lived **vob_server** process runs on the VOB host. The **vob_server** manipulates data in the VOB's storage pools in response to requests from client processes.

The **vob_server** is the only process that ever creates or deletes VOB data containers; only the VOB owner or a privileged user can modify VOB data containers and storage pools. For more information, see *The VOB Storage Directory* on page 105.

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

- The VOB is removed with the **rmvob** command.
- ClearCase is stopped on the VOB server host.
- The VOB server host is shut down and restarted.

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

db_server

A host's **db_server** processes manage VOB database transactions on that host in response to requests from client programs. Because client programs cannot access VOB databases directly, they must send database transaction requests to a **db_server** process when they need to create, read, or modify VOB data or metadata.

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

vobrpc_server

Each VOB server host runs one or more **vobrpc_server** processes for each of its VOBs. Each **vobrpc_server** process handles requests from **view_server** processes throughout the network. These requests can generate both metadata (VOB database) and file system data (storage pool) activity. The **vobrpc_server** accesses the VOB database in the same way as a **db_server**; it forwards storage pool access requests to the **vob_server**.

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

lockmgr

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

Unlike most other ClearCase services, the **lockmgr** is not started by the **albd_server**. Instead, it is started when the VOB host starts. Lock manager startup options can be changed if necessary to improve VOB server performance for certain configurations. For more information, see *Modify Lock Manager Startup Options* on page 291.

rwp

The Rational Web Platform **rwp** provides application support for the ClearCase Web interface. RWP servers are normally configured at install time and need no further administrative attention unless special configurations (for example, a configuration to support access by proxy) are required. A ClearCase community may have one or more **rwp** servers. For more information, see Appendix C.

Server Logs

Each ClearCase server process maintains a log on the host where it runs. Because an error returned by a command entered on the local host can generate a log entry on a remote host, ClearCase on UNIX provides two remote log retrieval tools: the **cleartool getlog** command and the **Server Logs** node of the ClearCase Administration Console. (On Windows, ClearCase server processes write to the local host's Windows event log, which can be viewed with the Windows event viewer on any Windows host.)

Logging on UNIX

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

abe_log	Used by the audited build executor (**abe**) during parallel **clearmake** builds (not supported on ClearCase LT)
albd_log	Used by the **albd_server**
db_server_log	Used by the **db_server**
error_log	General-purpose error log. Used by user programs such as **cleartool**
event_scrubber_log	Used by the **event_scrubber** program
export_mvfs_log	Used by the **export_mvfs** program (not supported on ClearCase LT)
install_log	Used by **install_release** (installation script)
lockmgr_log	Used by the **lockmgr** program
mntrpc_server_log	Used by **mntrpc_server** program, which performs MVFS mounts requested by **cleartool mount** (not supported on ClearCase LT)
msadm_log	Used by the MultiSite administrative server, which handles requests for mastership (not supported on ClearCase LT)
promote_log	Used by the **promote_server** (not supported on ClearCase LT)
scrubber_log	Used by the **scrubber** program
view_log	Used by the **view_server**
vob_log	Used by the **vob_server**
vob_scrubber_log	Used by **vob_scrubber** program
vobrpc_server_log	Used by **vobrpc_server**

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

As entries accumulate, log files grow. By default, the ClearCase scheduler periodically runs a job that renames log files to *logfile_name*.old and creates empty template files in their place.

Logging on Windows

On Windows, ClearCase programs write informational, warning, and error messages to the Windows application event log. The source of these messages is displayed by the

Windows event viewer as **ClearCase**. A typical event log entry includes the date and time of the error, the software module in which the error occurred, the current user, and an error-specific message.

On a ClearCase host where the MVFS is installed, the MVFS logs error and status messages to the file C:\mvfslogs. You can use the **MVFS** tab in the ClearCase program in Control Panel to change this pathname. By default, the ClearCase scheduler periodically runs a job that deletes MVFS log files more than seven days old. For information about describing and changing scheduled jobs, see the **schedule** reference page.

Starting and Stopping ClearCase

ClearCase is normally started and stopped when a ClearCase host starts up or shuts down, using the startup and shutdown conventions for the platform type. This section describes how to stop or start ClearCase services manually.

Note: No ClearCase server processes run on a ClearCase LT client.

On UNIX

When UNIX is bootstrapped, the ClearCase startup script *ccase-home-dir*/etc/clearcase is executed by the UNIX **init** program. The startup script does the following:

- Starts the host's **albd_server** process.
- Starts a **lockmgr** process if any VOBs are on the host.
- Performs additional setup tasks, such as mounting public VOBs.

The **root** user can run the **clearcase** script to stop or start ClearCase at any time. For example:

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

Stops ClearCase.

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

Starts ClearCase.

For more information, see the **init_ccase** reference page.

On Windows

Unless explicitly configured to be started manually, the **lockmgr** and **albd_server** services, as well as the MVFS if it is installed, are started when a ClearCase host starts. If you are a member of the local host's Administrators or Power Users groups, you can

use the **Services Startup** tab of the ClearCase program in Windows Control Panel to stop and start ClearCase on the host.

1. Click **Start > Settings > Control Panel**, and then open the ClearCase program.
2. Click the **Services Startup** tab. This tab displays the status of the **albd_server** and, if the host supports local VOBs, the lock manager. On hosts where the MVFS is installed, this tab also displays the status of the MVFS and the Credentials Manager, which manages the MVFS credentials cache.
   - Click **Stop ClearCase** to stop ClearCase services.
   - Click **Start ClearCase** to start ClearCase services.

You can also use the Windows **net start** and **net stop** commands:

- Type **net stop albd** to stop ClearCase services.
- Type **net start albd** to start ClearCase services.

Starting and Stopping the MVFS

Unlike other ClearCase services, the MVFS runs in the kernel and cannot be stopped or started independently of the operating system. To stop and restart the MVFS on UNIX or Windows, you must shut down and restart the computer.

ClearCase and Network-Attached Storage Devices

Network-attached storage (NAS) devices provide data storage resources to other hosts on a local area network by using a network file system protocol like the Network File System (NFS) on UNIX or the Common Internet File System (CIFS) on Windows. Any NAS device can provide storage for ordinary files that are created and used by ClearCase (networkwide release areas, for example, or the remote storage pools supported by ClearCase hosts running UNIX). In addition, several NAS devices have been certified by Rational to provide storage, when properly configured and used, for all VOB and view storage, including database storage.

Note: ClearCase LT does not support use of a NAS device to store VOBs or views.

Putting view and (especially) VOB storage on a certified NAS device can provide several advantages:

- **Enhanced scalability**. VOB or view storage can grow well beyond the physical limits typically imposed when the VOB or view is stored on a ClearCase server host.
- **Enhanced flexibility.** Separating the server host from its VOB or view storage simplifies the process of upgrading the server host.

- **Simplified administration.** NAS devices often have enhanced backup and restore features that can simplify backing up and restoring VOBs and views. When you use the **mkstgloc** command to create server storage locations on NAS devices, VOB and view creation and administration can be further simplified.
- **Simplified interoperation.** All certified NAS devices can be configured to support native interoperation with both UNIX and Windows hosts. You do not need to install any additional cross-platform file-access software on ClearCase client or server hosts if you are using a NAS device to provide VOB and view storage for a mixed network of Windows and UNIX computers. For more information, see *Cross-Platform File System Access* on page 57.

Caution: Every certified NAS device must be specially configured to support remote VOB or view databases. Configuration requirements for each certified device are described in the online *Platforms Guide*. If you do not configure a certified NAS device as described in that document, any VOB or view data stored on that device is at risk. If you put a VOB or view database on a NAS device that has not been certified for this purpose, the data is at risk. In some cases, using a NAS device for VOB or view storage can degrade ClearCase performance. For more information, see *Server Performance and NAS Devices* on page 295.

Special procedures may be required when creating, moving, backing up, or restoring VOBS and views on NAS devices. These procedures are described in the chapters of this manual that deal with VOB and view administration, and with backing up and restoring VOBs and views.

Configuring Network Access to the NAS Device

A NAS device used for VOB or view storage should be on a robust network, preferably on the same subnet as the ClearCase VOB or view server hosts that use it, but at most no more than one hop away. Network access to the NAS device can be configured as follows:

- UNIX VOB or view server hosts must mount the NAS device with an NFS hard mount (the default for most systems). If a soft mount is used, the VOB or view server process fails at startup and writes an error message in its log file.
- Windows hosts must establish access to the NAS device with the **net use** command (or **Map Network Drive** in Windows Explorer).

Note: You must use the NFS protocol to connect a UNIX VOB or view server host with a VOB or view database on a NAS device. You must use the CIFS protocol to connect a Windows VOB or view server host with a VOB or view database on a NAS device. Rational does not support using NFS software to connect Windows hosts to NAS devices used by ClearCase communities.

If You Use DHCP

The Dynamic Host Configuration Protocol (DHCP) is an Internet standard protocol that allows a host to obtain a temporary, dynamically assigned IP address from a DHCP server. DHCP is widely implemented on Windows computers and also on some UNIX computers. ClearCase is compatible with most DHCP configurations, although it has a few special requirements that you may need to be aware of.

ClearCase hosts cache the IP addresses of other ClearCase hosts (usually servers) that they access. If a cached address becomes invalid because the remote host has been assigned a new IP address by DHCP, ClearCase applications that rely on the cached address may fail. Because ClearCase server hosts are usually running and connected to the network, they are likely to keep the IP addresses that their clients have cached, which makes these sorts of failures unlikely. To further minimize the chance of such failures, we recommend that you adhere to the following guidelines if DHCP is in use at your site:

- If possible, assign fixed IP addresses to ClearCase server hosts.
- Set the DHCP lease time (the length of time for which a host can use a DHCP-assigned address) to be at least twice as long as any ClearCase host is likely to be down or off the network. The DHCP lease time is typically established by the DHCP server.
- Do not configure ClearCase hosts to obtain a new DHCP address when they restart.

If a ClearCase Host Has Multiple Network Interfaces

If any ClearCase host (client or server) has two or more network interfaces, you must create a file on that host which lists each of its host names.

- On UNIX, the file is /var/adm/rational/clearcase/config/alternate_hostnames.
- On Windows, the file is *ccase-home-dir*\var\config\alternate_hostnames.

For example, suppose that the NIS hosts database includes these entries for a UNIX host:

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

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

```
widget
widget-gte
```

Note that only the first host name in each hosts entry must be included in the file. The file must list each 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.

ClearCase and the NFS Automounter

This section discusses use of the NFS (UNIX) **automount** facility with ClearCase. UNIX hosts typically use the automounter to mount exported file systems, including VOB and view storage directories, on remote hosts. Because various ClearCase commands must derive a global pathname to a VOB or view storage directory that has been (or will be) mounted by the automounter, you may need to understand the details of how your automounter constructs the local mount points through which remote storage is accessed.

Note: Implementations of **automount** vary; consult your UNIX platform's reference information on **automount**.

Automounter Maps and Mount Points

You can use direct or indirect automount maps to access remote VOB or view storage. The heuristics that ClearCase uses to construct a global pathname assume that all hosts use a common automounter map. If you create a VOB or view on a host that does not use an automounter map, you must take additional steps to ensure network access to the VOB or view. See *UNIX Automounter Does Not Use the Hosts Map* on page 102 for more on this topic.

When constructing a global path to remote VOB or view storage, ClearCase attempts to access the remote host using a pathname that begins with one of the standard NFS mount points /net, /hosts, or /nfs. For example, to construct a global path to VOB storage on a host named **mars**, ClearCase starts with the pathname /net/mars, and then adds the name of the VOB storage directory specified in the **mkvob** command or VOB creation wizard. If it cannot access the specified storage directory using this path, it tries the other standard mount points.

Specifying Nonstandard Mount Points

You can use the environment variable CCASE_GPATH_HINTS to specify one or more mount points for ClearCase to try before it tries the standard mount points. Set this environment variable to a value that is a colon-separated list of mount points that ClearCase should try when constructing a global path to VOB or view storage on a remote UNIX host. For example, in the presence of the following setting:

CCASE_GPATH_HINTS="/vob_servers:/view_servers"

ClearCase would try to construct a pathname to a remote NFS directory starting with the mount point /vob_servers and then, if it could not construct a valid global path from that mount point, trying /view_servers. If none of the mount points listed in CCASE_GPATH_HINTS can be used to construct a valid global path, ClearCase tries the standard mount points.

Note: You can also set CCASE_GPATH_HINTS to a value of **""** (null) to force **cleartool** commands that create VOBs or views to use an explicit global storage path as specified on the command line. For more information, see the **env_ccase**, **mkvob**, **mkview**, and **mkstgloc** reference pages.

Specifying a Nonstandard Mount Directory

On most UNIX platforms, the automounter mounts remote file systems directly on one of the standard mount points. Some automounters must mount remote file systems under /tmp_mnt and then access them through a UNIX symbolic link to /tmp_mnt from /net or one of the other standard mount points. If a ClearCase host has an automounter that creates this kind of symbolic link to some other mount point, you must specify the name of that mount point in the file /var/adm/rational/clearcase/config/automount_prefix. For example, if the automounter uses /autom instead of /tmp_mnt, place this line in the automount_prefix file:

```
/autom
```

Licensing and License Administration

2

Rational ClearCase and Rational ClearCase LT require a license to run. This chapter describes licensing mechanisms and license administration procedures for both products.

ClearCase Licensing and Rational Common Licensing

ClearCase and ClearCase LT use different licensing mechanisms. ClearCase uses a proprietary floating license scheme. ClearCase LT uses Rational Common Licensing. The behavior of licenses (when they are required, how they are granted, how long they are valid, and so on) is similar in both systems. However, the two systems are separate (a ClearCase host cannot use a ClearCase LT license, and under normal circumstances, a ClearCase LT host cannot use a ClearCase license), and each scheme has different administrative requirements.

For information about ClearCase LT licensing, see *Rational Common Licensing for ClearCase LT* on page 27.

ClearCase Licensing

ClearCase uses an active-user floating license scheme. To use ClearCase, a user must obtain a license, which grants the privilege to use ClearCase on any host served by the same license server. 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 the license 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, as described in *License Database Format* on page 26. Whenever a user runs a ClearCase command or GUI, a license-verification check is made. If the user already has a license, the check succeeds and the command or GUI runs. If the user does not have a license:

1 ClearCase client software looks for the name of the license server host.

2. The command or GUI communicates with the license server process on the license server host, to see whether the user can get a license. (The license server process is actually the **albd_server**, performing these duties in addition to its other tasks.)
3. If a license is available, the license server grants the user a license and the command or GUI runs.
4. If no license is available, the license server returns a message to that effect and the command or GUI exists with an error message.

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 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.

Specifying a License Server Host

Each ClearCase host must be configured to obtain licenses from a specific license server host. This configuration usually happens at install time, and is normally taken from a site default specified by the ClearCase administrator. To reconfigure a ClearCase host to use a different license server, do one of the following:

- On UNIX, the license server host name is stored in the file /var/adm/rational/clearcase/config/license_host. You can edit this file with any text editor to specify the name of a new license server host.
- On Windows, the license server host name is specified on the **Licensing** tab of the ClearCase program in Control Panel as the value **Use license server on host:**.

When a ClearCase community includes both UNIX and Windows computers, the license server host for both types of computers can be either a Windows or a UNIX computer.

Note: If you are using a Windows computer to serve ClearCase licenses for both UNIX and Windows, the Windows license server host must be a member of a domain in which user accounts are defined for all UNIX users who will acquire licenses from this server. For more information, see *Common User and Group Names* on page 55.

Setting Up a License Server

The task of setting up a host to be a license server host is part of the ClearCase installation process. Initial license server setup is described in the *Installation Guide* for the ClearCase Product Family.

The License Database

The license database includes license keys and optional license management information.

- On Windows hosts, the license database is stored in the Windows Registry and should only be modified by using the **Licensing** tab of the ClearCase program in Control Panel.
- On UNIX hosts, the license database is a text file, /var/adm/rational/clearcase/license.db, which you can modify with any text editor

  Note: All lines in the license.db file must be terminated with a newline character.

Adding New Licenses to an Existing License Server Host

Some organizations purchase an initial set of ClearCase licenses and then purchase additional licenses later. Each time a new set of licenses is purchased, you receive a new license authorization code, called a license key, from Rational. The key is an alphanumeric string, which you may receive in e-mail or a fax. You must enter this string into the license database correctly, preserving the case of letters, or it will not be valid.

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 host, the name of the license server host is stored in the file /var/adm/rational/clearcase/config/license_host
   - On a Windows host, the name of the license server host is displayed in the ClearCase program in Control Panel. Click **Start > Settings > Control Panel**, and then start the ClearCase program. Click the **Licensing** tab to display the name of the license server host.
2. **Log on to the license server host.** On a UNIX host, only **root** can modify the license database. On a Windows host, you must have privileges to edit the host's Windows registry to change the license database**.**
3. **Add the licenses**.

- On UNIX, you can use any text editor to append the new license key to the end of the license.db file.
- On a Windows host, click **Start > Settings > Control Panel**, and then start 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 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 add 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. **Obtain the machine ID of the new license server host.** ClearCase licenses are tied to a specific license server host by that host's IEEE Media Access Control (MAC) address, a unique identifier associated with the host's network interface hardware. You can use the **clearlicense** command to display the machine identifier:

   clearlicense –hostid

   ```
   Current machine identifier: 00c02e0931cf
   ```

   On a Windows host, you can also click **Start > Settings > Control Panel**, and then start the **ClearCase** program. Click the **Licensing** tab to display the host's machine ID.

   If you need to obtain the machine ID of a host on which ClearCase is not yet installed, follow the instructions on the *License Registration Form*.

2. **Fill out the** *ClearCase Product Family License Registration Form*. A copy of this form appears at the back of the *Installation Guide* for the Rational ClearCase Product Family. Fax the form to Rational. Rational will send you a new license key.
3. **Log on to the new license server host.** On a UNIX host, only **root** can modify the license database. On a Windows host, you must have privileges to edit the host's Windows registry to change the license database**.**
4. **Add licenses to the new host's license database**.
   - On UNIX, you can use any text editor to append the new license string to the end of the license.db file.

- On a Windows host, click **Start > Settings > Control Panel**, and then start 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 in 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/rational/clearcase/config/license_host to contain the name of the new license server host. On a Windows host, click **Start > Settings > Control Panel**, and then start the ClearCase program. Click the **Licensing** tab and enter the new license server's name in the **Hostname of the license server** box. Click **Apply** or **OK**.

You can also use the ClearCase Administration Console to change the license server used by any ClearCase host in your network:

1 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**.

2 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 privileged user.

Note: When a user accesses multiple ClearCase hosts that use different license servers, the user acquires a license from each server. When a user accesses multiple ClearCase hosts that use the same license server, the user acquires a single license.

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 are changing the name of a license server host. It is only required if the license host ID (as reported by **clearlicense –hostid**) changes for any reason. Common reasons for such a change include moving the license server host function to a new computer or replacing the network interface hardware of an existing license server 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 *Installation Guide* for the Rational ClearCase ClearCase Product Family. Fax the form to Rational. Rational will send you the replacement license keys.
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* on page 23. If you are moving licenses to a new license server host, follow the procedure in *Setting Up Additional License Server Hosts* on page 24.

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 network host name. 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 25.

License Database Format

A license database contains several kinds of lines. A line can define a multiuser license, specify license priorities for individual users, or enable auditing of licensing activity.

License Set Definition Lines

A ClearCase Product Family license key consists of a single line of text, which defines a certain number of licenses. Most licenses are tied to a specific license server host and cannot be moved to any other host without first following the procedures outlined in *Moving Licenses to Another Host* on page 25. If the vendor field is **TEMPORARY**, you can move the license to any license server host.

The license file can contain any number of **–license** lines. All the lines are effectively combined into a single license; the maximum numbers accumulate to determine the total number of license slots.

User Priority Lines

The license file can contain any number of **–user** lines, each of which specifies one or more users (by name or by numeric ID). These lines are effectively concatenated into a single license priority list. The first user on the list has the highest priority; each successive user has a lower priority. Users who are not listed can use the products, but they share the lowest priority.

Excluded User Lines

The license file can contain any number of **–nuser** lines, each of which specifies one or more users (by name or by numeric ID). The specified users cannot obtain a license and thus cannot use the product.

–user and **–nuser** lines can be intermixed. If a user is named in both kinds of line, the first entry is used.

Audit-Enable Line

A line consisting of the single word **–audit** enables auditing of license activity. On UNIX, licensing audit messages are written to the file /var/adm/rational/clearcase/log/albd_log; on Windows, these messages are written to the Windows event log. On either platform, the following license events are logged:

- A user is granted a new license.
- A user is denied a license because all licenses are in use.
- A user entered a **clearlicense –release** command (the success or failure of the command is also logged).

Time-Out Line

By default, a ClearCase license granted to a user expires after 60 minutes of inactivity (60 minutes in which the user does not run a **cleartool** command or ClearCase GUI). A **–timeout** line changes the expiration interval to the specified number of minutes. The minimum interval is 30 minutes; there is no maximum interval.

Rational Common Licensing for ClearCase LT

Rational ClearCase LT uses Rational Common Licensing, which is based on the FLEXlm license management tool from GLOBEtrotter Software, Inc. The ClearCase LT installation procedure installs FLEXlm client software on ClearCase LT client hosts.

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

The *License Management Guide* for Rational Suite provides detailed descriptions of all aspects of Rational Common Licensing on Windows and UNIX platforms; it includes all the information a user or administrator needs to install, configure, and troubleshoot ClearCase LT licensing, as well as instructions for setting up a Rational license server host.

Floating and Node-Locked Licenses

Rational Common Licensing supports two types of licenses: floating and node-locked.

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

Note: ClearCase LT on UNIX platforms can use only the floating license type.

When a user runs a **cleartool** command or ClearCase LT GUI, the command or GUI attempts to obtain a license. This attempt always succeeds on a client with a valid node-locked license. If the client is configured to use a floating license, the attempt succeeds if the license server host can be reached over the network and a license is available. After a user obtains a floating license, running any ClearCase command or GUI renews the license. If the user doesn't enter a ClearCase command for a substantial period—by default, 30 minutes—the license expires and can be taken by another user.

Temporary, Permanent, and Term Licenses

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

ClearCase LT and Rational Suites Licenses

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

Installing and Configuring a License Server

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

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

A Rational license server hosted on either a Windows or UNIX computer can serve licenses for ClearCase LT on any platform. If you need to set up a license server host, see the *License Management Guide* for Rational Suite.

Installing and Configuring Windows Client Licenses

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

The License Key Administrator

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

Installing and Configuring UNIX Client Licenses

On UNIX platforms, ClearCase LT only supports the floating license type. Floating licenses require a license server host, and must be installed on that host by an administrator before they can be used. For information about how to obtain and install permanent ClearCase LT licenses, see the *License Management Guide* for Rational Suite. for information about how to specify the license server for a ClearCase LT host running UNIX, see the *Rational Suite UNIX Installation Guide*.

Administering the ClearCase Registry

3

Because it is a distributed application, Rational ClearCase provides a way for users to access shared resources without having to know specific host names, network paths, or any of the other information usually required to obtain access to a network resource. The ClearCase registry provides a central repository of information about:

- VOB and view locations
- Default values for typical client host settings

Every ClearCase host is assigned to (is a client of) a single registry host. In a ClearCase LT community, the registry server host is always the ClearCase LT server. In a ClearCase community, the registry server host is typically specified during the site preparation process and can be any computer on which ClearCase is installed. Clients use remote procedure calls (RPCs) to communicate with the registry server. These RPCs request information from the registry as needed and modify registry data when VOBs or views are created or moved or when community defaults are changed.

Every ClearCase community depends on its registry. If the registry server host becomes unavailable or registry data becomes corrupted, users cannot access artifacts under ClearCase control. This chapter introduces the ClearCase registry and describes typical registry administration procedures.

Note: The ClearCase registry is not related to the Windows registry in any way.

Differences Between the ClearCase and ClearCase LT Registries

Because ClearCase LT supports only a single server and does not support dynamic views, ClearCase LT registry administration is greatly simplified. The ClearCase LT registry does not support registry regions, cannot be administered directly from the ClearCase Administration Console, and cannot use ClearCase registry backup and switchover functionality. ClearCase LT administrators may wish to read some of the material in this chapter to gain a better understanding of the registry data model, but most of the material in this chapter does not apply to ClearCase LT.

Registry Data

The registry data model is a simple one, designed to provide the basic information needed to locate and retrieve shared resources. The registry holds several kinds of data:

- **Objects and tags.** Most ClearCase applications connect with VOBs and views by using the information in VOB or view objects and tags, which are created automatically as part of VOB and view creation. They can also be created, removed, and modified explicitly when necessary.
- **Other shared data.** The registry is also the repository for system configuration information such as recommended VOB and view storage locations and sitewide defaults for various host configuration parameters. These defaults are established by the administrator, usually at install time, but can be changed as needed.

An understanding of the registry data model is important for many administrative procedures.

Objects

Every VOB or view must have exactly one object entry in the registry. The object is created when the VOB or view is created, and it provides a persistent record of the information that enables VOB or view server processes to access VOB or view data.

A VOB object holds the following information:

- The name of the host on which the VOB's server processes run. The VOB's server processes run on the host where the VOB storage directory is located unless the VOB storage directory is located on a certified network-attached storage (NAS) device.
- The VOB's universal unique identifiers (UUIDs). Each VOB has a replica UUID and a family UUID. The UUIDs are generated when the VOB is created, stored in the VOB database, and copied into the VOB object entry by the **register** command (which is executed implicitly by **mkvob** and GUIs that create VOBs). If a VOB is replicated (with Rational ClearCase MultiSite), each replica in a family has its own replica UUID, but all replicas in a family have the same family UUID. If a VOB is not replicated, the family UUID is not used.
- A pathname to the VOB storage directory, expressed in a form that is valid on the VOB host itself. This host-local pathname is used by the VOB's server processes.
- VOB object attributes, which may include any of the following:
  - **UCM VOB** (the VOB is a UCM component)
  - **Project VOB** (the VOB is a UCM PVOB)

A view object holds the following information:

- The name of the host on which the view's server processes run. The view's server processes run on the host where the view storage directory is located unless the view storage directory is located on a certified NAS device.
- The view's UUID. Like a VOB UUID, the view UUID is generated when the view is created, stored in the view storage directory, and copied into the view object.
- A pathname to the view storage directory, expressed in a form that is valid on the host. This host-local pathname is used by the view's server processes.
- The user name of the view's owner.
- View object attributes, which may include any of the following:
  - **Snapshot view**
  - **UCM view** (sometimes displayed as **sumview**)
  - **Web view** (always paired with the **Snapshot view** attribute)

A VOB or view object is added to the registry when the VOB or view is created. The entry is updated whenever the VOB or view is reformatted (**reformatvob**, **reformatview**), because reformatting changes a VOB's replica UUID (but not its family UUID) or a view's UUID.

You can update or remove a VOB or view object manually by using the **cleartool register** and **unregister** commands. You can also use the ClearCase Administration Console. Typically, this is required only when you do any of the following:

- Move a VOB or view to another host
- Rename a VOB or view server host
- Clean up the registry after an incomplete or incorrect removal of a VOB or view

Tags

In addition to an object, every VOB or view must have a tag in the registry before it can be accessed by most ClearCase applications. VOB tags and view tags serve similar purposes and control access to VOBs and views in similar ways.

VOB Tags and VOB Access

Nearly every operation that accesses VOB data refers to the VOB by its tag. A VOB tag holds the following information:

- The name of the host on which the VOB's server processes run. The VOB's server processes run on the host where the VOB storage directory is located unless that directory is on a certified NAS device.

- The VOB's replica UUID, copied from the VOB object.
- A network pathname to the VOB storage directory, expressed in a form that is valid for all hosts that access the VOB. This pathname (referred to as a global path) is used by client programs that need network access to the VOB storage directory.
- Default mount options that are applied when the VOB is mounted for use with a dynamic view. For more information, see the **mount** reference page.
- The **Public** attribute if the VOB was created as a public VOB. For more information about the meaning of this attribute, see *Public and Private VOBs* on page 99.
- An optional description, which is displayed by various commands and GUIs.

A VOB must be mounted before it can be accessed by a dynamic view. All operations that mount a VOB refer to the VOB by its tag. A VOB without a tag cannot be mounted.

A VOB does not need to be mounted before it can be accessed by a snapshot or Web view. However, the view's load rules reference VOB tags, and no elements can be loaded into a snapshot or Web view from a VOB that does not have a tag.

In all views, VOB tags appear as file system directories. In a dynamic view, the tags are mounted as though they were network file systems. In snapshot and Web views, VOB tags are used to name the top-level directories that are loaded into the view. Naming rules for VOB tags depend on the platforms that will access the VOBs.

- Tags for VOBs accessed by ClearCase LT clients running Windows or UNIX have a single leaf, which must begin with either a slash or a backslash character (for example, \sources).
- Tags for VOBs accessed by ClearCase clients running Windows have only a single leaf, which must begin with a backslash character (for example, \sources).
- Tags for VOBs accessed by ClearCase clients running UNIX typically have two parts, a mount point and a leaf name (for example, in the VOB tag /vobs/sources, /vobs is the name of the mount point and /sources is the leaf name). Each client host must create the mount point locally before it can mount any VOB.

Note: It is possible to use UNIX VOB tags that have only a single leaf, but limitations on the number of file systems that can be mounted on the root directory of a UNIX computer make this impractical for most communities that use dynamic views. When single-leaf VOB tags are being used in a mixed Windows and UNIX environment, tags that differ only in their initial character—backslash or slash— are equivalent.

View Tags and View Access

Nearly every operation that accesses a view refers to the view by its tag. A view tag holds the following information:

- The name of the host on which the view's server processes run. These processes run on the host where the view storage directory is located unless that directory is on a certified NAS device.
- The view UUID, copied from the view object.
- A network pathname to the view storage directory, expressed in a form that is valid for all clients that will use the view. The pathname is used by client programs that need network access to the view storage directory.
- An optional description, which is displayed by various commands and GUIs.

A dynamic view must be started before it can be used. All operations that start a dynamic view refer to the view by its tag. A view without a tag cannot be started.

A snapshot or Web view does not have to be started. However, all operations that alter the contents of the view (update, for example, or checkout) must access the view database, which cannot be accessed if the view does not have a tag.

Like VOB tags, view tags are referenced by users and applications as though they were file system directories.

- On a UNIX client, each dynamic view appears as a subdirectory of the /view directory.
- On a Windows client, each dynamic view appears as a share under a special network name (\\view by default) as well as a directory under the client's MVFS drive (drive M by default).
- On both UNIX and Windows, snapshot views appear as directories in the local file system.

View tags must conform to the host platform's file-naming conventions with respect to length and character set.

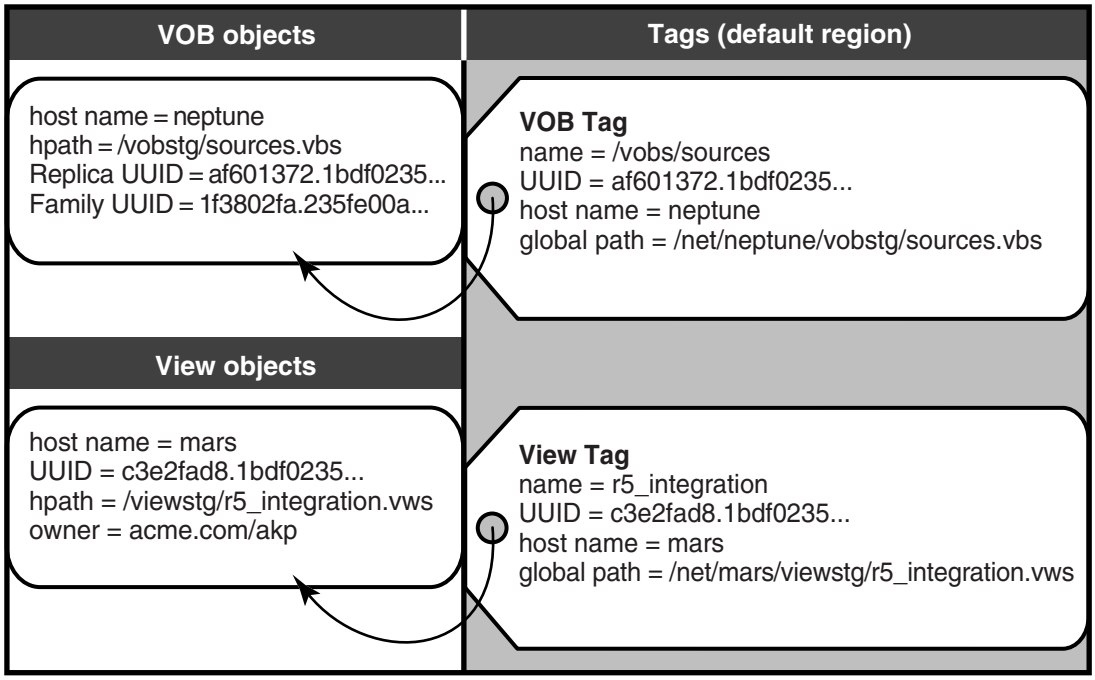
Registry Regions

A registry region is a tag namespace shared by a subset of registry clients. The ClearCase LT registry supports only a single region. The ClearCase registry allows you to create multiple regions. VOB and view tags in a ClearCase registry include the name of the registry region in which the tag is visible.

Each ClearCase client is a member of a single registry region and can access only those VOBs and views whose tags are visible in that region.

In both ClearCase and ClearCase LT, the registry initially has one region, as shown in Figure 1. All clients are members of this region. All tags are created in this region, and are visible to all clients of the registry host.

Figure 1 Registry with One Region

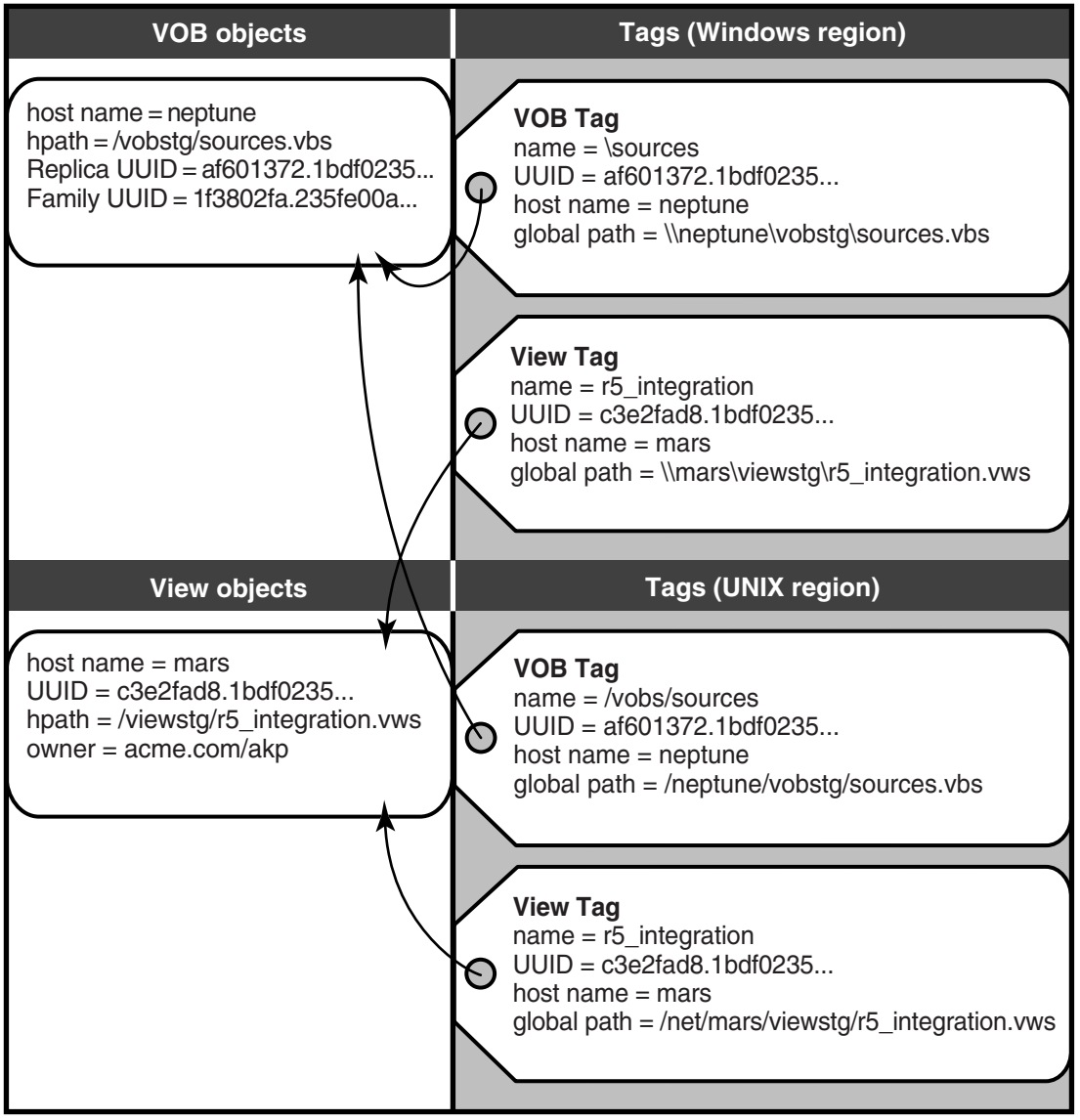


The most common reason for creating an additional region is to accommodate the different network file system naming conventions of Windows and UNIX operating systems when Windows hosts that use dynamic views need to access VOBs or views hosted on UNIX. (Snapshot and Web views do not require network file system access to VOB server hosts.)

Figure 2 illustrates a registry with two regions: one for Windows clients, in which global paths are expressed as UNC names, and another for UNIX clients, in which global paths are expressed as NFS pathnames.

Note: We recommend that you establish uniform naming standards when creating tags in multiple regions. For example, preserving the sources leaf name in the UNIX tag /vobs/sources and the Windows tag \sources makes it easier for users to know that the two tags refer to the same VOB.

Figure 2 Registry with Two Regions



Regardless of how many regions a registry has, each VOB or view tag in a region references a single underlying VOB or view object entry, which must exist before any tag can be created.

You may also need to create additional regions for any of the following reasons:

- Your network includes VOB or view server hosts that have multiple network interfaces and, therefore, multiple host names.
- You create VOB or view tags that include multibyte characters, and the registry serves clients that use different settings for operating system locale. Because the interpretation of multibyte characters is controlled by a host's locale setting (established on UNIX through the LOCALE environment variable and on Windows by using the Regional Options tool in Control Panel), all tags in a region must use the same character set, and all hosts in a region must use the same locale setting.
- You want to prevent access to specific VOBs or views from specific clients by assigning those clients to a region where tags for those VOBs or views do not exist.

A ClearCase host is typically assigned to a region at install time. You can change this assignment later if necessary by following the procedures in *To Assign a Host to a New Registry Region* on page 47. To display a host's region, use the host node of the ClearCase Administration Console, the ClearCase program in Control Panel, or the **cleartool hostinfo –long** command.

Note: A VOB must have a tag in the region to which its server host is assigned. A VOB can have tags in other regions as well.

Other Shared Data

The registry is also the central repository for other shared data, including information about server storage locations and site-wide defaults.

Server Storage Locations

A server storage location is a shared directory on a ClearCase server that appears in a list of recommended storage locations when a user creates a VOB or a view from a GUI, and can also be used with the **–stgloc** option to **mkvob** and **mkview**. Storage locations for VOBs or views can be created during server setup, or by using the **cleartool mkstgloc** command or the ClearCase Administration Console.

Because a server storage location requires a global path, it can be valid only for a single region. Each region may have zero or more storage locations for VOBs or views.

Use of server storage locations on appropriate devices—a dedicated VOB or view server host or a certified NAS device—can simplify the creation, management, and maintenance of VOBs and views.

For more information about server storage locations, see *Creating Server Storage Locations for VOBs* on page 95 and the **mkstgloc** reference page.

Site Defaults

The ClearCase registry stores default values for a number of ClearCase host properties. These values are established by the site preparation process. Some can be changed with the **cleartool setsite** command or the ClearCase Administration Console (see *Changing Site Defaults* on page 46). Site defaults include:

- cache sizes used by newly created dynamic views
- default text modes used by newly created views
- default behavior for certain commands (for example, whether **checkout** uses the **–reserved** option)

See the **setsite** reference page for the complete list.

Most site defaults can be overridden on individual clients, either by using command-line tools or GUIs such as the ClearCase program in Control Panel.

Note: Site defaults apply to all clients of a registry host, regardless of their region.

The Registry Server Host

Any ClearCase host can be a registry server. In a mixed community of UNIX and Windows hosts, the registry server can be either a UNIX computer or a Windows computer. Because clients use RPCs to communicate with the registry server, platform differences are unimportant.

Note: In a ClearCase LT community, the registry server is always the ClearCase LT server.

Registry requests are compact and registry lookups are efficient, so the registry server process (actually handled by the **albd_server**) does not consume significant computing resources. The registry database does not usually grow very large (2-4 MB of storage can accommodate registry data for thousands of VOBs and views). Because the registry is an integral part of most ClearCase operations, the registry server host must be stable, available, and accessible to all hosts in the network. We recommend that you set up a backup registry host (as described in *ClearCase Registry Backup and Switchover* on page 50) that can take over registry server duties if the primary registry host fails. (ClearCase LT does not support backup registry hosts.) We also recommend backing up registry data as part of your normal backup routine. See *Backing Up ClearCase Registry Data* on page 195.

Defining a ClearCase Registry Host

Note: This section does not apply to ClearCase LT. In a ClearCase LT community, the registry server is always the ClearCase LT server.

Because a ClearCase community is defined in part by use of a common registry, the registry server host is usually the first computer in the community on which ClearCase is installed. The ClearCase installation program automates most of this process, and by installing the registry server host first, you simplify the process of installing and configuring other hosts in your ClearCase community. When you upgrade to a newer release, we recommend that you upgrade the registry server host first.

A ClearCase registry server does not have to be configured with support for local VOBs and views, although many communities can designate a VOB or view host as a registry server without degrading overall performance.

To convert an existing ClearCase host to a registry server without reinstalling, use one of the procedures in this section.

Note: After you define a new registry host, you must establish a registry password for the new registry. Otherwise, you cannot perform any privileged operations, such as creating public VOB tags or changing site-wide defaults. For more information, see *Setting and Using the Registry Password* on page 41.

To Define a UNIX Computer as a Registry Host

Use the following procedure to define a UNIX computer as a registry host.

1. Log on to the host. You must log in as **root** to have privileges to stop or start ClearCase, or to create directories and files in the /var partition of most hosts.
2. Stop ClearCase.
3. Create the directory /var/adm/rational/clearcase/rgy
4. Create the file /var/adm/rational/clearcase/rgy/rgy_svr.conf. This file must contain exactly one line, consisting of the text string **master**.
5. Restart ClearCase.

To Define a Windows Computer as a Registry Host

Use the following procedure to define a Windows computer as a registry host.

1. Log on to the host as a member of the Administrators group.
2. Stop ClearCase.
3. Open the ClearCase program in Control Panel.

4 Click the **Registry** tab, and then click **Use local host as registry server**.

5 Restart ClearCase.

Setting and Using the Registry Password

Because registry changes can have a significant impact on a ClearCase community, some registry operations are protected by the registry password. To set or change the registry password, use the **rgy_passwd** command. There is no default registry password. In a newly created registry, all protected operations fail until the registry password is established. Once the registry password is established, it must be supplied in conjunction with any operation that creates a public VOB tag or changes a site default. Even ClearCase privileged users must supply this password when they perform protected registry operations.

The Registry Client List

Every registry server maintains a list of the clients that have contacted it within the past 30 days. This client list is displayed by commands and GUIs such as **cleartool lsclients** and the ClearCase Administration Console; it is also used by the **rgy_switchover** command, as described in *ClearCase Registry Backup and Switchover* on page 50. You cannot manually add clients to this list or remove them from it.

Guidelines for Using Multiple ClearCase Registries

A registry server host can serve only one registry. Unless you are running ClearCase LT, which supports only a single registry host, you can establish multiple registries, each served by its own host, if your site includes multiple communities that do not need to, or must not, share VOBs or views. But it is generally simpler and no less secure to use the registry regions mechanism to limit the set of VOBs and views that a group of clients can access. A registry server does not consume significant computing resources or become a performance bottleneck even when serving many clients and providing access information for many VOBs and views. Any performance or scalability advantage that may be gained from using multiple registries is not likely to be significant, and is likely to be outweighed by the greater administrative overhead.

ClearCase provides no tools to automate the sharing of data between registries, so if you choose to implement multiple registries at a site, each one must be administered separately. For example, you must manually create object and tag entries in the registry for all VOBs and views created on hosts that use a different registry server, and site defaults must be established and managed separately for each registry.

Assigning a ClearCase Host's Registry and Region

Note: This section does not apply to ClearCase LT, which supports only a single registry region.

Each ClearCase host is normally configured at install time with the name of its registry server and the name of the region of that registry in which it will look up tags. To change either or both of these assignments:

1. Specify the new registry host and region:
   - On a UNIX host, edit the file /var/adm/rational/clearcase/rgy/rgy_hosts.conf to contain the name of the new registry host and optional backup registry host; then edit the file /var/adm/rational/clearcase/rgy/rgy_region.conf to contain the name of a region in the new registry. You must be **root** to change these files.
   - On a Windows host, click **Start > Settings > Control Panel**, and then start the ClearCase program. On the **Registry** tab, type the name of the new registry server host into the **Use registry server on host** text box, and then type the name of the appropriate region into the **Windows Region** text box.
2. Stop ClearCase.
3. Restart ClearCase.

You can verify that the new region assignment is in effect by running the **cleartool hostinfo –long** command.

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 registry region. You cannot use the ClearCase Administration Console to stop or start ClearCase on a host.

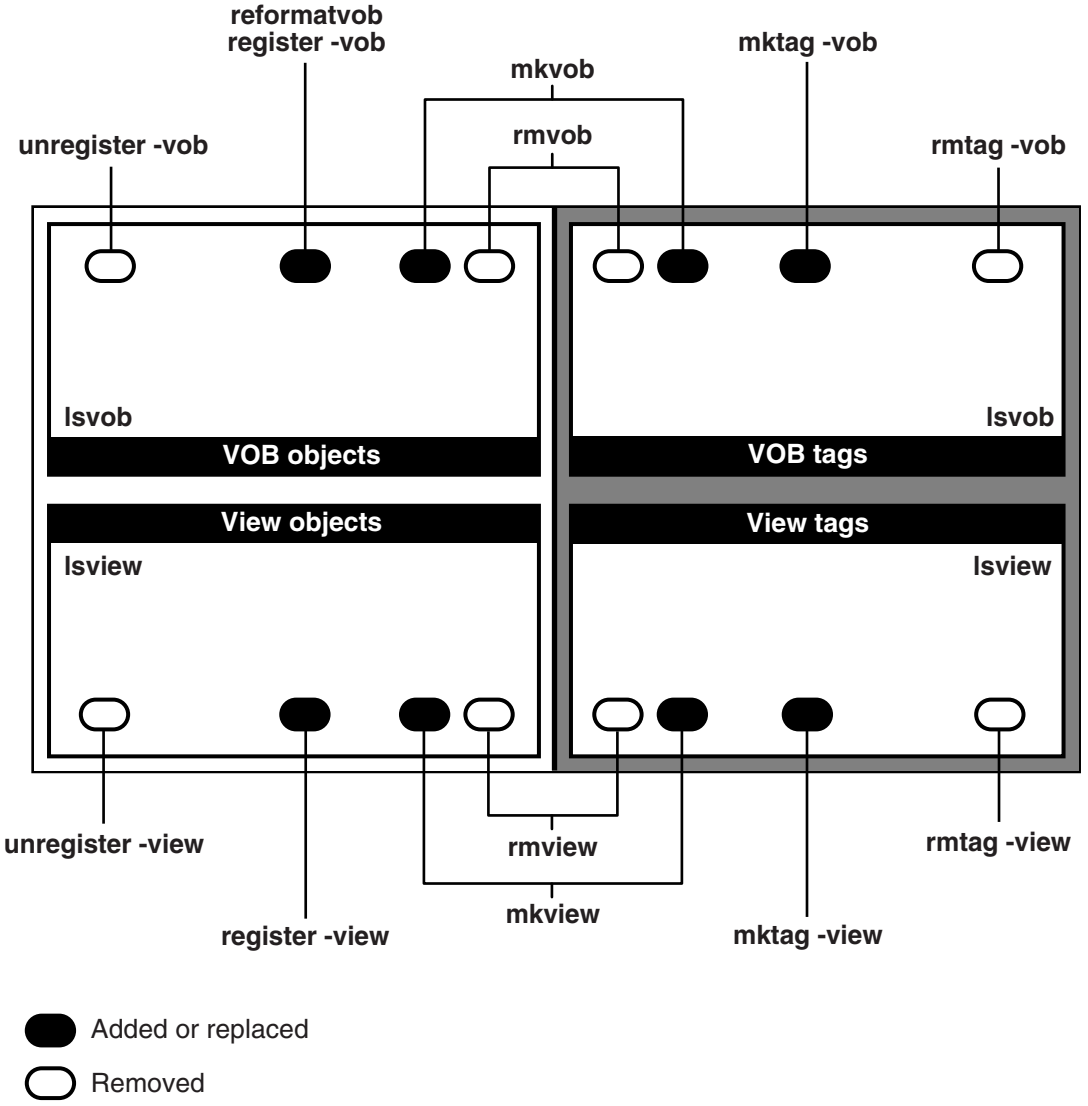
Managing Registry Data

ClearCase creates an object and a tag when you create a VOB or view. It removes the tag and the object when you remove a view. In many cases, no further administrative action is required. You may need to manipulate VOB and view objects and tags for any of the following reasons:

- To make a tag available in another registry region
- To change the name of a VOB or view
- After you move a VOB or view to a different host
- After you restore a VOB or view from backup media
- To clean up the registry after a VOB or view is removed by a nonstandard procedure

To inspect, create, and repair VOB and view objects and tags, use the **ClearCase Registry** node of the ClearCase Administration Console or any of the **cleartool** subcommands named in Figure 3, which shows how these commands affect registry data.

Figure 3 How cleartool Commands Affect Registry Data



Viewing Objects and Tags

Use the **VOB Objects** and **View Objects** subnodes of the **ClearCase Registry** node in the ClearCase Administration Console to inspect VOB and view objects:

1. Start 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 and view tags:

1. Start 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 the following registry information for a VOB or view:

- The tag
- The server host name
- The host-local storage access pathname
- The global pathname

For example:

cleartool lsvob –long /vobs/sources

```
  .
  .
Tag: /vobs/sources
  Global path: /net/neptune/vobstg/sources.vbs
  Server host: neptune
  .
  .
  Vob server access path: /vobstg/sources.vbs
  .
  .
```

cleartool lsview –long v5_integration

```
  .
  .
Tag: v5_integration
  Global path: /net/mars/viewstg/v5_integration.vws
  Server host: mars
  .
  .
  View server access path: /viewstg/v5_integration.vws
  .
  .
```

If a VOB or view is not listed in **lsvob** or **lsview** output, there is no tag for the VOB or view in the host's region (or, if the **lsview –region** option is used, no tag in the region specified).

Registering a VOB or View

Use the ClearCase Administration Console or **cleartool register** to create or change a VOB or view object. Common reasons to create or change an object include:

- Moving a VOB or view to a new host
- Moving a VOB or view to a new location on a host
- Renaming a VOB or view server host
- Any other operation that changes the host-local pathname of the VOB or view

To change the information in an object, use the ClearCase Administration Console to remove the object and create a new one, or use the **cleartool unregister** and **register** commands.

Creating, Removing, or Changing Tags

Use the ClearCase Administration Console or **cleartool mktag** to add, remove, or modify tags for existing VOBs and views. Common reasons to add or replace a tag include:

- Creating additional tags in other regions
- Renaming a VOB or view (a VOB or view's name is synonymous with its tag)
- Converting a private VOB to a public VOB or vice versa
- Any reason cited in *Registering a VOB or View*

To change the information in a tag, use the ClearCase Administration Console to remove the tag and create a new one, or use the **cleartool rmtag** and **mktag** commands.

To copy or move a tag to a different region, see *To Create Tags in a New Registry Region* on page 47.

Changing Site Defaults

You can use the ClearCase Administration Console to view or edit site-wide properties:

1. Start the ClearCase Administration Console.
2. Navigate to the **ClearCase Registry** node.
3. Click **Action > Properties** to view or change site defaults.

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.

Note: Some site defaults can be established only during site preparation and cannot be modified with the ClearCase Administration Console or the **setsite** command. Before you can change a site default, you must supply the registry password.

Adding a ClearCase Registry Region

Note: This section does not apply to ClearCase LT, which supports only a single registry region.

Take the following steps to add a new region to an existing ClearCase registry:

- Create the region
- Assign hosts to the region
- Populate the region with tags

To Create a New Registry Region

Use the ClearCase Administration Console to create a new region (in this example, the region is named **dev_win)**:

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:

cleartool mkregion –tag dev_win –tcomment "Windows ClearCase hosts"

To Assign a Host to a New Registry Region

A host's default registry server and region assignments are usually established when ClearCase is installed. To assign the host to a different region, follow the procedures in *Assigning a ClearCase Host's Registry and Region* on page 42.

To Create Tags in a New Registry Region

There are several ways to populate a newly created registry region with tags:

- The Region Synchronizer, a Windows GUI, makes it easy to import VOB tags and view tags from a UNIX region to a Windows region. To start the Region Synchronizer, click **Start > Programs > Rational Software > 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. (There is no region synchronizer application on ClearCase hosts running UNIX.)
- The ClearCase Administration Console provides a more general interface to regions and tags. It can be used to manipulate tags regardless of whether the registry host is running UNIX or Windows. You can use it to create and delete tags and to copy tags to another region. When you use the ClearCase Administration Console to manipulate tags, you must change the tags' global paths to make them valid for hosts in the new region. (The Region Synchronizer automates this operation for the specific case of importing a tag from a UNIX region to a Windows region.)
- You can use the **cleartool mktag** command.

Creating VOBs and Views in a ClearCase Registry That Has Regions

Every GUI and **cleartool** command that creates a VOB or view also creates an object and tag for it. The tag is created in the host's region (the region to which the host where you run the command is assigned). If you want the VOB or view to be available to hosts in other regions, you must create additional tags manually. These tags reference the object that was created along with the VOB or view, so you do not have to create a new object. Use the ClearCase Administration Console or the **cleartool mktag** command to create additional tags. For example:

After creating VOB /vobs/lib on **saturn** in the **dev_unix** region, run the following command to create a tag for the VOB in the **dev_win** region (used by Windows hosts):

```
cleartool mktag –vob –tag \lib –region dev_win –host saturn ^
–hpath /vobstg/lib.vbs –gpath \\saturn\vobstg\lib.vbs \\saturn\vobstg\lib.vbs
```

The **mktag** command in this example can be executed on a UNIX or a Windows host. However, it will not be able to validate the global path \\saturn\vobstg\lib.vbs unless it is executed on a Windows host.

Removing a Registry Region

To remove a registry region, reassign each host in the region to another region, as described in *To Assign a Host to a New Registry Region* on page 47, and 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 remove the region with the **rmregion** command:

cleartool rmregion –tag *region-name* **–rmall**

Renaming a Registry Server Host

ClearCase client hosts cache the name and IP address of their registry server host. If you rename the registry server host, you must change each client host's registry server host assignment, and then stop and restart ClearCase on each host to replace the cached registry server host name with the new one. To rename the network's registry server host:

1. Stop ClearCase on each client host.
2. Rename the registry server host. (Use the tools that the host's operating system provides.)
3. Change each client host's registry server host assignment, as described in *Defining a ClearCase Registry Host* on page 40, to specify the registry server's new host name.
4. Restart ClearCase on all hosts.

Registry Administration Guidelines

Consider the following guidelines regarding registry administration:

- The ClearCase Administration Console, which runs on Windows but can administer registries on Windows or UNIX, provides a graphical display of ClearCase registry data, which can help you understand the structure and contents

of registry data even if you prefer to use **cleartool** commands to manipulate this data.

- You cannot access a VOB or view (even to remove it) unless it has an object and a tag. Use the ClearCase Administration Console, **lsvob**, or **lsview** to see whether the tag is missing (no listing) or the object entry is missing (no storage paths appear in the **–long** output).

  If you cannot access a VOB or view, verify that it has a tag. If the VOB or view 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 **lsvob –long** or **lsview –long**. If the output includes incorrect pathnames (identical local and global pathnames, for example), correct the object and/or tag by using the ClearCase Administration Console (use the **VOB Objects** or **View Objects** nodes and the **VOB Tags** or **View Tags** nodes of the appropriate region). You can also use the **cleartool register** and **mktag** commands.
- To be used with dynamic views, VOB and view storage directories must be exported (on UNIX) or shared (on Windows). If a dynamic view cannot access a VOB or view storage directory on UNIX through the global path in its tag, verify that the directory is exported and, if it includes symbolic links to other directories, that they are exported too. If a dynamic view cannot access a VOB or view storage directory on Windows through the global path in its tag, verify that the directory is shared and that the share allows read and write access to all groups that access the VOB.
- Always use ClearCase utilities to remove VOBs and views. Do not use operating system utilities such as **rm**, **rmdir**, or GUIs such as Windows Explorer. They do not clean up registry entries.
- Do not remove tags before you delete a VOB or view storage directory; use **rmvob** or **rmview**, which removes the tags for you.
- When performing any of these tasks, execute the command on the host where the VOB or view storage directory resides, especially when you use the **–host –hpath** and **–gpath** options together:
  - Creating a VOB or view
  - Creating a VOB tag or view tag
  - Registering or unregistering a storage directory
  - Reformatting a VOB

  Doing so enables the command or GUI to validate the host-local pathname, which it cannot do when the command is executed from a remote host.

- Run the **rgy_check** utility periodically to diagnose problems and clean up obsolete or stranded registry entries

ClearCase Registry Backup and Switchover

Note: This section does not apply to ClearCase LT.

Access to the registry is critical for nearly all ClearCase operations. Because registry data is such an important resource, ClearCase allows you to designate any ClearCase host as a backup registry host and periodically copy registry data to it with the **rgy_backup** command. If the primary registry host fails, you can run the **rgy_switchover** program, which promotes the backup registry host to primary registry host and reconfigures client hosts to use it.

This section describes two scenarios that involve the failure of a registry host, one that assumes you have been using **rgy_backup** and one that assumes you have not (but have been backing up the primary registry host's file system, as described in *Backing Up ClearCase Registry Data* on page 195).

The ClearCase schedule service runs the **rgy_backup** command periodically on all hosts. On a ClearCase host that is not configured as a backup registry host, the command does nothing. On a ClearCase host that is configured as a backup registry host, the command copies the primary registry host's registry database and client list to the backup registry host, where it can be used if the host is promoted to primary registry host. For more information, see the **rgy_backup** reference page and *The ClearCase Scheduler* on page 217.

Note: To run the **rgy_backup** command, you must have write permission to the backup registry database directory.

To Define a Backup Registry Host

You can use the **–backup** option to the **rgy_switchover** command to define a new backup registry host. You can also define the backup registry host manually by using one of the following procedures.

Note: A backup registry host must be running the same release of ClearCase as the primary registry host.

To Define a UNIX Computer as a Backup Registry Host

Use the following procedure to define a UNIX computer as a backup registry host.

1. Log on to the host. You must log on as **root** to have privileges to stop or start ClearCase, or to create directories and files in the /var partition of most hosts.
2. Stop ClearCase.
3. Edit the file /var/adm/rational/clearcase/rgy/rgy_hosts.conf. The first line of this file is the name of the host's registry server. Add a second line that is the host name of the host itself.
4. Restart ClearCase.

The host becomes a backup registry host and begins running **rgy_backup** to back up the registry data on the primary registry host specified in the first line of /var/adm/rational/clearcase/rgy/rgy_hosts.conf.

To Define a Windows Computer as a Backup Registry Host

Use the following procedure to define a Windows computer as a registry host.

1. Log on as a local Administrator.
2. Stop ClearCase.
3. Click **Start > Settings > Control Panel**, and then start the **ClearCase** program. Click the **Registry** tab, and then type the name of the current host into the **Backup registry host** field.
4. Restart ClearCase.

The host becomes a backup registry host and begins running **rgy_backup** to back up the registry data on the primary registry host specified in the **Use registry server on host** box of the **Registry** tab.

To Promote a Backup Registry Host to Primary Registry Host

This scenario assumes that the **rgy_backup** program has been executing successfully on backup registry host **rgy2**. When the primary registry host **rgy1** fails, you can use **rgy_switchover** to promote **rgy2** to primary registry host. When **rgy1** becomes available again, you use **rgy_backup** and then **rgy_switchover** to return **rgy1** to the role of primary registry host.

To promote the backup registry host.

1. **Run rgy_switchover.** Make **rgy2** the primary registry host. Make **rgy3** the new backup registry host:

   rgy_switchover –backup rgy3 rgy1 rgy2

2 **Handle unreachable clients.** Record the names of any client hosts for which the switchover fails. You must reconfigure these hosts manually, following the procedure in *Defining a ClearCase Registry Host* on page 40, when they become available.

Note: ClearCase client hosts installed with no support for local VOBs or views cannot be reconfigured by **rgy_switchover** and always appear on the list of unreachable clients.

3 **Configure a new backup registry host**. While the primary registry host is down and the backup registry host is functioning as the primary registry host, we recommend that you configure an additional backup registry host to provide continuing **rgy_switchover** capability. Follow the procedures in *To Define a Backup Registry Host* on page 50.

To Restore the Primary Registry Host

Because registry contents can change frequently, you cannot simply restart a failed registry host after you have been using a backup registry host. Use this procedure if you have switched to a backup registry host (**rgy2** in this example) and want to restore **rgy1** as primary registry host.

1 When host **rgy1** becomes available again, deactivate its registry server. Follow the procedures in *Defining a ClearCase Registry Host* on page 40 to reconfigure **rgy1** to recognize **rgy2** as the registry host.

2 Make **rgy1** a backup registry host in preparation for its promotion to primary registry host:

   a Log on to **rgy1**. You must log on as **root** on UNIX or a local Administrator on Windows to have privileges to change the host's registry server status.

   b 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, Click **Start > Settings > Control Panel**, and then start the ClearCase program. 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**.

   c Stop ClearCase.

   d Restart ClearCase.

3 **Run rgy_backup manually on rgy1.** This updates the registry database on **rgy1** with the contents of the registry database on **rgy2.**

4 **Run rgy_switchover.** Make **rgy1** the primary registry host. Make **rgy2** the backup registry host by running the following command:

rgy_switchover –backup rgy2 rgy2 rgy1

5 **Handle unreachable clients.** Follow the procedure in *To Promote a Backup Registry Host to Primary Registry Host* on page 51.

Moving the Registry to a Host Not Configured for Registry Backups

If your primary registry host fails and you do not have a backup registry host, you can restore registry data that you have backed up from the primary registry host (using an ordinary file system backup tool) to another host, and then configure that host as a primary registry host.

In this scenario, primary host **rgy1** has failed and prospective registry host **rgy2** is not configured as a backup registry host. You must restore a file system backup of the rgy directory from **rgy1** to the rgy directory on **rgy2** and then reconfigure all clients of **rgy1** to use **rgy2** as their registry host. The following procedure, in which you restore registry database files from an ordinary file system backup and then run the **rgy_switchover** command, can partially automate this task.

1 **Restore rgy1 registry files from backup to rgy2**. Retrieve the ClearCase rgy directory and client_list.db file from backup and load its files into rgy/backup. When you are finished, the backup subdirectory includes all the registry data files. A registry host stores its client_list.db file in the same directory as the rgy directory (not in the rgy directory itself). If you do not have this file, **rgy_switchover** cannot reconfigure clients for you; you must reconfigure clients to use **rgy2** as their registry host by following the procedure in *Assigning a ClearCase Host's Registry and Region* on page 42.

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 files in the rgy directory, one file name per line. List only the name of each file, not the path to the file. For example:

```
bbase_object
bbase_tag
regions
.
.
.
view_object
view_tag
vob_object
vob_tag
vob_tag.sec
```

3 **Promote rgy2 to primary registry host.** Use the procedure described in *To Promote a Backup Registry Host to Primary Registry Host* on page 51.

ClearCase Network Administration in Mixed Environments

4

Whenever a ClearCase community includes users who access a common set of VOBs and views from both UNIX and Windows computers, most or all of the following requirements may need to be addressed:

- Common user and group names and common primary group memberships on both Windows and UNIX for all users and groups that access a common set of VOBs and views
- Cross-platform file system access, which is required when a dynamic view (MVFS) hosted on a Windows client must access a VOB or dynamic view hosted on a UNIX computer
- Client configuration issues and file naming conventions that address the differences between Windows and UNIX in areas such as case sensitivity, line termination in text files, and characters that are allowed in file names

If you are the administrator of a ClearCase community whose computers all run Windows or all run UNIX, you can skip this chapter.

Common User and Group Names

The identity with which a user logs on to the operating system of a ClearCase host establishes the credentials that control the user's access rights to artifacts in VOBs and views. Because many of these artifacts store a user name and one or more group names to identify the user and groups that can modify the artifact, each user who accesses a VOB or dynamic view from UNIX and Windows computers must have the same user name on both UNIX and Windows. In addition, the user's primary group and all other groups that access VOB objects must have the same name on both UNIX and Windows. Chapter 5 has more information about how credentials are used when determining access rights in VOBs and views.

Note: In ClearCase credentials, group membership is important only for file system objects such as directory and file elements. For VOB metadata such as labels and branches, only the owner's name is important. Group membership is not considered

How ClearCase Compares User and Group Names

The user and group name comparisons that ClearCase makes are case-insensitive on Windows and case-sensitive on UNIX. Passwords are not considered.

Each type of name (user or group) is compared with others in its namespace, but the relationship of the user and group names is not considered during the comparison.

Note: UNIX and Windows place different restrictions on the length of user and group names and on the characters allowed in them. The user and group names in your ClearCase community must be acceptable in both environments.

Credentials Mapping

When a process running on one OS type (UNIX or Windows) requests access to an object in a VOB or view on a different OS type, the process's credentials (user and group name) are mapped to a set of credentials on the other OS type.

The names in the credentials are compared and, if they can be matched, the combined *user.group* credentials are evaluated on the other OS to determine whether the requested access is allowed. If no matching user or group name can be found, the requested access will fail unless the object has access permissions for user **nobody** or group **nobody**.

Note: If a mixed-case or uppercase Windows user or group name fails a case-sensitive comparison on UNIX, the name is converted to lowercase and the comparison retried.

Credentials mapping is handled by each client process with the help of a **credmap_server** process that runs on each ClearCase server host.

UNIX Credentials for the ClearCase Server Process User

The ClearCase server process user account is a special Windows domain account (see *Domain User and Group Accounts* on page 320) under whose identity the **albd_server** runs on Windows computers. Mixed environments in which dynamic views on Windows access VOBs on UNIX require a UNIX user account to which the ClearCase server process user's credentials can be mapped. (No UNIX server processes run with the identity of this account. It is used only for credentials mapping when the ClearCase server process user requests access to a VOB or dynamic view on UNIX.)

You may create a new account for this purpose or use an existing one. Either way, the following guidelines apply:

- The account must be a member of each group that owns a VOB or dynamic view on UNIX that will be accessed from Windows.

- The account's primary group should have the same name as the primary group designated for Windows ClearCase users. (You must create this group on UNIX if it doesn't already exist.)

  Note: On Windows, the ClearCase server process user is a member of a single group, the ClearCase administrators group, that has special ClearCase privileges but no special privileges in the operating system. Even though some ClearCase server processes run as **root** on UNIX, we do not recommend mapping the ClearCase server process user account to **root** on UNIX. Privileged user status is not required for this account, and mapping it to **root** may create a security risk.

- The account can have any name that is convenient, unless either of the following conditions apply:

  - Windows computers use an NFS client to access VOBs and views on UNIX and that NFS client does not allow you to map the ClearCase server process user account to a UNIX account with a different name. (For more information, see *NFS Client Products* on page 297.)

  - UNIX VOB and view server hosts run an SMB server to provide Windows computers with access to VOBs and views and you do not want to map the ClearCase server process user account to an existing UNIX account. (For more information, see *SMB Server Products* on page 305.)

  In either of these cases, the UNIX account must have the same user name as the ClearCase server process user on Windows.

Cross-Platform File System Access

Because dynamic views must use a native network file system to access VOB or view data on the server host, cross-platform file system access is required whenever a Windows computer accesses a VOB or dynamic view hosted on UNIX. ClearCase provides its own file transfer mechanism, the ClearCase File Service (CCFS), for use by snapshot and Web views. CCFS is used whenever a snapshot or Web view on one OS type accesses a VOB on the other OS type. For more information, see *The ClearCase File Service* on page 60.

Network File System Protocols Used for VOB and View Access

When a dynamic view on Windows needs to access a VOB or dynamic view on UNIX, it must use one of the following network file system protocols to do so:

- **NFS.** The NFS (Network File Service) protocol is the native protocol supported by most UNIX computers for network file system access. NFS client products for

Windows use the NFS protocol to access UNIX file systems. You install an NFS client product on each Windows computer from which you want to access UNIX VOBs and views.

- **SMB.** The SMB (Server Message Block, also referred to as the Common Internet File System or CIFS) protocol is the native protocol that Windows computers use for network file system access. SMB servers that run on UNIX computers allow Windows computers to access UNIX VOBs and views by using native Windows protocols. You install an SMB server product on each UNIX VOB or view server that you will access from a Windows client.

Note: Because Rational does not support any NFS server solution for Windows or SMB client for UNIX, a dynamic view on a UNIX computer cannot access a VOB or shared dynamic view hosted on a Windows computer.

Table 1 lists the protocols that ClearCase clients can use to access VOB data. Table 2 lists the protocols that ClearCase clients use to access view storage. In these tables, protocols native to their respective computer platforms are labeled Native. The ClearCase File Service (CCFS) is included in ClearCase. All other protocols require third-party software support.

Table 1 Protocols for ClearCase Client Access to VOB Data

Client platform	Access to VOB data on UNIX	Access to VOB data on Windows
Windows (dynamic views)	Third-party NFS or SMB	Native SMB
Windows (snapshot views)	CCFS, third-party NFS or SMB	Native SMB
UNIX (dynamic views)	Native NFS	Unsupported
UNIX (snapshot views)	Native NFS	CCFS

Table 2 Protocols for ClearCase Client Access to View Data

Client platform	Access to view data on UNIX	Access to view data on Windows
Windows (dynamic views)	Third-party NFS or SMB	Native SMB
Windows (snapshot views)	Third-party NFS or SMB	Native SMB
UNIX (dynamic views)	Native NFS	Unsupported

Table 2 Protocols for ClearCase Client Access to View Data

Client platform	Access to view data on UNIX	Access to view data on Windows
UNIX (snapshot views)	Native NFS	See note.

Note: When a view has been created with the **–ngpath** option, ClearCase hosts that use snapshot views but cannot run a local **view_server** process use the native ClearCase RPC mechanism to access a **view_server** on another host. This configuration is typical of ClearCase LT.

Table 3 lists the protocols used by a **view_server** process to access VOB data.

Table 3 Protocols for view_server Access to VOB Data

View server platform	Access to VOB data on UNIX	Access to VOB data on Windows
Windows	CCFS, third-party NFS, or SMB	Native SMB
UNIX	Native NFS	CCFS

Network-Attached Storage and Cross-Platform File Access Tools

Rational supports several network-attached storage devices for storage of VOB and view data. These devices can be configured to host VOB or view storage and provide native file system access to both UNIX and Windows computers without the need to install any other software on ClearCase client or server hosts. For more information, see *ClearCase and Network-Attached Storage Devices* on page 15.

In addition to these NAS devices, Rational supports two categories of third-party cross-platform file-access tools for use with ClearCase:

- SMB servers for UNIX platforms, which provide Windows clients with native SMB/CIFS access to file systems on UNIX hosts.
- NFS client programs for Windows, which enable Windows clients to use the NFS protocol to access the file system of a UNIX computer.

Any of these tools can be used to enable a dynamic view on a Windows computer to access a VOB hosted on a UNIX computer and to enable a Windows computer to use a shared dynamic view hosted on a UNIX computer.

For more information about supported cross-platform file system access tools, see Appendix A.

The ClearCase File Service

The ClearCase File Service is a TCP/IP-based mechanism that enables cross-platform file transfers between VOB servers and snapshot views. It supports access by snapshot views on Windows computers to VOB data on UNIX computers and access by snapshot views on UNIX computers to VOB data on Windows. Web views also use CCFS for the same types of access.

Note: When snapshot or Web views access VOBs on the same platform type, they always use the platform's native network file system and do not use CCFS.

If a ClearCase client uses only snapshot or Web views, no third-party cross-platform file-access solution is needed to access VOB data.

When CCFS is enabled, file transfers between snapshot view clients and VOB servers (for example, operations such as checking out, checking in, and creating and updating snapshot views) take place over a standard TCP/IP connection. File transfers between a VOB server and a snapshot view's view server also use this TCP/IP connection.

CCFS is always enabled on a UNIX computer running ClearCase. It may be enabled as needed on Windows by using the ClearCase program in Control Panel.

Enabling CCFS on Windows

CCFS is disabled by default on Windows computers running ClearCase. To enable CCFS on a Windows computer:

1. Click **Start > Settings > Control Panel**, and then start 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.
3. Click **OK**.
4. Shut down and restart the computer.

CCFS and Remote Windows View Server Hosts

ClearCase hosts that use snapshot views but cannot run a local **view_server** process use CCFS as their sole file transfer mechanism when accessing UNIX VOBs. Snapshot views on these clients require another Windows computer to run the **view_server** processes and host the view storage directories associated that support these views.

CCFS must be enabled on any Windows computer that performs this role. A Windows computer configured to use an NFS client product to access UNIX VOBs cannot provide **view_server** support for ClearCase clients that cannot run a local **view_server** process.

Case-Sensitivity

UNIX and Windows observe different conventions for case-sensitivity in file name lookup: UNIX is case-sensitive; Windows is not. These operating system conventions, which are also typically observed by applications and users, can cause problems when users working on UNIX and Windows computers share a common set of VOBs and views. There are two aspects of case-sensitivity in ClearCase:

- Regardless of how you configure ClearCase, all ClearCase applications that refer to pathnames in VOBs, including **cleartool** and **clearmake**, are case-sensitive on both UNIX and Windows.
- The MVFS on UNIX is always case-sensitive. The MVFS on Windows can be configured to support various case-sensitivity and case-preservation options.

In general, you can use ClearCase on either Windows or UNIX with little consideration for case-sensitivity issues. If you encounter unexpected behavior in a mixed environment, read this section.

General Recommendations

To avoid common problems related to differences in case-sensitivity, follow these general recommendations:

- In VOBs and views, avoid creating file or directory names that differ only in character case. A case-insensitive MVFS cannot distinguish between two file names that differ only in their capitalization, and attempts to access such files produce indeterminate results.
- Configure the MVFS on Windows to be case-insensitive (this is the default setting).
- If you use an NFS client or SMB server product, disable any automatic case conversion features it may have (See Appendix A).

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

Case-Sensitivity and the MVFS

The multiversion file system (MVFS) is a feature of ClearCase that supports dynamic views. Because it is a file system, its case-sensitivity behavior is important and may need to be reconfigured in some mixed networks. For more information, see *The Multiversion File System* on page 170.

Note: ClearCase LT does not support the MVFS.

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

On Windows, the native file systems are case-insensitive and case-preserving. Because native Windows file systems perform case-insensitive file lookup, the MVFS, in its default configuration on Windows, also performs case-insensitive file lookup. Native Windows file systems preserve case on file-creation operations. The MVFS is configured to convert file names to lowercase on Windows.

To reconfigure the MVFS on Windows to use different case-sensitivity options, use the **MVFS** tab in the ClearCase program in Control Panel:

- 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 character case in the names of all files it creates.

Changes to MVFS options do not take effect until you restart your computer.

Note: Windows applications that do not specify file names with the correct case may fail in a case-sensitive MVFS.

When to Use a Case-Preserving MVFS

In the default mode (**Case Insensitive MVFS** enabled, **Case Preserving** disabled), the MVFS converts to lowercase the names of view-private files created in a dynamic view. This behavior can interfere with the operations of tools that intentionally create files with mixed-case names. For example, if a Java development environment creates a file

named util.JAR in a case-insensitive MVFS, it will be written as util.jar. Development tools that reference the file as util.JAR will not find it. To correct this kind of problem, configure the MVFS to use case-preserving mode (clear **Case-Insensitive MVFS** and select **Case-Preserving**).

Case-Sensitivity in Snapshot and Web Views

Snapshot and Web views use the native file system on the host where the snapshot view directory is located to read and write files. On Windows, these file systems perform case-insensitive file lookup and generally preserve case when creating file names. On UNIX, file lookups are always case-sensitive, and filename creation is always case-preserving.

Configuring Text Modes for Views

UNIX and Windows observe different conventions for writing line terminations in text files. UNIX utilities and applications normally terminate lines with a single <LF> (line feed, or new line) character; Windows utilities and applications 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.

These different conventions can create line-termination problems in text files that are edited on both UNIX and Windows platforms. For example, a file that contains

```
abc
def
ghi
```

would look like this if it were created by a Windows editor like Notepad and read by a UNIX editor like **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 the UNIX editor and read by the Windows editor:

```
abc■def■ghi
```

To better support parallel development in mixed 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 applies only 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 default value stored in the ClearCase registry determines which text mode is used when a new view is created. This default can be overridden on the command line and in GUIs that create views. For details about setting the default, see the **setsite** reference page.

You can create a view in one 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 always created in **transparent** text mode. To create a view in **transparent** text mode regardless of the site default, clear the **Use interop (insert_cr) text mode** check box in the **Advanced** options of the View Creation Wizard, or use the **–tmode transparent** option to the **mkview** command.

  If all developers at your site use the same development platform (Windows or UNIX) or use tools that are compatible with either line-termination convention, all views should be created in **transparent** text mode.

- **insert_cr text mode.** In a view created in **insert_cr** text mode, ClearCase inserts a <CR> character before every <LF> character. To create a view in **insert_cr** text mode, select the **Use interop (insert_cr) text mode** check box in the **Advanced** options of the View Creation Wizard or use the **–tmode insert_cr** option to the **mkview** command.

- **strip_cr text mode.** In a view created in **strip_cr** text mode, ClearCase strips the <CR> character from every <CR><LF> sequence. To create a view in **strip_cr** text mode, use the **–tmode strip_cr** options to the **mkview** command. You cannot create a view in **strip_cr** mode from the View Creation Wizard.

In a snapshot view created in either **insert_cr** or **strip_cr** text mode, ClearCase adds or removes the <CR> characters when it updates the view. In a dynamic view, ClearCase adds or removes the <CR> characters when 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.

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 –properties –full** command.

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; a user who edits 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 to those platforms that require different line-termination conventions.

If most of your users are editing text files 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 users are editing text files on Windows:

- Windows clients should use views created in **transparent** text mode.
- UNIX clients should use views created in **strip_cr** text mode.

Regardless of the policy you adopt, it is important to maintain a consistent combination of client platform, view text mode, and element. For example, if a user on UNIX creates a version of an element in a view that has a **strip_cr** or **insert_cr** text mode and another user on UNIX creates a version of the same element in a **transparent** text mode view, the two versions will be difficult to compare or merge.

Enabling Interop Text Mode Support in Older VOBs

VOBs created by recent releases of Rational ClearCase are compatible with views in any text mode. VOBs created with older versions of ClearCase are compatible only with views in transparent text mode until you run the **msdostext_mode** command on the VOB.

Only the VOB owner or a 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 support for **strip_cr** and **insert_cr** text modes, so that this information is recorded for newly created versions.

With the **–d** option, **msdostext_mode** disables support for **strip_cr** and **insert_cr** text modes.

The **msdostext_mode** command does not convert or modify files in any way. It affects only the information recorded for text file versions in the VOB database.

To Determine Whether a VOB Supports Interop Text Modes

To determine whether a VOB supports interop text modes, use the following command:

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, run this command.

cleartool dump vob:\pc_src

If the output of this command contains a line similar to this one, 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 as follows:

1. Synchronize all VOB replicas.
2. Lock all replicas, specifying **–nusers** *vob_owner*, where *vob_owner* is the user name of the VOB owner of the replica.
3. As *vob_owner*, run **msdostext_mode** on each replica.
4. Unlock all replicas.

Differences in Supported Character Sets

Various characters that are allowed in UNIX file names are not allowed in Windows file names. File names that include these characters are not recognized by the MVFS on Windows and cannot be loaded into a Windows snapshot view. VOB element names that include these characters are not visible in Windows views. Table 4 lists these characters.

Table 4 Characters Not Allowed in Windows File Names

?	*	/	\	|	<	>

VOB and View Access Controls

5

This chapter describes how Rational ClearCase controls access to VOB and view data.

Fundamentals of VOB and View Access Control

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

Users and Groups

ClearCase does not have its own implementation of user and group accounts. Instead, the identity with which a user logs on to the operating system of a ClearCase host establishes the user's ClearCase credentials. Because these credentials are evaluated whenever and wherever a user requests access to an object under ClearCase control, operating system definitions for user names, group names, and each user's group memberships must be consistent on every ClearCase host. This consistency is usually achieved by means of a networkwide account database such as a Windows domain or the UNIX Network Information System (NIS).

Note: In environments where users access a common set of VOBs and views from UNIX and Windows hosts, this consistency must extend to both platform types (user and group names as well as each user's group memberships must be the same on UNIX as they are on Windows). For more information, see *Common User and Group Names* on page 55.

The Primary Group

A user can be a member of one or more groups, one of which is distinguished as the user's primary group. We recommend that all ClearCase users in a community be members of the same primary group. This can be a group that already exists, or one that you create expressly for use by the ClearCase community.

The primary group is defined in different ways on Windows and UNIX:

- On UNIX, the primary group is defined in the user's entry in the NIS **passwd** database.
- On Windows, the primary group is specified when the user's domain account is created. However, if the user is a member of multiple groups, the primary group name is not always returned when an application requests it. Any ClearCase user who is a member of multiple groups must set the user environment variable CLEARCASE_PRIMARY_GROUP to the name of a group of which he is a member and that has been designated the primary group for ClearCase users on Windows (see *Setting the ClearCase Primary Group* on page 321).

Limitations When a User Belongs to More Than 32 Groups

If a user is a member of more than 32 groups, ClearCase recognizes only the first 32 groups (in numerical order by GID on UNIX or SID on Windows). If the user environment variable CLEARCASE_GROUPS exists for any user, ClearCase will consider the semicolon-separated list of group names specified in the value of this variable first when determining the list of groups to which the user belongs.

Privileged Users and Groups

A typical ClearCase community includes two classes of user:

- **Ordinary users** have rights to modify or delete ClearCase resources (VOBs, views, and the objects they contain) that they create or that are assigned to any group of which they are a member.
- **Privileged users** have unrestricted rights to create, modify, and delete all ClearCase resources.

In ClearCase LT, all privileged operations must be executed directly on the ClearCase LT server host.

- On a ClearCase host or ClearCase LT server running UNIX, the privileged user is the **root** user. (Some limitations apply to a **root** user logged on to a remote host. These limitations are described in detail in *Restricted Privileges for Remote root* on page 71.)
- On a ClearCase host running Windows, the privileged user is any member of the ClearCase administrators group, described in *Domain User and Group Accounts* on page 320.
- On a ClearCase LT server running Windows, the privileged user is any member of the Administrators group on the ClearCase LT server host.

The ClearCase Server Process User

On ClearCase hosts running Windows, the **albd_server** program runs with the identity of a special user account, created during ClearCase site preparation, known as the ClearCase server process user. This user is a member of the ClearCase administrators group and therefore has privileged user status. For more information about the ClearCase server process user, see *Domain User and Group Accounts* on page 320. (There is no ClearCase server process user on ClearCase LT.)

Restricted Privileges for Remote root

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

- When a user logged on as **root** attempts to access a view on a remote UNIX host, the user's identity is interpreted as **nobody.nobody** (unidentified user, unidentified group). This interpretation is typical of NFS implementations on UNIX and provides a level of security comparable to that provided by NFS. Because ClearCase does not support any mount option that controls how requests for access by remote **root** are treated, it does not allow view access by remote **root** unless the view specifies explicit access rights for **nobody.nobody**.
- Operations that change the user or group ownership of a view-private file, such as the UNIX **chown** command, fail when run by remote **root**.

User Processes

When a process requests access to VOB or view data, ClearCase evaluates the process's credentials to determine whether the requested form of access is authorized. The following process credentials are important in making this determination:

- **User.** The name of the user who starts the process.
- **Primary group.** The primary group of the user who starts the process.
- **Supplemental group list.** 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 collectively as the process's groups.

ClearCase Objects

The following ClearCase objects are subject to access control:

- VOBs

- Elements and versions
- Types and instances of types, such as labels, branches, and attributes
- Unified Change Management objects, such as projects, activities, and streams
- VOB storage pools
- Views
- In dynamic views, view-private files, view-private directories, and derived objects

Each object 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, which 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.

Protection Modes

The protection mode for a ClearCase object is summarized in Table 5. Information about an object's protection mode usually takes the form of a single-character abbreviation of each Boolean value in the protection mode; these abbreviations appear in Table 5.

Table 5 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 a directory element named `lib`. Suppose the owner and group of `lib` 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 (**akp**) and group (**clearusers**), as follows:

cleartool describe lib

```
...
  Element Protection:
    User : akp         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 6.

Table 6 Protection Mode Digits for a ClearCase Object (Part 1 of 2)

Digit	Read permission?	Write permission?	Execute permission?
0	No	No	No
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes
4	Yes	No	No
5	Yes	No	Yes

Table 6 Protection Mode Digits for a ClearCase Object (Part 2 of 2)

Digit	Read permission?	Write permission?	Execute permission?
6	Yes	Yes	No
7	Yes	Yes	Yes

A sequence of three digits 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 VOB and View Data

Whether a process has access to an object in a VOB or view 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 the object. See the descriptions in *Access Control for VOBs and VOB Objects* on page 75 and *Access Control for Views and View Objects* on page 81.

In certain cases, ClearCase grants a process access to an object only if the process has access to one or more containing objects as well. For example, to create a view-private file in a dynamic view, the user must have write permission for the directory that will contain the file as well as for the view itself.

Access Control for VOBs and VOB Objects

VOBs and the objects they contain are subject to 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 VOB created on Windows. On UNIX, it contains the group list of the VOB owner.

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

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

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

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

Permission to Create VOBs

Any user can create a VOB.

Permission to Delete VOBs

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

Permission to Read VOBs

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

Permission to Write VOBs

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

Permission to Execute VOBs

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

Access Control for Elements

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

- **Owner.** The initial owner is the user ID of the process that creates the element.
- **Group.** The initial group is determined differently on UNIX and Windows hosts.
  - On UNIX, it is the primary group ID of the process that creates the element.
  - On Windows, it is the primary group ID of the process that creates the element if that group is on the VOB's group list. Otherwise, it can be any group that appears on both the group list of process that creates the element and the group list of the VOB.

  On both Windows and UNIX, 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. If your **umask** is 0, a directory element initially has read, write, and execute permission for all user categories. Otherwise, the initial read, write and execute permissions are determined by the value of your **umask**.
- On Windows, a file element initially has only read permission for all user categories. You must explicitly add execute permission for an element by using the **cleartool chmod** command. A directory element initially has read, write, and execute permission for all user categories.

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

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

After an element is created, the element owner, the VOB owner, or a 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, it has a single element: the VOB root directory. This element is the container for all 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 additional elements. To create an element, a process must also have permission to check out a version of the directory element that will contain the new element. See *Permission to Write Elements*.

Permission to Delete Elements

Only the element owner, the VOB owner, or a privileged user can delete an element. Deleting an element with the **cleartool rmelem** command is not the same as removing the element's name from a version of a directory using **cleartool rmname**. For more information, see *Permission to Write Elements*.

The creator of a version, the element owner, the VOB owner, or a 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 VOB and View Data* on page 74.

Permission to Write Elements

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

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

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

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

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

When a directory element is checked out, you can modify the directory by creating elements or by removing elements from it. Removing an element's name from a version of a directory with 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 VOB and View Data* on page 74. In addition, two special cases can restrict permission to execute an element:

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

  cleartool chmod +x command.exe

  If you add an executable program to source control, you cannot execute it while it is checked in unless you take this step.

- On UNIX, when a VOB is mounted with the **nosuid** mount option, checked-in setuid executables cannot run.

Access Control for Other VOB Objects

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

- Metadata types, such as label types, branch types, and attribute types
- Unified Change Management objects, such as projects, activities, 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 a 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 a 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 a privileged user.

Permission to Delete Other VOB Objects

The owner of the object, the owner of the VOB, or a 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 instance can delete that instance.

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 a privileged user can change a type or a storage pool.

Locks on VOB Objects

Access controls on VOB objects are intended to provide 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 a 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 while it is locked.

Locking Type Objects

You can lock type objects to prevent 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 branches other than **main**.
- Locking a label type prevents anyone from creating or moving an instance of that label type. Labeling all the element versions used in a particular release, and then locking that label, provides an easy way to recreate the release later.

Access Control for Views and View Objects

Views mediate user access to VOB data. 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 before you can 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 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 establish access rights to files and directories in the snapshot view, including copies of element versions. Creating, deleting, or modifying elements in a snapshot view requires the process to have the appropriate permissions for those elements.

Note: On UNIX hosts, ClearCase treats view access requests from a remote **root** user as requests from the user **nobody.nobody**. See *Restricted Privileges for Remote root* on page 71 for details.

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, 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 in Windows Explorer or ClearCase Explorer 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 UNIX **umask** 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 v5_integration

  ```
  * v5_integration          /net/mars/viewstg/v5_integration.vws
  .
  .
  .
  Owner: akp        : rwx (all)
  Group: clearusers : rwx (all)
  Other:            : r-x (read)
  ```

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

  cd /net/mars/viewstg
  ls –ld v5_integration.vws

  ```
  drwxrwxr-x  6  akp  clearusers  512  Nov 10 11:29
  v5_integration.vws
  ```

  The output of this command shows the view's owner (`akp`), group (`clearusers`), and protection mode (`drwxrwxr-x`). On some UNIX systems, 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 a 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 78. 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 85.

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 VOB and View Data* on page 74.

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 78. 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 85.

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 VOB and View Data* on page 74.

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 78. 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 86.

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 VOB and View Data* on page 74.

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 determine access to those directories and files.

In a dynamic view, the initial owner, group, and protection mode for a view-private file are determined differently on UNIX and Windows.

Initial Owner, Group, and Protection Mode on UNIX

On UNIX, the initial owner, group, and protection mode for a view-private file are determined using the following rules:

- **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 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 UNIX **umask** 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.

Initial Owner, Group, and Protection Mode on Windows

On Windows, the initial owner, group, and protection mode for a view-private file are determined using the following rules:

- **Owner.** The initial owner is the user of the process that creates the file or directory.
- **Group.** The initial group is assigned in one of two ways based on the group of the process that creates the file or directory:
  - If the process's primary group is the same as the VOB's group, that group is assigned.
  - Otherwise, the process's group list is compared with the VOB's supplementary group list and the first group that appears on both lists is assigned.
- **Protection mode.** A view-private file or directory initially has read, write, and execute permission for all users.

  You can use the **cleartool describe** command or the **Properties of File** or **Properties of Directory** dialog box in ClearCase Explorer or Windows Explorer to display the owner, group, and protection mode for a view-private file or directory.

You cannot change the owner or group of a view-private file or directory. You can use the **Read-only** check box in Windows Explorer **Properties** dialog box or the **attrib +R** (equivalent to mode 777) and **attrib –R** (equivalent to mode 555) 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 83.

If the containing directory is an element version, the process must have write permission for the element. See *Permission to Write Elements* on page 78. 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 83.

If the containing directory is an element version, the process must have write permission for the element. See *Permission to Write Elements* on page 78. 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 83.

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 VOB and View Data* on page 74.

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 83.

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 VOB and View Data* on page 74.

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 83.

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 VOB and View Data* on page 74.

Access Control for Derived Objects

A derived object (DO) is a file created in a dynamic view by **clearmake** or **omake** or by 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 83.

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 and becomes 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 a 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 VOB and View Data* on page 74.

A winked-in DO has the owner, group, and protection mode it had as a shared DO. If you change a winked-in DO in the view, such as changing its permissions or writing to it, ClearCase converts the DO to a view-private copy.

ClearCase and Native File System Permissions

ClearCase maintains data for VOBs and views in ordinary directories, the VOB storage directory and the view storage directory, in the native file system on each VOB and view host. ClearCase manages all access to these directories; users never read from or write to these directories directly. File system protections are managed so that a user who has no rights to access a VOB object also has no rights to access any file in which the object's data is stored.

Warning: Never change native file system permissions on any part of any VOB storage directory or view storage directory.

If you have inadvertently changed permissions on a VOB or view storage directory, VOB or view access may be severely restricted, but you may be able to repair the damage. See *Repairing Storage Directory ACLS on NTFS* on page 255.

A snapshot view directory is a native file system directory that contains copies of element versions as well as other 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.

VOB Administration 6

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

Introduction to VOBs and VOB Administration

Any ClearCase development environment requires one or more VOBs. Because VOBs are the principal repository for software artifacts under ClearCase control, VOB administration is one of the ClearCase administrator's most important tasks.

VOB administration tasks include the following:

- Configuring VOB hosts
- Creating VOBs
- Enabling access to VOB data
- Importing data into a VOB from another configuration management system
- Backing up and recovering VOB data
- Managing VOB storage
- Monitoring VOB integrity

Access to VOB Data and Metadata

A VOB is a repository for data (versioned artifacts like files and directories) and metadata (branches, labels, event records, and so on). VOB data must be accessed through a view. Most VOB metadata can be accessed directly by using ClearCase GUIs and **cleartool** commands.

VOB data can be created by users or imported from another configuration management system. To create, import, or access VOB data, you must use a view, and you must have rights to access the VOB and the objects in it. Chapter 5 provides detailed information about VOB and view access controls. Chapter 9 provides more information about views.

VOB metadata can be created in several ways:

- As an indirect result of a command that creates a VOB or an object in a VOB. For example, commands that create VOBs also create VOB attributes and UUIDs; the **cleartool mkbranch** command creates a branch and local instance of a branch type if needed.
- As the direct result of a command that creates or manipulates a specific type of metadata, for example the **cleartool mk***object***type** commands (for example, **mkeltype** and **mkbrtype**), which create specific type objects that are described in detail in Chapter 7.

The **cleartool describe** command (and its graphical counterpart), as well as commands like **lstype** and **lsvob** provide ways to view and, in some cases, edit VOB metadata.

VOB Attributes

All VOBs have the same on-disk directory structure, and all have similar administrative requirements. However, certain VOBs may perform special functions, as determined by attributes attached to the VOB object when the VOB is created.

- A VOB with the **UCM VOB** (sometimes displayed as **sumvob**) attribute holds components that are part of a UCM project.
- A VOB with the **Project VOB** attribute holds metadata about UCM artifacts such as projects, baselines, folders, and components.

A VOB that has both attributes stores UCM project metadata and UCM components; a VOB that has neither attribute is an ordinary VOB. Ordinary VOBs and UCM component VOBs are permanent repositories for versioned artifacts. They account for the bulk of the VOBs in any ClearCase community.

Any VOB can also be part of an administrative VOB hierarchy. For more information about administrative VOB hierarchies, see Chapter 7.

In a small ClearCase community, a single VOB can be an administrative VOB, a PVOB, and either an ordinary or a UCM component VOB. It is far more common for a community to require one or more administrative VOBs and a number of ordinary VOBs or, if UCM is in use, a PVOB and a number of component VOBs. Larger communities may require multiple administrative VOBs or PVOBs.

VOB Schema Versions

Every VOB has a database schema version that denotes the format of the VOB database and determines the types of data and metadata that the VOB can store. ClearCase supports two VOB schema versions:

- Schema version 53, which is supported on all platforms except ClearCase LT for Windows.

- Schema version 54, which is supported on all Windows platforms and many UNIX platforms. Schema version 54 provides better support for Windows security identifiers and is required when using ClearCase in a Microsoft Active Directory environment. Schema version 54 also provides support for VOB database files larger than 2 GB.

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

To Change the VOB Schema Version

To upgrade an existing schema version 53 VOB to schema version 54:

1. Reinstall ClearCase on the VOB host and specify during installation that you want the host to support VOB schema version 54. (Some UNIX platforms do not support this schema version.)
2. After the reinstallation is complete, all VOBs on the host are inaccessible until you reformat them with the **cleartool reformatvob** command.

If you move a schema version 53 VOB to a host that supports schema version 54, the VOB will be inaccessible until you reformat it.

Note: You cannot reformat a VOB to a lower-numbered schema version.

VOB Feature Levels

A feature level is an integer that defines the set of ClearCase features that a VOB supports. Whenever a ClearCase release introduces features that require support in the VOB database, you must raise the feature level of a VOB before clients can use the new features when accessing data in that VOB. The primary purpose of feature levels is to help manage ClearCase release upgrades when VOBs are replicated (with ClearCase MultiSite) to server hosts that run different ClearCase releases.

Every ClearCase release is associated with a feature level. For information on the feature levels that this release supports, see the *Release Notes* for Rational ClearCase and ClearCase MultiSite.

To Display the VOB Feature Level

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

To display the feature level of a replica family, use the command **cleartool describe replica:***replica-name***@***vob-tag*. For example:

cleartool describe replica:tokyo@\dev

```
replica "tokyo"
  created 20-Aug-02.13:35:37 by akp (akp.user@boron)
...
  feature level: 2
...
```

To display the feature level of a VOB family, use the command **cleartool describe vob:***vob-tag*. For example:

cleartool describe vob:/vobs/dev

```
versioned object base "/vobs/dev"
  created 15-Aug-01.14:19:03 by jjp (jjp.user@mars)

...
 VOB family feature level: 2
...
```

Note: Before you set the feature level for a newly created replica, its value is recorded as `unknown`. For example, if you use the **describe** command to show the properties of a new replica, the output looks like this:

cleartool describe replica:sanfran_hub@/vobs/dev

```
...
  feature level: unknown
```

To Change the VOB Feature Level

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

1. Log on to the VOB host as the VOB owner or privileged user.
2. Issue the **chflevel** command with the **–auto** option. The command lists each VOB served by the host. It then offers to raise the feature level of each unreplicated VOB that is not at the highest feature level supported by the release of ClearCase installed on the host.

When you use MultiSite, each replica in a VOB family has a feature level. Replicas in the same family can have different feature levels. The VOB family itself has a feature level, which must be less than or equal to the lowest replica feature level found among members of the VOB family. You cannot raise the feature level of a VOB family above the lowest feature level of any replica in that family. For more information about feature levels and VOB replicas, see the *Administrator's Guide* for Rational ClearCase MultiSite.

VOB Server Host Configuration Guidelines

A host on which a **vob_server** process runs is a VOB host. A ClearCase LT community has a single VOB host, the ClearCase LT server. A ClearCase community may have many VOB hosts. Selecting and configuring an appropriate VOB host are crucial to obtaining satisfactory ClearCase performance.

As described in *ClearCase Server Processes* on page 8, a VOB host runs a number of server processes for each VOB it supports. Because an active VOB host may have to support dozens, or even hundreds, of such processes, and support network access from many clients, we recommend that you follow the guidelines in this section when specifying the hosts's physical memory, storage, processor, and network interface hardware. In addition, you may want to use the performance tuning procedures described in Chapter 16.

Note: Windows VOB hosts should run the server, rather than the workstation, variant of the operating system (Windows 2000 Server, for example). The workstation variants of these operating systems limit the number of concurrent client connections and are not appropriate for most VOB host applications.

Physical Memory

The minimum recommended main memory size is 128 MB or half the size of all the VOB databases the host will support, whichever is greater. To this amount, add:

- 7 MB of memory per VOB, regardless of VOB database size
- 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-effective) way to make VOB access faster and to increase the number of concurrent users without degrading performance.

Note: We do not recommend running any **view_server** processes on a VOB host (see *Minimize Process Overhead* on page 289).

Disk Capacity

A VOB database (and, on Windows, the entire VOB storage directory) must fit in a single disk partition. VOB databases tend to grow significantly as development proceeds and projects mature. Although there is no rule of thumb for estimating how large a VOB will be, Appendix F provides some guidelines that may be useful is establishing VOB server disk capacity requirements.

We recommend a high-performance disk subsystem, one with high rotational speed, low seek times, and a high mean time between failures. If possible, use a RAID or similar system 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 289.

Processor Capacity

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.

Network Connectivity

Nearly every access to a VOB places a load on the VOB host's network interface; a high-bandwidth (100 MB/sec. or greater) network connection to the VOB host is important. Multiple network interfaces to a VOB host can further improve its network accessibility, but some operating systems require each such connection to have a separate host name, which in turn requires the use of multiple regions in the ClearCase registry (see Chapter 3).

Adjusting UNIX Kernel Resources

You may need to adjust the following kernel resources on a VOB host running UNIX:

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

You may also find it beneficial to readjust kernel resources for the VOB host after ClearCase has been up and running for some time. For more information, see Chapter 16.

Creating VOBs

Any user with access to a suitable server host can create a VOB. In many communities, users create their own VOBs for casual use (testing, for example, or to hold artifacts that are accessed by only a few users) without the need for administrative intervention.

VOBs that contain significant components or other artifacts and are used by most members of the community are typically created and managed by administrators. The integrity and availability of such VOBs is critical to the work of the community, so additional planning and administrative support may be required.

Access Requirements for VOB Storage

A VOB storage directory must be writable by anyone who is authorized to create a VOB in it. In addition, storage directories for VOBs that will be accessed by dynamic views must be accessible over the network.

- On UNIX hosts, the partition where the VOB storage location is created must be exported so that other UNIX clients can mount it. If you plan to use an SMB server product to make the VOB accessible to Windows clients, the SMB server must be configured for that purpose, as described in Appendix A.
- On Windows hosts, the directory (folder) must be shared. Newly created shares have few access restrictions. If you modify the ACL of a share that holds one or more VOB storage directories, you must preserve full access rights for all users who need access to the VOB or view. In addition, you must grant full access to the ClearCase administrators group.

If a VOB will never be accessed by a dynamic view, you can create it on a partition that is not exported or shared.

Creating Server Storage Locations for VOBs

Server storage locations allow administrators to designate specific hosts and disk volumes as preferred locations for VOB storage. Server storage locations are specific to a registry region, because they must be accessible using a global path. You can specify one or more VOB storage locations for each region you have defined. After a server storage location for VOBs has been created, it can be used by **mkvob –stgloc** and the VOB Creation Wizard. (Server storage locations can also be created for views by using similar procedures.)

Server storage locations for VOBs should be created on a disk partition that has plenty of room for VOB database growth and is accessible to all hosts in the region. A VOB host that uses a supported network-attached storage device can be an excellent choice for a VOB server storage location.

There are several ways to create a VOB server storage location:

- When you install ClearCase server software on a Windows computer and specify during installation that you want to create storage locations on that server, the Server Storage Configuration Wizard runs when you log on after the post-installation restart. You can use the wizard to create storage locations on that computer.
- On a UNIX or Windows computer, you can use the **cleartool mkstgloc** command to create server storage locations.
- On ClearCase LT, the server setup wizard creates server storage locations as part of server setup.

You can also use the ClearCase Administration Console to add, change, or remove server storage locations:

1. Start the ClearCase Administration Console.
2. Navigate to the **Storage Locations** subnode of the **ClearCase Registry** node and do any of the following:
   - To create a new storage location, click **Action > New > Server Storage Location**.
   - To change the properties of an existing storage location, select one and click **Action > Properties**.
   - To remove an existing storage location, select one from the list and click **Action > Remove Server Storage Location**.

Note: If a server storage location for VOBs will be accessed by dynamic views, it must provide adequate network access, as defined in *Access Requirements for VOB Storage* on page 95.

Creating Server Storage Locations on a NAS Device

Creating server storage locations on a NAS device provides an easy way to start taking advantage of the device's storage capacity. This example uses the **cleartool mkstgloc** command to create a VOB storage location named **ccnasvobstg** in a UNIX region.

```
cleartool mkstgloc –vob –host vobsvr1 –gpath /net/nasdevice/vobstg/nasvobstg \
-hpath /net/nasdevice/vobstg/nasvobstg ccnasvobstg \
/net/nasdevice/vobstg/nasvobstg
```

Use **cleartool mkstgloc –view** with similar options to create a server storage location for views.

The ClearCase processes that support VOBs or views in a server storage location that resides on a NAS device run on the host specified in the **–host** option to **mkstgloc**.

Choose a host that has an appropriate configuration to service requests for access to the VOBs or views you intend to create in a NAS-based server storage location.

Note: You cannot use the Server Storage Configuration Wizard to create a storage location on a NAS device.

To Create a VOB on a VOB Server Host

You can use command line or GUI tools to create a VOB. The GUI tools provide complete procedural guidance and integrated Help. Before using the command line, see the reference pages for the **mkvob** command.

1. **Choose a location for the VOB storage directory**. You can use an existing VOB server storage location, create a new server storage location, or use any other directory that has the proper characteristics for VOB storage as described in *Access Requirements for VOB Storage* on page 95.

2. **Log on to the VOB host**. Unless the VOB storage will be created on a supported NAS device, the VOB host is the host where the VOB storage directory you selected in Step 1 resides.

   We recommend that you log on as a member of the same primary group as all other users who must access the VOB. The identity of the user who creates a VOB is used to initialize VOB access permissions. If a VOB is created by a user whose primary group is different from the primary group of other users who must access the VOB, you must edit the VOB's supplementary group list before those users can access VOB data. For more information, see *Adjusting the VOB's Group Ownership Information* on page 102.

   Note: Although you can create a VOB from a remote host of the same type (Windows or UNIX), we recommend that you run VOB creation commands on the VOB host itself to simplify the process of deriving the local storage path.

3. **Choose a VOB tag.** As described in *Tags* on page 33, a VOB tag is a unique name by which all hosts in a ClearCase registry region refer to a VOB. It's a good practice to choose a tag name that indicates what the VOB contains or how it is used.

4. **Create the VOB.** This example explains how to create a VOB by using the **cleartool** command line. You can also use various ClearCase GUIs, which provide their own procedural guidance.

   When you run the following command on Windows host **pluto**, it creates a VOB with the VOB tag \lib whose storage is in the directory shared as \\pluto\vobstg.

   cleartool mkvob –co "library sources" –tag \lib \\pluto\vobstg\lib.vbs

   ```
   Host-local path: c:\vobstg\lib.vbs
   Global path: \\pluto\vobstg\lib.vbs
   ```

```
VOB ownership:
  owner vobadm
  group dev
```

The comment is stored in the new VOB's database and displayed by **cleartool describe** and various GUIs.

For more information about options you may need to use when creating a VOB, see *To Create a VOB on a NAS Device* on page 98, *To Create a VOB on a Remote Windows Host* on page 99, and *Public and Private VOBs* on page 99.

5 **Verify that the VOB is accessible from typical hosts in the region.** When it is first created, a VOB has no elements other than the VOB root directory and a special directory called lost+found (see *The lost+found Directory* on page 112). You can use a dynamic view, a snapshot view, or a Web view to verify VOB access:

- To use a dynamic view, mount the VOB, activate the dynamic view, and from within the view run a command like this one:

  cleartool ls *vob-tag*

  You should be able to see the VOB root and lost+found directories

- To use a snapshot or Web view, add the VOB to the view's load rules and verify that the VOB root directory can be loaded into the view.

Note: Because a dynamic view requires network access to the file system of the VOB host, a VOB that is accessible from a Snapshot or Web view may not be accessible from a dynamic view. For more information, see *Access Requirements for VOB Storage* on page 95.

To Create a VOB on a NAS Device

When you use a certified NAS device to provide VOB storage, we recommend that you create one or more VOB storage locations on your NAS device (as described in *Creating Server Storage Locations on a NAS Device* on page 96) and then using **mkvob –stgloc** to create VOBs in this storage location. For example, in a Windows region where you have created a server storage location named **nasvobstg**, the following command creates a VOB with the tag \nasvob in that storage location:

cleartool mkvob –tag \nasvob –stgloc nasvobstg

If you do not use a server storage location, you must specify the host name, global path, host-local path, and VOB storage pathname on the **mkvob** command line. For example, the following command creates a VOB with the tag /vobs/nasvob. The VOB is served by a **vob_server** process running on ClearCase host **vobsvr1** (a UNIX computer) and has its storage on NAS device mounted by **vobsvr1** at /net/nasdevice.

cleartool mkvob –tag /vobs/nasvob –host vobsvr1
**–gpath /net/nasdevice/vobstg/nasvob.vbs **
–hpath /net/nasdevice/vobstg/nasvob.vbs /net/nasdevice/vobstg/nasvob.vbs

To Create a VOB on a Remote Windows Host

To create a VOB on a remote Windows host, ClearCase must have access to information stored in the remote host's Windows registry. Remote access to the Windows Registry can be restricted by setting security on the key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePipeServers\Winreg

Because this key is not normally present on Windows workstations, the underlying security on the individual keys controls access.

On Windows server hosts, the key is preset by default and may prevent workstations from reading the server's registry. When this is the case, attempts to create VOBs on the server fail with the error `Invalid argument`. To correct this problem, either remove the key or modify the security on it.

The ClearCase Doctor program warns you if this registry key will prevent creation of VOBs on a remote host and offers to modify it.

Public and Private VOBs

To provide control over how a VOB is mounted, each VOB tag is designated as public or private when it is created. VOBs themselves are neither public nor private. The distinction rests entirely in the tag. The public/private status of a VOB tag is only important to dynamic views, which can access only those VOBs that are mounted. Snapshot and Web views can access any VOB that has a tag, regardless of whether it is mounted.

On ClearCase LT, which does not support dynamic views, VOBs are never mounted, and the public/private distinction is meaningless.

Note: Any user can mount any VOB, public or private. The private designation means only that a VOB is not mounted by a **cleartool mount –all** command and must be mounted explicitly by name.

Public VOB tags are treated somewhat differently on Windows and UNIX:

- 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

If you use the **–persistent** option to **cleartool mount**, the specified VOBs are mounted each time you log on. To force all public VOBs to be mounted in this way, use the following command:

cleartool mount –all –persistent

You can also arrange to have a VOB mounted when you log on by selecting the **Reconnect at Logon** check box in the **Mount** dialog box when you first mount the VOB from the ClearCase shortcut menu.

Note: On both Windows and UNIX, when UCM is in use, all of a project's VOBs are mounted when a user starts a dynamic UCM view.

To Create a Public VOB

To create a public VOB, you must specify **–public** on the **mkvob** command line or select the **Make this a public VOB** check box in the VOB Creation Wizard. You must also supply the ClearCase registry password.

If you specify **–public** but do not know the ClearCase registry password (or the ClearCase registry password has not been established), the VOB is created without a tag, and you must create the tag in a separate operation. For more information the ClearCase registry password, see *Setting and Using the Registry Password* on page 41 and the **rgy_passwd** reference page.

Linking a VOB to an Administrative VOB

Any VOB can be linked to an administrative VOB from which it derives definitions of type objects such as branch types and label types. Type objects are a fundamental part of the ClearCase data model. To simplify type administration, we recommend that you establish an administrative VOB hierarchy into which new VOBs can be placed. For more information about administrative VOBs and global types, see Chapter 7.

Note: A PVOB is the administrative VOB for all UCM component VOBs that are included in projects defined in that PVOB. Every UCM component VOB is created with an **AdminVOB** hyperlink to its PVOB. If a project uses multiple PVOBs, they must each have an **AdminVOB** hyperlink to a common PVOB in which shared components and streams are defined.

The VOB Creation Wizard allows you to specify the administrative VOB for each new VOB you create. If you create a VOB with **mkvob**, you must specify the VOB's administrative VOB in a subsequent **mkhlink** command that creates an **AdminVOB** hyperlink from the VOB to its administrative VOB.

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

Replacing a VOB Server Host for a NAS Device

When you locate VOB storage on a NAS device, you can easily replace the VOB host for any VOB on the NAS device with another VOB host of the same architecture without actually moving the VOB storage.

The following procedure replaces the UNIX VOB host for the VOB /vobs/libpub with a different ClearCase host of the same architecture, **vobsvr1**. The procedure locks the VOB, removes the old tag and object, and then creates a new object and tag, specifying the replacement host and the existing storage.

1. **Lock the VOB**.
2. **Unmount the VOB** as needed on dynamic view hosts.
3. (If applicable) **Disable VOB snapshots on the current host.** If VOB database snapshots are enabled for the VOB, disable them with this command:

   vob_snapshot_setup rmparam /vobs/libpub
4. **Remove the VOB tag and unregister the VOB.** Use the ClearCase Aministration Console or the following commands:

   cleartool rmtag –vob –all /vobs/libpub

   cleartool unregister –vob /net/nasdevice/vobstg/libpub.vbs
5. **Terminate the VOB's server processes on the current host**.
   - On a UNIX host, search the process table for the **vob_server** and **vobrpc_server** processes that service the VOB. Run the UNIX **ps** command and search its output for processes associated with **libpub.vbs**. Use the UNIX **kill** command to terminate any such processes.
   - On a Windows host, stop and restart ClearCase.
6. **Register the VOB and create a new VOB tag**. Use the ClearCase Administration Console or the following commands.

   **cleartool register –vob –host vobsvr1 –hpath /net/nasdevice/vobstg/libpub.vbs **
   –gpath /net/nasdevice/vobstg/libpub.vbs /net/nasdevice/vobstg/libpub.vbs

   **cleartool mktag –vob –replace –host vobsvr1 –tag /vobs/libpub **
   **–hpath /net/nasdevice/vobstg/libpub.vbs –gpath **
   /net/nasdevice/vobstg/libpub.vbs /net/nasdevice/vobstg/libpub.vbs

7 **Unlock the VOB.**

8 **Mount the VOB** as needed on dynamic view hosts.

9 (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.

Troubleshooting VOB Access Problems

This section provides information on diagnosing and correcting common problems that prevent access to VOBs.

Incorrect Global Path Information

When you create a VOB, the tool that you use attempts to verify that the local and global pathnames you supply are correct. By running the tool on the VOB host, you maximize the chances for this verification to succeed. Because the tool can test the validity of the global and local paths to the VOB storage as seen from the server host, it is likely that any other host will be able to access the VOB as long as all of the following are true:

- It is the same type of host (UNIX or Windows).
- It has network connectivity with the VOB host.
- The VOB storage directory meets the criteria defined in *Access Requirements for VOB Storage* on page 95.

UNIX Automounter Does Not Use the Hosts Map

If the automounter on a UNIX host does not use the hosts map, the heuristics that help **mkvob** and **mkview** construct a global path to VOB or view storage may fail. On such hosts, you must use the **–host**, **–hpath**, and **–gpath** options to **mkvob** and **mkview** to ensure that the global path, stored in the VOB or view tag, is valid for all hosts in the registry region. For more information about ClearCase and the automounter, see *ClearCase and the NFS Automounter* on page 18.

Adjusting the VOB's Group Ownership Information

Even though a VOB is accessible, problems may arise during **checkout** or **mkelem** (**Add to Source Control**) operations when all prospective users of the VOB do not belong to the same group. (For detailed information about user identity and VOB access rights, see Chapter 5.) A VOB's group is initially the same as the primary group of the

user who created the VOB. A VOB is fully accessible only to those users who are members of that group.

If your community 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 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. For more information and examples, see the **protect** reference page.
- Users cannot modify an element (by checking in a new version, for example) unless they belong to the element's group. The element's group can be any group the user belongs to.

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

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

Enabling Setuid Execution on UNIX

By default, the viewroot and VOB file systems are mounted with setuid and setgid disabled; this is the recommended configuration. If you must enable hosts to mount these file systems with setuid and setgid enabled (for example, if you run setuid tools from a VOB), you can either configure your release area with the **site_prep** script to make this the default for all hosts, or you can run *ccase-home-dir*/etc/utils/change_suid_mounts to enable it for a single host.

The effects of running change_suid_mounts on a host do 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 the *Installation Guide*.

Making a VOB Inaccessible

To make a VOB inaccessible, you may lock it or remove its tag.

To Lock a VOB

Locking a VOB prevents it from being modified. A locked VOB can be mounted for read-only access by dynamic views and can have its elements loaded into a snapshot or Web view, but no elements can be checked out or checked in, and no metadata can be modified. Locking a VOB also flushes all pending transactions to disk and prepares the VOB for backup.

Use the ClearCase Aministration Console or the **cleartool lock** or **unlock** commands to lock or unlock a VOB. To list locks on a VOB, use the ClearCase Administration Console or the **cleartool lslock** command.

To Remove a VOB Tag

Removing a VOB's tag makes the VOB inaccessible. Use the ClearCase Administration Console or the **cleartool rmtag** command to remove a VOB tag. You can also unregister the VOB, which prevents any new tags from being created. Use the ClearCase Administration Console or the **cleartool unregister** command to unregister a VOB.

Removing a VOB

Removing a VOB destroys all of its data. Do not remove a VOB unless the data it contains is no longer valuable.

Caution: Replicated VOBs have special requirements. To remove a replicated VOB, follow the procedures documented in the *Administrator's Guide* for Rational ClearCase MultiSite.

To remove a VOB:

1. Verify that the VOB is not in use. Clients that use dynamic views should unmount the VOB. Users with snapshot views should remove the VOB from their load rules.
2. Remove the VOB. You can use the **rmvob** command or the ClearCase Administration Console. To use the ClearCase Administration Console, navigate to the VOB storage node for the VOB (a subnode of the host node for the host where the VOB storage directory resides) and then click **Action > All Tasks > Remove VOB**.
3. Stop and restart ClearCase on the VOB host. This terminates the VOB's server processes.

The VOB Storage Directory

This section describes the contents of a VOB storage directory. An understanding of these contents can be useful to anyone responsible for VOB storage maintenance.

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

A VOB storage directory can reside on a VOB host or on a supported network-attached storage device. The directory structure is visible when you use native operating system utilities to list the VOB storage directory and its subdirectories and files, which include the following:

.pid	A one-line text file that lists the process ID of the VOB's associated **vob_server** process.
admin	A directory that contains administrative data related to the amount of disk space the VOB is using. Use **cleartool space –vob** or the ClearCase Administration Console to display this data.
vob_oid	A one-line text file that lists the VOB's object identifier (OID) expressed as a universal unique identifier (UUID). This UUID is the same for all the replicas in a VOB family (ClearCase MultiSite). This value is also stored in the VOB's database; do not change it.
replica_uuid	A one-line text file that lists the replica UUID of this particular replica of the VOB. Different replicas created with ClearCase MultiSite have different replica UUIDs.
.identity	On UNIX hosts, a subdirectory whose files establish the VOB's owner and group memberships. See *Access Control for VOBs and VOB Objects* on page 75.
identity.sd	On Windows hosts, a binary data file that contains the security descriptor for the VOB storage directory.
groups.sd	On Windows, security descriptors of the VOB's supplementary groups.
s	A subdirectory in which the VOB's source storage pools reside.
d	A subdirectory in which the VOB's derived object storage pools reside.
c	A subdirectory in which the VOB's cleartext storage pools reside.

db	A subdirectory containing the files that implement the VOB's embedded database.
vob_server.conf	A text file read by the **vob_server** at startup. It contains the setting for deferred deletion of source containers; deferred deletion is activated if you have turned on semi-live backup. See the **vob_snapshot_setup** reference page for more information.
.hostname	A text file that records the name of the VOB's server host.
.msadm_acls	Stores the MultiSite administration server's ACL

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

VOB Storage Pools

The c, d, and s subdirectories of the VOB storage directory contain the VOB's storage pools. On UNIX, a pool can be a symbolic link to a directory on another disk volume or host. On Windows, a pool must be a subdirectory of the VOB storage directory. The storage pools in these directories hold data container files, which store versions of elements, shared derived objects, and cached cleartext. Depending on the element type, versions of an element may be stored in separate containers or may be interleaved in a single file as version-to-version deltas (differences).

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

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

You can create additional pools as needed using the **Pools** subnode of a VOB node of the ClearCase Administration Console or the **cleartool mkpool** or **chpool** commands. Each storage pool holds data containers of one kind (source, cleartext, or DO). For more about managing storage pools, see *Creating Additional Storage for a VOB* on page 113.

Note: On ClearCase LT, VOBs may use only the default 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 generating individual cleartext versions from deltas in the container. It also updates the source container with a new delta when a new version is checked in.

Source pools are accessed by GUIs and **cleartool** commands such as **checkout** and **checkin**, as well as by any program that opens a file or directory in a dynamic view (whether or not the file or directory is checked out). If a cleartext version of the element is available, it is used. If it is not available, it is 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 one version of an element. These pools are caches that accelerate access to element types such as **text_file** and **compressed_text_file**, for which all versions are stored in a single container.

For example, the first time a version of a **text_file** element is required, the **text_file_delta** type manager constructs the version from the element's source data container. The version is cached as a cleartext data container (an ordinary text file) in a cleartext storage pool. On subsequent accesses, ClearCase looks first in the cleartext pool, and only constructs a new version if the requested version cannot be found there.

Cleartext pools are periodically scrubbed to remove versions that have not been recently accessed. For more information, see *Scrubbing VOB Storage Pools* on page 226.

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 in a dynamic view by a build tool such as **clearmake** or **clearaudit**.

DO storage pools contain data containers only for the derived objects that have been promoted to the VOB by the **winkin** command. 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.

Like cleartext pools, DO storage pools are periodically scrubbed to remove extraneous DOs.

Note: The DO pools of ClearCase LT VOBs are always empty because ClearCase LT does not support derived objects.

VOB Database

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

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

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

The db directory contains these files:

vob_db.dbd	A compiled database schema, used by embedded database routines for database access. The schema describes the structure of the VOB database.
vob_db_schema_version	A schema version file, used by embedded database routines to verify that the compiled schema file is at the expected revision level.
vob_db.d0*n* vob_db.k0*n*	Files in which the database's contents are stored.
vista.*	Database control files and transaction logs.
db_dumper (UNIX)	Backup copy of *ccase-home-dir***/etc/db_dumper**. reformatvob invokes this copy of **db_dumper** only if it cannot invoke *ccase-home-dir***/etc/dumpers/db_dumper.***num*, where *num* is the schema version of the VOB.

db_dumper (Windows)	A copy of *ccase-home-dir***bin****db_dumper.exe**. This is an executable program, invoked during the **reformatvob** command's dump phase. Each VOB has a copy of **db_dumper** so that it can always dump itself. (Typically, a VOB needs to be dumped after a newer release of ClearCase has been installed on the host; with this strategy, the *ccase-home-dir***bin****db_dumper** program in the newer release need not know about the older VOB database format.)
vob_db.str_file	Database string file that stores long strings.

Preserved Database Subdirectories

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

The .identity Directory

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

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

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

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

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

VOB Storage Management

VOB storage grows in proportion to the number of developers who use the VOB and the rate at which they create and change artifacts in it. VOB storage management is an important administrative responsibility. It includes three principal tasks:

- Monitoring the storage each VOB consumes
- Controlling VOB growth by removing unneeded data and metadata. Much VOB data and metadata must be preserved for an extended period, but some of it loses value quickly and can be safely removed.
- Creating additional storage capacity by adding new storage pools to a VOB, relocating data from one VOB to another, or moving an entire VOB to another partition or host.

Using the Scheduler

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

- Daily data gathering on disk space used by VOBs
- Weekly data gathering on disk space used by shared derived objects
- Daily scrubbing of VOB storage pools by the **scrubber** utility
- Weekly scrubbing of VOB databases by the **vob_scrubber** utility
- Daily and weekly execution of jobs that you can customize as needed

In many cases, the default set of jobs running on the default schedule are adequate to monitor and manage VOB storage growth. For information about modifying the default set of jobs and the default schedule, see Chapter 12.

Monitoring VOB Storage

ClearCase provides command-line and GUI tools that display information about disk space used by VOBs.

- In the ClearCase Administration Console, the VOB storage directory node for each **VOB** subnode of a ClearCase host node 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 and also lists views that hold references to these DOs.
- The **cleartool space –vob** command shows current and historical disk space use for a VOB.

- The **cleartool dospace** command shows disk space used by shared derived objects in the VOB. The display also lists views that hold references to these DOs.

Note: These tools depend on one or more scheduled jobs that generate the VOB space data they display. If these jobs are not enabled or have not been run, these tools do not produce any meaningful output. For more information about scheduled jobs, see Chapter 12.

Scrubbing

ClearCase includes two utility programs, **scrubber** and **vob_scrubber**, that you can use to control the growth of the VOB storage pools and database. The ClearCase scheduler runs these programs for all VOBs on a host, on a daily or weekly schedule, using default settings that specify what should be scrubbed. The default settings, while conservative, are often adequate to manage the growth of moderately active VOBs. To learn more about scrubbing schedules and parameters and how to tune them to the needs of individual VOBs, see *Scrubbing to Control VOB Storage Growth* on page 225.

Removing Unneeded Versions from a VOB

Scrubbing removes only those artifacts that can be regenerated. Scrubbing never removes elements or versions. Because elements and versions are historical data, you should approach their removal with extreme caution. Removing entire elements with **cleartool rmelem** is particularly dangerous:

- Even if an element is no longer needed for ongoing work, you may need it to reproduce and maintain earlier work.
- **rmelem** removes the element's name from all directory versions in which it was ever cataloged. This erasing of history means that the element does not appear in listings or comparisons of old directory versions. Removing an element's name from a VOB directory, by using the **rmname** command, preserves its history but removes its name from subsequent versions of the directory.
- A mistake can be costly; the procedure for recovering a single element from backup is cumbersome and cannot be used in a UCM component. (See *Restoring an Individual Element from Backup* on page 207.)

If you need to reclaim disk space in source pools, 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 that meet the following criteria:

- 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, such as version labels, attributes, or hyperlinks

rmver has options that allow you to force removal of versions that do not meet these criteria. See the **rmver** reference page for more information.

You can also use the **cleartool relocate** command to move entire VOB directories and all the element versions they contain from one VOB to another. For more information, see *Relocating Elements to Another VOB* on page 141.

The lost+found Directory

Each VOB includes a special directory element, lost+found, which is used to hold elements that become stranded when they are not cataloged in any directory version in the VOB. An element can become stranded when you do any of the following:

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

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

element_leaf_name.*UUID*

For example, an element named util.c could have the name
util.c.41a00000bcaa11caacd0080069021c7

The lost+found directory has several unique properties:

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

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

- If you need an element in lost+found, move it to another directory by using **cleartool mv**.
- You can use **cleartool rmelem** to remove elements from lost+found if you are sure they are no longer needed. See the **rmelem** reference page for additional cautions about using **rmelem**.

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

Creating Additional Storage for a VOB

Note: This section does not apply to ClearCase LT, where VOBs may use only the default pools.

VOBs provide flexible pool configuration options that may help with VOB storage management. This section describes:

- Creating additional storage pools in a VOB storage directory
- Using UNIX symbolic links to create additional pools in other directories or on other hosts
- Assigning elements to specific pools

Creating Additional Storage Pools

After you have created a VOB, you can create as many additional storage pools as you need and adjust their contents as necessary with the **Pools** subnode of a VOB node of the ClearCase Administration Console or the **cleartool mkpool** or **chpool** commands. On UNIX hosts, you can also create remote storage pools.

Creating Remote Storage Pools on UNIX Hosts

On UNIX hosts, which support symbolic links to file systems on other UNIX hosts or NAS devices, you can distribute a VOB's storage pools to provide additional VOB storage capacity. These remote pools can be on another host (which need not have ClearCase installed) or on another disk partition on the VOB host.

There are several requirements to consider when deciding whether to use remote storage pools:

- Backup and recovery can be more complicated for a VOB that uses remote storage pools. See *If the VOB Has Remote Storage Pools* on page 190.
- VOB tags for VOBs with remote storage pools must include additional information if they are to be accessed by Windows computers. See *Windows Tags for UNIX VOBs with Remote Storage Pools* on page 303.
- The remote location must be accessible over the network by all hosts that use the VOB. A remote host with multiple network interfaces (and multiple host names) may not work in this role.
- Unless the entire VOB storage directory is located on a certified NAS device, the VOB database (`db` subdirectory) must be a subdirectory of the VOB storage directory. It cannot be a symbolic link to a directory on another computer.

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: checked-in versions of file elements. Traffic to and from these pools is relatively light, but data integrity is very important. We recommend that you keep source storage pools 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 and restore procedures. These concerns typically override performance considerations, because losing a source pool means that developers must recreate the lost versions. If you decide to use remote source pools, choose a robust file server for which you provide frequent, reliable data backups.
- **Cleartext pools.** These pools probably get the heaviest traffic (assuming that many of your file elements are stored in delta or compressed format), but the data in them is expendable, because it can be reconstructed from the data in the source pools. The ideal host for cleartext pools is a computer with a fast file system.
- **Derived object pools.** These pools can become quite large if derived objects are maintained for numerous releases and platforms. Derived objects, like cleartext, can usually be regenerated from the information in the VOB database and source pools, but regeneration of older derived objects can sometimes be difficult. The DO pools of ClearCase LT VOBs are always empty, because ClearCase LT does not support derived objects.

  Note: If you check in a derived object, it becomes a versioned object and is no longer stored in a derived object pool.

Changing Elements' Source Pool Assignments

Because all newly created elements inherit the pool assignments of their parent directories, all elements in a VOB use the default storage pools unless you create new pools and reassign elements to them. On UNIX hosts, you may be able to combine remote pools with pool reassignments to better manage storage consumption. For example, if a VOB's storage directory is being filled up with large versioned DOs, you may want to create a new DO pool on a remote volume and reassign those elements to the new pool.

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 element moves all its data containers to the new pool. Changing the source pool of a directory element assigns all new elements created within that directory to the new source pool. Existing elements remain in their current pools.

Example: Assigning All Files in a Directory to a New Pool

The following example shows how to create a new, remote source storage pool for a VOB on UNIX and reassign all current and future elements in a particular directory to this new pool.

1 **Create the new storage pool**. Specify a global pathname for the remote pool. The pathname must be 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 subdirectory.

cd libbgr

cleartool find . –type f –exec 'cleartool chpool –force bgrsrc2 $CLEARCASE_PN'
```
Changed pool for "./Makefile" to "bgrsrc2".
Changed pool for "./getcwd.c" to "bgrsrc2".
  .
  .
Changed pool for "./strut.c" to "bgrsrc2".
```

3 **Reassign the directory element to the new pool.** This ensures that all newly created file and directory elements in this directory use the new pool.

cleartool chpool bgrsrc2 .
```
Changed pool for "." to "bgrsrc2".
```

Example: Moving an Existing Storage Pool to Another Disk

Use the following procedure to move an existing storage pool to another partition on a UNIX host:

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**.

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 by 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

Verify that the symbolic link is valid for all hosts that will access the VOB.

5 **Unlock the VOB.**

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 the VOB has a tag in a Windows region, you must modify the tag to account for the relocated storage pool. See *Windows Tags for UNIX VOBs with Remote Storage Pools* on page 303.

VOB Datatypes and Administrative VOB Hierarchies

7

There are many types of VOB metadata. Some are unique to a particular VOB, but many must be managed consistently across a group of VOBs that store related artifacts (components of a UCM project, for example). This chapter introduces VOB metadata and describes how to use administrative VOB hierarchies to simplify sharing of type objects among related VOBs.

VOB Datatypes

A VOB is a repository for data (versioned artifacts like files and directories) and metadata (branches, labels, event records, and so on). Some metadata types are stored as VOB objects; other metadata is stored as records or annotations attached to these objects. A general understanding of VOB datatypes, which are described in this section, can help place many administrative tasks in context.

Some VOB datatypes are created automatically by ClearCase commands. Others must be created explicitly by users or (more often) administrators. You must use commands like **cleartool describe** and various ClearCase GUIs to access VOB metadata.

The VOB Object and Replica Objects

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

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

If a VOB is replicated, it also has a replica object, which is created by the **mkreplica** command.

File System Objects

A VOB database keeps track of files, directories, and VOB symbolic links by using the following file system objects in the VOBs:

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

Event Records

Nearly every operation that modifies the VOB (**checkout** and **checkin**, for example) creates an event record in the VOB database. See the **events_ccase** reference page for more information.

Shareable Derived Objects

A VOB's database stores information about all the shareable derived objects (DOs) created at pathnames within the VOB. For each DO, the database catalogs this information:

- the name of DO and the name of its containing directory
- the DO's unique identifier
- shopping information for the DO (used by **clearmake** and **omake**)

DOs are not created on ClearCase LT because they only be created and accessed through dynamic views.

Configuration Records

A VOB database stores configuration records (CRs) associated with shareable derived objects and DO versions (derived objects that have been checked in). Each CR documents a single target rebuild, which is typically the result of the execution of one build script.

Like DOs, CRs are not created on ClearCase LT because they only be created and accessed through dynamic views.

Type Objects

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

A VOB can store several kinds of type objects:

Type	Mnemonic	Description
Element type	**eltype**	Instances are elements.
Branch type	**brtype**	Instances are branches.
Hyperlink type	**hltype**	Instances are VOB hyperlinks that connect two related objects.
Trigger type	**trtype**	Instances are triggers.

VOBs also store type objects that are used only to modify instances of other type objects:

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

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

cleartool describe brtype:v4_patch

Instances of Type Objects

After a type object is created, users can create instances of the type. Creating an instance of a type creates a reference to the type object. For example, attaching the label **BASELEVEL_4.2** to a particular version does not make a copy of the **BASELEVEL_4.2** type object. Instead, it establishes a connection between the element version and the label type object. The following table describes how type objects are used by **cleartool** commands that create instances of types.

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

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

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

Scope of Type Objects

The scope of a type object is defined when the object is created. Local type objects can only be used to create instances in the VOB in which they are defined. Global type objects can be used to create instances in any VOB that is part of an administrative VOB hierarchy beneath the VOB in which the global type is defined. For more information, see *Administrative VOB Hierarchies and Global Types* on page 121.

Note: Trigger type objects cannot be global types.

Predefined and User-Defined Type Objects

Each VOB is created with a set of predefined type objects. You can create additional type objects as needed with any of the following the **cleartool** commands:

- **mkattype** creates or modifies an attribute type object
- **mkbrtype** creates or modifies a branch type object
- **mkeltype** creates or modifies an element type object
- **mkhltype** creates or modifies a hyperlink type object
- **mklbtype** creates or modifies a label type object
- **mktrtype** creates or modifies a trigger type object

The reference page for each of these commands lists all predefined type objects associated with the command. (For example, the **mkeltype** reference page lists all of the predefined element types.) You can also use the ClearCase Administration Console or the **cleartool lstype** command to list the type objects defined in a given VOB.

Changing an Element's Type

You can use **chtype** to convert an element from one type to another (for example, from **file** to **text_file**). Typically, you change an element's type to change the way its versions are stored. For example, versions of an element of type **file** are stored in separate data containers in a VOB source pool. Converting the element to type **text_file** causes all its versions to be stored in a single data container, as a set of deltas (version-to-version differences), which saves disk space.

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

Administrative VOB Hierarchies and Global Types

Managing type objects that are used in a group of related VOBs is greatly simplified by creating administrative VOB hierarchies that can share globally defined type objects. After a global type object is created in a VOB, instances of the type can be created in any other VOB that links to it with an **AdminVOB** hyperlink. This section introduces global types and administrative VOB hierarchies.

Administrative VOB Hierarchies

A VOB is an administrative VOB if it meets both of these criteria:

- It contains at least one global type.
- It is the target of an **AdminVOB** hyperlink from another VOB.

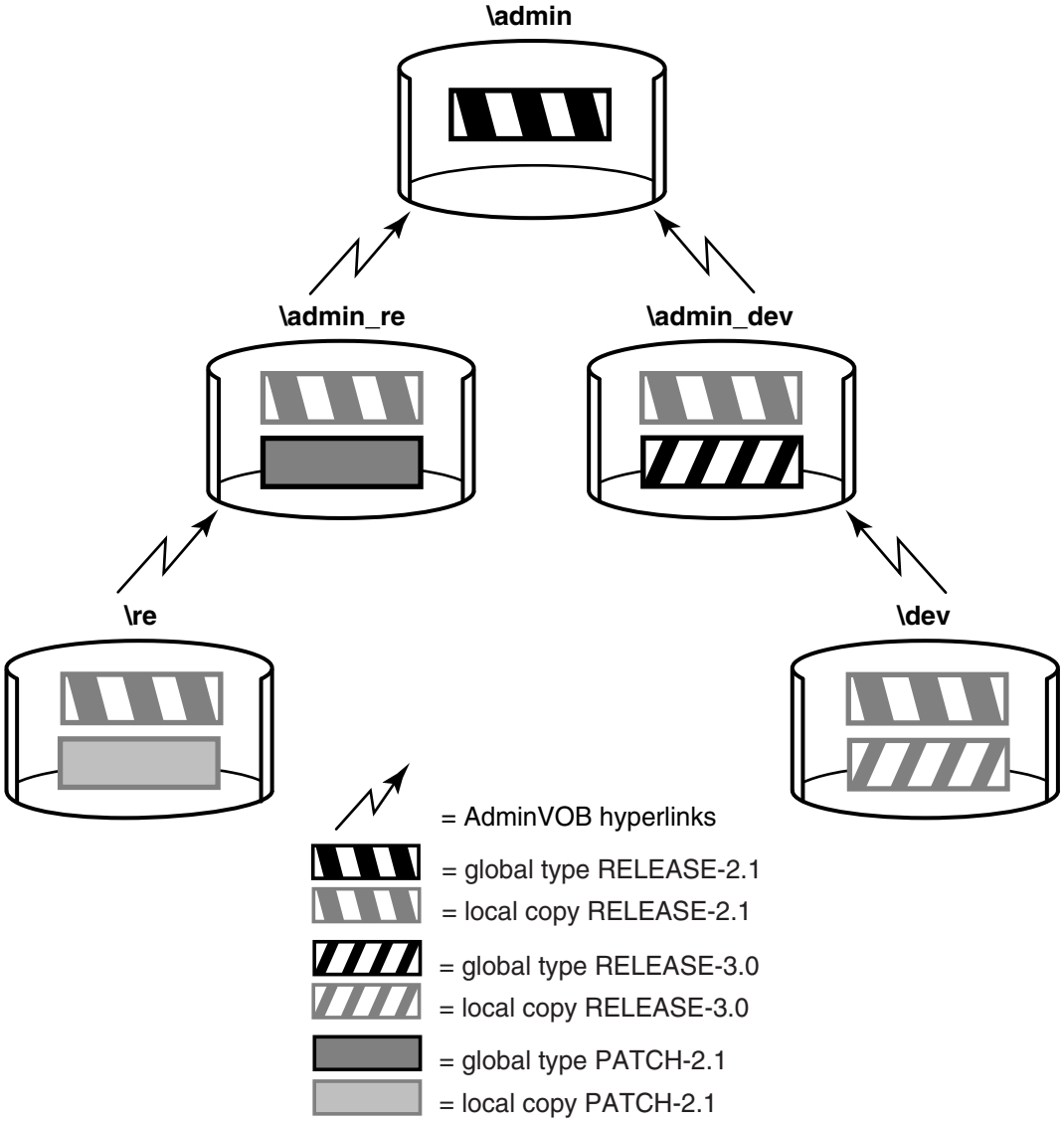
An administrative VOB hierarchy includes at least one administrative VOB and one or more VOBs that have an **AdminVOB** hyperlink to it or to another VOB that has an **AdminVOB** hyperlink to it.

Every VOB in an administrative VOB hierarchy has exactly one parent and zero or more children. Administrative VOB hierarchies can have any structure that does not violate these constraints or create a circular relationship. In the example shown in Figure 4, the administrative VOB **\admin** defines global types used by all VOBs in the hierarchy, while lower-level administrative VOBs **\admin_dev** and **\admin_re** define additional global types specific to the needs of particular teams.

All UCM component VOBs are created as part of an administrative VOB hierarchy, with an **AdminVOB** hyperlink to a PVOB in which UCM type objects such as projects, folders, and streams are defined. If multiple PVOBs are in use, they must each have an **AdminVOB** hyperlink to a common PVOB in which shared UCM type objects are defined.

Note: If you are using UCM, you do not normally need to create administrative VOB hierarchies manually as described in this section.

Figure 4 Administrative VOB Hierarchy



You can add a VOB to an existing administrative VOB 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 a VOB to an administrative VOB hierarchy:

1 Remove the **AdminVOB** hyperlink at the point where you want to add the new VOB. For example, if you want to add a VOB between **\admin** and **\admin_re**, use **cleartool describe** to list the **AdminVOB** hyperlink and **cleartool rmhlink** to remove it.

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 Link the new VOB to its superior in the hierarchy. The following command creates an **AdminVOB** hyperlink from the VOB **\admin_lb** to the VOB **\admin**.

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

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

3 Link the VOB you removed from the hierarchy in Step 1 back into the hierarchy:

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

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

This also reconnects the VOB **\re** with the hierarchy, because its link to **\admin_re** is still intact.

Listing an AdminVOB Hyperlink

To list an **AdminVOB** hyperlink, use the ClearCase Administration Console or the **cleartool describe** command. The **describe** command shows any **AdminVOB** hyperlink from a VOB to its superior in the administrative VOB hierarchy. The following examples show **AdminVOB** hyperlinks.

- 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 links from two lower-level VOBs

cleartool describe vob:\admin

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

- A VOB that is in the middle of a hierarchy (has an **AdminVOB** hyperlink pointing to its superior in the hierarchy and has an **AdminVOB** hyperlink pointing to it from a lower-level VOB)

  cleartool describe vob:\admin_dev

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

To display the hyperlink ID, use **cleartool 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 VOB Hierarchies

The following restrictions apply to administrative VOB hierarchies:

- A VOB can have at most one **AdminVOB** hyperlink pointing to an administrative VOB. The **mkhlink** command prevents the creation of a second **AdminVOB** hyperlink from a VOB.
- Local types may not eclipse (have the same type and name as) global types. An attempt to add a VOB to an administrative VOB hierarchy will fail if the VOB defines a local type that would eclipse one already defined as global in the hierarchy. Use the **–acquire** option to **mkhlink** to force the local type to become an instance of the global type. See the **mkhlink** reference page for more information.
- All VOBs in a hierarchy should be backed up and restored as a group. If they are not, VOBs in one part of the hierarchy may hold instances of types that are not defined in their administrative VOB (because the administrative VOB was restored from a backup taken before the type was created). For more information, see *Restoring One or More Members of a Set of Related Databases* on page 213.
- If the administrative VOBs from which an ordinary VOB derives a type object definition is not accessible, instances of that type cannot be created or manipulated

in the ordinary VOB. For more information, see *If an Administrative VOB Becomes Unavailable*.

If an Administrative VOB Becomes Unavailable

If an administrative VOB or PVOB becomes unavailable, attempts at other VOBs in the hierarchy to create instances based on a now-inaccessible global type definition produce the following error:

`cleartool: Error: Unable to access administrative VOB` *AdminVOB-tag* `of` *VOB-tag*

In addition, the output of **cleartool describe** for a VOB with an **AdminVOB** hyperlink to an unavailable VOB may not show the hyperlink.

Note: An administrative VOB does not have to be mounted to be an active part of a hierarchy, but it must be registered and must have a tag.

Removing a VOB from an Administrative VOB Hierarchy

You can break the relationship between a VOB and its administrative VOB by removing the **AdminVOB** hyperlink and all **GlobalDefinition** hyperlinks from the VOB to the administrative VOB. You must remove all such hyperlinks to sever the connection. The following sections describe how to remove the hyperlinks by using **cleartool** commands. You can also use the ClearCase Administration Console.

Note: You cannot use these procedures on a UCM component VOB or PVOB. If you need to remove a UCM component VOB or PVOB from an administrative VOB hierarchy, call Rational Customer Support.

Removing the AdminVOB Hyperlink

To remove the **AdminVOB** hyperlink:

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 VOB to global types in the administrative VOB:

1. Determine the names of all local copies of global attribute, branch, element, hyperlink, and label types:

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

   ```
   Tested          local copy
   Revision        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

When you remove an administrative VOB with the **rmvob** command or by using the ClearCase Administration Console, related hyperlinks are cleaned up automatically:

- All **AdminVOB** hyperlinks to the VOB are removed.
- All **GlobalDefinition** hyperlinks connecting to global types in the VOB are removed.
- All local copies (in other VOBs in the hierarchy) of global types defined in the VOB are converted to ordinary types.

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. Examples in this section use the **cleartool** command line. You can also use the **Metadata** subnode of any VOB node in the ClearCase Administration Console to manipulate metadata, including type objects, in that VOB.

Creating a Global Type

Most **cleartool** type-creation commands (see *Predefined and User-Defined Type Objects* on page 121) support a **–global** option, which creates a global type object. 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".
```

Unless you use the **–acquire** option to the appropriate type-creation command, you cannot create a global type if any VOB in the hierarchy defines a type object with the same name. Before you create a new global type, check for potentially eclipsing local types and use the **–acquire** option to convert these to local copies of the global type.

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

You can also use the **–replace**, **–global**, and **–acquire** options to convert an existing ordinary type to a global type.

Auto-Make-Type Operations

In general, creating a local instance of a global type creates a local copy of the global type. This action is referred to as an auto-make-type operation. Auto-make-type operations occur in any of the following cases:

- An operation that creates attributes, branches, elements, hyperlinks, or labels creates a local copy of the global type.

- A checkout operation that creates a branch (because of an auto-make-branch rule) creates a local copy of the global branch type.
- An operation that attaches an attribute or a hyperlink to a local copy of a type creates the local copy if it does not already exist.

If the global type has supertypes, the auto-make-type operation creates local copies of the supertypes and then fires any type-creation triggers associated with them. It also sets the permissions and ownership of the local copy to match those of the global type.

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".
```

Note: 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, because built-in actions cannot cause cascading triggers. Therefore, if you attempt to create such a trigger in a VOB that does not contain a local copy of the global type, the **mktrtype** command fails.

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 VOB that does not contain a local copy of the type. To describe the local copy, use the **–local** option.

The following command describes the global label type **REL6**:

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 the global label type **REL6**:

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

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

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

cleartool describe –local lbtype:NOLOCAL@\dev

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

Listing Global Types

By default, the **lstype** command lists global types associated with local copies, even if you use **–invob** to specify a VOB in which the global type is not defined. The output also includes global types from all administrative VOBs above this VOB in the administrative VOB hierarchy, even if the specified VOB does not currently contain local copies of the type. To show only those types that are defined (or exist as copies) in a specific VOB, use the **–local** option.

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

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

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

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

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

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

Listing the 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 specified 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 the Protection of a Global Type

Changing the protection of a global type or 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 any VOB in a hierarchy in which the type is defined, even if the 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 that prevented the protection from being changed 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 new instances of a locked global 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 attempt to lock a type in a 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.

Copying a Global Type

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

- The source VOB of the original type and destination VOB of the copy are 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 an 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 type.

Renaming a Global Type

Renaming a global type renames all local copies of the type. 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 attempt to rename a type in a VOB that does not contain a local copy of the specified type, **rename** 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, see the **rename** reference page.

Changing the Scope of a Type

To convert an existing ordinary type to a global type, enter the appropriate **mk***object***type** command with the options **–replace**, **–global**, and **–acquire**. This converts the type to a global type, and also converts any identical types in other VOBs in the hierarchy to local copies of the type. For example:

1 A VOB and its administrative 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 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 other label type 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 the appropriate **mk***object***type** command with the options **–replace –ordinary**. This converts 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**, defined in VOB \admin, to an ordinary type, enter the following command:

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

You can also use the **–replace** option to change the scope of a type if you created the type or are a privileged user.

Removing a Global Type

Removing a global type or any of its copies removes all local copies and the global type itself. The **rmtype** command lists all VOBs that have local copies of the global type, then prompts for confirmation of the removal. You must use the **–rmall** option when removing a global type 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".
```

Note: If you enter a **rmtype** command in a VOB that does not contain a local copy of the global type, **rmtype** 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 that prevented the local copy from being removed and enter the **rmtype** command again.

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

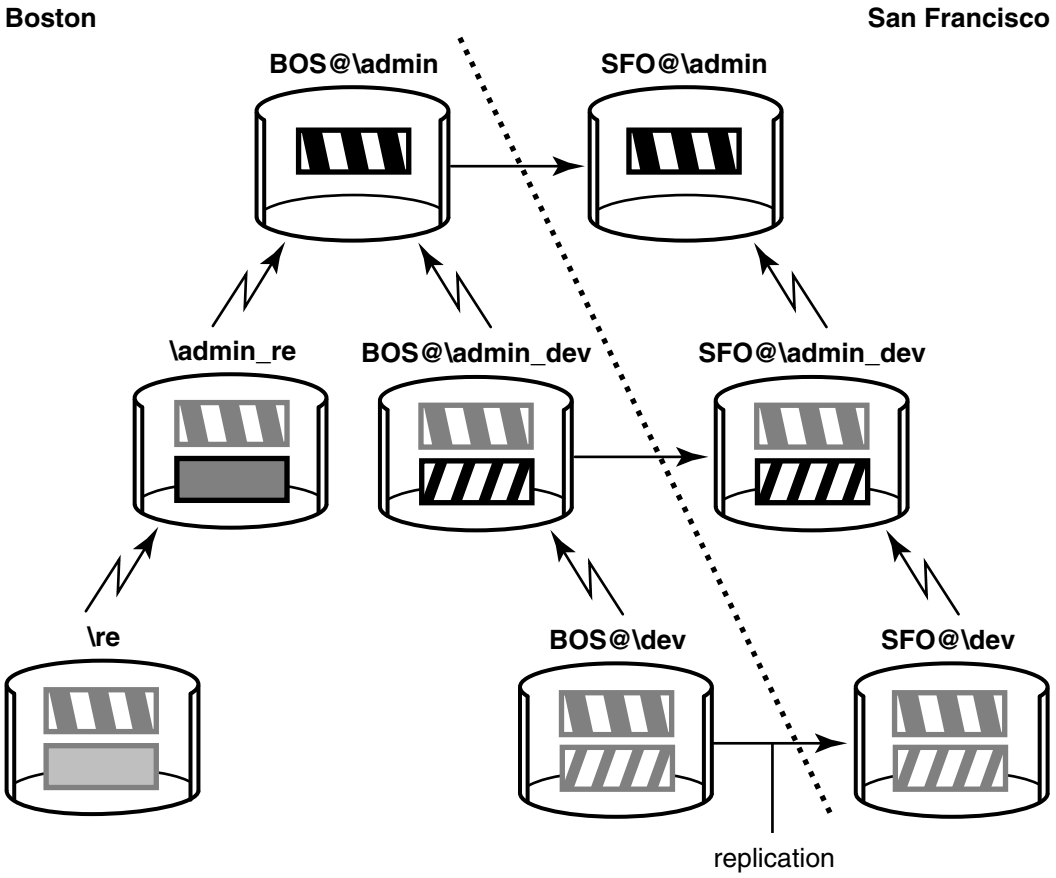
Cleaning Up Global Types

Use **checkvob –global** to check and fix global types that are in an inconsistent state. For more information, see *Using checkvob to Find and Fix Broken Hyperlinks* on page 241.

Replicated Administrative VOB Hierarchies

When you use administrative VOB hierarchies (including those created by UCM) in a Rational ClearCase MultiSite environment, special considerations apply to replication. If a VOB that is part of an administrative VOB hierarchy is replicated, other members of the hierarchy (or at least those members further up in the hierarchy) must also be replicated. In the example shown in Figure 5, \re has an **AdminVOB** hyperlink to \admin, \re is replicated, so all sites that have a replica of \re must also have a replica of \admin.

Figure 5 Replication Requirements of Administrative VOB Hierarchies



If you replicate a VOB that is part of an administrative VOB hierarchy, the **mkreplica –export** command prints a reminder that you must replicate all VOBs in the hierarchy above the VOB you are replicating. The output lists these 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 VOB are linked to global type objects in its administrative VOB hierarchy, we recommend that you synchronize all members of the hierarchy at the same time. If you do not, users may have trouble accessing type objects.

Note: In Figure 5, the VOB \admin_re has an **AdminVOB** hyperlink to \admin but is not replicated. In this situation (an administrative VOB is replicated but one or more lower-level members of its hierarchy are not), **cleartool** type-creation commands produce messages of the form:

```
Error: Unable to find replica in registry for VOB with object ID:"<VOB-oid>"
Error: Unable to locate versioned object base with object id:"<VOB-oid>"
```

These errors are not important if you did not intend to replicate the VOB referred to in the message, and they do not prevent creation of the type as specified.

Global Types and Mastership

You can create a local copy of a global type only if the global type is mastered by a replica at the current site. If the global type is not mastered at the current site, you can create instances of the type in a local replica only if the replica already 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 of the type. This restriction prevents creation of conflicting types at multiple sites.

If a VOB at your site does not contain a local copy of the type, you must:

1. Create a local copy at the site that masters the type.
2. Export an update packet from the replica at the mastering site to the replica at your site.
3. Import the packet at your site.

For example (see Figure 5), an engineer at your site (San Francisco) tries to apply the **RELEASE-2.1** label to a version in the \dev VOB. The command fails because the label type is a global type mastered at a different site and no local copy exists in \dev.

cleartool mklabel –nc RELEASE-2.1 \dev\file.txt

```
cleartool: Error: Type must be mastered in original replica "SFO" to
use copy type.
cleartool: Error: Unable to create label "RELEASE-2.1" on
"\dev\file.txt" version "/main/3".
```

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

1. Determine the tag of the administrative VOB where the global type is defined.

   cleartool describe vob:\dev

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

2. Determine which replica of the administrative VOB masters the type. 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, in addition to the replica name, a `master replica` line that displays the VOB tag of the master replica.

cleartool describe –long lbtype:RELEASE-2.1@\admin

```
label type "RELEASE-2.1"
  created 03-Aug-02.12:29:00 by Pete Sharon (pds.user@argon)
  master replica: BOS@\admin
  instance mastership: shared
  owner: pds
  group: user
  scope: global
  constraint: one version per branch
  Hyperlinks:
    GlobalDefinition@43@\admin <- lbtype:RELEASE-2.1@\admin
```

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

cleartool cptype –c "forced local copy" lbtype:RELEASE-2.1@\admin \ lbtype:RELEASE-2.1@\dev

```
Copied type "RELEASE-2.1".
```

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

multitool syncreplica –export –fship SFO@\dev

5 At your site, import the update packet.

multitool syncreplica –import –receive

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

cleartool mklabel –nc RELEASE-2.1 \dev\file.txt

```
Created label "RELEASE-2.1" on "\dev\file.txt" version "/main/3".
```

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 does not change the mastership of the global type or any other local copies.

For example, the global label type **RELEASE-3.0** is mastered by the San Francisco replica in the VOB family **\admin_dev**, and the local copy in the VOB **\dev** is mastered by the San Francisco replica in the VOB family **\dev**:

cleartool describe –fmt "%n\n %[master]p\n %[type_scope]p\n" lbtype:RELEASE-3.2@\admin_dev

```
RELEASE-3.2
  SFO@\admin_dev
  global
```

cleartool describe –local –fmt "%n\n %[master]p\n %[type_scope]p\n" \ lbtype:RELEASE-3.2@\dev

```
RELEASE-3.2
  SFO@\dev
  local copy
```

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

multitool chmaster BOS@\admin_dev lbtype:RELEASE-3.2@\admin_dev

```
Changed mastership of label type "RELEASE-3.2" to "BOS@\admin_dev"
```

cleartool describe –fmt "%n\n %[master]p\n %[type_scope]p\n" \ lbtype:RELEASE-3.2@\admin_dev

```
RELEASE-3.2
  BOS@\admin_dev
  global
```

cleartool describe –local –fmt "%n\n %[master]p\n %[type_scope]p\n" \ lbtype:RELEASE-3.2@\dev

```
RELEASE-3.2
  SFO@\dev
  local copy
```

If you enter a **chmaster** command in a 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 BOS@\admin lbtype:DOC_SOURCE@\dev

```
multitool: Error: Label type not found: "DOC_SOURCE".
```

For more information about mastership, see the **chmaster** reference page and the *Administrator's Guide* for Rational ClearCase MultiSite.

Moving VOBs and Relocating VOB Data

8

Rational ClearCase provides tools for moving data and metadata from one VOB to another and for moving entire VOBs from one disk or host to another. You may need to use these tools to reorganize VOB directories, redistribute data storage, or rebalance server loads.

Relocating Elements to Another VOB

The **cleartool relocate** command moves directory elements and the elements they contain from one VOB to another. The VOBs can be on the same server host or on different server hosts, but they must have tags in the same registry region. A **relocate** operation is appropriate when you need to reorganize VOB directories to reflect changes in component architecture or organizational structure or when you need to move groups of elements out of a VOB to reduce its size, or provide better load balancing across VOB servers.

Note: You cannot use **relocate** in a UCM component VOB or PVOB. Before you perform any **relocate** operation, read the **relocate** reference page.

What Does relocate Do?

The **relocate** command moves elements from one VOB (the source VOB) to another VOB (the target VOB). In many cases, the target VOB is created for this purpose. The **relocate** command normally does all of the following:

- Moves elements from one VOB to another. For each element, the following data and metadata are moved:
  - The element's data containers
  - The element's event history (some minor events are lost)
  - Bidirectional hyperlinks attached to the element
  - Related VOB database records
  - The config records associated with versioned derived objects
- Preserves the source VOB's namespace by creating appropriate symbolic links to the target VOB.

- Copies metadata types associated with moved elements to the target VOB, as necessary (label, attribute, element, trigger, and hyperlink types).
- Creates relocation 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.
- Creates a log of its activities—by default, in the file relocate.log.*date-time* in the directory where you run **relocate**.

relocate does not do the following:

- Relocate elements when either the source or the target VOB is a UCM component VOB or PVOB.
- Move view-private files and nonversioned DOs stored in relocated directories.
- Move elements to a new location in the same VOB. (Use **cleartool mv** for this purpose.)

Steps to Take Before Relocating Elements

Take the following steps before relocating elements:

1. **(MultiSite only) Coordinate with administrators of other replicas**. Elements to be relocated must be inactive at all replica sites. See the section about replicated VOBs in the **relocate** reference page.
2. **(Dynamic views only) Move view-private files out of directories that you plan to relocate.** View-private files that are not moved become stranded after relocation is complete. Stranded files can be recovered by running the **recoverview** command, but it is easier to move them before running **relocate**.
3. **Resolve all checkouts in directories that you plan to relocate.** The **relocate** command aborts if it encounters a checked-out version in any directory it is trying to relocate. To find and resolve (check in or cancel) these checkouts, use the **Referenced Views** subnode of the **VOBs** node in ClearCase Administration Console. You can also use the **cleartool lsdo** and **lscheckout** commands.
4. **Remove any rmelem triggers.** Unless it is running in **–update** mode, **relocate** uses **rmelem** to remove elements from the source VOB. If a trigger is attached to **rmelem**, this step fails and you must remove these elements manually (see *Errors While Removing Elements from the Source VOB* on page 144).
5. **Establish a working view.** Use a view whose config spec selects the branch (typically **main**) on which the move is to occur.

The view in which you run **relocate** must be able to check out elements in both the source and target VOBs, so its config spec must include a **CHECKEDOUT** rule. After you run **relocate**, you may need to use this view (or a view with the same config spec) to clean up, as described in *Problems with Existing Views After Relocating Elements* on page 262.

6 **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]
```

Handling Borderline Elements

When an element is cataloged in more than one VOB directory (usually because it has been moved with **cleartool mv**), **relocate** classifies it as a borderline element if any directory in which it is cataloged does not appear in the set of elements selected by the view in which you run **relocate**. To find borderline elements, run **relocate** with the **–qall** option,

When **relocate –qall** encounters a borderline element, it produces output like this:

```
Element "main.c" is located outside the selection tree as "main.c"
in directory "\sources\libs",located inside the selection tree as
"main.c" in directory "\sources\proto\libs".
Do you want to relocate it? [no]
```

We recommend that you either expand the selection set to relocate all versions of the element or leave the element in place by accepting the default response to the prompt (**no**). If you relocate a borderline element, views that reference any version of the element that was not relocated cannot see the element. You may be able to correct this problem manually, as described in *Problems with Existing Views After Relocating Elements* on page 262.

Handling Errors During a Relocate Operation

The following conditions are the most frequent causes of failed relocate operations:

- Checked-out files in directories to be relocated
- Locked type objects in the target VOB
- Triggers on **rmelem** in the source VOB

In most cases, you can fix the reported error and restart **relocate** with the same command line; the operation continues from the point of the error. Errors that occur when elements are being removed from the source VOB require manual intervention.

Errors Involving Locked Types or Checked-Out Files

You can restart **relocate** after errors such as locked types or checked-out files. When you encounter one of these errors:

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 the checkout.
2. **Restart relocate.** When invoked with an identical command line, **relocate** resumes processing from the interrupt point if all of 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.

Errors While Removing Elements from the Source VOB

After **relocate** has recreated elements in the target VOB, it removes the elements from the source VOB (unless the **–update** option was used). If this step fails (typically because of a trigger on **rmelem**), you must remove the elements manually, as follows:

1. **Find element OIDs from log file.** Examine the relocate log file and find the lines that specify the object IDs (OIDs) of the elements that **relocate** tried to remove but could not.
2. **Remove the elements.** For each element reported in the log file, run this command:

   cleartool rmelem oid:*OID-reported-in-log-file*
3. If necessary, remove the trigger that prevented **relocate** from removing the elements.

To avoid these errors, remove any **rmelem** triggers on elements before you relocate them.

After Relocating Elements

Although **relocate** leaves behind symbolic links to keep the VOB namespace consistent, always check existing views, build scripts, triggers, and any other tools that require access to the relocated elements. Update these views and tools to access relocated elements at their new location, rather than relying on symbolic links.

Note: Because nonversioned DOs are not relocated (they are instead removed, as if by **rmdo**), they must be rebuilt. Relocating any element causes any DOs that include the element as a dependency to be rebuilt.

For more information about troubleshooting problems that may arise after relocating elements, see *Problems with Existing Views After Relocating Elements* on page 262.

Moving VOBs

You cannot use an ordinary file system copy utility to copy a VOB from one location to another. You must follow special procedures that maintain the integrity of VOB data and preserve permissions and ownership on VOB storage directories. The remainder of this chapter presents procedures for moving VOBs. The following scenarios are covered:

- Moving a VOB to another disk partition on Windows or to another Windows host in the same domain.
- Moving a VOB to a Windows host in a different domain.
- Moving a VOB to another disk partition on UNIX or to another UNIX host with the same architecture (binary data format).
- Moving a VOB to a UNIX host with a different architecture.
- Moving a VOB from a Windows host to a UNIX host.
- Moving a VOB from a UNIX host to a Windows host.

This chapter also describes special considerations 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.

Important Steps to Take When Moving Any VOB

The procedures described in this section differ in various ways, but the following considerations apply when moving any VOB.

- Unless you are moving a VOB between UNIX and Windows, or from one Windows domain to another, make sure that the ownership and access control information for the VOB storage directory is preserved when the directory is copied. Most file system copy utilities, especially those that run on Windows, do not preserve this information. If this information is changed during the copy step of the VOB move procedure, the VOB cannot be used in its new location until you

fix the protections on the VOB storage directory. ClearCase provides several programs that can correct damaged file system protections on VOB storage directories. For more information, see *Repairing Storage Directory ACLS on NTFS* on page 255.

- All of the VOB move procedures in this section preserve the original VOB storage directory. After you are certain that the VOB can be accessed at its new location and that its contents are intact after the move, delete the old VOB storage directory to free the storage it would otherwise consume.
- If database snapshot backups are enabled for the VOB, use the **vob_snapshot_setup** program to disable them before you begin the move and enable them again after the move is complete.
- If you move a VOB formatted with schema version 53 to a host that supports schema version 54, you must reformat the VOB after the move is complete. You cannot move a VOB formatted with schema version 54 to a host that supports schema version 53. For more information, see *VOB Schema Versions* on page 90.
- Scrub the VOB's pools to reduce their size before moving the VOB.
- Ask users who are working in dynamic views to unmount the VOB before you begin any VOB move procedure. They can remount it after the procedure is complete.

Caution: Never register more than one copy of a VOB in a ClearCase registry. The procedures described in this chapter copy the VOB storage directory but do not register the copy before the original has been unregistered. If more than one copy of a VOB is registered in the same registry, even if the copies have tags in separate regions, you will encounter severe VOB errors and potential data loss. For more information about the ClearCase registry, see Chapter 3.

Special Considerations for Replicated VOBs

If you are moving a VOB replica to a new host, take the following steps first:

1 **Verify that the replica masters its own replica object.** For a replica named **portland**, check which replica has mastership of the **portland** replica object:

```
cleartool describe replica:portland@\libpub
...
master replica: west
```

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 we recommend 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 from 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.** After the last export from the replica's current location, 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).

Moving a VOB on Windows

This section describes two common procedures:

- Moving a VOB to a host in the same Windows domain. You can also use this procedure to move a VOB to another partition on a Windows host.
- Moving a VOB to a host in another Windows domain.

Moving a VOB Within a Domain

The following procedure describes how to move the VOB **\libpub** from storage directory C:\ClearCaseStorage\VOBs\libpub.vbs on VOB server host **\\sol** to a storage directory shared as **vobstg** on VOB server host **\\vobsvr01**.

1 **Log on to the VOB's current server as the VOB owner or privileged user.** In this example, the VOB's current server is **\\sol**.

2 **Lock the VOB**.

3 **Stop ClearCase** on the VOB server host.

4 **Copy the VOB storage directory, preserving all ownership information**. You must use a copy utility that preserves ownership information contained in the VOB storage directory ACLs. See *Preserving NTFS ACLs When Copying a VOB or*

View Storage Directory on page 257. We recommend that you use the ClearCase utility *ccase-home-dir*\etc\utils\ccopy.exe for this purpose.

```
C:\ClearCaseStorage\VOBs> net use E: \\vobsvr01\vobstg

C:\ClearCaseStorage\VOBs> ccopy libpub.vbs E:\libpub.vbs
```

5 **Restart ClearCase** if you have copied the VOB storage directory to a new location on the same VOB server host.

6 **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console or the following commands (this example applies to the destination on server **vobsvr01**):

```
cleartool register –vob –replace \\vobsvr01\vobstg\libpub.vbs

cleartool mktag –vob –replace –tag \libpub \\vobsvr01\vobstg\libpub.vbs
```

7 **Unlock the VOB.**

8 **Verify that all clients can access the VOB at the new location.**

Moving a VOB to a Different Domain

On Windows, VOBs formatted with schema version 54 store Windows security identifiers (SIDs) to represent users, groups, and resources (hosts). When you move a VOB to a different domain, these SIDs become invalid and must be changed (mapped) to SIDs that are valid in the new domain. ClearCase includes a utility program, **vob_sidwalk**, that provides a flexible means of mapping SIDs after you move a VOB to a different domain. We strongly recommend that you review the **vob_sidwalk** reference page before continuing with this procedure.

The following procedure moves the VOB **\libpub** from storage directory C:\ClearCaseStorage\VOBs\libpub.vbs on VOB server host **\\sol**, which is in the OLD domain, to a storage directory shared as **vobstg** on VOB server host **\\vobsvr-new**, which is in the NEW domain. To execute this procedure, you must be able to log in to both the OLD and NEW domains as the VOB owner of **\libpub** or as a privileged user.

1 **Verify that the VOB is formatted with schema version 54**. Lower-numbered schema versions do not support moving a VOB to a different domain. You can use the ClearCase Administration Console or the **cleartool describe** command to determine a VOB's schema version. For more information about VOB schema versions and how to change them, see *VOB Schema Versions* on page 90.

2 **Log on to the VOB server host as the VOB owner or privileged user.**

3 **Lock the VOB.** This ensures that no new VOB objects are created while you complete Step 4.

4 **Generate a SID file** that lists the names of users and groups associated with objects in **\libpub**. Run **vob_siddump** to generate a SID file in comma-separated-value (csv) format:

*ccase-home-dir***etc\utils\vob_siddump \libpub ^**
C:\ClearCaseStorage\VOBs\libpub.vbs\libpub.csv

We suggest creating the SID file in the VOB storage directory so that it is available on the new VOB host after the storage directory has been moved. (You need it in Step 10.)

5 **Stop ClearCase** on the VOB server host.

6 **Copy the VOB storage directory** to the new location.

```
C:\ClearCaseStorage\VOBs> net use E: \\vobsvr-new\vobstg

C:\ClearCaseStorage\VOBs> xcopy libpub.vbs  E:\libpub.vbs /s
```

Note: Because the existing VOB storage directory ACLs are invalid in the new domain, you may use **xcopy** or another copy utility that does not preserve ACLs for this step.

7 **Fix the VOB storage directory protections.** Log on to the VOB server host in the new domain (**\\vobsvr-new** in our example) as the VOB owner of **\libpub** or as a privileged user. Run the **fix_prot** utility. In this example, **vobadm** is the name of the new VOB owner, **ccusers** is the name of the VOB's new principal group, and V:\vobstg\libpub.vbs is the host-local pathname of the VOB storage directory on **\\vobsvr-new**:

*ccase-home-dir***etc\utils\fix_prot –root –r –chown vobadm –chgrp ccusers ^**
V:\vobstg\libpub.vbs

8 **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console or the following commands:

cleartool register –vob –replace \\vobsvr-new\vobstg\libpub.vbs

cleartool mktag –vob –replace –tag \libpub \\vobsvr-new\vobstg\libpub.vbs

If **\\vobsvr-new** is not in the same registry region as **\\sol**, you do not need to use the **–replace** option to **cleartool register** and **cleartool mktag**, but you should remove the old registration and tag for \libpub, which will be invalid after the move.

9 **Lock the VOB.** Although the VOB is now registered and has a tag, it will not be usable until you complete this procedure. If you are concerned that users may try to access the VOB before it is ready, lock it now.

10 **Create a map file.** Open the SID file generated in Step 4 (\\vobsvr-new\vobstg\libpub.vbs\libpub.csv). It may be easier to edit this file if you use a spreadsheet program that can read the comma-separated-value format. This example shows one line of such a file. It includes a header row for clarity. The SID string has been truncated to save space.

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
OLD\akp	USER	NT:S-1-2-21-532...	IGNORE	USER		137

For each line in the file, replace the string **IGNORE** in the **New-name** field with a string made up of the new domain name and the user name from the **Old-name** field; then delete the last three fields (**Type**, **New-SID**, and **Count**). In this example, old domain's name is OLD and the new domain's name is NEW, so the line would change, as shown here:

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
OLD\akp	USER	NT:S-1-2-21-532...	NEW\akp			

Although this example shows a user name that is the same in the old and new domains, the procedure can also be used to map a user or group name from the old domain to a different user or group name in the new domain.

After you have edited all the rows of the SID file, save it as a comma-separated-value file and use it as the mapping file required when you run **vob_sidwalk –map**. Each line of the mapping file must have exactly four fields, separated by commas. The example row created in this step would look like this in .csv form:

```
OLD\akp,USER,NT:S-1-2-21-532...,NEW\akp
```

Note: You can reassign ownership of any object in a VOB to the VOB owner by placing the string **DELETE** in the **New-name** field. You may also reassign ownership of all objects in a VOB to the VOB owner without creating a mapping file. See *Reassigning Ownership to the VOB Owner* on page 337.

11 **Test the map file.** Run **vob_sidwalk** without the **–execute** option. The list of mappings in the map file libpub-map.csv is written to the SID file (libpub-test.csv in this example), but no changes are made to the VOB.

*ccase-home-dir***\etc\utils\vob_sidwalk –map ^**
\\vobsvr-new\vobstg\libpub.vbs\libpub-map.csv \libpub libpub-test.csv

12 **Unlock the VOB.** If you are concerned that users may try to access the VOB before this procedure is complete, lock the VOB again for all users except yourself (**cleartool lock –nusers** *you*). You must have write access to the VOB to complete this procedure.

13 **Update user and group identities stored in the VOB.** When you are satisfied that the map file is correct, run **vob_sidwalk**. In this example, libpub-map.csv is the map file created in Step 10:

*ccase-home-dir***etc\utils\vob_sidwalk –execute –map ^**
\\vobsvr-new\vobstg\libpub.vbs\libpub-map.csv \libpub libpub-exec.csv

vob_sidwalk remaps ownership as specified in the map file and records the changes made in libpub-exec.csv.

14 **Recover file system ACLs.** While you are still logged on to **\\vobsvr-new** as the VOB owner or privileged user, use **vob_sidwalk** with the **–recover_filesystem** option to apply the correct ACLs to the VOB storage directory.

*ccase-home-dir***etc\utils\vob_sidwalk –recover_filesystem \libpub recov.csv**

vob_sidwalk logs changes made during this step to the file recov.csv

15 **Verify that all clients in the new domain can access the VOB.** Unlock the VOB if it is still locked.

16 **Verify that all ClearCase users in the new domain can access objects in the VOB.** Users should be able to create new objects as well as change or remove objects that they own.

Note: If the user's name in the new domain is not the same as in the old domain, the user loses rights (for example, the right to remove a version that you created) associated with the creator of a version or a branch. These operations can still be executed by a more privileged user (VOB owner, member of the ClearCase administrators group).

Moving a VOB on UNIX

This section describes two common procedures for moving VOBs on UNIX:

- Moving a VOB to a UNIX VOB server host that has the same architecture (binary data format). You can also use this procedure to move a VOB to another partition on a UNIX host.
- Moving a VOB to a UNIX host that has a different architecture.

For clarity, the procedures in this section use an example:

- The current location of the VOB storage directory to be moved is /vobstore/libpub.vbs, on a host named **sol**.
- The VOB tag is /vobs/libpub.
- The new location for the VOB storage directory is /vobstore2/libpub.vbs. The example includes these cases:
  - The new location is also on **sol**.
  - The new location is on another host, **vobsvr04**.

If the VOB Has Remote Pools

Any of the procedures in this section can be used to move a VOB that has remote pools. Before you begin the move, determine whether the VOB has any remote pools and verify that the pools will be accessible after the move.

1 **Determine whether the VOB has any remote 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/vobsvr04/ccase_pools/s_2"
```

The output of **lspool** shows that this VOB has one remote pool, **s_2**.

2 **Verify that the remote pools are accessible on the target host**. 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 use the same global pathnames used by the VOB's current host to access each remote storage pool:

- If you are moving the VOB to another location on the same host, these global pathnames will continue to be valid.
- If you are moving the VOB to a different host, log on to that host and verify that all the remote storage pools can be accessed from that host.

Consolidating Remote Pools

If you are moving a VOB that has remote pools from UNIX to Windows (see *Moving a VOB from UNIX to Windows* on page 159), you must consolidate the remote pools before you move the VOB. You can also use this procedure to consolidate a VOB's remote pools after the VOB has been moved to a NAS device (see *Moving a VOB to Network-Attached Storage* on page 163).

1 **Log on to the VOB server host.** Log on as the VOB owner or privileged user.

2 **Find the remote pools.** Go to the VOB storage directory and determine the locations of all remote pools and the links that point to them. In this example, the UNIX **find** command shows a single symbolic link to a remote pool.

```
cd /vobstg/libpub.vbs
find . –type l –exec ls –l {} \;
lrwxrwxrwx   1 root           12 Dec 30  1999
d/ddft_2 ->/net/vobsvr5/pools/libpub/d/ddft_2
```

3 **Replace each remote pool with a local directory.** For each remote pool, replace the link with a local copy of the pool. You must preserve file and directory protection and ownership information during this operation. The UNIX commands in this step remove the symbolic link d/ddft_2 and replace it with the contents of the link's target, /net/vobsvr5/pools/libpub/d/ddft_2. (Note that if the target had a different terminal leaf, you would need to ensure that the contents were copied into a local directory named ddft_2.)

```
rm d/ddft_2

cd /net/vobsvr5/pools/libpub/d; tar –cf – ddft_2 | \
(cd /vobstg/libpub.vbs; tar –xBpf–)
```

4 **Verify that the VOB has no remote pools.** Stop and restart ClearCase on the VOB server host; then use the **cleartool lspool** command to verify that the VOB has no remote pools.

```
cleartool lspool –long –invob /vobs/libpub
```

The output of **lspool** should list no link targets.

5 **Modify the VOB tag.** If the consolidated VOB is not being moved to a new host (for example, if the VOB storage is being moved to a NAS device but the VOB server host remains the same) and has a tag in a Windows region, the tag must be modified to remove the split pool map. Use the ClearCase Administration Console's **Registry Regions** node. The **Properties** page for the VOB tag has a **Mount Options** tab that allows you to edit the split pool map if you are logged in as a member of the ClearCase administrators group. If you cannot use the ClearCase Administration Console, use **cleartool rmtag** and **mktag** to remove the VOB tag and recreate it without a split pool map.

6 **Verify that users can access the consolidated pools.** After you test the VOB, you may delete the old remote pool storage.

Note: Check and modify your VOB backup procedures after you consolidate remote pools. Verify that the newly consolidated pools are backed up with the rest of the VOB and that the old remote pools are no longer backed up. If you restore backups that were made before the pools were consolidated, the remote pools are recreated, and the restored VOB is not usable.

If the VOB Is Exported for Non-ClearCase Access

You can use any procedure in this section to move a VOB that has been exported for non-ClearCase access. Before you begin, stop any export views that export the VOB. After the move is complete, restart the export views.

For more information, see *Setting Up an Export View for Non-ClearCase Access* on page 356.

Moving a VOB Between UNIX Hosts That Have the Same Architecture

You can use the following procedure to move a VOB to another disk partition on a UNIX host, or to another UNIX host that has the same architecture.

1. **Log on to the VOB server host.** Log on as the VOB owner or privileged user.
2. **Lock the VOB**.
3. **Stop ClearCase** on the VOB server host.
4. **Copy the VOB storage directory**. The procedure you use depends on whether you're moving the VOB within the same disk partition or to another disk partition.

   If you are moving the VOB to another disk partition, use **tar** or a similar command to copy the entire VOB storage directory, but not the remote storage pools, to the new location. For example, to move a VOB to a different disk partition on the same host:

   cd /vobstore

   tar –cf – libpub.vbs | (cd /vobstore2 ; tar –xBpf –)

   To move a VOB to a different host:

   cd /vobstore

   tar –cf – libpub.vbs | rsh vobsvr04 'cd /vobstore2 ; tar –xBpf –'

   Note: The **–B** option to the **tar** command may not be supported on all UNIX platforms. Also, the **rsh** command may have a different name, such as **remsh**, on some platforms. See the reference pages for your operating system.

   If you are moving the VOB storage directory within the same disk partition, use the UNIX **mv** command.
5. **Restart ClearCase** if you have copied the VOB storage directory to a new location on the same VOB server host.

6 **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console or the following commands (this example applies to the destination on server **sol**):

cleartool register –vob –replace /net/sol/vobstore2/libpub.vbs

cleartool mktag –vob –replace –tag /vobs/libpub /net/sol/vobstore2/libpub.vbs

7 **Unlock the VOB.**

8 **Verify that all clients can access the VOB at the new location.**

Moving a VOB Between UNIX Hosts That Have Different Architectures

Use the following procedure to move a VOB to a UNIX host that has a different architecture. The procedure is similar to that described in *Moving a VOB Between UNIX Hosts That Have the Same Architecture*, but it includes the additional steps required to dump the VOB database before it is moved and then reformat it on the target host.

1 **Log on to the VOB server host.** Log on as the VOB owner or privileged user.

2 **Dump the VOB database** with the **cleartool reformatvob –dump** command. (This also locks the VOB.)

cleartool reformatvob –dump /vobstore/libpub.vbs

reformatvob –dump marks the VOB database as invalid. It cannot be used until it is processed by a **reformatvob –load** command.

3 **Copy the VOB storage directory**. First, verify that the target location exists and is writable. Then, copy the VOB storage directory to the new host.

cd /vobstore

tar –cf – libpub.vbs | rsh vobsvr04 'cd /src/vobstore ; tar –xBpf –'

Note: The **–B** option to the **tar** command may not be supported on all UNIX platforms. Also, the **rsh** command may have a different name, such as **remsh**, on some platforms. See the reference pages for your operating system.

4 **Terminate the old VOB's server processes**. You can either stop and restart ClearCase on the VOB server host or search the process table for the **vob_server** and **vobrpc_server** processes that service the old VOB. Use **ps –ax** or **ps –ef**, and search for the VOB storage directory name (libpub.vbs in our example); then use the UNIX **kill** command to terminate any such processes.

5 **Log on to the new VOB server host.** Log on as the VOB owner or privileged user.

6 **Recreate the VOB database from the dump files**:

cleartool reformatvob –load /src/vobstore/libpub.vbs

7 **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console or the following commands (this example applies to the destination on server **sol**):

cleartool register –vob –replace /net/sol/vobstore2/libpub.vbs

cleartool mktag –vob –replace –tag /vobs/libpub /net/sol/vobstore2/libpub.vbs

8 **Unlock the VOB.**

9 **Verify that all clients can access the VOB at the new location.**

Moving a VOB to a Different OS Type

When you move a VOB from a Windows host to a UNIX host or vice versa, all user identity information stored in the VOB storage directory and VOB database must change. The binary data format of the VOB database must change as well. To accommodate these requirements, you must take a number of additional steps beyond those required in most other VOB move scenarios:

- Run **vob_sidwalk** before the move to capture information about ownership of VOB objects.
- Use **reformatvob** to dump the VOB database into a portable form.
- Copy the VOB storage directory, which includes the dumped database, to the new host.
- Use **reformatvob** to load the VOB database in the proper binary format.
- Reset the file system protections on the VOB storage directory after the move.
- Remap the SIDs (or, on UNIX, UIDs and GIDs) of owners of objects in the VOB.

These steps, along with others required by all VOB moves, are described in this section.

Schema Version Compatibility

Before a VOB can be moved from a Windows host to a UNIX host, the VOB must be formatted with schema version 54 and the UNIX host must be configured to support VOB schema version 54. Some UNIX hosts do not have this capability. For more information, see *VOB Schema Versions* on page 90.

vob_sidwalk and **vob_siddump** are not compatible with VOB schema version 53. **vob_sidwalk** is installed only on hosts that are configured to support local VOBs and views and to support VOB schema version 54. **vob_siddump**, which is not restricted to operating on local VOBs, is installed on all hosts.

Moving a VOB from Windows to UNIX

For clarity, the procedures in this section use an example:

- The current location of the VOB storage directory to be moved is C:\ClearCaseStorage\libpub.vbs on Windows host **vobsvr-nt**. The VOB tag for this VOB is \libpub.
- The new location for the VOB storage directory is /vobstg/libpub.vbs on UNIX host **vobsvr2**. VOB tag /vobs/libpub will be created for this VOB.

To move a VOB from Windows to UNIX:

1. **Log on to the Windows VOB server host as the VOB owner or privileged user**.
2. **Lock the VOB.** This ensures that no new VOB objects are created while you complete Step 3.
3. **Generate a SID file** that lists the names of users and groups associated with objects in **\libpub**. Run the **vob_siddump** utility as shown in this example:

   *ccase-home-dir***\etc\utils\vob_siddump –raw_sid \libpub ^**
   C:\ClearCaseStorage\libpub.vbs\libpub.csv

   We suggest creating the SID file in the VOB storage directory so that it is available on the new VOB host after the storage directory is moved. You need the SID file in Step 15.
4. **Dump the VOB database.** Use the **cleartool reformatvob** command:

   cleartool reformatvob –dump C:\ClearCaseStorage\libpub.vbs

   reformatvob –dump marks the VOB database as invalid. It cannot be used until it is processed by a **reformatvob –load** command.
5. **Copy the VOB storage directory**. Use any file system copy utility to copy the entire VOB storage directory to the UNIX host. This example assumes that the target UNIX host **vobsvr2** is running an SMB server and has shared its **\vobstg** partition.

   ```
   C:\ClearCaseStorage\VOBs> net use E: \\vobsvr2\vobstg

   C:\ClearCaseStorage\VOBs> xcopy libpub.vbs E:\libpub.vbs /s
   ```

   Note: Because ACLs are not supported on the UNIX host, you may use **xcopy** or another copy utility that does not preserve ACLs for this step.
6. **Terminate the VOB's server processes on Windows**. Stop and restart ClearCase on the Windows VOB server host (**\\vobsvr-nt** in our example).
7. **Log on to the UNIX VOB server host.**

8 **Update VOB owner identity information.** Use the **fix_prot** utility as shown here to create a new .identity directory for the VOB. (You must be **root** to run **fix_prot** in this way.) This example sets the VOB's owner to **vobadm** and the VOB's primary group to **ccusers**:

*ccase-home-dir***/etc/utils/fix_prot –root –recurse –chown vobadm –chgrp ccusers ^ /vobstg/libpub.vbs**

9 **Recreate the VOB database from the dump files**:

```
# cleartool reformatvob –load /vobstg/libpub.vbs
```

10 **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console or the following commands:

```
cleartool register –vob –replace /net/vobsvr2/vobstg/libpub.vbs

cleartool mktag –vob –replace –tag /vobs/libpub /net/vobsvr2/vobstg/libpub.vbs
```

11 **Create a map file.** Open the SID file generated in Step 3 (/vobstg/libpub.vbs/libpub.csv). It may be easier to edit this file if you use a spreadsheet program that can read the comma-separated-value format. This example shows one line of such a file. It includes a header row for clarity. The SID string has been truncated to save space.

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
OLD\akp	USER	SID:3.0105037...	IGNORE	USER		137

For each line in the file, replace the string **IGNORE** in the **New-name** field with a user or group name that is valid on the UNIX VOB server host; then delete the last three fields (**Type**, **New-SID**, and **Count**).

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
OLD\akp	USER	SID:3.0105037...	akp			

Although this example shows a user name that is the same on UNIX as it was on Windows, the procedure can also be used to map a Windows user or group name to a different user or group name on UNIX.

After you edit all the rows of the SID file, save it as a comma-separated-value file and use it as the mapping file required when you run **vob_sidwalk –map**. Each line of the mapping file must have exactly four fields, separated by commas. The example row created in this step looks like this in .csv format:

```
OLD\akp,USER,SID:3.0105037...,akp
```

Note: You can reassign ownership of any object in a VOB to the VOB owner by placing the string **DELETE** in the **New-name** field. You may also reassign ownership of all objects in a VOB to the VOB owner without creating a mapping file. See *Reassigning Ownership to the VOB Owner* on page 337.

12 **Test the map file.** Run **vob_sidwalk** without the **–execute** option. The list of mappings in the file libpub-map.csv is written to the SID file (libpub-test.csv in this example), but no changes are made to the VOB.

ccase-home-dir/**etc/utils/vob_sidwalk –map /vobstg/libpub.vbs/libpub-map.csv \ /vobs/libpub /libpub-test.csv**

13 **Unlock the VOB.** If you are concerned that users may try to access the VOB before you complete this procedure, lock the VOB again for all users except yourself (**cleartool lock –nusers** *you*).

14 **Update user and group identities stored in the VOB.** When you are satisfied that the map file is correct, run **vob_sidwalk**. In this example, libpub-map.csv is the map file created in Step 11:

ccase-home-dir/**etc/utils/vob_sidwalk –execute –map \ /vobstg/libpub.vbs/libpub-map.csv /vobs/libpub /libpub-exec.csv**

vob_sidwalk makes the changes specified in the map file and records the changes that were made in a new SID file, libpub-exec.csv.

15 **Update the VOB's group list and container protections**. Run **vob_sidwalk** with the **–recover_filesystem** option:

ccase-home-dir/**etc/utils/vob_sidwalk –recover_filesystem /vobs/libpub recov.csv**

vob_sidwalk logs changes made during this step to the file recov.csv

16 **Verify that all clients can access the VOB at the new location.** Unlock the VOB if it is still locked.

17 **Verify that all ClearCase users in the new domain can access objects in the VOB.** Users should be able to create new objects and change or remove objects they own.

Moving a VOB from UNIX to Windows

For clarity, the procedures in this section use an example:

- The current location of the VOB storage directory to be moved is /vobstg/libpub.vbs on UNIX host **vobsvr2**. The VOB tag for this VOB is /vobs/libpub.

- The new location of the VOB storage directory is C:\ClearCaseStorage\VOBs\libpub.vbs on Windows host **vobsvr-nt**. VOB tag **\libpub** will be created for this VOB.

To move a VOB from UNIX to Windows:

Note: If the VOB has remote storage pools, you must first consolidate those pools under the VOB root directory. UNIX symbolic links are not supported on Windows, so the entire VOB storage directory must reside on a single partition on the Windows host. See *Consolidating Remote Pools* on page 152 for the procedure.

1 **Log on to the VOB server host.** Log on as the VOB owner or privileged user.

2 **Lock the VOB.** This ensures that no new VOB objects are created while you complete Step 3.

3 **Generate a SID file** that lists the names and UIDs/GIDs of users and groups associated with objects in /vobs/libpub. Run **vob_siddump** utility as shown in this example:

*ccase-home-dir***/etc/utils/vob_siddump /vobs/libpub ^**
/vobstg/libpub.vbs/libpub.csv

We suggest creating the SID file in the VOB storage directory so that it is available on the new VOB host after the storage directory is moved. You will need this file in Step 15.

4 **Dump the VOB database**. Use the **cleartool reformatvob** command:

cleartool reformatvob –dump /vobstg/libpub.vbs

reformatvob –dump marks the VOB database as invalid. It cannot be used until it is processed by a **reformatvob –load** command.

5 **Copy the VOB storage directory**. Use any file system copy utility to copy the entire VOB storage directory to the Windows host. This example assumes that the UNIX host **vobsvr2** is running an SMB server and has shared its **\vobstg** partition. On the Windows host, run these commands:

```
C:\ClearCaseStorage\VOBs> net use E: \\vobsvr2\vobstg

C:\ClearCaseStorage\VOBs> xcopy E:\libpub.vbs libpub.vbs /s
```

Note: Because ACLs are not supported on the UNIX host, you can use **xcopy** or for this step.

6 **Terminate the VOB's server processes on UNIX**. You can either stop and restart ClearCase on the VOB server host or search the process table for the **vob_server** and **vobrpc_server** processes that service the old VOB. Use **ps –ax** or **ps –ef**, and

search for the VOB storage directory name (libpub.vbs in our example); then use the UNIX **kill** command to terminate any such processes.

7 **Fix the VOB storage directory protections.** Log in to the Windows VOB server host as the VOB owner of **\libpub** or as a privileged user and run the **fix_prot** utility. In this example, **vobadm** is the name of the new VOB owner, **ccusers** is the name of the VOB's new principal group, and C:\ClearCaseStorage\VOBs\libpub.vbs is the host-local pathname of the VOB storage directory:

*ccase-home-dir***etc\utils\fix_prot –root –r –chown vobadm –chgrp ccusers ^ C:\ClearCaseStorage\VOBs\libpub.vbs**

8 **Recreate the VOB database from the dump files**. Use the cleartool **reformatvob** command:

cleartool reformatvob –load C:\ClearCaseStorage\VOBs\libpub.vbs

9 **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the ClearCase Administration Console or the following commands (which assume that C:\ClearCaseStorage is shared as \\vobsvr-nt\ClearCaseStorage):

cleartool register –vob –replace ^ \\vobsvr-nt\ClearCaseStorage\VOBs\libpub.vbs

cleartool mktag –vob –tag \libpub ^ \\vobsvr-nt\ClearCaseStorage\VOBs\libpub.vbs

10 **Lock the VOB.** Although the VOB is now registered and has a tag, it cannot be usable until you complete this procedure. If you are concerned that users may try to access the VOB before it is ready, lock it now.

11 **Create a map file.** Open the SID file generated in Step 4 of this procedure (\\vobsvr-nt\ClearCaseStorage\VOBs\libpub.vbs\libpub.csv). It may be easier to edit this file if you use a spreadsheet program that can read the comma-separated-value format. This example shows one line of such a file. It includes a header row for clarity.

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
akp	USER	UNIX:UID-1247	IGNORE	USER		137

For each line in the file, replace the string **IGNORE** in the **New-name** field with a domain-qualified name of the user or group to which the old name should be mapped; then delete the last three fields (**Type**, **New-SID**, and **Count**).

Old-name	Type	Old-SID	New-name	Type	New-SID	Count
akp	USER	UNIX:UID-1247	NEW\akp			

Although this example shows a user name that is the same on Windows as it was on UNIX, this procedure can also be used to map a UNIX user or group name to a different user or group name on Windows.

After you have edited all the rows of the SID file, save it as a comma-separated-value file and use it as the mapping file required when you run **vob_sidwalk –map**. Each line of the mapping file must have exactly four fields, separated by commas. The example row created in this step looks like this in .csv format:

```
akp,USER,UNIX:UID-1247,NEW\akp
```

Note: You can reassign ownership of any object in a VOB to the VOB owner by placing the string **DELETE** in the **New-name** field. You can also reassign ownership of all objects in a VOB to the VOB owner without creating a mapping file. See *Reassigning Ownership to the VOB Owner* on page 337.

12 **Test the map file.** Run **vob_sidwalk** without the **–execute** option. The list of mappings in the file libpub-map.csv is written to the SID file (libpub-test.csv in this example), but no changes are made to the VOB.

*ccase-home-dir***\etc\utils\vob_sidwalk –map^**
\\vobsvr-nt\ClearCaseStorage\VOBs\libpub.vbs\libpub-map.csv ^
\libpub libpub-test.csv

13 **Unlock the VOB.** If you are concerned that users may try to access the VOB before this procedure is complete, lock the VOB again for all users except yourself. You will need write access to the VOB to complete this procedure.

14 **Update user and group identities stored in the VOB.** When you are satisfied that the map file is correct, run **vob_sidwalk** with the **–execute** option. In this example, libpub-map.csv is the map file you created in Step 10:

*ccase-home-dir***\etc\utils\vob_sidwalk –execute –map ^**
\\vobsvr-nt\ClearCaseStorage\VOBs\libpub.vbs\libpub-map.csv \libpub
libpub-exec.csv

vob_sidwalk remaps ownership as specified in the map file and records the changes that were made in a new SID file, libpub-exec.csv.

15 **Recover file system ACLs.** Finally, while you are still logged in to **\\vobsvr-nt** as the VOB owner or privileged user, use **vob_sidwalk** with the **–recover_filesystem** option to apply the correct ACLs to the VOB storage directory.

*ccase-home-dir***etc\utils\vob_sidwalk –recover_filesystem \libpub recov.csv**

vob_sidwalk logs changes made during this step to the file recov.csv

16 **Verify that all clients in the region can access the VOB.** Unlock the VOB if it is still locked.

17 **Verify that all ClearCase users on Windows can access objects in the VOB.** Users should be able to create new objects as well as change or remove objects they own.

Note: If the user's name on Windows is not the same as it was on UNIX, the user loses rights (for example, the right to remove a version that you created) associated with the creator of a version or a branch. These operations can still be executed by a more privileged user (VOB owner, member of the ClearCase administrators group).

Moving a VOB to Network-Attached Storage

All procedures for moving VOBs refer to the architecture, or binary data format, of the VOB host. VOB database formats are architecture-dependent, and the database must be reformatted if the VOB is moved from a host of one architecture to a host of another architecture.

When a VOB database is stored on a NAS device, the binary data format is determined by the architecture of the VOB server host, not the NAS device. You need to reformat a VOB database only if you are changing the architecture of the server host. If you move a VOB from one NAS device to another—even if the device comes from a different manufacturer—you do not have to reformat the VOB database as long as the architecture of the VOB server host does not change.

Moving a VOB That Has No Remote Pools

To move a VOB that has no remote pools from a ClearCase host running UNIX to a NAS device, follow the procedures in *Moving a VOB on UNIX* on page 151, but remember to use the **–host**, **–hpath**, and **–gpath** options to the **register** and **mktag** commands. The example here has been modified to register the VOB **libpub** that has been moved to /net/nasdevice/vobstg and is served by the UNIX host **ccvobsvr1**.

cleartool register –vob –replace –host ccvobsvr1
**–hpath /net/nasdevice/vobstg/libpub.vbs **
–gpath /net/nasdevice/vobstg/libpub.vbs /net/nasdevice/vobstg/libpub.vbs

**cleartool mktag –vob –replace –tag /vobs/libpub –host ccvobsvr1 **
**–hpath /net/nasdevice/vobstg/libpub.vbs **
–gpath /net/nasdevice/vobstg/libpub.vbs /net/nasdevice/vobstg/libpub.vbs

A similar change is required when moving a VOB from a Windows VOB server host to a NAS device. Follow the procedure described in *Moving a VOB on Windows* on page 147, but modify the **register** and **mktag** commands as shown here:

cleartool register –vob –replace –host sol –hpath \\sol\vobstore2\libpub.vbs ^
–gpath \\sol\vobstore2\libpub.vbs \\sol\vobstore2\libpub.vbs

cleartool mktag –vob –replace –tag \libpub –host ccvobsvr1 ^
–hpath \\sol\vobstore2\libpub.vbs –gpath \\sol\vobstore2\libpub.vbs ^
\\sol\vobstore2\libpub.vbs

Moving a VOB That Has Remote Pools

A VOB hosted on a UNIX platform can have one or more of its storage pools on a remote UNIX host accessed through a UNIX symbolic link. If you move a VOB with this configuration to a NAS device, you may want to consolidate the remote pools by replacing the symbolic links with local directories. As long as the remote pools are accessible to the VOB server host, you do not need to consolidate them, but doing so simplifies VOB backups and other administrative tasks. See *Consolidating Remote Pools* on page 152.

View Administration

9

This chapter describes the operational details and storage requirements of views. It also describes the MVFS (multiversion file system) that supports dynamic views. The **mkview** reference page has additional information about views.

Introduction to Views and View Administration

Any ClearCase development environment requires one or more views. A view provides a workspace where users access versions of file and directory elements that are under ClearCase control. Views can also contain view-private file system objects (such as ordinary files and directories) that are not under ClearCase control.

A typical view is created and used by an individual or perhaps by a small group working on a common task. Other views—for example, UCM integration views—may be created by a project leader or administrator and shared by many users. Various administrative responsibilities are associated with views:

- View creation and access control
- Backing up and recovering views
- Moving, removing, and managing the storage used by views

There are three kinds of views:

- Dynamic views (not supported by ClearCase LT) use the MVFS to provide transparent access to versions of elements in the VOB and also view-private objects. Dynamic views also support ClearCase build-auditing and build-avoidance tools and can contain derived objects (DOs). Developers work in dynamic views when they want immediate access to the latest versions of elements on a given branch or when they need to take advantage of audited builds and build-avoidance. Dynamic views provide more functionality, require more frequent use of the network, and need more administrative support than other views.
- Snapshot views use the host's native file system to hold copies of versions of specified elements and also view-private objects. Snapshot views do not support build-auditing and build-avoidance tools. Developers work in snapshot views when they want a simplified environment for editing, compiling, and debugging

(even when disconnected from the network), do not need immediate access to the latest versions of elements, and do not need audited builds and build-avoidance. Snapshot views provide less functionality, require less frequent use of the network, and need less administrative support than dynamic views.

- Web views are similar to snapshot views, but are created and accessed through the ClearCase Web interface.

Dynamic Views

Note: Dynamic views are not supported by ClearCase LT.

A dynamic view is an MVFS directory that enables dynamic access to VOB elements. Dynamic views contain nearly all the artifacts created during the software development process, whether or not they are under ClearCase control. These artifacts include the following:

- 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)
- Shareable and unshared derived objects built by users who work in the view (stored in the view's private storage area) and configuration records that describe these derived objects
- Shared derived objects built in another view and winked in to this view (stored in VOB storage pools)
- View-private objects: miscellaneous files, directories, and links that are not under ClearCase control and appear only in this view (stored in the view's private storage area).

The View Root

Every dynamic view on a host appears as a subdirectory of the host's view root directory. On UNIX computers, the default view root directory is /view. On Windows computers, the default view root directory is \\view, which is mapped by default to drive M.

In addition to the various view storage directories, the view root contains the special file .specdev. If this file is missing or damaged, attempts to access dynamic views on the host generate this error message:

```
cleartool: Error: Unable to open file "viewroot": ClearCase object not
found.
```

To Change the Default View Root

If you need to change the default view root (to avoid a conflict with an existing directory or network name), use one of the following procedures:

- On UNIX, edit the file *ccase-home-dir*/etc/clearcase to change this line, which defines the UNIX viewroot

  ```
  VIEWPATH="/view"
  ```

- On Windows, use the ClearCase program in Control Panel. The view root is specified as **View network name** on the **MVFS** tab.

The View Storage Directory

A dynamic view is implemented as a standard directory, whose root is called the view storage directory. The view storage directory contains these files and subdirectories, all of which are created and modified ClearCase commands and should never be modified in any other way:

.access_info	A file of view access event information that is periodically updated by the **view_server**.
.pid	A one-line text file that lists the process ID of the view's **view_server** process.
admin	A directory that contains administrative data related to the amount of disk space a view is using. Use **space –view** to list this data.
config_spec	A file that stores the view's current config spec, in the form displayed by **catcs**.
.compiled_spec	A modified version of config_spec, which includes accounting information.
.identity	On UNIX, a subdirectory whose zero-length files establish the view's owner and group memberships.
identity.sd	On Windows, a binary data file that contains the security descriptor for the view storage directory.
.s	A subdirectory that implements the view's private storage area.
db	A subdirectory containing the files that implement the view's embedded database.

.view	A file that lists the view's universal unique identifier (UUID) and other attributes

View-Private Storage

Subdirectory .s of the view storage directory is the root of a subtree that implements the view's private storage area. On UNIX platforms, .s can be either a local directory or a UNIX symbolic link to a directory on another UNIX computer or network-attached storage device.

The private storage area holds several kinds of objects:

- **View-Private objects.** A view-private object is an ordinary file system object, such as a file, directory, or UNIX link, contained in a view. View-private objects exist only within the view's private storage area and are not stored in a VOB.

  Note: In a dynamic view on UNIX, you cannot create any of the UNIX special file types (sockets, named pipes, character device files, or block device files).

- **Checked-out versions.** A checked-out version is an editable copy of a version of an element in a VOB. A checked-out version is very much like a view-private file, except that there is a corresponding object in the VOB: a placeholder version with the special CHECKEDOUT version label.

- **Unshared derived objects.** An unshared derived object (DO) is a data container created by **clearmake**, **omake**, or any program invoked by **clearaudit**. When the DO becomes shareable, a corresponding DO is created in the VOB.

  Note: Even after the DO becomes shared, a copy of it remains in the view's private storage area. DOs can consume a great deal of disk space and should be scrubbed when they are no longer needed. For more information, see the **winkin** and **view_scrubber** reference pages.

- **Nonshareable derived objects.** A nonshareable derived object is a data container created in a dynamic view by **clearmake**, **omake**, or any program invoked by **clearaudit**. No information about the DO is stored in the VOB database. When you use **winkin** or **view_scrubber –p** to convert a nonshareable DO to a shareable DO, the command promotes the DO's data container to the VOB, and removes the data container from view storage.

- **Configuration records.** The file view_db.crs_file in the .s subdirectory stores the configuration records of derived objects built in the view.

- **Stranded files.** The directory lost+found in the .s subdirectory contains stranded files. A file becomes stranded when there is no VOB pathname through which it can be accessed. For example:

- A VOB can become temporarily unavailable, for example, by being unmounted.
- A VOB can become permanently unavailable, for example, by being deleted.
- A VOB directory can become permanently unavailable when it is deleted with a **rmelem** command.

For more information about recovering stranded files, see the **recoverview** reference page.

View Database

The view database subdirectory, db, contains these files, all of which are created and modified by ClearCase commands and should never be modified in any other way:

view_db.dbd	A compiled database schema, used by embedded database routines for database access. The schema describes the structure of the view database.
view_db_schema_version	A schema version file, used by embedded database routines to verify that the compiled schema file is at the expected revision level.
view_db.d0*n* view_db.k01	Files in which the database contents are stored.
vista.*	Database control files and transaction logs.
view_db.crs_file	Stores the configuration records of nonshareable and unshared derived objects. This file resides in subdirectory .s of the view storage directory, allowing it to be remote. Compressed copies of the configuration records are cached in a view-private file, .cmake.state, located in the directory that was current when the build started. These copies speed configuration lookup during subsequent builds in the view.

The view database keeps track of the objects in its private storage area: view-private objects, checked-out versions, nonshareable derived objects, and unshared derived objects.

How a Dynamic View Selects Versions

Each time you access an element in a dynamic view, the view's **view_server** process evaluates the view's config spec to determine which version of the element to make available. The view server follows these steps when resolving element names to versions.

1. Application software (for example, an editor or compiler) references a pathname. The MVFS, which processes all pathnames within VOBs, passes the pathname to the appropriate **view_server** process.
2. The **view_server** attempts to locate a version of the element that matches the first rule in the config spec. If this fails, it proceeds to the next rule and, if necessary, to succeeding rules until it locates a matching version.
3. When it finds a matching version, the view server selects it and has the MVFS pass a handle to that version back to the application.

How a Dynamic View Manages Derived Objects

In addition to view-private files, a dynamic view's private storage area contains derived objects built in that view by **clearmake (**and, on Windows, **omake).** Nonshareable and unshared derived objects typically consume the most disk space in a view's private storage area. When a derived object is created, 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:

1. The configuration record is moved to the appropriate VOB. If the build script creates derived objects in several VOBs, each VOB database gets a copy of the same configuration record.
2. The data container is copied (not moved) to the VOB's DO 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 may find it worthwhile to remove redundant containers from views with the **view_scrubber** utility.

The Multiversion File System

Note: The MVFS is not supported by ClearCase LT. Snapshot and Web views do not use the MVFS.

The multiversion file system (MVFS) is a feature of ClearCase that supports dynamic views. Dynamic views use the MVFS to present a selected combination of local and remote files as if they were stored in the native file system. The selected files are

versions of VOB files and view-private files. To use the MVFS, you must activate a view and mount one or more VOBs. (VOBs are mounted as a file systems of type MVFS.)

The MVFS is installed as an extension to a host's native operating system. 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 computers, the MVFS is a file system driver. It is loaded by the Service Control Manager at system start up. When a user logs on to a Windows host where the MVFS is installed, the user's credentials are cached for use in determining access rights to objects under ClearCase control. The Credentials Manager service periodically checks the credential cache and deletes the credentials of users who have logged off since the last credentials check.

Supported File Types

The MVFS supports the following file types:

- Files
- Directories
- Symbolic links (on UNIX only)

You cannot create other file types, such as UNIX special files, within a dynamic view.

The MVFS and Audited Builds

When you build software in a dynamic view and a build tool (for example, a compiler) references a pathname, the MVFS passes the pathname to the appropriate **view_server** process. During build script execution, **clearmake** works with the MVFS to audit low-level system calls that reference ClearCase data, recording every instance when a file or directory is created, deleted, or opened for read access. Calls that involve the following objects are monitored:

- Versions of elements used as build input
- View-private files (for example, checked-out element versions) used as build input
- Files created within VOB directories during the build

Known Limitations of the MVFS on Windows

On Windows, 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.
- When an 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 that 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.
- On Windows, alternate, or short (8.3-compliant), names on are not supported in a VOB name space.

  Any alternate name may already exist in another version of the given directory, which could cause ClearCase to return the wrong file data, so the MVFS does not return an alternate name to applications that request one.

  You can run applications that require short names in the 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 using a VOB symbolic links to create 8.3-compliant names as needed

The MVFS and Case-Sensitivity

On Windows, the MVFS can be configured to perform case-sensitive or case-insensitive name lookups. In the default configuration, the MVFS performs case-insensitive name lookups, which is the behavior that Windows applications expect. When using dynamic views in mixed networks of Windows and UNIX computers, you may need to change case-sensitivity setting of the MVFS on Windows. See *Case-Sensitivity* on page 61 for details.

Running Executables in the MVFS

If you add an executable to source control and want to run it in a dynamic view, you must give it explicit execute permission. Use the **cleartool protect** command, as shown in this example:

cleartool protect –chmod +x *executable_filename*

MVFS Performance

The MVFS has several caches that it uses to provide improved performance. For many users, the default cache sizes provide the best balance between MVFS performance and memory requirements. You can change most of theses caches to tune the MVFS to better serve special needs. For more information, see *Examining and Adjusting MVFS Cache Size* on page 272.

Snapshot Views

A snapshot view uses a host's native file system to hold versions of file and directory elements that have been loaded from a VOB. After the snapshot view has been created, versions of file and directory elements must be copied, or loaded, into the snapshot view directory. The snapshot view directory is usually on the local host, but can be located on any host that the view server can access over the network, even if that host does not have ClearCase installed.

In addition to copies of versions, the snapshot view directory can contain view-private objects. It also contains some files that ClearCase creates to help manage the view.

Any snapshot view storage directory contains two subdirectories:

- **View database.** The `db` subdirectory contains the binary files managed by the embedded database management system. 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. Periodically, the ClearCase scheduler runs a job that generates this data.

Snapshot views cannot contain derived objects. You can run **clearmake** and **omake** in a snapshot view, but their build auditing and build avoidance features are disabled.

A snapshot view must have an associated **view_server** process, which can run on the local host or another host.

Note: Because a ClearCase LT client cannot run a **view_server** process, the **view_server** process for a ClearCase LT snapshot view runs on the ClearCase LT server.

For administrative purposes, it is important to know whether the view storage directory of a snapshot view is collocated (is a subdirectory of the snapshot view directory). For example, when you back up or move a snapshot view, you must include both the snapshot view directory and the view storage directory. For more information, see *Moving a View* on page 177.

Snapshot View Directory

The snapshot view directory uses the host's native file system (unlike a dynamic view, which uses the MVFS). In addition to copies of elements, the root of this directory (referred to as the snapshot view's root directory) contains the following files and subdirectories, which are created and modified ClearCase commands and must never be modified in any other way:

view.dat	A read-only text file used to identify the current directory as part of a view. (.view.dat on UNIX.)
view.stg or a generated directory name	A directory used to maintain the view. (.view.stg on UNIX.) For a collocated view storage directory only, the default name of this directory is named view.stg; for any other view storage directory, this directory has a generated name.

View Storage Directory

The view storage directory contains the following files and subdirectories, which are created and modified by ClearCase commands and must never be modified in any other way:

.access_info	A file of view access event information that is periodically updated by the view server.
.pid	A one-line text file that stores the process ID of the view's **view_server** process.
admin	A directory that contains administrative data related to the amount of disk space a view is using. Use **cleartool space –view** to list this data.
config_spec	A file that stores the view's current config spec, in the form displayed by **catcs**.
.compiled_spec	A modified version of config_spec, which includes accounting information.
.identity	On UNIX, a subdirectory whose zero-length files establish the view's owner and group memberships.
identity.sd	On Windows, a security descriptor created for views stored in a FAT or FAT32 file system. Views stored in NTFS file systems include security descriptors in the file system ACL and do not need this file.
.s	A subdirectory that implements the view private storage area.
db	A subdirectory containing the files that implement the view's embedded database.

.view	A file that holds the view's universal unique identifier (UUID) and other attributes
view_db.state	A file that records the current state of the view.

View Database

The view database subdirectory, db, contains these files, all of which are created and modified ClearCase commands and must never be modified in any other way:

view_db.dbd	A compiled database schema, used by embedded database routines for database access. The schema describes the structure of the view database.
view_db_schema_version	A schema version file, used by embedded database routines to verify that the compiled schema file is at the expected revision level.
view_db.d0*n* view_db.k01	Files in which the database's contents are stored.
vista.*	Database control files and transaction logs.

For a snapshot view, the view database keeps track of the loaded VOB objects and checked-out versions in the view.

How a Snapshot View Selects Versions

A snapshot view's config spec and load rules determine which versions of elements are loaded into the snapshot view directory by the **cleartool update** command. Each time you update the view, the **view_server** evaluates the view's config spec, selects a version of each element defined in the load rules, and copies a new version of any element loaded in the snapshot view directory if the element does not match the selected version. It uses the following method to resolve element names to versions.

1. The view server attempts to locate a version of the element that matches the first rule in the config spec. If this fails, it proceeds to the next rule and, if necessary, to succeeding rules until it locates a matching version.
2. When it finds a matching version, the view server selects it, updates the view's database, and passes a handle to that version to the CCFS server, which copies it into the snapshot view directory.

Creating a View

Views are typically created by users. The process is simple, can be automated by GUIs on UNIX and Windows, and does not usually require administrative intervention. In some cases, an administrator may be responsible for creating shared views or for establishing policies that govern view creation. For more information about creating views, see the **mkview** reference page.

Remote View Storage

Like VOB storage, view storage may be located on a remote host (one that is not running the view's associated **view_server** process) by using any of the following techniques:

- You can create a snapshot view that does not have collocated view storage (that has the view storage directory on the view server host and the snapshot view directory on another host).
- You can locate some or all of a view's storage on a certified network-attached storage device.
- On a UNIX host, a view's private storage area (but not its database) can be located remotely and accessed through a UNIX symbolic link. This arrangement resembles the remote VOB storage pool facility, discussed in *Creating Remote Storage Pools on UNIX Hosts* on page 113, but the facility for views is less elaborate.

  Note: A Windows computer cannot use a UNIX view that has symbolically linked view-private storage (**mkview –ln**).

Creating a View on a NAS Device

If you want to create views on a NAS device, we recommend that you first create a server storage location for this purpose (see *Creating Server Storage Locations on a NAS Device* on page 96). The following command creates a dynamic view in a server storage location named **ccnasviewstg**.

cleartool mkview –tag *viewtag* **–stgloc ccnasviewstg**

You can also use the **mkview** command with **–host**, **–hpath**, and **–gpath** options. The following example creates a dynamic view on a NAS device. The view is served by a **view_server** process running on the ClearCase host **ccviewsvr-ux** (a UNIX computer) and has its storage on a NAS device mounted by **ccviewsvr-ux** at /net/nasdevice.

**cleartool mkview –tag nasview –host ccviewsvr-ux **
**–gpath /net/nasdevice/viewstg/nasview.vws –hpath **

**/net/nasdevice/viewstg/nasview.vws **
/net/nasdevice/viewstg/nasview.vws

Moving a View

This section presents procedures for moving a view to another location on the same host or to another host. You do not need to back up the view before moving it because these procedures do not destroy the old view storage directory until the move is complete and the view is accessible in its new location.

Caution: When you move a view storage directory, use copy or backup software that does not change file and directory ownership and access control information. This is especially important on Windows hosts that use an NTFS file system. For more information, see *Preserving NTFS ACLs When Copying a VOB or View Storage Directory* on page 257. In addition, if you are moving a snapshot view, the utility you use to copy the snapshot view directory must preserve the modification times and ownership of loaded files and directories. Otherwise, these files and directories are reported as hijacked after the move is complete.

Determining Whether a Dynamic View on UNIX Has Remote Private Storage

The procedures in this section do not move any remote pools that may be associated with views on UNIX hosts. Before you move a dynamic view on a UNIX host, determine whether the view has symbolically linked private storage. (Snapshot views do not support symbolically linked view-private storage.) Use the UNIX **ls** command to list the view's .s directory. For example:

ls –ld /net/mars/viewstg/v5_integration.vws/.s

```
...   .s -> /public/view_aux/v5_integration
```

The symbolic link indicates that the private storage area is remote. If you intend to use the same storage for the moved view, it must be accessible from the new view host.

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 location.

Moving a View to a Host with the Same Architecture or to a NAS Device

Use this procedure to move a view to another partition on the same host, to another host with the same architecture, or to a NAS device.

1 **Log on to the view host**. Log on as the view owner or privileged user.

2 **Deactivate the view.** To keep the view inactive while it is being moved, use the **cleartool chview** *view_tag* command with the **–readonly** option to prevent users from updating the view database.

cleartool chview –readonly r5_integration
```
Properties: readonly
```

3 **Stop the view's server process**. Use the **cleartool endview** *view_tag* command with the **–server** option:

cleartool endview –server r5_integration

4 **Ensure that the view cannot be reactivated**. Remove its tag and then unregister it. Use the ClearCase Administration Console or the following commands:

cleartool rmtag –view –all v5_integration

cleartool unregister –view /net/mars/viewstg/v5_integration.vws

Note: This step and Step 6 are not necessary for a snapshot view whose view storage directory will not be moved.

5 **Copy the view**. If you are moving a dynamic view, copy the entire view storage directory (but not any symbolically linked private storage) to the new location. For dynamic views, the procedures are similar on UNIX and Windows:

- On UNIX, you can use a utility like **tar** utility to copy a view storage directory. The following example copies the view storage directory **v5_integration.vws** to a host named **venus**:

  cd /viewstg

  tar –cf – v5_integration.vws | rsh venus 'cd /viewstg ; tar –xBpf –'

  Note: If you are not logged on as **root**, the **–p** option to the **tar** command, which preserves file and directory protections critical to the view, may be ignored. The **–B** option, which supports copying over the network, may not be supported on some UNIX platforms. See the reference pages for your operating system.

- On Windows, you must use a copy utility that preserves file and directory ownership and access control information (see *Preserving NTFS ACLs When Copying a VOB or View Storage Directory* on page 257) and you must run the copy utility on the host to which the view is being moved. The following example uses the ClearCase **ccopy** utility to copy the view storage directory **v5_bugfix.vws** from a host named **mars**:

  `C:\ClearCaseStorage\views>` **net use E: \\mars\vws**

  `C:\ClearCaseStorage\views>` **ccopy E:\v5_bugfix.vws v5_bugfix.vws**

- On UNIX or Windows, if you are moving a snapshot view, you can move the snapshot view directory, the view storage directory, or both. If you are moving a snapshot view on Windows, see *Special Procedures for Moving Snapshot Views on Windows* on page 180.

6 **Register the view at its new location and create a new view tag.** Use the ClearCase Administration Console or the following commands:

cleartool register –view /net/venus/viewstg/v5_integration.vws

**cleartool mktag –view –tag v5_integration **
/net/venus/viewstg/v5_integration.vws

Note: If you are moving the view to a NAS device, you must use the **–host**, **–hpath**, and **–gpath** options to the **cleartool register** and **mktag** commands. This example, assumes a NAS device mounted at /net/nasdev on ClearCase host **venus**:

**cleartool register –view –host venus –hpath **
**/net/nasdev/viewstg/v5_integration.vws **
**–gpath /net/nasdev/viewstg/v5_integration.vws **
/net/nasdev/viewstg/v5_integration.vws

**cleartool mktag –view –view –tag v5_integration –host ccviewsvr1 **
**–hpath /net/nasdev/viewstg/v5_integration –gpath **
/net/nasdev/viewstg/v5_integration /net/nasdev/viewstg/v5_integration

7 **Reactivate and test the view.** While you are still logged on as the view owner, access the view and verify the following:

- You can check out versions and, in a dynamic view, create view-private files.
- All view-private objects were moved.
- No files appear as hijacked (snapshot views only).

8 **Update VOB references to this view.** The view's old location is still recorded by all VOBs that the view has accessed. To update this information, go to the view check out any element from each VOB that holds such a reference. (You can cancel the checkout immediately if you want).

9 **Delete the old view storage directory.** After you verify that the moved view is accessible and working correctly, delete the old view storage directory using any operating system utility.

10 **Create additional tags as needed**. If the view has tags in other network regions, replace them with tags that reference the new location.

Special Procedures for Moving Snapshot Views on Windows

When you move a snapshot view on Windows, you must preserve both the ACLs of the view storage directory and the modification times of loaded elements in the snapshot view directory.

You can use the Windows **xcopy** utility to copy the snapshot view directory, but you must use the ClearCase **ccopy** utility (described in *Preserving NTFS ACLs When Copying a VOB or View Storage Directory* on page 257) to copy the view storage directory. In the following example, the view storage directory view.stg is a subdirectory of the snapshot view directory v5_integration.vws.

```
C:\ClearCaseStorage\views> net use E: \\boron\viewstg

C:\ClearCaseStorage\views> xcopy /e/v/i/r/h/k v5_integration.vws E:\v5_integration.vws

C:\ClearCaseStorage\views> rmdir E:\v5_integration.vws\view.stg

C:\ClearCaseStorage\views> ccopy v5_integration.vws\view.stg E:\v5_integration.vws
```

xcopy /k preserves the modification time of loaded elements but allows the parent directory's ACL to be inherited by the copied snapshot view storage directory, which can cause problems with view access. **ccopy** preserves NTFS ACLs as required by ClearCase, but does not preserve file modification times.

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.

Moving a view to a UNIX host with a different architecture requires all the steps needed to move a view to another host of the same architecture and additional steps to convert the view database files to the binary format supported by the new host.

To move the view:

1. **Log on the view host**. Log on as the view owner or a privileged user.
2. **Deactivate the view.** To keep the view inactive while it is being moved, use the **cleartool chview** *view_tag* command with the **–readonly** option to prevent users from updating the view database.

   cleartool chview –readonly r5_integration
   ```
   Properties: readonly
   ```
3. **Stop the view's server process**. Use the **cleartool endview** *view_tag* command with the **–server** option:

   cleartool endview –server r5_integration

4 **Ensure that the view cannot be reactivated**. Remove its tag and then unregister it. Use the ClearCase Administration Console or the following commands:

cleartool rmtag –view –all v5_integration

cleartool unregister –view /net/mars/viewstg/v5_integration.vws

Note: This step and Step 7 are not necessary for a snapshot view whose view storage directory will not be moved.

5 **Dump the view's database.**

cleartool reformatview –dump /net/mars/viewstg/v5_integration.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.

6 **Copy the view**. If you are moving a dynamic view, copy the entire view storage directory (but not any symbolically linked private storage) to the new location. If you are moving a snapshot view, you can move the snapshot view directory, the view storage directory, or both. The following example uses the UNIX **tar** utility to move a view storage directory to a host named **venus**:

cd /viewstg

tar –cf – v5_integration.vws | rsh venus 'cd /viewstg ; tar –xBpf –'

Note: If you are not logged on as **root**, the **–p** option to the **tar** command, which preserves file and directory protections critical to the view, may be ignored. The **–B** option, which supports copying over the network, may not be supported on some UNIX platforms. See the reference pages for your operating system.

7 **Register the view at its new location and create a new view tag.** Use the ClearCase Administration Console or the following commands:

cleartool register –view /net/venus/viewstg/v5_integration.vws

**cleartool mktag –view –tag v5_integration **
/net/venus/viewstg/v5_integration.vws

8 **Recreate the view database from the dump files.**

cleartool reformatview –load –tag sv5_integration

9 **Reactivate and test the view.** While you are logged on as the view owner, access the view and verify the following:

- You can check out versions and, in a dynamic view, create view-private files.
- All view-private objects were moved.
- No files appear as hijacked (snapshot views only).

10 **Update VOB references to this view.** The view's old location is still recorded by all VOBs that the view has accessed. To update this information, go to the view check out any element from each VOB that holds such a reference. (You can cancel the checkout immediately if you want).

11 **Delete the old view storage directory.** After you verify that the moved view is accessible and working correctly, delete the old view storage directory using a file system utility like **rmdir** or Windows Explorer. On the new view host, delete the backup view database (typically named db.dumped).

12 **Create additional tags as needed**. If the view has tags in other network regions, replace them with tags that reference the new location.

Replacing a View Server Host for a NAS Device

When you locate view storage on a NAS device, you can easily designate a different ClearCase host of the same architecture to run the **view_server** process that manages access to the view without actually moving the view storage.

The following procedure replaces the UNIX view server host for dynamic view **V4.1_Int**. It stops the view, removes the old view tag and object, and creates a new object and tag that specify the replacement host and the existing storage.

1 **Log on to the view's server host**. Log on as the view's owner.

2 **Deactivate the view.** Use the **cleartool endview** command to stop the view and terminate the view's **view_server** process:

cleartool endview –server V4.1_Int

3 Delete the existing view tag. You can use the ClearCase Administration Console or the following command.

cleartool rmtag –view V4.1_Int

4 Create a new view object and tag specifying a new view server host and the existing storage. You can use the ClearCase Administration Console or the following commands.

**cleartool register –view –replace –host ccviewsvr1 **
**–hpath /net/nasdevice/viewstg/v4.1_int.vws **
–gpath /net/nasdevice/viewstg/v4.1_int.vws /net/nasdevice/viewstg/v4.1_int.vws

**cleartool mktag –view –tag V4.1_Int –host ccviewsvr1 **
**–gpath /net/nasdevice/viewstg/v4.1_int.vws **
–hpath /net/nasdevice/viewstg/v4.1_int.vws /net/nasdevice/viewstg/v4.1_int.vws

5 **Reactivate the view** on the replacement view server host.

cleartool startview V4.1_Int

Displaying the Properties of a View

You can use the ClearCase Administration Console or the **cleartool lsview** command to display the properties of a view. Properties include information about the view's creation date, last modification date, and owner. If you use the ClearCase Administration Console or the **–long** and **–full** options to **lsview**, you can get additional information, including:

- The view's text mode
- The view's protections
- Whether or not the view's ownership and permissions are consistent
- When and by whom the config spec was last updated
- Whether the view is a dynamic view or a snapshot view
- Whether the view is read-only or writable
- For a dynamic view, whether it creates shareable or nonshareable derived objects
- When and by whom view-private data was last accessed
- When and by whom a view-private derived object was last updated
- For a dynamic view, when and by whom a derived object was last created, promoted, or winked-in

This information can help you decide whether a view is still in use, or contains artifacts that may still be needed.

Note: Changes to any of these properties may not be reflected immediately by the ClearCase Administration Console or **cleartool lsview**, although in most cases view property update latency is short (less than 30 seconds).

Taking a View Out of Service

To take a view out of service temporarily, stop the view's **view_server** process and remove the view's tag. Use **cleartool endview** to stop the view's **view_server** process.

cleartool endview –server alh_main

After the view's server process has stopped, remove the view tag by using the ClearCase Administration Console or the **cleartool rmtag** command.

Restoring the View to Service

To restore a view to service create a tag for the view by using the ClearCase Administration Console or the **cleartool mktag** command. The view's server process starts when a client references the view tag.

Removing a View

To permanently remove a view (including its entries in the ClearCase registry and all references to the view held by any VOBs), 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 **cleartool rmview** command.

If you have tried to remove a view by simply removing its storage directory, use the following procedure to remove references to the view still held by VOBs:

1. Run the **cleartool describe** 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 **lsview –long** to find the view UUID.
2. If the view tag still exists, remove it 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 command:

   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. Remove references to this view from each VOB that holds them. 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.

Backing Up Critical ClearCase Data

10

Ensuring frequent, reliable backups of essential data is a critical task for any ClearCase administrator. Like all databases, VOBs and views have special backup and restore requirements. Restore procedures are described in Chapter 11. This chapter provides backup procedures for VOBs, views, and the ClearCase registry.

Because data in multiple repositories is often related, especially if you use UCM, this chapter also provides suggestions for implementing a backup routine that recognizes and tries to preserve those relationships.

Requirements for VOB and View Backup Tools

VOB and view data is stored in ordinary files and directories. It can be backed up with any tool that is appropriate for file system backups as long as the tool captures all the necessary data and you follow the procedures in this chapter. When selecting a VOB or view backup tool, consider these requirements and recommendations:

- **The backup tool must preserve NTFS ACLs.** On Windows platforms where VOB or view storage is located on an NTFS file system, the NTFS ACLs on the VOB or view storage directory are important. Unless your VOB and view storage is on a FAT filesystem (not recommended), any Windows backup tool you use must be able to back up and restore NTFS ACLs.
- **The backup tool should back up files even if they are open for writing.** On Windows and UNIX, VOB server processes keep VOB database files open for writing even when the VOB has been locked for backup. Unless you lock the VOB and stop ClearCase on the VOB host during VOB backup, the program you use to back up VOBs must be able to back up files that are open for writing. Many common Windows backup tools, as well as file system copy utilities such as **copy** and **xcopy**, do not have this capability and will skip files that are open for writing, which renders the backup useless.
- **The backup tool should preserve file access times.** On some UNIX platforms, the **tar** utility, which is often used for backups, resets file access times. This can disrupt DO and cleartext storage pool scrubbing patterns (see *Scrubbing VOB Storage Pools*

on page 226) and may prevent these pools from ever being scrubbed. The UNIX **cpio** utility may be a better choice for VOB backup.

Note: Any backup procedure in this chapter can be used for a VOB or view that has its database on a network-attached storage (NAS) device. If you locate a VOB storage directory on a NAS device that has been certified for use with ClearCase, you may be able to take advantage of device-specific backup utilities that can dramatically reduce VOB lock time. For information about certified NAS devices and their backup utilities, see the *Platforms Guide* in Help.

VOB Backup Strategies

When planning a VOB backup strategy, you must consider several things:

- The backup tool must capture all of the necessary data.
- The VOB must be locked during the backup. A locked VOB can defer or disrupt many development activities. If backups cannot be scheduled for off hours, your backup strategy may need to focus on reducing the duration of each VOB lock to a practical minimum.
- A VOB may be part of a group of VOBs that are connected by hyperlinks and should be backed up and restored together.
- If UCM and Rational ClearQuest are in use, a PVOB and a ClearQuest database are likely to hold references to each other. These references may become invalid when the PVOB or the ClearQuest database is restored.

Note: When choosing backup tools and strategies for VOBs, it is helpful to understand the file system layout of VOB storage directories, as described in *The VOB Storage Directory* on page 105.

Choosing Between Standard and Snapshot Backups

Your backup strategy may be based on either or both of the following:

- A standard backup strategy that uses the tools you normally use to back up the file systems of VOB server hosts
- A snapshot backup strategy that uses the **vob_snapshot** utility to reduce VOB lock time

The standard VOB backup procedure requires you to lock the VOB, back up the entire VOB storage directory, and then unlock the VOB. The VOB must remain locked for the duration of the backup. If you use a backup utility that cannot capture files open for writing, ClearCase must be stopped as well.

We recommend that you use the standard backup procedure on ordinary or UCM component VOBs whenever users can accept the duration of required locks. Locking a VOB prevents checkins, checkouts, and other operations that affect VOB data and metadata. These include UCM deliver operations and any UCM rebase operations that require merging. You can still work with checked-out or (in a snapshot or Web view) hijacked versions while a VOB is locked. **clearmake** and **omake** are designed to cope with locked VOBs. They can be configured to "sleep" when they are unable to open an object in a locked VOB, and then retry (see the **clearmake** reference page). When you use a standard backup procedure:

- You can use a backup tool that cannot back up files that are open for writing, but only if you first stop ClearCase on the VOB host. The time required to stop and start ClearCase may not add significantly to the time it takes to back up the VOB. While ClearCase is stopped, no VOBs or views on the host are accessible.
- No extra disk space is required if you are backing up to a different physical device.
- The VOB restore procedure is simplified. Because the VOB database and storage pools are backed up as a unit, data loss at restore time is unlikely.

Snapshot backups, which copy only the VOB database, can significantly reduce the duration of a VOB lock, especially for larger VOBs. The **vob_snapshot** utility helps automate the most critical parts of a snapshot backup and can easily be run as a ClearCase scheduled task. When you use **vob_snapshot**:

- You must use a backup tool that can back up files that are open for writing. After **vob_snapshot** is finished and the VOB has been unlocked, you must back up the remainder of the VOB storage directory. If the VOB is in use while this backup is in progress, it is likely that one or more files in the VOB storage directory will be open for writing. If they are not backed up, data may be lost when the VOB is restored.
- More disk space is required. A copy of the VOB database must be written to disk, ideally on the local host. 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 improve the chances 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. Because the VOB storage pools and the VOB database are backed up at different times, they 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 by the **checkvob** utility.

- 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. *Minimizing Data Loss with checkvob -force -fix* on page 237 explains these risks in more detail.

Note: PVOBs that do not contain any components have no data in their pools and can be backed up by **vob_snapshot** without risk of data loss. We recommend that you use **vob_snapshot** to back up PVOBs.

Backing Up VOBs with vob_snapshot

To enable database snapshots, run **vob_snapshot_setup**. This command causes the ClearCase scheduler to run **vob_snapshot** periodically (daily, by default) on the VOB database. The **vob_snapshot_setup** and **vob_snapshot** reference pages explain these operations. Whenever you enable a VOB for snapshot backups, you must also take additional steps to ensure that the backup is complete and can be recovered with little or no data loss:

- Back up the entire VOB storage directory, including any remote storage pools (see *If the VOB Has Remote Storage Pools* on page 190), as soon as possible after each **vob_snapshot** completes. You do not have to lock the VOB during this backup. If you complete this backup within 30 minutes of the time **vob_snapshot** completes, you are less likely to encounter data loss on restore. You can reduce backup time by excluding the VOB database (`db` directory) itself from the backup. (It has already been backed up by **vob_snapshot**.)
- Back up the snapshot itself. This is especially important if the snapshot is copied to a location on the same physical device as the VOB itself. If that device fails and the snapshot has not been backed up to other media or copied to a different device, the backup is useless.

Deferred Deletion of Source Containers

Deferred deletion is enabled by the **vob_snapshot_setup** program, and is intended to optimize chances that a pool backup that follows a **vob_snapshot** within 30 minutes will include all source containers to which the database snapshot holds a reference. When a container is replaced by new version data (for example, during a checkin), the new container is created and the old one is deleted. If deferred deletion is enabled, the old container is added to a list of pending deletions and removed after 30 minutes.

Deferred deletion increases disk space requirements in a active VOB. On UNIX platforms, you can execute the **kill –HUP** command on the **vob_server** process to send deferred deletion statistics to its log file.

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

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

Backing Up VOBs with Standard Backup Procedures

If you do not use the **vob_snapshot** utility, you must follow the procedures in this section to ensure the integrity of VOB backups. Any VOB backup procedure requires these steps:

1. Lock the VOB for all users.
2. Back up the entire VOB storage directory, including any remote pools. (If necessary you may exclude certain subdirectories of the VOB storage directory from the backup. See *What to Back Up if You Cannot Back Up Everything* on page 191 for details).
3. Unlock the VOB.

Determining the Location of the VOB Storage Directory

To determine the location of a VOB storage directory, use the ClearCase Administration Console or the **cleartool lsvob** command.

cleartool lsvob –long /vobs/sources

```
Tag: /vobs/sources
  Global path: /net/neptune/vobstg/sources.vbs
 .
 .
 .
VOB on host: neptune
VOB server access path: /vobstg/sources.vbs
```

If your backup program runs locally, it probably accesses the VOB by using the `VOB server access path`. If your backup program runs over the network, it probably uses the `Global path`. Specify the appropriate pathname for your backup program.

Locking and Unlocking the VOB

A VOB backup is valid only if the VOB has been locked for all users before the backup begins.

Caution: Regardless of your chosen backup tools or strategy, you must lock the VOB for all users. A VOB locked with the **–nusers** option does not perform a database checkpoint. Without this checkpoint, the backup is invalid.

You must be the VOB owner or privileged user to lock or unlock a VOB. Several GUIs support VOB locking, as do the **cleartool lock** and **unlock** commands.

- You can lock or unlock a VOB on any UNIX or Windows host from the ClearCase Administration Console:
  - **a** 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.
  - **b** Click **Action > Properties**. In the **Properties of VOB** dialog box, click the **Lock** tab.
- On a Windows host, you can lock or unlock a VOB in Windows Explorer. Click **ClearCase > Properties of VOB** and then click the **Lock** tab.
- On the command line, use the **cleartool lock** and **unlock** commands.

If the VOB Has Remote Storage Pools

If a VOB on a UNIX host has remote storage pools, you must back them up too, regardless of the backup strategy (standard or **vob_snapshot**) you employ.

Use the **lspool** command in any view to determine which storage pools are remote:

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 the **lspool –long** command to list the pool storage global pathname of every pool. 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"
      .
      .
```

Note: Many UNIX backup utilities, such as **tar** and **cpio**, include options to follow symbolic links. If the remote host is accessible at backup time, these utilities can copy both the symbolic link and its contents. They may be useful when backing up VOBs that have remote pools.

What to Back Up if You Cannot Back Up Everything

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 7 to determine the relative importance of the various directories.

Table 7 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. (Not used on ClearCase LT)
Cleartext storage pools	Optional
Administrative directory	Optional

DO Pool Backup

Note: ClearCase LT does not support derived objects, so the DO pools of ClearCase LT VOBs are always empty and do not need to be backed up.

Derived object storage pools do not exist unless you use the MVFS and programs such as **clearmake**, **omake**, or **clearaudit**. 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, DO storage pools may contain many objects that are frequently reused. Loss of a DO storage pool at this stage may significantly increase the time required for the next complete system build, but not for subsequent builds, which can reuse the newly created DOs.
- Mature projects may refer to DOs that are hard to rebuild (they may require compilers that are no longer available, or views that have been deleted). These DOs are valuable, and should be backed up regularly or, better yet, versioned as elements.

If you do not back up DO pools, we recommend backing up at least the pool root directory (d\ddft, for example) and pool_id file. This prevents post-restore pool root check failures from **checkvob**.

Caution: 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) leaves 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. Use **cleartool rmdo** and/or **checkvob**.

Cleartext Pool Backup

Backing up cleartext storage pools is not important, because type managers recreate cleartext data containers as necessary.

If you do not back up cleartext pools, we recommend backing up at least the pool root directory (c\cdft, for example) and pool_id file. This prevents post-restore pool root check failures from **checkvob**.

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 this directory. By default, the scheduler stores data for the previous 30 days. This historical data cannot be recreated. If the data is important to you, back up the admin directory.

Incremental Backups Are Not Recommended

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.) This mechanism defeats the purpose of most incremental backup schemes, which typically back up only files that have changed since a previous backup. A series of incremental backups of a VOB would capture every container and then would restore containers that had been purposely deleted, filling the VOB storage directory with useless data.

For example, suppose that one or more new versions of a particular element are created each day for a week. Each day's incremental backup saves a different source container for that element. If you restore from such a backup at the end of the week, you restore all those containers to the storage pool, even though only one of them (the most recent) corresponds to the current state of the VOB database. A subsequent run of **checkvob** would report many unreferenced data containers, which you would then have to delete.

Given this situation, we recommend that you avoid incremental backups of VOB storage or, if this is not possible, perform only a few incremental backups before the next full backup.

Backing Up a View

The contents of views, unlike that of VOBs, can usually be reconstructed easily. With the exception of changes to checked-out versions and other view-private files, the contents of any view can be recovered by recreating the view. Regular backups of views can still be important, especially if users are not in the habit of checking in their work regularly. Backing up a view is similar to backing up a VOB, but simpler:

- Views do not have multiple storage pools. A dynamic view has a single private storage area, its .s subdirectory. Unless the view storage is on a UNIX host, where it may be implemented as a symbolic link to a remote host, or on a NAS device, this directory must be local to the host where the view server runs. Element versions loaded into a snapshot or Web view reside in a single directory and its subdirectories (there is no .s directory).
- We do not recommend making partial backups of a view storage directory or snapshot view directory. All the data in these directories is important, especially modified checked-out files and other view-private files, which are not recorded in the VOB.

Caution: Any utility you use to back up a view storage directory must be able to back up files that are open for writing. View database files are typically held open for write while ClearCase is running, even if the view is inactive or read-only. A backup that skips these files has limited value if any. Unless your backup software is known to capture files open for write access (something many Windows utilities cannot do), you must stop ClearCase on the view host before performing a backup.

A snapshot or Web view may have a view storage directory that is in a different location, perhaps on a separate host, from the directory where the downloaded files are stored. (Web views almost always have this configuration, as do snapshot views on ClearCase LT.) You must back up both directories.

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 r5_integration

```
Tag: r5_integration
    Global path: /net/mars/viewstg/r5_integration.vws
 ...
 ...
View on host: mars
View server access path: /viewstg/r5_integration.vws
 ...
```

2 **Ensure integrity and consistency of the backup**. To keep the view inactive while it is being backed up, use the **chview** command to prevent users from updating the view database.

cleartool chview –readonly r5_integration

```
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 **If the view is on a UNIX host, determine whether it has remote storage.** You can use the UNIX **ls** command:

cd /viewstg/r5_integration.vws

ls –ld .s

```
...  .s -> /net/ccsvr04/viewstore/r5_integration.stg
```

A symbolic link indicates remote private storage area.

4 **Enter the backup commands.** For a dynamic view or for a snapshot view whose view storage directory is a subdirectory of the snapshot view directory, back up the entire view storage directory. Use the local path or the network path listed by **lsview** in Step 1.

If you are backing up a snapshot or Web view and its view storage directory is not a subdirectory of the view directory, back up both the view directory and the view storage directory.

Note: When you back up and restore a snapshot view, use a utility that preserves the modification times and ownership of all files and directories in the view. If you do not, loaded files become hijacked.

5 If you made the view read-only in Step 2, make it writable.

cleartool chview –readwrite r5_integration
```
Properties: readwrite
```

6 If you renamed the view storage directory in Step 2, rename it to its original name.

Backing Up ClearCase Registry Data

ClearCase registry data is kept in a group of files in the rgy directory on the registry server host. These are ordinary files that can, and should, be backed up regularly.

- On UNIX hosts, the registry directory is /var/adm/rational/clearcase/rgy.
- On Windows hosts, the registry directory is *ccase-home-dir*\var\rgy.

On both Windows and UNIX, all files in the registry directory should be backed up and restored together. In addition to backing up the rgy directory, we recommend that you back up the registry client list, which is not located in the rgy directory.

- On UNIX hosts, the client list is stored in /var/adm/rational/clearcase/client_list.db.
- On Windows hosts, the client list is stored in *ccase-home-dir*\var\client_list.db.

When you back up registry data, we recommend using a backup utility that can back up files that are open for writing.

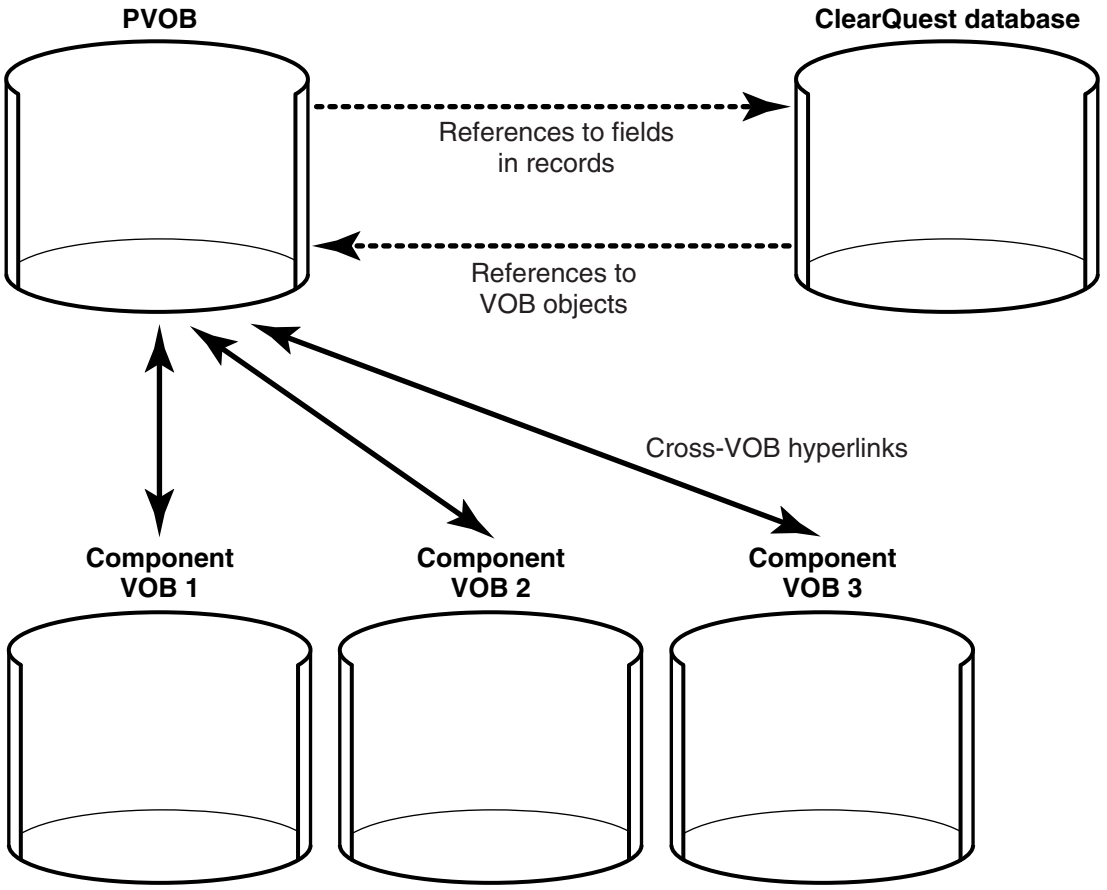
Backing Up Related Databases Together

Data in multiple repositories is often related, especially when UCM is in use. These are typical relationships:

- UCM PVOBs and the component VOBs they reference are connected by hyperlinks.
- Administrative VOB hierarchies (see Chapter 7) are connected by hyperlinks.
- In an environment that uses UCM and Rational ClearQuest, PVOBs store references to records in ClearQuest databases, and ClearQuest databases store references to objects in PVOBs.

Figure 6 illustrates these relationships in a simple configuration.

Figure 6 Database Relationships



UCM operations such as joining a project, making an activity, delivering or rebasing a stream are likely to change the data in multiple related databases and, in some cases, the relationships themselves. Even if UCM is not in use, changes in an administrative VOB hierarchy (for example, creating a global type and a local copy) can affect multiple VOBs with a single operation. The longer the interval between the time that the first member group of related databases is backed up and the time that the last member is backed up, the greater the chance that these relationships will be skewed when any of the backups are restored.

An ideal backup strategy for a configuration like the one in Figure 6 would consist of these steps:

1 Lock all the VOBs and the ClearQuest database.

2 Back up all the VOBs and the ClearQuest database.

3 Unlock all the VOBs and the ClearQuest database.

In all but the simplest configurations (those with a few small VOBs and a small ClearQuest database), it is unlikely that this procedure could be completed in a reasonable length of time. Even if all of the related databases could be backed up in a single event, it is unlikely that they would be restored together. It is far more likely that one or two databases from the set would be restored into an environment where the others were intact, which would again result in skew between the restored and intact databases. Many types of skew between databases can be reconciled by using tools and procedures described in *Restoring One or More Members of a Set of Related Databases* on page 213.

A more practical backup strategy would consist of the following steps:

1 Back up the component VOBs. If possible, use the **vob_snapshot** tool, which minimizes backup time for each VOB.

2 Back up the ClearQuest database.

3 Back up the PVOB using **vob_snapshot**. PVOBs that do not also contain components have no data in their pools and can be backed up by **vob_snapshot** without risk of data loss. If your configuration includes multiple PVOBs that have a common administrative VOB, back up all the PVOBs and the administrative VOB as a unit. (Lock all the PVOBs and their administrative VOB, back them up, then unlock them.)

This strategy maximizes the chances that, on restore, the PVOB is newer than the ClearQuest database and component VOBs, which in turn simplifies the reconciliation of database skew.

Whatever your backup strategy, we recommend that you do the following:

- Back up all members of a related group of databases in the shortest amount of time possible.
- Back up the entire group frequently. On restore, a backup performed yesterday will generate far less skew than a backup performed last week.

Determining Database Relationships

In an environment where UCM is not in use, database relationships are determined primarily by the structure of administrative VOB hierarchies. Use the ClearCase Administration Console or **cleartool describe** to display the **AdminVOB** hyperlinks that point to or from a VOB. (As described in *Administrative VOB Hierarchies and Global Types* on page 121, a VOB may have exactly one **AdminVOB** hyperlink pointing to its

superior in an administrative VOB hierarchy and may have zero or more **AdminVOB** hyperlinks pointing to it from other VOBs in the hierarchy).

In a UCM environment, database relationships are determined by the organization of the project. All the modifiable components in a project, as well as its PVOB and any ClearQuest databases it uses are related and should be backed up together. (Nonmodifiable components should be backed up also, but because the references held to these components by the PVOB are likely to be valid for longer intervals, you may not need to back up these components on the same schedule as the others.)

You can use the Project Explorer or **cleartool lsproject** to display the components in a project.

Restoring Critical ClearCase Data

11

If you have implemented a sound backup strategy for ClearCase data, it should be a simple matter to restore that data if necessary. Like all databases, ClearCase repositories have special backup and restore requirements. The backup requirements are described in Chapter 10. This chapter provides procedures for restoring VOBs, views, and the ClearCase registry.

Because data in multiple repositories is often related, especially if you use UCM, this chapter also describes procedures for detecting and correcting inconsistencies (or skew) between existing and restored members of a group of related repositories.

Before You Restore

Whenever you restore a VOB, a view, or (if you use Rational ClearQuest in a UCM environment) a ClearQuest database, you are introducing an artifact from an earlier time into an environment in which an artifact's history is often critical. Before you restore any of these databases, consider the implications of the restore on other databases in your community.

- After you restore an ordinary VOB or UCM component VOB from a **vob_snapshot** backup, you must check and fix any discrepancies between the VOB's database and its pools. *Restoring a VOB from Backup with vob_restore* on page 200 and *Restoring a VOB from Backup Without vob_restore* on page 205 provide details on this procedure.

  If you need to restore individual elements that have been inadvertently removed from a VOB with the **rmelem** command, follow the procedure described in *Restoring an Individual Element from Backup* on page 207.

  Note: When restoring VOBs, it is helpful to understand the file system layout of VOB storage directories, as described in *The VOB Storage Directory* on page 105.

- After you restore an ordinary VOB or UCM component VOB, you must resynchronize the VOB with any views that hold references to it. See *Synchronizing VOBs and Views After You Restore* on page 210 for the procedures.

- After you restore a view, you must resynchronize the view with any VOBs to which it holds references. See *Synchronizing VOBs and Views After You Restore* on page 210 for the procedures.
- If you use UCM and Rational ClearQuest, a number of relationships between VOB, PVOB, and ClearQuest databases must be reconciled after any of these databases is restored. These relationships, along with procedures for detecting and correcting discrepancies in them, are described in *Restoring One or More Members of a Set of Related Databases* on page 213.

Restoring a VOB from Backup with vob_restore

The **vob_restore** utility automates many of the routine steps involved in restoring a VOB from backup. This utility is especially useful in conjunction with the **vob_snapshot** backup tool, and is also useful with a standard backup (see *VOB Backup Strategies* on page 186).

vob_restore handles the following tasks:

- Deactivating the VOB and stopping its server processes.
- Copying backup data to the VOB's currently registered storage directory or another directory you specify on the local host.
- Optionally merging a database snapshot taken by **vob_snapshot** with the restored data. This step also validates the integrity of the database snapshot and ensures that the database snapshot is the correct one for the VOB being restored.
- Creating a new VOB object and tag in the registry if the VOB is not being restored to its currently registered location.
- Reactivating the VOB.
- Running **checkvob** to check for inconsistencies between the VOB's database and its pools. This step is critical if you use **vob_snapshot** and recommended in any case.

Note: You cannot use **vob_restore** to restore a VOB that is located on a network-attached storage device. For more information, see *Restoring a VOB on Network-Attached Storage* on page 207.

Scenarios Supported by vob_restore

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

- The pathname of a target directory to which the VOB will be restored

- The pathname of a source directory, from which backed-up data will be copied to the target
- The (optional) pathname of a directory that holds a VOB database backed up with **vob_snapshot**

This information, along with the VOB tag you supply on the **vob_restore** command line, allows **vob_restore** to determine which of the following scenarios to use:

- An in-place restore, which restores the backup to the VOB's current storage directory. This scenario has two variations:
  - In the variation that uses the least disk space, both the target and source directories are the same as the VOB's currently registered storage directory. Because you cannot copy the source data into the storage directory of a VOB that is registered and possibly still active, **vob_restore** unregisters the VOB, stops its server processes, and prompts you to copy the source data to the target. If you have also supplied a snapshot backup path, **vob_restore** verifies that the database snapshot is valid and then merges the snapshot with the data you have copied to the target.
  - If you have adequate disk space available on the VOB host or on the network, consider the variation of the in-place restore in which the target and source directories are different and the source pathname specifies a temporary staging area where you have placed the backup data you want to restore. In this variation, **vob_restore** manages the entire process of restoring the VOB.
- A restore-and-move, which restores the backup to a different location on the local host. In this scenario, the source and target pathnames are usually the same and become the VOB storage directory after restoration is complete.

In either scenario, the following rules apply:

- The VOB to be restored must be registered and have a tag in the host's current region.
- The target directory must be local to the host on which you are running **vob_restore**. It cannot be on a remote host or a NAS device.
- If the target has remote pools, you must restore them manually.

Running vob_restore

This section summarizes the steps required to restore a VOB when you use the in-place restore scenario. Both variants are described.

Note: Many **vob_restore** prompts allow you to quit. If you quit at any of these prompts, **vob_restore** saves its current state in a restart file and displays the name of that file on

standard output. You must use this file name as the *restart_path* argument to **vob_restore** to restart after you have quit.

1. **Log on to the VOB host.** Log on as a privileged user.
2. **Verify that the target volume has adequate disk space**. In most cases, an in-place restore does not require more disk space than is available in the VOB's currently registered storage directory. If you are restoring to a different location, you can determine the space needed for the VOB in any of several ways:
   - If the VOB's admin directory is intact, you can check the VOB storage node of the ClearCase Administration Console or use the **cleartool space** command.
   - If the VOB's admin directory has been destroyed or has data that is not current, you can use ordinary file system utilities such as **dir**, **ls**, or **du** to get a rough idea of the space consumed by the VOB storage directory.
3. **Exit any view context.** You cannot run **vob_restore** in a view.
4. **Run vob_restore.** Unless you are restarting an incomplete **vob_restore**, the command takes a single argument that specifies the VOB tag:

   vob_restore *vob-tag*

   If you are restarting an incomplete **vob_restore**, you must also supply the **–restart** *restart_path* option. **vob_restore** displays some introductory information and requests confirmation to proceed.
5. **Enter the target path.** This is the path to which you want to restore the VOB. On a UNIX computer, specify a host-local path (for example, /vobstg/libpub.vbs). On a Windows computer, specify a UNC path (for example, \\ccsvr01\vobstg\libpub.vbs).
6. **Enter the source path.** This is the path where **vob_restore** can find the backup data. If you use the variant of the in-place restore that requires a temporary staging area, enter its pathname. Otherwise, enter the pathname of the VOB's currently registered storage directory (the same pathname you entered in Step 5), but do not copy the backup data to that path until you are prompted by **vob_restore** (see Step 16).
7. **Enter the path at which vob_restore can find the database snapshot.** If you are restoring a VOB backed up with **vob_snapshot**, supply the snapshot pathname now. Otherwise, press RETURN to skip this step.
8. **Specify whether the backup is currently accessible.** If the backup data is available at the path you entered in Step 6, enter **yes** at this prompt. Otherwise, enter **no**.

9 **Review the summary information and decide whether to proceed.** If the information is correct and you're ready to proceed with the restore, enter **yes**. Otherwise, enter **no**.

10 **Check the VOB database.** If the backup and optional database snapshot are accessible, **vob_restore** may prompt you to allow it to validate the integrity of the database if either of these conditions exist:

- You are restoring a database snapshot (you entered a path in Step 7)
- You are using a standard backup (you did not enter a path in Step 7) and **vob_restore** detects that the database was not locked during backup.

If the database fails this test, **vob_restore** cannot continue. (If you answered **no** in Step 8, the database check will be deferred until you have loaded the backup in Step 16.)

11 **Restore remote pools.** If the VOB is on a UNIX host, it may have remote storage pools. **vob_restore** prompts you to restore them now. Copy them from the backup to their target locations. Press RETURN when the remote pools have been restored, or, if there are no remote pools, to continue.

12 **Verify that the database snapshot should replace the database in the backup.** If you are restoring a VOB backed up with **vob_snapshot**, **vob_restore** prompts you to confirm that it can overwrite the database in the backup with the one in the snapshot. To do this, enter **yes**. Otherwise, enter **quit** and rerun this procedure specifying **no** in Step 7.

13 **Remove the VOB tag and unregister the VOB.** Before it can continue, **vob_restore** must remove the VOB tag and unregister the VOB. **vob_restore** prompts you to remove any tags that may exist in other registry regions and to unregister the VOB in any other registries. Press RETURN when you have completed this task or, if there are no other tags or registries, to continue.

14 **Verify that it is safe to shut down ClearCase**. **vob_restore** has to stop the VOB's server processes before the restore can proceed.

15 **Remove the existing VOB storage directory.** After it has shut down ClearCase, **vob_restore** prompts you to remove the existing VOB storage directory. Press RETURN when the directory has been deleted.

16 **Copy the backup data to the target**. If you responded **no** in Step 8, **vob_restore** prompts you to load the backup now. Press RETURN when the backup has been loaded.

Note: On UNIX, each VOB storage directory includes a subdirectory 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 permissions when you copy the backup data to the target:

- If you use **tar**, specify the **–p** option when restoring the VOB. You must run the **tar** command as **root**. Otherwise, the **–p** flag is ignored.
- If use **cpio**, no special options are required.

17 **Run checkvob.** After the target directory is populated with the backup data and optional database snapshot, **vob_restore** prompts you to run **checkvob**, a utility that can detect and optionally fix discrepancies between the VOB database and storage pools. You can specify the types of pools to check and whether **checkvob** should try to fix any problems that it finds. These problems are more likely to occur in a VOB that has been restored with a database snapshot, because the database and the pools have been backed up at different times. Chapter 12 and the **checkvob** reference page contain more information about **checkvob**. We strongly recommend that you read this information, especially if you are going to run **checkvob** in **–fix** mode.

Caution: If you decide to run **checkvob** in **–fix** mode, **vob_restore** unlocks the VOB, then locks it again specifying **–nusers** *username* where *username* is the name of the user running **vob_restore**. If the VOB is replicated, this user must not make any modifications to the VOB other than those made by **checkvob –fix**. Otherwise, replication failures and replica divergence will occur.

18 **If the VOB is replicated, run the MultiSite restorereplica command.** You must perform this processing while the VOB is still locked.

19 **While the VOB is still locked, run some consistency checks.** For example, in a view with the default config spec, access the restored VOB and verify that all expected elements and versions are present.

20 **Unlock the restored VOB**. **vob_restore** always leaves the VOB locked when it exits.

The restored VOB is now ready for use, but you may need to take two additional steps:

- If necessary, resynchronize the VOB and views. See *To Synchronize Dynamic Views* on page 211.
- If you have restored the VOB to a different location and the VOB has tags in other regions, replace those tags, as described in *Creating, Removing, or Changing Tags* on page 45.

Restoring a VOB from Backup Without vob_restore

In most cases, the easiest way to restore a 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**.

1. **Log on to the VOB host.** Log on as a privileged user.
2. **Verify that the target volume has adequate disk space**. In most cases, an in-place restore requires no more disk space than is available in the VOB's currently registered storage directory. If you are restoring to a different location, you can determine the space needed for the VOB in any of several ways:
   - If the VOB's admin directory is intact, you can use the VOB storage node of the ClearCase Administration Console or the **cleartool space** command.
   - If the VOB's admin directory has been destroyed or has data that is not current, you can use ordinary file system utilities such as **dir**, **ls**, or **du** to get a rough idea of the space consumed by the VOB storage directory.
3. **Stop ClearCase.** This ensures that the existing VOB's server processes are stopped.
4. **Remove the existing VOB storage directory.** Use any file system utility (**rmdir**, **rm**) or a GUI such as Windows Explorer.
5. **Restart ClearCase.** Starting ClearCase makes other VOBs and views on the host available.
6. **Load the backup**. Recreate the VOB storage directory if necessary; then restore the VOB storage directory's contents from the backup. Be sure to use a copy utility that preserves file and directory protections.

   On UNIX, each VOB storage directory includes a subdirectory 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 permissions when you copy the backup data to the target:
   - If you use **tar**, specify the **–p** option when restoring the VOB. You must run the **tar** command as **root**. Otherwise, the **–p** flag is ignored.
   - If use **cpio**, no special options are required.

   On Windows, follow the procedures in *Preserving NTFS ACLs When Copying a VOB or View Storage Directory* on page 257.

   Note: On both Windows and UNIX, if the VOB was locked when it was backed up (the most likely case), it will be locked when it is restored. If the VOB was not locked when it was backed up, the backup is probably worthless.

7 **Fix storage directory protections if necessary**. If your Windows backup tool does not back up and restore ACLs correctly, you may need to fix them now. See *Repairing Storage Directory ACLS on NTFS* on page 255 for details.

8 **Restore remote pools.** If the VOB is on a UNIX host and has any remote storage pools, restore them now. You can simply copy them from the backup to their target locations. Remote pools must be in place when you run **checkvob** in Step 10.

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

# cleartool register –vob  /vobst_aux/flex.vbs

# cleartool mktag –vob –replace –tag /vobs/flex  /vobst_aux/flex.vbs
```

10 **Run checkvob.** This utility can detect and optionally fix discrepancies between the VOB database and storage pools. These problems are more likely to occur in a VOB that has been restored with a database snapshot, because the database and the pools have been backed up a different times. Chapter 12 and the **checkvob** reference page contain more information about **checkvob**. We strongly recommend that you read this information before you run **checkvob**.

Caution: Before you run **checkvob** in **–fix** mode, you must unlock the VOB, then lock it again specifying **–nusers** *you*. If the VOB is replicated, you must not make any modifications to the VOB other than those made by **checkvob –fix**, and you must lock the VOB again for all users as soon as **checkvob** exits. Otherwise, replication failures and replica divergence will occur.

11 **If the VOB is replicated, run the MultiSite restorereplica command.** You must perform this processing while the VOB is still locked.

12 **While the VOB is still locked, run some consistency checks.** For example, in a view with the default config spec, access the restored VOB and verify that all expected elements and versions are present.

13 **Unlock the restored VOB**.

The restored VOB is now ready for use, but you may need to take two additional steps:

- If necessary, resynchronize the VOB and views. See *To Synchronize Dynamic Views* on page 211.
- If you have restored the VOB to a different location and the VOB has tags in other regions, replace those tags, as described in *Creating, Removing, or Changing Tags* on page 45.

Restoring a VOB on Network-Attached Storage

You cannot using **vob_restore** to restore a VOB whose database is stored on a NAS device. To restore a VOB to a NAS device, use the procedure described in *Restoring a VOB from Backup Without vob_restore* on page 205. When you register the restored VOB and create a tag for it, you must supply the **–host**, **–hpath**, and **–gpath** options to the **cleartool register** and **mktag** commands. For example, if the VOB had been restored to the /vobst_aux directory on a NAS device, you would use the following **register** and **mktag** commands:

**cleartool register –vob –host ccvobsvr1 –hpath /net/nasdevice/vobst_aux/flex.vbs **
–gpath /net/nasdevice/vobst_aux/flex.vbs /net/nasdevice/vobst_aux/flex.vbs

**cleartool mktag –vob –replace –host ccvobsvr1 –tag /vobs/flex **
**–hpath /net/nasdevice/vobst_aux/flex.vbs **
–gpath /net/nasdevice/vobst_aux/flex.vbs /net/nasdevice/vobst_aux/flex.vbs

Restoring an Individual Element from Backup

If you mistakenly delete an element with **rmelem**, you can restore it from a VOB backup by using the procedure presented in this section.

Note: This procedure cannot be used to restore an element to a UCM component.

Removing a directory element does not remove the file elements cataloged within it; file elements exist independently of directory elements. In many cases, removing a directory element moves the files it contains to the VOB's lost+found directory. Look there first if you've accidentally removed a directory element.

To restore a single element from a backup of a non-UCM VOB:

1. **Deactivate the VOB from which the element was deleted.** Unmount it if it is mounted, and then remove its tag and unregister it. This step is critical. In Step 2, you restore a temporary copy of this VOB from backup. The temporary copy and the original cannot both be registered in the same registry.
2. **Temporarily restore the most recent backup of the VOB.** Use the procedure in *Restoring a VOB from Backup with vob_restore* on page 200 or *Restoring a VOB from Backup Without vob_restore* on page 205. Do not restore the backup to the storage directory used by the original VOB. The restored VOB is used only long enough to retrieve the deleted element, so you can restored it to any convenient temporary location on a VOB server host.
3. **Register and create a tag for the restored VOB** if you did not use **vob_restore** in Step 2.

4 **Create a new temporary VOB** to hold the element you need to restore. Because the original VOB and the restored backup cannot be active at the same time, this temporary VOB serves as a staging area to which the element you recover can be copied.

5 **Retrieve the deleted element from the restored VOB.** Use **cleartool relocate** to relocate the element from the backup VOB to the temporary VOB. For more information, see the **relocate** reference page and *Relocating Elements to Another VOB* on page 141.

6 **Remove the restored VOB** after you verify that the element you need has been relocated to the temporary VOB. Use **rmvob** or the ClearCase Administration Console.

7 **Re-register and recreate the tag for the original VOB.**

8 **If you are using a dynamic view, mount the VOB.**

9 **Relocate the element from the temporary VOB to the original VOB.** Use **cleartool relocate**.

10 **Verify that the element is accessible in the original VOB.**

11 **Remove the temporary VOB.** Use **rmvob** or the ClearCase Administration Console.

Note: This procedure restores the element to the version of its directory that is selected by the view in which Step 5 takes place. It does not restore the element to earlier versions of the directory.

Restoring a View from Backup

Use the following procedure to restore a view from backup. This procedure applies to views stored on ClearCase hosts and to views stored on certified NAS devices.

1 **Log on to the view host.** Log on to the host where the view storage directory (or, for a snapshot view, the snapshot view directory) resides.

2 **Check disk space availability**. Make sure that the view's disk partition has enough free space to hold the backup copy. If necessary, delete the view storage directory (or, for a snapshot view, the snapshot view directory), or use other means to make enough space available.

3 **Stop the view server.** Use **endview –server** *view-tag* to stop the **view_server** process for the view named in *view-tag*.

4 **Remove the original view if it still exists.** Use the ClearCase Administration Console or the **rmview** command. If you plan to restore the view to its currently registered storage directory and use its current tag, you can use an operating system utility like **rm** or **rmdir** to remove the entire view storage directory.

Note: If you are manually removing a snapshot view and its view storage directory is not a subdirectory of the snapshot view directory (the normal configuration on ClearCase LT), remove both the snapshot view directory and the view storage directory.

5 **Make sure that the restored view has the correct ownership.** Depending on your platform and your backup tool, you may have to take one or more precautions now to ensure that the view is restored with the correct owner and group identities.

- On UNIX, if you use **tar**, specify the **–p** option when restoring the VOB. You must run the **tar** command as **root**. Otherwise, the **–p** flag is ignored. If you use **cpio**, no special options are required.
- On Windows, if your backup tool or copy utility does not preserve NTFS ACLs, you may need to fix them after the restore. See *Repairing Storage Directory ACLS on NTFS* on page 255 for details.

Note: When you restore a snapshot view, the utility you use to restore the backup must preserve the original modification times and ownership of all files and directories in the view. If it does not, all loaded files are reported as hijacked when the view is activated.

6 **Restore the backup.** If you are restoring a snapshot view, you must restore both the snapshot view directory and the view storage directory, which may not be a subdirectory of the snapshot view directory. You can restore a view to its old location or 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).

7 **Register and create a tag for the view if necessary.** If you have restored the view to its currently registered location, the existing view object and view tag remain valid. Otherwise, use the ClearCase Administration Console or the **cleartool register** and **mktag** commands to register the view and create a tag for it.

Note: If you are restoring a view to a NAS device, you must use the **–host**, **–hpath** and **–gpath** options to **register** and **mktag**. See *Restoring a VOB on Network-Attached Storage* on page 207 for an example.

8 **Clean up any post-restore skew.** The restored view database will not reflect any changes made in the VOB since the view was backed up. For a discussion of the

problems that may arise and steps you may need to take to address them, see *Synchronizing VOBs and Views After You Restore*.

The view is now ready for use.

Synchronizing VOBs and Views After You Restore

VOBs and views maintain persistent references to each other. These references are backed up and restored along with the VOBs and views themselves. When a VOB is restored from backup, it may be out of sync with any view that holds a reference to it (a checked-out or loaded version from the VOB, for example). When a view is restored from backup, it may be similarly out of sync with any of the VOBs to which it holds a reference. This skew can cause a variety of problems, primarily in views, but also in VOBs.

- **Problems in the view.** Whenever a view is restored or a VOB referenced by a view is restored, any of the following problems can occur in the view:
  - View-private files and directories can become stranded because the VOB directory element that they were cataloged in no longer exists 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.
  - DO promotions can fail because the VOB records information about unshared DOs that don’t exist.
  - Winked-in DOs become inaccessible when the VOB no longer contains the shared DOs. This can occur if the VOB loses its DO pools and the shared DOs are removed by **checkvob** fix processing.
- **Problems in the VOB.** Whenever a dynamic view is restored, DOs in VOBs referenced by the view may not be scrubbed because they have out-of-date reference counts (the restored view no longer holds the references to the DO that are recorded in the VOB).

This section describes how to correct post-restore skew and identify or eliminate the problems.

To Find Views with References in a VOB

When a VOB is restored, resynchronize it with 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.

To Synchronize Dynamic Views

After restoring a dynamic view or a VOB that is referenced by a dynamic view, run **cleartool recoverview –synchronize** to recover stranded view-private files. After a view is restored, synchronize it with all of the VOBs it references. After a VOB is restored, you can restrict the **recoverview** synchronization to the restored VOB. For more information, see the **recoverview** reference page.

To Reconcile Missing Checkouts

To find and reconcile missing checkouts in either a snapshot or dynamic view:

- In 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**.
- In a snapshot view, run **cleartool ls –recurse –view_only**.
- Look for files that are marked `checked out but removed`.
  - If the file is from a VOB that has been restored, a user had probably checked out the element and resolved the checkout after the backup completed. If the checkout was to a dynamic view and was resolved with a checkin, the changes made in the checked-in version have been lost if a more recent backup of the VOB is not available. If the checkout was to a snapshot or Web view and was resolved with a checkin, the changes may still be present in the snapshot view directory. If the checkout was canceled, any changes made to the checked out file made may still be in the view in a view-private file with a .keep suffix.
  - If the checkout was in a view that was restored, any view-private modifications since the checkout have been lost.
  - In either case, the user must resolve the checkout by canceling it with **uncheckout** or by recreating the version data and checking it in.

To Remove Eclipsing View-Private Files

To find eclipsing view-private files in either a snapshot or dynamic view:

- In 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 –other –long**.
- In a snapshot view, run **cleartool ls –recurse –view_only**.
- Look for elements that are marked `eclipsed`.
  - If the element is in a VOB that was restored, a user had probably checked out the element after the VOB was last backed up.
  - If the element was checked out to a view that was restored, a user had probably checked out the element and then canceled the checkout or 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 renamed so that a new version of the element can be checked out to hold these changes.

Remove Invalid Winked-In DOs

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.

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, follow the procedures in *Reestablish the Consistency of a Dynamic View's Derived Object State*.

Reestablish the Consistency of a Dynamic View's Derived Object 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 builds. As a result, errors occur during hierarchical builds that reuse those late-arriving DOs to construct higher-level targets. In the typical

case, **clearmake** displays an `INTERNAL ERROR` message and writes an entry of the following form in the error_log file:

```
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_cache.msg.c@@24-May.20:51.42929
  .
  .
  .
```

2 **Remove the DOs that have no config record.** Use an operating system command like **rm** or **del**.

rm bldr_cr.msg.o bldr_cr.o

Restoring One or More Members of a Set of Related Databases

As described in *Backing Up Related Databases Together* on page 195, there are relationships among VOBs that you must consider when backing them up and, especially, when restoring them. These relationships are most complex in a UCM environment, in which dozens of VOBs are connected by cross-VOB hyperlinks of various kinds, and where there may also be a ClearQuest database that holds references to objects in one or more PVOBs. Whenever one or more of these databases is restored from backup, skew (inconsistencies) between databases is a likely result.

This section describes several situations in which skew is introduced when a member of a set of related databases is restored from backup. Three of these situations apply only to UCM environments:

- A component VOB is restored from backup.
- A ClearQuest database is restored from backup.
- A PVOB is restored from backup.

This section also describes some more general considerations for environments that have administrative VOB hierarchies but do not use UCM.

Note: Many of the procedures described in this section use the **cleartool checkvob –ucm** command. For more information, see the **checkvob** reference page.

Restoring a Component VOB

When you restore one of a project's component VOBs from backup, the following types of skew may be introduced:

- Change sets and baselines defined in the PVOB may reference versions and perhaps elements that do not exist in the component (because they were created after the component VOB was backed up).
- If the VOB contains multiple components, components that were created after the most recent backup are referenced in the PVOB but do not exist in the component VOB.

To analyze and, if necessary, fix these types of skew, use **cleartool checkvob –ucm**:

1. Restore the component VOB by using any of the procedures in this chapter.
2. Establish a working view. This step is recommended, but not essential. If you run **checkvob –ucm** in a view, you do not have to specify the PVOB on the command line, and **checkvob** can report names of elements that have problems and have a version that is selected by the view. (Versions that have problems but are not selected by the view are reported by numerical ID).
3. In the view, go to the root directory of the PVOB.
4. Run **checkvob –ucm** to detect skew between the PVOB and the restored component. The following command checks a component named **lib**:

   cleartool checkvob –ucm –component lib

   Note: If a component VOB contains more than one component, all components in the VOB are checked when you use **checkvob –ucm –component**.

Restoring a ClearQuest Database

When you restore a project's ClearQuest database from backup, the following types of skew may be introduced:

- Activities defined in the PVOB may reference fields and/or records that do not exist in the ClearQuest database.
- Any activity and stream names and/or titles that have been changed since the ClearQuest database was backed up will be different in the PVOB and the ClearQuest database.

To analyze and, if necessary, fix these types of skew, use **cleartool checkvob –ucm** and specify the **–crm_dbname** option:

1. Restore the ClearQuest database. Use the procedures specified by the database vendor. Verify that the restored database is working and accessible over the network.
2. Run **checkvob –ucm** to detect skew between the PVOB and the restored ClearQuest database. The following command line checks all entries for a project named **Release-6** in a ClearQuest database named **CQ-UCM-DB**:

   cleartool checkvob –ucm –crm_dbname CQ-UCM-DB –project Release-6

Restoring a PVOB

When you restore a PVOB from backup, the following types of skew may be introduced:

- Versions, baselines, and label types used by component VOBs may not exist in the PVOB.
- Activities in the ClearQuest database may reference objects that do not exist in the PVOB.

To analyze and, if necessary, fix these types of skew, use **cleartool checkvob –ucm** to check references to the PVOB held by each of the project's component VOBs, then run **checkvob –ucm** with the **–crmonly** option to check references to the ClearQuest database held by the PVOB:

1. Restore the PVOB by using any of the procedures in this chapter.
2. Run **checkvob –ucm** to detect skew between each component VOB and the restored PVOB. The following command line checks for skew between a component VOB whose tag is \lib and its PVOB:

   cleartool checkvob –ucm vob:\lib

   checkvob uses information stored in the component VOB to find the PVOB. You must run a command like this one for every component VOB in the project.
3. Run **checkvob –ucm –crmonly** to detect skew between the restored PVOB and the ClearQuest database. The following command checks all ClearQuest database references held by a PVOB whose VOB tag is \projects:

   cleartool checkvob –ucm –crmonly vob:\projects

Restoring a Member of an Administrative VOB Hierarchy

Note: If you are restoring a UCM component or PVOB, do not use the procedures in this section before you run **checkvob –ucm**. Doing so may remove damaged cross-VOB hyperlinks that **checkvob –ucm** can repair.

Although we recommend that you always back up members of an administrative VOB hierarchy as a group, it is likely that you will restore only a few members of a group at a time (for example, members stored on a server whose disk has failed). Whenever you restore some, but not all, members of an administrative VOB hierarchy, there is the possibility of a mismatch in global type information. For example, if an administrative VOB is backed up before a new global type is created, VOBs that are lower in the hierarchy will exhibit problems related to the unavailable global type after that backup is restored. To prevent such problems:

- After you restore an administrative VOB from backup, check for and fix any broken hyperlinks in that VOB and all other members of the hierarchy.
- After you restore any member of an administrative VOB hierarchy from backup, check for and clean up any broken hyperlinks from that VOB to its administrative VOB.

To remove broken hyperlinks, use **checkvob –hlink**. For more information, see *Using checkvob to Find and Fix Broken Hyperlinks* on page 241 and the **checkvob** reference page.

Periodic Maintenance 12

Periodic attention to storage management and repository integrity is an important administrative function. This chapter describes VOB and view storage management tasks, the **checkvob** utility, and the ClearCase scheduler, which you can use to automate many periodic maintenance chores.

The ClearCase Scheduler

The ClearCase scheduler provides a way to automate periodic maintenance on hosts that are configured with support for local VOBs and views. The scheduler's work is defined in terms of jobs and tasks.

- A job is a data structure that defines the execution schedule for a task and stores any exit status associated with the most recent execution of the task. Each job runs a single task.
- A task is an executable program or script. A task may be executed by more than one job and, because a job can pass job-specific arguments to a task, the operations performed by a task can be customized to suit the schedule on which the job runs the task.

Unlike the general-purpose scheduled-execution utilities (**cron**, **at**, ...) associated with the operating systems on which ClearCase runs, the scheduler is integrated with the ClearCase management infrastructure and supports these functions:

- Centralized management with **cleartool** or the ClearCase Administration Console
- Access control based on user and group identity
- Flexible scheduling based on time of day, period, or execution of another task
- A default set of tasks and jobs that run on every host
- Custom tasks and jobs

The scheduler is managed by the **albd_server**. The **schedule** program is an interface to the job and task registries that provide the scheduler with work to do. The default set of tasks and jobs is appropriate for many host systems and may need no modification. To modify jobs or tasks, take one or more of the following steps:

- Modify the scheduler access control list (ACL), if necessary.

- Add jobs to the default daily and weekly local tasks
- Define new tasks and jobs for them to execute.

Managing the Scheduler Access Control List

schedule uses a single access control list (ACL) that determines who is allowed access to the scheduler's task and job registries and to the scheduler ACL itself.

The ACL consists of a list of entries. Each entry assigns an access type to an identity. Four types of identity exist: **Everyone**, **Domain**, **Group**, and **User**. A domain is a Windows domain on hosts running Windows and an NIS domain on hosts running UNIX. Each group and user is qualified by a domain name. In a Windows domain, a group must be a global group, and a user must be a domain account.

Note: UNIX hosts that are not part of an NIS domain can use the string **<unknown>** in place of the domain name in an ACL entry.

Each identity may have one of three access types. Table 8 shows the access types and their implications for access to the schedule and access to the ACL itself.

Table 8 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

Although each identity can have only one access type, 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. When logged on locally, the privileged user always has **Full** access. On a remote host, access rights for all users (even privileged users) are determined by the scheduler ACL. Thus, to change the default ACL, you must be logged on to the host where the scheduler is running, and you must be a privileged user.

To view or edit the scheduler 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 that contains a representation of the current ACL using the ACL-definition syntax documented on the **schedule** reference page.

You can also create a text file that contains ACL entries in the scheduler's ACL-definition syntax, and then use the following command to replace the entire ACL with the ACL in the file (named acldef.txt in this example):

cleartool schedule –set –acl acldef.txt

Creating a Task

A task has two components:

- A job (executable program or script) to be executed when the task runs
- A definition for the task in the scheduler's task registry

A set of standard tasks and a schedule for running them is installed on every host that supports local VOBs and views. You cannot create, change, or delete any standard tasks. You can, however, define new tasks. You can also customize two predefined tasks, one of which runs 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

To create a new task:

1 Create an executable program that can run in the scheduler's execution environment (see the **schedule** reference page.). We recommend placing the program in the scheduler's tasks directory.

2 If you want the program to run daily, you can run it from the existing task ccase_local_day. If you want the program to run weekly, you can run it from the existing task ccase_local_wk. Both tasks are in the scheduler's tasks directory, and are scripts with a .sh suffix on UNIX and a .bat suffix on Windows**.** 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 task_registry file in the scheduler's tasks directory. To add a task to this file, use a text editor.

Tasks are defined with a job-definition syntax that is documented on the **schedule** reference page. A task definition includes these essential components:

- A unique numeric ID used by a job to refer to the task
- A unique name for the task
- The pathname to the executable program associated with the task

Caution: Place new task definitions at the end of the task registry file. Do not alter or delete any of the standard tasks defined in that file.

4 If you have added a task to the task registry, you must create a new job to run the task. See *Creating a Job* on page 221. 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.

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.

Caution: Edit only tasks that have been added at your site. Do not alter or delete any of the standard tasks defined in the task registry. 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 *Creating a Job* on page 221.

To change a task definition, use a text editor to edit the file task_registry in the scheduler's tasks directory. When you edit a task, you must use the task-definition syntax documented on the **schedule** reference page.

Deleting a Task

Before you delete a task definition, you must remove all references to the task in all scheduled jobs. See *Creating a Job* on page 221. To delete a task definition, use a text editor to edit the file task_registry in the scheduler's tasks directory.

Caution: Delete only tasks that have been added at your site. Do not alter or delete any of the standard tasks defined in the task registry.

Managing Jobs

You can create, delete, or edit jobs run by the scheduler. You can also run a scheduled job immediately. To manage a scheduled job on any Windows or UNIX host, use the ClearCase Administration Console. The **My Host** node and each host snap-in has a **Scheduled Jobs** node from which you can manage scheduled jobs on the host. If the scheduler ACL supports write access (see *Managing the Scheduler Access Control List* on

page 218), 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.

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 *Creating a Task* on page 219.
- Any arguments that must be passed to the executable associated with the task when it runs. For a standard 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.
- Job-related events that trigger e-mail notifications and recipients for the notifications. See *Specifying Job Notifications* on page 222.
- Whether the scheduler should delete the job after it runs.

To create a new job, use ClearCase Administration Console:

1. 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.

You can also use the following command:

cleartool schedule –edit –schedule

This command opens in a text editor a file that contains definitions for all currently scheduled jobs. To create a new job, add a definition using the job-definition syntax documented on the **schedule** reference page. You cannot specify any read-only job properties, such as **LastCompletionInfo**. The job runs in the environment described on the **schedule** reference page.

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:

- Navigate to the **Scheduled Jobs** node of the host on which you want to specify a job's schedule.
  - 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.
  - To specify the schedule for an existing job, select a job and click **Action > Properties**. In the 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 on the **schedule** reference page.

Specifying Job Notifications

The scheduler can send e-mail notifications to recipients you specify. You can also determine particular events that trigger notifications, such as the start of a job or the end of a job that fails.

Note: Job notifications require the scheduler to contact an SMTP mail server. On Windows, you must specify the name of this server on the **Options** tab of the ClearCase program in Control Panel on 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 **Details** pane and click **Action > Properties**. In the dialog box, click the **Settings** tab to specify notification events and recipients.

Or use the following command:

cleartool schedule –edit –schedule

This command opens in a text editor a file that contains definitions for all currently scheduled jobs. To specify the notification information for a new or existing job, edit the job's **NotifyInfo** property using the job-definition syntax documented on the **schedule** reference page.

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 **Details** 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 **Details** 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*

The job name is case-sensitive and must be quoted if the name contains spaces. For example:

cleartool schedule –get –job "Weekly Log Scrubbing"

```
Job.Begin
       Job.Id: 7
       Job.Name: "Weekly Log Scrubbing"
       Job.Description.Begin:
.
.
.
Job.End
```

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 by using the job-definition syntax documented on the **schedule** reference page.

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 **Details** pane and click **Action > Properties**. This command opens a dialog box in which you can edit properties of the job. You cannot edit any read-only job properties.

Or use the following command:

cleartool schedule –edit –schedule

This command opens in a text editor a file that contains definitions for all currently scheduled jobs. Edit the properties of the job using the job-definition syntax documented on the **schedule** reference page. 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 *pname* represents your file of job definitions:

cleartool schedule –set –schedule *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 **Details** 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, which is described on the **schedule** reference page.

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 **Details** pane and click **Action > Delete Job**.

Or use the following command:

cleartool schedule –delete *job-id-or-name*

Scrubbing to Control VOB Storage Growth

VOB storage, like any storage, must be managed to keep it from uncontrolled growth. Two utilities, the **scrubber** and the **vob_scrubber**, help manage the growth of VOB storage. Table 9 describes the function of each of these utilities.

Table 9 Default scrubber and vob_scrubber actions

Program	Removes (default)	Which are created by
scrubber	Cleartext containers that have not been referenced for more than 96 hours	Any command that accesses a version of an artifact in a VOB
	Derived objects that have not been referenced for more than than 96 hours	**clearmake**, **omake**, **clearaudit** (dynamic views only)
vob_scrubber	Certain event records (see **vob_scrubber** reference page for details)	Any command that creates or modifies elements, versions, or other metadata.
	Certain oplog entries (see **vob_scrubber** reference page for details)	Any modification of data or metadata in a replicated VOB.

Both utilities run on a default schedule with default parameters on any computer that hosts one or more VOBs. You may need to reconfigure scrubbing parameters and schedules, as described in this section, if the default ones are not appropriate.

Scrubbing VOB Storage Pools

Several basic rules the govern operations of the **scrubber**:

- Cleartext pools are scrubbed to control their size. These pools are essentially caches; scrubbing unneeded data containers in a cleartext pool has little or no effect on performance.
- Derived object pools are scrubbed to delete DO data containers that are no longer being used by any view. Scrubbing also removes the corresponding derived objects from the VOB database.
- Source pools are never scrubbed.

In the default configuration, the **scrubber** runs nightly as part of the Daily VOB Pool Scrubbing task. You can also run it manually.

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 by using the ClearCase Administration Console or a **mkpool –update** command. For more information, see *Adjusting Default Scrubbing Parameters* on page 227.

Scrubbing VOB Databases

Almost every change to a versioned artifact 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 VOB 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.

Note: Deleting event records and other metadata from the VOB database, using the **vob_scrubber** or any other mechanism (**rmver**, **rmelem**, **relocate**, and so on), increases the amount of free space in the database, but does not reduce the disk space used by the VOB database. A regularly scrubbed VOB database grows slowly and should not

require further intervention to keep growth under control. However, if you must occasionally force a reduction in the size of a VOB database, scrub it, and then run the **reformatvob** command.

Adjusting Default Scrubbing Parameters

Typically, you adjust a VOB host's default scrubbing parameters in response to either of these two common problems:

- **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 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 did not previously experience. Before you make any scrubbing adjustments on a VOB host, analyze its scrubber_log file. The **scrubber** reference page explains how to read this file.

Note: If you use a 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 during the previous 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 simple examples of adjustments to scrubbing parameters.

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/sources

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/sources

cleartool mkpool –derived new_do_pool

```
Comments for "new_do_pool":
```

pool for DOs created in /vobs/sources/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**. Use the **lsvob** command:

cleartool lsvob /vobs/sources

```
* /vobs/sources /net/mars/vobstg/sources.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 scheduler that invokes the **scrubber** utility for your new pool. For example:

```
scrubber –e –p new_do_pool /net/mars/vobstg/sources.vbs
```

This script invokes the **scrubber** on the derived object storage pool `new_do_pool`. The **–e** option removes DOs with a reference count of zero.

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 about tasks and jobs, see *The ClearCase Scheduler* on page 217.

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/mars/vobstg/sources.vbs**

Scrubbing Less Aggressively

If your scrubbing regimen takes too long (perhaps spilling over into the work day), you can make the starting time for the 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 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 default Daily VOB Pool Scrubbing job to follow your new job instead. The default Daily VOB Snapshots job follows Daily VOB Pool Scrubbing, so you must change this job to follow your new job.

About checkvob

VOBs have complex internal and external relationships that can sometimes be disrupted by system or network failures or when a VOB is restored from backup. **checkvob** is a **cleartool** subcommand that can find, and in many cases fix, a variety of problems in these relationships. For example:

- Inconsistencies between a VOB's database and its pools
- Improper protections on storage pools
- Improperly configured **AdminVOB** hyperlinks
- Local types that eclipse global types
- Broken cross-VOB hyperlinks, including those that link UCM components to PVOBs.

You can use **checkvob** for a variety of purposes:

- When you restore a VOB from backup media without using **vob_restore** (the **vob_restore** script runs **checkvob** as part of its normal operation)
- When you restore a ClearQuest database used by a UCM project
- When you move a VOB
- As part of your periodic VOB maintenance routine

Replicated VOB Considerations

Note: This section does not apply to ClearCase LT.

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; in a VOB replica restoration scenario, **checkvob** must be able to run before **restorereplica**.) To synchronize replicas after running **checkvob –fix**, 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 use MultiSite to repopulate this database with version data from another replica. 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 that 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.

Using checkvob to Find and Fix Internal VOB Inconsistencies

As described in *The VOB Storage Directory* on page 105, a VOB storage directory contains a VOB database and a number of storage pools. Any operation that modifies VOB data makes changes to the database and at least one pool. When inconsistencies arise between the contents of a VOB database and the contents of its pools, VOB data integrity may be damaged.

There are two main causes of such inconsistencies:

- A VOB is restored from a backup that backed up the database and the pools at different times. Backups made with the **vob_snapshot** utility fall into this category. In the most common case, the pools are backed up after the database, and may contain data that the database does not reference.
- A system or network failure prevents a ClearCase operation from completing properly, which can leave unreferenced data containers in the pools.

This section describes how **checkvob** operates. See also the **checkvob** reference page.

Individual File Element or DO Processing

When processing individual elements (no **–pool** option), **checkvob** takes the following steps for file elements:

1. Check the element for **–data** (missing container) and **–protection** problems.
2. If running in **–fix** mode, fixes protection and missing container problems:

   Note: Fix processing is blocked if the VOB, source pool, or element is locked.

   a. Locks the element.

   b. Fixes missing protection problems.

   c. Fixes missing data container problems by scanning the pool for missing or misplaced containers and reconstructing containers as necessary.

   d. Updates the VOB database to reference the reconstructed containers.

   e. For missing version data, updates the VOB database to dereference lost versions with the equivalent of **rmver –data**.

   f. Applies minor events to the element's event history.

   g. Move alternate (unreferenced) containers for this element to pool's lost+found directory.

h Unlock the element.

checkvob takes a similar approach to processing DOs:

1 Checks for **–data** and **–protection** problems.

2 If running in **–fix** mode, fixes protection and missing container problems:

   a Fixes missing protection problems.

   b For each missing container, removes the DO with the equivalent of **rmdo**.

Pool Mode Processing

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 Checks pool roots. Checks pool names, locations, and pool identity information. Each pool must have a valid pool_id file. Problems found at this stage cannot be fixed by **checkvob** and must be directed to Rational Customer Support.

2 For source pools, checks the installed type managers to verify that they support the **get_cont_info** method, which **checkvob** requires when processing elements.

For source and DO pools only:

3 Scans the VOB database for missing (referenced, but not found) or misprotected containers.

4 Generates a list of problem VOB objects.

5 If **–fix** is specified, processes each VOB object on the list as described in *Individual File Element or DO Processing* on page 231.

6 Scans for unreferenced data containers.

7 If **–fix** is specified, moves each unreferenced container to the pool's lost+found directory.

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 failures, system crashes, failed or aborted cleanup operations, operating system bugs, disk

failure, network reconfiguration events, and so on. Another class of events includes accidental or malicious delete, move, rename, or change-protection operations on 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 time, the time when a still-operational VOB was locked, or the 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 than the pools.
  - The pools are newer than the database.
  - 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.

Pool Name and Protection Problems

checkvob can detect the following problems with pools.

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:** Create the **pool_id** file if it is missing from an existing pool whose name and location are recorded in the VOB database. **checkvob** cannot fix any other problems related to pool names or location. It skips to the next pool kind as soon as it finds a pool of the current kind that has any problem other than a missing **pool_id** file.

Source or DO Pool: Misprotected Container on Windows

- **Description:** Pool has an incorrect FAT RO attribute or NTFS 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 log on as a privileged user, and take the following steps:

1. In Windows Explorer, click **File > Properties**. On the **Security** tab, take ownership of all files and directories in the VOB's storage directory.

   Note: If you have identified only a small number of affected files, take ownership of these files only, to avoid a time-consuming **checkvob** operation in Step 6.

2. For all files and directories in the VOB storage directory, use the **Security** tab to grant full rights to the ClearCase administrators group and the VOB owner.
3. Log out.
4. Log on as the VOB owner.
5. Take ownership of all files and directories in the VOB storage directory.
6. Run **checkvob –force –fix –protection** to fix protections on storage pools:

Missing and Unreferenced Data Containers

checkvob works to reconcile the contents of VOB storage pools 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 for alternate containers from which to reconstruct the missing ones.
- **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 properties of an unreferenced container before moving it to the appropriate pool's lost+found directory.

Whenever the VOB database and storage pools have different reference times, **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, and can correct the problem in fix mode.

Source Pool: Missing Container

- **Description:** The VOB database references a source pool data container that does not exist at the expected location.
- **Causes:** Any of the following, depending on whether the database or the pool has the more recent reference time:
  - (Database newer) The database records a checkin that is not found in the (older) pool.
  - (Pool newer) Pool stores updated, renamed container with new checkin data, but the older database references the 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:** Examine any unreferenced containers. If a container is found that has the correct data (or a subset or superset of the correct data), create a new container and copy in the referenced (subset of) data. If **–force** is specified or the `fix?` prompt is accepted, update VOB database to reference reconstructed containers. Display a list of versions that could not be recovered.

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:** Any of the following, depending on whether the database or the pool has the more recent reference time:
  - (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 and the containers unreferenced.
- Restoring a pool from incremental backups.
- Unexpected events.

- **Fix processing: checkvob** usually moves an unreferenced container to the pool's lost+found subdirectory. It does not move unreferenced containers that fit 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 **–fix** and **–debris** without specifying **–data**, **checkvob** performs **–data** check processing to identify debris that may be needed, but does not perform **–fix** processing.
  - **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 the **vob_snapshot_setup** reference page), 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 the pool's lost+found directory. These versions should be presumed lost. Contact Rational Customer Support for more information.

DO Pool: Missing Container

- **Description:** VOB database references a DO pool data container that does not exist at the expected location.
- **Causes:** Any of the following, depending on whether the database or the pool has the more recent reference time:

- (Database newer) The database references a recently promoted DO, but the older pool does not include its container.
- (Pool newer) The older database references a DO that has been scrubbed from the newer pool. See the **scrubber** reference page.
- Unexpected events.

- **Fix processing:** Using the equivalent of **rmdo**, **checkvob** adjusts the database to remove the reference the DO container.

DO Pool: Unreferenced Container (Debris)

- **Description:** Source 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:** Any of the following, depending on whether the database or the pool has the more recent reference time:
  - (Database newer) The pool includes a DO that was subsequently scrubbed.
  - (Pool newer) A DO was promoted to the DO pool, but the older database does not record the promotion.
  - Unexpected events.
- **Fix processing:** Container moved to pool's lost+found directory.

Minimizing Data Loss with checkvob -force -fix

During **–fix** processing, **checkvob** uses the following algorithm to recover missing data by examining the contents of unreferenced containers. No changes are made to the database unless both the **–force** and **–fix** options are specified or you accept a `fix?` prompt.

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 might 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 alternate containers 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: If no alternate containers with the right type manager are found, **checkvob** constructs the container one version at a time from whatever sources are available, including containers maintained by other type managers and versions in cleartext pools.

2 If container reconstruction is incomplete, collect and report a list of missing versions.

3 If **–force** is specified or the `fix?` prompt is accepted, update VOB database:

a Adjust the database to reference reconstructed containers.

b Adjust the database (with the equivalent of **rmver –data)** to dereference lost version data.

4 Move all of the element's alternate containers to pool's lost+found directory.

Note: Because (unreferenced) alternate containers are moved to lost+found now, rather than during **checkvob**'s subsequent debris processing pass, you have an opportunity to reclaim disk space from lost+found if the disk fills up during reconstruction. (Reconstruction can consume substantial disk space.) For example:

- A **chtype binary_delta *.gif** operation (from element type **file**) is not recorded in 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 storage pools. **checkvob** may have to find and clone hundreds, or thousands, of unreferenced data containers.

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 with 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]
  ```

  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 24-hour-old backup, you can reasonably expect to lose version data created in the past 24 hours. In this case, you can run:

  checkvob –force –fix –data –pool –source *VOB-stg-pname*

  and respond to the `Allow missing data created since:` prompt to direct **checkvob** to allow the loss of data created and recorded in the VOB database after that time.

  If **checkvob** cannot find an expected container with data that was created before the specified time, it records this fact in the output log but does not accept the data loss until you 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** asks 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.

Note: checkvob does not find or fix corrupted data containers. A container with the right identity information at the right location is considered healthy by **checkvob**.

Using checkvob to Find and Fix Problems with Global Types

In addition to identifying and correcting internal inconsistencies in a VOB, checkvob can identify and correct problems with an administrative VOB hierarchy. For example:

- VOBs with multiple **AdminVOB** hyperlinks
- 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

To find and, optionally, fix the global types problems in the hierarchy:

1. Run **checkvob –global** in any VOB in the hierarchy. If **checkvob** detects that the VOB has more than one **AdminVOB** hyperlink, it stops and prompts you to correct the problem.
2. To correct the problem, use **cleartool describe –long** to list VOB hyperlinks, and then use **rmhlink** to remove all but one of them. In this example, the VOB **\sources** has two AdminVOB hyperlinks, one of which must be removed before **checkvob** can continue checking the hierarchy.

   cleartool describe –long vob:\sources
   ```
   versioned object base "\sources"
   ...
     Hyperlinks:
       AdminVOB@2@\sources -> vob:\projects
       AdminVOB@3@\sources -> vob:\admin1
   ```
   cleartool rmhlink AdminVOB@3@\sources
   ```
   Removed hyperlink "AdminVOB@3@\sources".
   ```
3. After the hierarchy has been corrected, run **checkvob –global** again to check for problems with global types in the hierarchy. **checkvob** can detect the following types of problems:
   - Eclipsing local types
   - Eclipsing local locks
   - Protection mismatches
   - Name mismatches
4. Review the log file, and then run **checkvob –global –fix** to correct any problems. With the **–fix** option, **checkvob** does the following:
   - Changes the name of each local copy to match the name of its global type.
   - Attempts to acquire eclipsing local copies and eclipsing ordinary types. 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.
   - Changes ownership of each local copy to match the ownership of its global type.

   Unless you specify **–force**, **checkvob** prompts you before attempting to fix any problems it detects:

Using checkvob to Find and Fix Broken 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 at one end of a cross-VOB hyperlink is unavailable. **checkvob** prompts you before deleting each partially unavailable hyperlink that it detects. Use **–force** to suppress these 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.

Using checkvob in a UCM Environment

UCM environments, especially those that are integrated with Rational ClearQuest, can create complex relationships among a project's VOBs and its ClearQuest databases. **checkvob** has a special set of capabilities enabled by the **–ucm** option that can detect and correct many of the problems that may arise when one or more members of a related set of databases are restored from backup or are otherwise affected by a system or network failure. *Backing Up Related Databases Together* on page 195 describes ways in which you can schedule VOB and ClearQuest database backups to minimize restore-time inconsistencies between them. *Restoring One or More Members of a Set of Related Databases* on page 213 describes ways in which you can use **checkvob –ucm** to find and fix problems after restoring one or more members of a set of related databases (or any time an unexpected system or network event has caused an inconsistency in database relationships in a UCM environment.)

Note: Do not run **checkvob –hlinks** before you run **checkvob –ucm**. **checkvob –hlinks** may remove hyperlinks that **checkvob –ucm** can fix.

View Storage Maintenance for Dynamic Views

View storage maintenance for dynamic views includes two principal tasks:

- Removal of unneeded views. Views can outlive their usefulness and should be regularly considered as candidates for removal. Versioned artifacts and DOs in any view can be completely reconstructed from information stored in VOBs, although view-private files cannot. In addition to consuming storage, inactive dynamic views can have an adverse impact on build performance if they are still being searched for candidate DOs.

- Removal of unneeded view-private objects. 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 review, remove, and clean up their own views periodically.

Note: Snapshot views, which use the host's native file system and do not support creation of DOs, do not require any special storage maintenance procedures.

Getting Information on View Contents

The ClearCase Administration Console displays information on disk space used by 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 hold 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 by 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 disk space used by views:

- Daily data gathering on disk space used by views
- 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, see *The ClearCase Scheduler* on page 217.

Scrubbing View-Private Storage

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. (The container is copied, not moved, because 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, build scripts replace old (and potentially redundant) DO data containers with newer ones.
- 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.

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.

See the **view_scrubber** reference page for more information.

Cleaning Up a View Manually

Users can remove unneeded files from their views by using tools the operating system supplies for removing files. To list view-private files by the pathnames at which they appear in VOBs, use the **Private Files** subnode of the view storage node in the ClearCase Administration Console or use the **lsprivate** command:

cleartool lsprivate

```
\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]
  .
  .
  .
```

Do not remove any files that are listed as `[checkedout]`. If any files are listed as `not available - deleted perhaps?`, use the following procedure to recover them.

Note: 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.

1 **Decide which stranded files to delete**. Stranded files were once cataloged in VOB or view directories that are not currently accessible; in some cases, they may never become accessible again. See 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-private files are to be deleted.

2 **Collect the UUIDs of stranded objects**. 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 **util.cxx**:

```
<VOB-beeb313c.0e8e11cd.ad8e.08:00:69:06:af:65>
     <DIR-375b5ca0.0e9511cd.ae20.08:00:69:06:af:65>\util.cxx
```

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.
```

3 **Move stranded files to the view's lost+found directory**. To remove a set of stranded files, use **recoverview** to transfer them to the view's lost+found directory. This command transfers all stranded view-private files created in the same directory as util.cxx:

cleartool recoverview –dir 375b5ca0.0e9511cd.ae20.08:00:69:06:af:65 –tag r2integ

```
Moved file ccsvr03:\vws\integ\.s\lost+found\57FBB6DF.0418.util.hxx
Moved file ccsvr03:\vws\integ\.s\lost+found\2203B56D.00C2.util.cxx
```

In this example, **recoverview** transfers two files, util.hxx and util.cxx, to the lost+found directory.

4 **Delete the files from the lost+found directory.** You can now use a standard file removal command to delete the stranded files:

cd \vws\integ\.s\lost+found

```
c:\vws\integ\.s\lost+found> del 57FBB6DF.0418.util.hxx
c:\vws\integ\.s\lost+found> del 2203B56D.00C2.util.cxx
```

Importing Data 13

Rational ClearCase includes several utilities for importing data from other configuration management systems or directly from the file system itself. This section presents examples of how to import data into ClearCase.

Note: You cannot run any of the **clearimport** procedures described in this chapter 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*.

Importing Data from Microsoft SourceSafe

This example demonstrates the use of command line tools to import data from SourceSafe to a newly created VOB. It describes a sample SourceSafe configuration, shows how to import it, and examines how the import 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 the Example SourceSafe Configuration

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

Figure 7 Sample SourceSafe Configuration

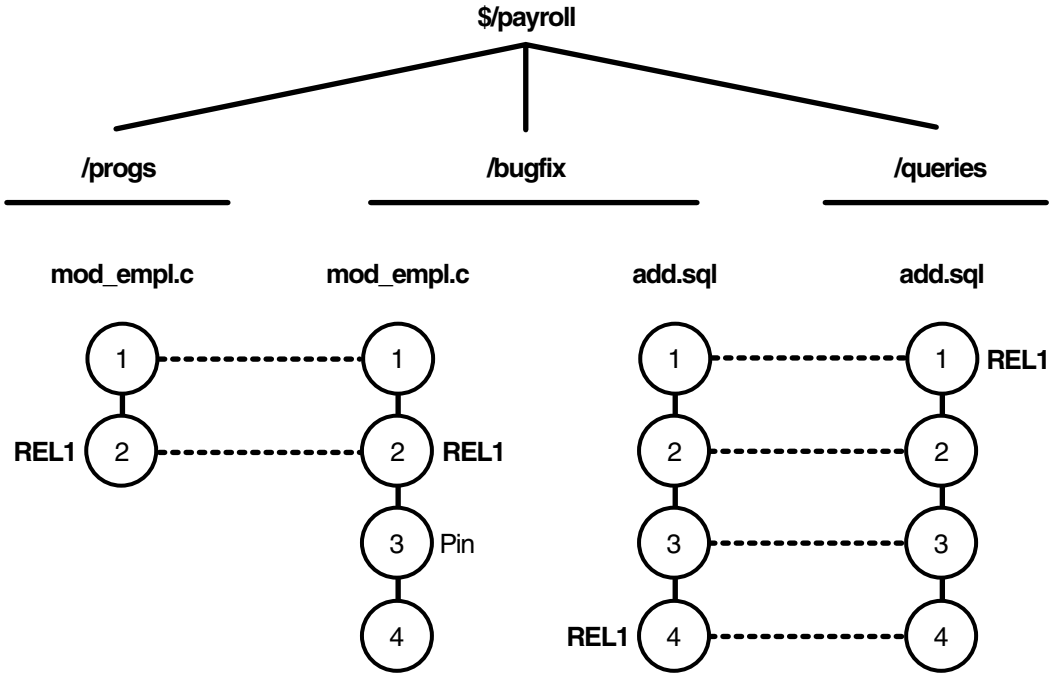


Figure 7 shows the configuration hierarchy in SourceSafe, which consists of these projects:

- **$/payroll** project
- **/progs** subproject for storing source code
- **/bugfix** subproject where developers store defect fixes
- **/queries** subproject for storing SQL queries

The payroll configuration contains the following objects:

- **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 share.
- **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 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.

clearexport_ssafe recognizes and handles most of these objects during the export phase of the conversion process.

Setting Your Environment

Before you start the conversion process, verify that the Windows PATH and TMP environment variables are set correctly, and that the SourceSafe current project has been specified.

- Verify that the PATH environment variable includes the location of the SourceSafe executables.
- Set the TMP environment variable to a location where **clearimport** can safely store temporary files. For example:

  set TMP=c:\temp

- 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 VOB elements. By default, **clearexport_ssafe** names the data file cvt_data and stores it in your current working directory. To specify a different name and storage location for the data file, use the **–o** 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.

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 "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 ...
.
.
.
```

Running clearimport

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

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**.

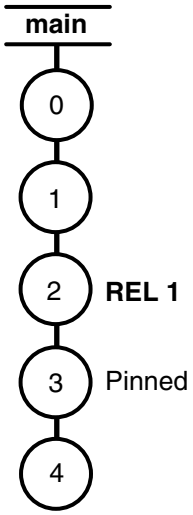
clearimport c:\datafiles\paycvt

```
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"
.
.
.
Closing directories.
```

Examining the Results

After **clearimport** finishes populating the VOB, examine the version trees of the new VOB elements to verify that **clearexport_ssafe** and **clearimport** converted the SourceSafe configuration as you expected. In ClearCase Explorer or Windows Explorer, open the folder, select an element, and view its version tree. Figure 8 shows the version tree for the **mod_empl.c** element from the SourceSafe **/bugfix** project.

Figure 8 Version Tree of Imported Element



- **Branches.** In the SourceSafe 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.
- **Version numbers.** As it does with all elements, ClearCase creates version 0 at the root 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.
- **Pins.** ClearCase does not have a feature equivalent to a SourceSafe pin. Because pins sometimes perform the same function as labels, the conversion process maps pins to labels. In the SourceSafe 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.

Importing RCS Data

This example demonstrates the use of command line tools to import data from a UNIX RCS source tree to a newly created VOB. The root of the RCS tree is /usr/libpub on a host where the empty VOB has already been activated, at /vobs/libpub.

Creating the Data File

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

   cd /usr/libpub

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

   clearexport_rcs

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

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

Running clearimport

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

2. **Go to the target VOB.** Change to the root directory of /vobs/libpub:

   cd /vobs/libpub

3. **Run clearimport.** Run **clearimport** on the data file, cvt_data, to populate /vobs/libpub:

```
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.

Importing PVCS Data

This example demonstrates the use of command line tools to import data from PVCS to a newly created VOB. The PVCS tree is located at C:\libpub, on a host where the empty VOB has been activated, at \libpub.

Creating the Data File

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

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

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

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

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

Running the Conversion Scripts

1. **Use any view. clearimport** ignores the config spec. Using a dynamic view improves the performance of the import operation.
2. **Go to the target VOB.** Change to the root directory of the \libpub VOB:
3. **Run clearimport.** Run **clearimport** on the datafile, **cvt_data**, to populate \libpub:

```
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 about the ClearCase exporters and importer, see the **clearexport_ccase**, **clearexport_pvcs**, **clearexport_rcs**, **clearexport_sccs**, **clearexport_ssafe**, **clearexport_cvs**, **clearimport** and **clearfsimport** reference pages.

Troubleshooting 14

This chapter includes procedures that can be useful if you need to do any of the following tasks:

- Fixing protection problems on NTFS VOB or view storage
- Using an existing view after you relocate elements that it references
- Finding a physical container created by the MVFS for file storage
- Editing a UNIX crontab to prevent recursive traversal of the /view directory

Repairing Storage Directory ACLS on NTFS

We recommend that you use NTFS-formatted disks for holding VOB and view storage directories on Windows computers. NTFS file system objects are protected by security descriptors, which contain ownership information and access control lists (ACLs). FAT file systems do not support ACLs, so objects in FAT file systems can only be protected by the **readonly** attribute. This attribute is available in both NTFS and FAT, but it is not enforced and can be removed easily.

On NTFS, a VOB or view storage directory's ownership (its owner and primary group ID) is determined from the security descriptor on the directory root. On FAT file systems, this information is stored in the file identity.sd in the storage directory root. (For compatibility, the identity.sd file is also created on NTFS file systems). On both FAT and NTFS, the file groups.sd holds the supplementary VOB group list.

VOB and View Storage Directory ACLs

Rational 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, sources.vbs, created by user **NT_WEST\ccase_adm**, whose primary group is **user**. The ClearCase administrators group is named **clearcase**.

Note: As annotated in the example, ClearCase LT uses the built-in **LocalSystem** identity (`NT_AUTHORITY\SYSTEM`) wherever ClearCase uses the ClearCase administrators group. On a ClearCase LT server, the name `NT_AUTHORITY\SYSTEM` would appear where the name **clearcase** appears in this example. In both ClearCase

and ClearCase LT, the first ACL entry (`NT_AUTHORITY\NETWORK`) is present only on VOB storage directories, not on view storage directories, and is never present when the VOB storage directory is on a NAS device.

cacls c:\vobstore\sources.vbs

```
NT AUTHORITY\NETWORK:(OI)(CI)(DENY)(special access:)  (on VOB storage only)
              DELETE
              FILE_WRITE_DATA
              FILE_APPEND_DATA
              FILE_WRITE_EA
              FILE_WRITE_ATTRIBUTES

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 LT uses the built-in identity NT AUTHORITY\SYSTEM)
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 (ClearCase LT uses the built-in identity NT AUTHORITY\SYSTEM)
BUILTIN\Administrators:(OI)(CI)F
```

Preserving NTFS ACLs When Copying a VOB or View Storage Directory

Whenever you copy a VOB or view storage directory to another location in an NTFS volume, or to another NTFS volume, you must use a copy program that preserves the storage directory ACLs.

- In most cases, we recommend using the **ccopy** utility (*ccase-home-dir*\etc\utils\ccopy) when copying a VOB or view storage directory. Although **ccopy** copies all of the necessary ACL information, it does not copy the full security descriptor of an object, and therefore effectively grants the user who runs it full access to the copied directory. If someone other than the VOB or view owner is copying a VOB or view storage directory, it may be more appropriate to use **xcopy /o** or, on Windows NT, **scopy** instead.
- On Windows 2000 or Windows XP, you can use **xcopy** with the **/o** option.

  Because Windows 2000 and Windows XP allow a directory to inherit ACLs from its parent directory, the copied directory may inherit some entries from the ACL of its parent directory, which can cause problems with VOB and view access.
- On Windows NT, you can use the **scopy** command, from the Windows NT Resource Kit, with the **/o** option.

  Note: When using **xcopy** or **scopy**, you must supply any additional options required to copy subdirectories, and you must be logged on as a user with rights to create the directory and set the ACLs on the target host. The Administrators and Backup Operators groups typically have these rights.

After you copy the VOB or view storage directory, run **fix_prot –l**, as described in *Utilities for Fixing Protection Problems* on page 258, to check the ACLs on the target directory. If **fix_prot –l** shows any errors, follow the procedures in *Fixing Protection Problems* on page 261 to correct them.

Causes of Protection Problems

This section describes typical scenarios that can lead to incorrect storage protections and suggests how to avoid these situations.

Copying the Storage Directory Incorrectly

If you do not use the appropriate copy utility, as described in *Preserving NTFS ACLs When Copying a VOB or View Storage Directory* on page 257, when copying a VOB or view storage directory, the ACLs on the copy will not be correct. For information about fixing protection problems caused by copying the directory, see *Fixing Protection Problems* on page 261.

Converting the File System to NTFS

If you create the storage directory on a disk partition formatted as FAT or FAT32 and later convert that partition to NTFS, the storage directory protection will become 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 correcting storage directory protection, see *Fixing Protection Problems* on page 261.

Editing Permissions

If you edit a file permission, for example, by using Windows Explorer or **cacls**, ClearCase users will begin having access and protection problems.

Caution: Never click **OK** in the Windows Explorer **File Permissions** dialog box if the changes will affect VOB or view storage. Doing so changes the order of access control entries even if you have not changed and file permissions.

To detect VOB storage directory protection problems, use **checkvob**:

cleartool checkvob –protections –pool *vob-stg-pname*

where *vob-stg-pname* is pathname of the VOB storage directory. For information about fixing protection problems caused by editing file permissions, see *Fixing Protection Problems* on page 261. For more information, see the **checkvob** reference page.

Utilities for Fixing Protection Problems

ClearCase includes three utility programs for finding and fixing VOB and view storage directory protection problems:

- **vob_sidwalk** fixes storage directory protections for schema version 54 VOBs. It can also be used to change ownership on VOB objects. For more information, see the **vob_sidwalk** reference page.
- **fix_prot** fixes storage directory protections for views and also for schema version 53 VOBs.
- **lsacl** (Windows only) displays NTFS ACLs for file system objects

All of these utilities are installed in the *ccase-home-dir*/etc/utils directory.

fix_prot

fix_prot [**–f**·**orce**] { **–root** [**–r**·**ecurse**] [**–recover** {**–chown** *user* | **–chgrp** *group* } | **–replace**·**_server_process_group** | [**–r**·**ecurse**] [**–type** { **d** | **f** }]
[**–chown** *user*] [**–chgrp** *group*] [**–chmod** *permissions*] } *pname* …

Options

–f·orce
Do not prompt for confirmation.

–r·ecurse
Recursively fix protections.

–root
Specifies that *pname* is a storage directory root.

–type
Specifies the type of file system objects to fix. Use **d** for directories and **f** for files. If **–type** is not specified, **fix_prot** operates on both directories and files.

–chown
Specifies the new owner. Mandatory with **–root**.

–chgrp
Specifies the new primary group. Mandatory with **–root**.

–chmod
Specifies the new access rights: `owner`, `group`, `other` (world). Both symbolic and numeric codes are valid, such as `go-x` (symbolic) or `0644` (absolute).

–recover
(Windows only) Restores correct file system ACLs in a VOB or view storage directory based on the information in the identity.sd and groups.sd files. Not applicable to schema version 54 VOBs, for which **vob_sidwalk –recover_filesystem** performs this function.

–replace·_server_process_group
(Windows only) Replaces ACL entries for the ClearCase administrators group. Use this option if you have changed the ClearCase administrators group name or have moved a view to a new domain that has a different SID for this group.

Examples

- To create a UNIX .identity directory after moving a VOB from Windows to UNIX. The VOB has been moved to the storage directory /vobstg/winvob.vbs. The new owner is **vobadm** and the new group is **ccusers**. You must run this command as **root**.

  *ccase-home-dir***/etc/utils/fix_prot –root –recurse –chown vobadm –chgrp ccusers \ /vobstg/winvob.vbs**

- To repair ACLs damaged when a view storage directory was copied to NTFS directory C:\ClearCaseStorage\int.vws using a copy utility that did not preserve ACLs:

*ccase-home-dir***etc\utils\fix_prot –root –recover C:\ClearCaseStorage\int.vws**

- To display the protection modes ClearCase associates with the directory E:\vobstg\sources.vbs:

  *ccase-home-dir***etc\utils\fix_prot E:\vobstg\sources.vbs**

  ```
  drwxr-xr-x   MYDOMAIN\me    MYDOMAIN\mygroup
  E:\vobstg\sources.vbs
  ```

Note: fix_prot may return an error message that begins with the following text:

```
fix_prot: Error: unknown style protections on directory: The data is
invalid.
```

If it does, you must rerun **fix_prot** and specify all of the **–chown**, **–chgrp**, and **–chmod** options and also supply an absolute *permissions* value, either in numeric or *ID=rights* (**o=rwx**) form to **–chmod**. For example:

*ccase-home-dir***etc\utils\fix_prot –chown** *owner* **–chgrp** *group* **–chmod 0666** *pname*

lsacl

lsacl [**–s** | **–l**] [**–n**] { [**–f** *path-name*] | [**–r** *registry-value-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.

–f
: Reads a security descriptor from a file; allows you to display the contents of the identity.sd and groups.sd files.

–r
: (Windows only) Reads a security descriptor from a Windows registry value. Specify *registry-value-name i*n the form *RootKey**ValueName* where:

  RootKey is one of **HKLM**, **HKCU**, **HKCR**, **HKU**, **HKCC**. **HKLM** is assumed if *RootKey* is omitted.

ValueName is the full pathname of a subkey of the specified RootKey

Note: You can also use the Windows **cacls** utility to display an NTFS ACL, but **cacls** cannot read a security descriptor from a *.sd file.

Fixing Protection Problems

The following sections describe how to fix the protection problems described in *Causes of Protection Problems* on page 257.

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 *stg-pname*, 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: NT_WEST\bob (User) (non-defaulted)            (Owner)
Group: NT_WEST\usersnt (Group) (non-defaulted)       (Primary group)
ACL (revision 2):
0: allowed
SID: NT_WEST\user (Group)                            (Supplementary group)
rights (00000000)

1: allowed
SID: NT_WEST\tester (Group)                          (Supplementary group)
rights (00000000)

===== stg-pname\groups.sd
Owner: NT_WEST\bob (User) (non-defaulted)            (Owner)
Group: NT_WEST\usersnt (Group) (non-defaulted)       (Primary group)
ACL (revision 2):
Empty ACL: all access denied                         (No supplementary
                                                     group)
```

3. Run **fix_prot –root** to remove the supplementary group list.

   *ccase-home-dir***etc\utils\fix_prot –r –root –chown** *owner* **–chgrp** *group* ^
   stg-pname

   If you are fixing view storage, you are finished.
4. If you are fixing VOB storage, log on as the VOB owner and continue.
5. If the VOB has a supplementary group list, run this command:

   cleartool protectvob –add_group *group-name[,...] vob-stg-pname*

6 To remove the cleartext containers, run this command:

scrubber –e –k cltxt *vob-stg-pname*

This step must precede Step 7 because **checkvob** cannot fix cleartext containers.

7 To fix the storage pool's protections, run this command:

cleartool checkvob –force –fix –protections –pool *vob-stg-pname*

Problems with Existing Views After Relocating Elements

Although **relocate** (see *Relocating Elements to Another VOB* on page 141) creates symbolic links to preserve a VOB's historical namespace, you may need to make additional adjustments after relocating elements so that existing views and tools can access them.

Cleanup Guidelines

To clean up after relocating file or directory elements:

- **Use the view that was used for the relocate operation.** When adjusting config specs, modifying build rules, and so on, use the same view (or a view with the same config spec), that you used when running **relocate**.

  In the source VOB, **relocate** completes its operation by checking in the parent directories of all relocated elements. Because this directory version does not include symbolic links to relocated elements, working in a view that selects this version allows you to discover which tools, build rules, or other processes will fail when they cannot access relocated elements.

- **Check important and historical views to see whether symbolic links work.** As described in *Problems with Symbolic Links to Relocated Elements* on page 263, the inherent limitations of VOB symbolic links may make it necessary for you to modify the links or the config specs of views that reference them. Some kinds of config specs are more 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 the one on which the **relocate** operation was performed. **relocate** creates symbolic links only for versions on the branch that was selected by the view in which it ran.

- **Fix broken views.** After you find the problem config specs, use one of the techniques listed in *Modifying Old Target Directory Versions to See Relocated Elements* to make relocated elements visible again.
- **Recover any stranded view-private files.** Run **recoverview –sync** on each view used to access the source VOB. Any stranded view-private files and DOs are moved to the view's **lost+found** directory.

  Note: cleartool lsprivate lists stranded objects by their object IDs (OIDs).

Problems with Symbolic Links to Relocated Elements

VOB symbolic links provide a powerful mechanism for linking VOB objects, but they have some limitations that may affect views that access elements through the symbolic links left behind by **relocate**:

- 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 about a VOB symbolic link object, not 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 to the element from the source VOB are broken.
- Broken symbolic links—those with nonexistent targets—are not visible to most UNIX commands. This includes links that have accurate pathname information but point to elements not selected by the current view. The **cleartool ls** command lists these objects.

There are several ways to address these limitations:

- Change the view's config spec to reference elements at their actual pathnames in 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.

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.

Caution: 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 an event history. To use these commands, you must be the VOB owner or privileged user. 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 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. Site-specific factors such as the contents of the NIS host map, VOB mounting and naming conventions on UNIX, as well as drive assignment, disk sharing, and naming conventions on Windows can sometimes lead to an incorrect guess.

To help identify the sources and targets for symbolic links, **relocate** connects a symbolic link object to the target element object with a hyperlink of type **HyperSlink**. Use the **cleartool describe** command on the symbolic link to display information about this hyperlink, which can help to repair it if necessary.

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

When **relocate** checks in a new version of the destination directory, only that version catalogs the relocated elements. Now, consider this scenario:

- You have a view that selects a previous version of the target directory (\new@@\main\2, for example).
- You want the view to be able to see the newly relocated element dir1 at its new location, \new\dir1, rather than its old one, \bigdir\dir1.

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

1. Log on as a 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

   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"
   ```

Modifying Newest Version of Source Directory to See Relocated Elements

In some **relocate** scenarios, 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 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"
```

Locating MVFS Data Containers

Note: This section does not apply to ClearCase LT.

This section demonstrates how to use standard operating system utilities and the **mvfsstorage** command to find where MVFS files are physically stored. Locating this storage may help resolve file access problems in dynamic views.

Scenario

This section 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 e-mail message to disk.

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-99.20:40:30 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-99.20:17:04 by (allison.mon@phobos)
  element type: file

View private file "ralph_msg"
  Modified: Wednesday 02/03/99 21:39:49
```

Determining the Full Pathnames of Files

Use standard operating system commands (**ls** and **pwd** on UNIX; **dir**, **attrib**, or Windows Explorer) to show the full pathname of a file:

ls –l ‘pwd‘/cmd.c

```
-rw-rw-r--   1 allison  mon     211 Feb  2 12:03 /proj/monet/src/cmd.c
```

attrib cmd.c

```
C:\proj\monet\src\cmd.c
```

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 specific VOB in the ClearCase Administration Console, where you can manage VOB storage.

Where Is the View?

The **pwv** and **lsview** commands show the location of the view storage:

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 resides.

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 specific view in the ClearCase Administration Console, where you can manage view storage.

Where Are the Individual Files?

The data containers for all MVFS files are stored within a VOB or view storage directory, as shown in Table 10.

Table 10 Storage Directories for MVFS Files

Kind of file	Storage directory
Version (checked-in)	VOB source storage pool (s subdirectory of the VOB storage directory).

Table 10 Storage Directories for MVFS Files

Kind of file	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 *ccase-home-dir*/etc/mvfsstorage utility locates the physical storage for an MVFS file.

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 places 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
```

Element types that store each version in a separate container do not use the cleartext pool. Instead, programs access the data container in the source pool.

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 Remote Pools on UNIX

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 the UNIX **ls** command to determine whether a view's private storage area (subdirectory .s) is local:

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 VOB and file system link and directory objects as follows:

- For a link, **mvfsstorage** indicates the storage directory 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** displays the directory pathname.

Preventing Recursive Traversal of the UNIX View Root Directory

Note: This section does not apply to ClearCase LT.

On any UNIX host where the MVFS is installed, the /view directory functions as the mount point for the MVFS namespace. Because this /view directory causes the root directory to contain itself recursively, commands that traverse the entire directory, starting at the system root, loop infinitely. Many UNIX systems run a daily **cron** script that cleans up the file system. If the script uses a UNIX **find /** command, it runs the risk of looping infinitely.

Note: Any command that operates on the entire UNIX file system, beginning at /, may loop in this way, regardless of whether it is run from the command line, a script, or a GUI, and regardless of whether it is run by **root** or another user.

The ClearCase installation script analyzes the **crontab** file of the **root** user and modifies entries in this file to prevent the recursive traversal problem. If it cannot complete this modification, it displays a warning message.

After installation, verify the correctness of these changes, and then modify the **crontab** entries of other users as needed to remove any command that operates on the entire file system beginning at the system root. Detailed information on platform-specific options is in the online *Platforms 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.

Improving Client Host Performance

15

This chapter presents techniques for improving ClearCase performance on the client host. Administrative responsibilities related to client host performance include the following:

- Establishing standards for client host configurations (memory, processing power, network interface characteristics, and local storage).
- Understanding MVFS cache sizes and how to adjust them.
- Understanding view caches and how to adjust them.
- Establishing sitewide defaults for cache sizes.

Note: Most of the performance tuning procedures described in this chapter deal with the MVFS. They do not apply to ClearCase LT clients, or to ClearCase clients that run only snapshot or Web views.

Client Host Configuration Guidelines

ClearCase client software makes demands on host memory, CPU, and storage subsystems that are comparable to the demands made by similar workstation applications. Any computer configured to support such applications will deliver good performance on the majority of ClearCase client tasks.

Because Rational 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 has a negative impact on the responsiveness 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, needs 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.

Examining and Adjusting MVFS Cache Size

ClearCase hosts that use dynamic views may benefit from MVFS cache tuning. Default MVFS cache sizes scale automatically based on host memory and are appropriate for a wide range of client needs, but you may want to change default MVFS cache sizes to improve performance in specific cases.

Note: All cache tuning requires an intelligent trade-off between specific performance benefits that may result from larger caches and the potential for degraded application performance due to the dedication of more physical memory to caches. Cache tuning can help reallocate the MVFS's use of available memory in ways that reflect a particular host's patterns of use, but it cannot compensate for inadequate physical memory.

To examine MVFS cache sizes and utilization, use the **cleartool getcache –mvfs** command. There are two ways to resize MVFS caches:

- The easiest (and recommended) way is to adjust the MVFS scaling factor, an integer value from which most other MVFS cache sizes are derived. Use **cleartool setcache –mvfs –persistent –scalefactor** to adjust the scaling factor on UNIX and Windows (you must use the **–persistent** option when setting the scaling factor). On Windows, you can also use the ClearCase program in Control Panel. The **MVFS Performance** tab allows you to set the scaling factor and several other cache sizes.
- You may also adjust selected values for individual caches by using **setcache –mvfs** or the **MVFS Performance** tab of the ClearCase program in Control Panel. This method provides finer control, although it requires extensive analysis and testing, and can actually degrade performance if not done correctly. We do not recommend method unless there are unusual circumstances that make values derived from the scaling factor unsuitable.

Note: Changes you make with **setcache –mvfs** are normally temporary and revert to their default values when the MVFS is restarted. **setcache** includes a **–persistent** option that enables changes you make in the scaling factor or individual caches to persist across restarts. Changes made on the **MVFS Performance** tab are always persistent.

Examining MVFS Cache Statistics

Use the **cleartool getcache** command to display information about MVFS cache statistics on a host. **getcache –mvfs** presents information about file and directory handle caches and also about attribute caches. File and directory handle cache statistics are presented in the form *current-number-on-list*/*list-capacity* and (*percentage-of-list-capacity*). Attribute cache statistics show each caches's size and total cache misses since the last MVFS restart. 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%)

Current MVFS cache enable settings:
Attribute cache:                enabled
Close-to-open revalidation:     enabled
Name cache:                     enabled
Readlink cache:                 enabled
Root version cache:             enabled
VOB freelist low-water mark:  1620
Cleartext freelist low-water mark:1768
Readdir block cache size:       4
MVFS scaling factor:            1

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%)
```

Note: You may also need to use the **mvfsstat** (see *mvfsstat* on page 284) and **mvfstime** commands to determine the effectiveness of a cache before manipulating its size.

Adjusting the MVFS Scaling Factor

The default MVFS scaling factor is set automatically, as shown in Table 11, based on available physical memory installed in the host.

Table 11 How Memory Size Affects the MVFS Scaling Factor

Memory size (MB)	<24	24-512	512-2048	>2048
Scaling Factor	0	1	2	*size*/1024

In Table 11, the value of *size* excludes memory required by the operating system kernel, and for *size* values greater than 2048 MB, is rounded down to the nearest multiple of 1024 MB. For example, if a host has 4096 MB (4 GB) of physical memory and the kernel consumes 100 MB, the scaling factor is set at 3.

You can use **setcache –mvfs –persistent –scalefactor** to override the default scaling factor setting. On Windows, you can also use the ClearCase program in 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**.

Increasing the scaling factor by one requires about 500 KB of additional kernel memory (200 KB nonpageable, 300 KB pageable).

Note: If you use **setcache** to request more kernel memory than is currently available, the request will fail with a `not enough space` error. If this happens, you must request a smaller cache size or else terminate unnecessary processes that are consuming kernel memory.

Changing the value of the scaling factor scales all of the MVFS cache sizes proportionally as shown in Table 12.

Table 12 How the MVFS Scaling Factor Affects Individual MVFS Cache Sizes

MVFS cache name	scaling factor = 0	scaling factor = 1	scaling factor = 2	scaling factor = *f* (*f* > 2)
DNC File Cache	800 bytes	1600 bytes	2400 bytes	800(*f*+1) bytes
DNC Directory Cache	200 bytes	400 bytes	600 bytes	200(*f*+1) bytes
DNC ENOENT Cache	800	1600	2400	800(*f*+1)
Cleartext Freelist Max (UNIX)	900	1800	2700	900(*f*+1)
Cleartext Freelist Max (Windows)	900	1800	1800	1800
RPC Handle Cache	5	10	15	5(*f*+1), $f \leq 2$ 10(*f*+1), $f > 2$
mnode Freelist Max	901	1802	2703	0.22 *f*
mnode Cache	4096	8192	12,288	4096(*f*+1), f≤4; 2048(*f*+7), *f*>4
readdir Block Cache	2	4	6	2(*f*+1), $f < 2$ 6, $f \geq 2$

Setting Individual Caching Parameters

Although we recommend using the scaling factor to make MVFS cache adjustments, you can also change the sizes of individual caches. Before you take this approach, there are several considerations that you should take into account:

Note: All cache tuning involves making intelligent trade-offs that optimize caching of frequently-referenced objects without adversely affecting overall performance. Effective MVFS cache tuning for a given application requires extensive analysis and testing, as well as a detailed knowledge of how that application uses the MVFS.

- It is possible to adversely affect MVFS performance by changing individual caching parameters.
- Using the scaling factor to change cache sizes greatly simplifies cache tuning and is effective in the majority of cases.
- On multiprocessor systems, caches must sometimes be locked. Larger caches incur greater locking overhead, which has an impact on performance

Adjust cache sizes gradually and check the cache utilization with **cleartool getcache –mvfs**. If you change the MVFS cache configuration on a host, also consider reconfiguring each view on that host to increase its cache size. See *Reconfiguring a View* on page 281.

Table 13 describes each adjustable caching parameter and provides recommendations for setting its size. It also shows the relationship between the fields in **getcache** output and various **setcache** options.

Table 13 MVFS Cache Information (Part 1 of 3)

getcache output field name	Description	Adjustment mechanism:	
		setcache option	Control Panel value (Windows)
`Mnode freelist`	An mnode is a data structure used by the MVFS to represent a file or directory. The scaling factor determines the total number of mnodes allocated when the MVFS starts. An mnode on the VOB free list was used recently but is currently idle. By keeping more mnodes on the free list, the time needed to reopen the associated file is reduced.	**–vobfreemax**	**Maximum number of mnodes to keep on the VOB free list**

Table 13 MVFS Cache Information (Part 2 of 3)

getcache output field name	Description	Adjustment mechanism:	
		setcache option	Control Panel value (Windows)
`VOB freelist low-water mark`	Minimum number of mnodes to keep on the VOB free list. When the number of mnodes on the VOB free list reaches the specified maximum, the number of mnodes on the list is reduced to this value	**–vobfreemin**	**Minimum number of mnodes to keep on the VOB free list**
`Cltxt freelist`	The cleartext free list is a collection of pointers to file objects in a storage pool. Caching these pointers speeds reopening of an MVFS-resident file, but consumes additional resources. (On a Windows computer, these resources include open file descriptors on a network connection if the storage is on a remote computer.)	**–cvpfreemax**	**Maximum number of mnodes to keep on the cleartext free list**
`Cleartext freelist low-water mark`	Minimum number of mnodes to keep on the cleartext free list. When the number of mnodes on the cleartext free list reaches the specified maximum, the number of mnodes on the list is reduced to this value.	**–cvpfreemin**	**Minimum number of mnodes to keep on the cleartext free list**
`DNC Files`	The DNC file and directory caches hold pointers to recently accessed files and directories. The ENOENT cache holds the names of files and directories for which a recent lookup failed. When presented with a file or directory name, the MVFS first looks in these caches. If it cannot find a matching entry, it must make an RPC to the view server. Each cache entry consumes approximately 100 bytes. The directory cache usually has the greatest impact on performance.	**–regdnc**	**Maximum number of entries for files**
`DNC Directories`		**–dirdnc**	**Maximum number of entries for directories**
`DNC ENOENT`		**–noentdnc**	**Maximum number of entries for non-existent names**

Table 13 MVFS Cache Information (Part 3 of 3)

getcache output field name	Description	Adjustment mechanism:	
		setcache option	Control Panel value (Windows)
`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 when you perform large builds with many simultaneously active processes, you may want to increase this cache size to reduce the time required to perform an RPC. After an RPC handle is added to this cache, it remains available until the MVFS is restarted. 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**	**Maximum number of rpc handles**
`Readdir block cache size`	The readdir block cache is used to cache directory contents. If you routinely work with very large directories, you may want to increase the size of this cache.	**–readdir_blocks**	**Maximum readdir cache blocks per directory**

Minimizing Attribute Cache Misses

The MVFS caches attributes of file and directory objects to avoid having to look them up in the VOB. A full attribute cache is not necessarily ineffective if its hit rate is high enough. We recommend maintaining a hit rate of 85% or better. The sections that follow describe each cache-miss category and suggest ways to reduce the miss count.

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 checks with the **view_server** when a view-private file is reopened and refreshes cached file attributes if the file was modified by another MVFS client.

This behavior can be disabled on a per-host (not a per-view) basis. If you are certain that a host will not access view-private files in views on any other host, you can disable close-to-open consistency checking on that host to reduce these cache misses.

Use **setcache –mvfs –ncto** to disable close-to-open consistency checking. Use **setcache -mvfs –cto** to re-enable it. (Use the **–persistent** option if you want the change to persist across reboots.)

Generation Misses

Generation misses occur during a parallel build. If you are doing parallel builds, you cannot avoid these misses.

Cache Time-out Misses

Cache time-out misses occur when a file's attributes have not been verified recently. The MVFS periodically revalidates a file's attributes to ensure that it does not cache stale data.

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 between 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

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*

  You must be a privileged user to do this.

- In the registry (either at VOB creation with **mkvob** or later by using **mktag**). For example:

  cleartool mktag –vob –tag *vob-tag* **–replace –options** *mount-options* . . .

Note: The time-out values specified in these mount options affect the view's metadata latency (the delay before changes to VOB metadata become visible in a dynamic view other than the one in which the changes were made). Longer time-out values improve performance at the expense of greater latency. Shorter time out values decrease latency, but also have an impact on view performance because the caches must be refreshed more frequently.

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 running **setcache –mvfs –vobfreemax**.

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 a file name mapped to a particular file object, but the attribute cache on that object needs to be refreshed because of a time-out or a parallel-build-induced generation miss).

These misses cannot be completely eliminated, but as with cache time-out misses, they can be reduced by adjusting minimum/maximum cache lifetimes.

View Caches

To improve its performance, the **view_server** process associated with a view maintains a cache. The default size of this cache is adequate for typical views, but a larger cache can further improve the performance of views in which very large software systems are built by **omake** or **clearmake**. The view server maintains the following caches, consisting mostly of VOB data, to respond faster to RPCs from clients:

- An object cache, which facilitates retrieval of objects such as versions and branches
- A directory cache, which facilitates file system directory read access
- A stat cache, which stores file attributes
- A name cache, which stores names and accelerates name lookups

When a **view_server** process is started, it chooses its cache size from the first one of these that yields a value:

- A per-view persistent value set when the view was created (**cleartool mkview –cachesize**) or modified with the **cleartool setcache –view –cachesize** command
- A per-host persistent value set with **setcache –view –host**
- A site-wide default set with **setcache –view –site**
- A default value: 512 KB on 32-bit platforms, 1 MB on 64-bit platforms

The view cache size represents the total number of bytes available for allocation among the individual view caches.

For information about setting the cache size for a view and for a site, see the **setcache** reference page. For information about displaying cache information, see the **getcache** reference page.

Obtaining View Cache Information

To examine a view's cache information, use **cleartool getcache –view**. The view server prints information about the view cache sizes and hit rates. For example:

cleartool getcache –view r5_integration

```
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
```

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

The view cache includes these types of subcaches:

- **Lookup**. Stores data used to accelerate mapping names to objects in the view.
- **Readdir.** Stores data used to accelerate read-directory operations (for example, **ls** or **dir**) by the view.
- **Fstat.** 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.

Note: On UNIX computers, you can force a **view_server** process to write cumulative cache-performance (and other) statistics to its log file, /var/adm/rational/clearcase/log/view_log and then reset all the statistics accumulators to zero, by running the following command on the host where the view server executes:

kill –HUP *view_server-process-ID*

Analyzing the Output

getcache –view 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 an anomalous, 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*.

Reconfiguring a View

Use **setcache –view** 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.
- Larger cache sizes generally improve view performance, though as cache sizes approach 4 MB, the performance improvement usually becomes less noticeable.
- 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:

Reference Information

This section provides reference information on the **mvfsstat** and **mvfscache** utilities. These utilities are intended primarily to help Rational Customer Support diagnose problems with the MVFS.

No locks or mastership restrictions apply for either of these utilities. However, certain options require privileged user status.

mvfscache

mvfscache manipulates a host's MVFS caches. Nearly all of its options are redundant with similar options provided by **setcache –mvfs** or **getcache –mvfs**. We recommend that you use those commands whenever possible.

Synopsis

- Determine cache status:

  mvfscache [*cache_name*]

- Control cache operation:

  mvfscache { **–e** *cache_list* | **–d** *cache_list* | **–f** *cache_list* }

- Display name cache contents:

  mvfscache –p [**–n** *name*] [**–v** *dbid*] [**–i**]

Options and Arguments

DETERMINING CACHE STATUS. With no options or arguments, **mvfscache** displays the enabled/disabled status of all MVFS caches. If you don't use any of the options, but specify a cache name as an argument, **mvfscache** does not display any output; it returns an appropriate exit status:

0 specified cache is enabled

1 specified cache is disabled

CONTROLLING CACHE OPERATION. Use one of the following options to control a cache, a set of caches, or cache-related behavior.

–e *cache-list*

(Requires privileged user status) Enables the specified caches and (with **cto**) cache-related behavior. The *cache-list* must be comma-separated, with no white space, and can include one or more of the following keywords.

attr Attribute cache. Caches information about recently accessed file system objects.

name Name cache. Caches name lookup translations for recently accessed files and directories.

noent Name-not-found cache. Caches names of files and directories recently looked up but not found.

slink Symbolic link text cache. Caches the contents of recently accessed symbolic links.

rvc VOB root version cache. Caches VOB mount point data for each dynamic view.

cto (cache-related behavior) Close-to-open consistency behavior. Forces a"get file info"-type operation to the view server on every operating-system **open** operation.

Disabling this behavior may boost performance if **mvfsstat** or **mvfstime** shows heavy **cto** activity and the user is not sharing views. However, disabling this behavior may result in consistency loss.

–d *cache-list*

(Requires privileged user status.) Disables the specified caches and cache-related behavior. The syntax is the same as for **–e**.

–f *cache-list*

Flushes the specified caches. Use this option only under direction from Rational Support. The *cache-list* can include any of the following keywords (comma-separated, no white space).

mnode mnode freelist cache. Flushes the `attr` and `slink` caches, open freelist files, and mnode storage for all freelist mnodes.

name Name cache. (Also flushes name-not-found entries.)

rvc VOB root version cache.

lcred Global credentials cache for cleartext lookup permissions.

DISPLAYING NAME CACHE CONTENTS. Use **–p** by itself or with one or more of **–n**, **–v**, and **–i**. The name cache contains the name lookup translations for recently accessed files and directories. The first line of a name lookup translation has this form:

VOB-tag *view*:*directory-dbid* *name* ==> *view*:*lookup-dbid*

–p
Prints the contents of the name cache.

–n *name*
Prints only the entries in the name cache that match *name*.

–v *dbid*
Prints only the entries in the name cache that match *directory-dbid* (database-ID for the directory in which *name* is found) or *lookup-dbid* (database-ID for the result of the lookup).

–i
Includes invalidated name cache entries in the output. These are entries that have been marked as bad and are not used in lookups, but are retained for statistical purposes. This helps determine how often invalid entries are replaced with new data. Invalidations usually happen when a change in the VOB requires the MVFS to refresh its cache.

Examples

- Determine the status of all caches.

  mvfscache
  ```
  Attr: on
  Name: on
  Noent: on
  Rvc: on
  Slink: on
  Cto: on
  ```

- Clear busy mount points, to prepare for unmounting VOBs.

  mvfscache –f mnode

mvfsstat

The **mvfsstat** command displays MVFS use and operating statistics, including cumulative statistics on MVFS cache use, RPC statistics, cleartext I/O counts, vnode

operation counts, and VFS operation counts. This data is useful for evaluating file system performance and determining whether MVFS cache sizes require adjustment.

Synopsis

- UNIX:

  mvfsstat [**–icrvVhalzAdF**] [**–o** *outfile*] [*time*] [*count*]

- Windows:

  mvfsstat [**–iIcrvVhalzAdF**] [**–o** *outfile*] [*time*] [*count*]

MVFS Cache Statistics

The **–c** option reports on the use of the host's MVFS caches. This report is cumulative, covering the entire period since the MVFS was started. The following sections describe the particular statistics that are useful in tuning MVFS performance on a ClearCase client host.

Directory Name Lookup Cache (dnlc)

The `dnlc` section reports on use of a name-lookup cache that maps pathnames to ClearCase identifiers. Note that the value precedes the keyword. For example, `1301984 dot` means that the reported value of the `dot` statistic is 1301984.

Cache Hits. The `hit` line reports on the number of times an entry type was found in the cache (hit):

`dot`	Number of times the current working directory was looked up (always a cache hit)
`dir`	Number of times a directory object entry was found in the cache
`reg`	Number of times a file object entry was found in the cache
`noent`	Number of times a cached `File not found` (`ENOENT`) entry was found

This cache has low hit rates (around 50%) for activities that walk a large tree—for example, a **find** command, or a recursive **clearmake** that examines many files and determines that nothing needs to be built.

Cache Misses. The `miss` line reports on total cache misses. The `events` value is the number of cache misses that occurred because of a significant VOB event, a time-out of the entry, or vnode recycling. Cache misses can occur because there was no entry in the cache. The total number of cache misses equals the `events` value plus the number of misses occurring because there was no entry in the cache.

Cache Additions. The `add` line reports on cache misses that occurred because a new entry was being added to the cache. The additions are categorized as directory entries (`dir`), file entries (`reg`), and ENOENT entries (`noent`).

Attribute Cache. The `attr` section reports on use of a cache of **stat** returns (UNIX) or object status inquiry records (Windows). This cache generally has hit rates comparable to those for the directory name lookup cache.

Options and Arguments

time

Time in seconds between samples. Display deltas on each sample. If you omit this option, only the absolute values of all information are printed.

count

Number of samples. If omitted, defaults to “infinite”.

–o *outfile*

Writes the output to *outfile*.

–i

Displays cleartext I/O counts and wait times.

–I

Displays count and wait times for Windows I/O Request Packets that the MVFS has processed.

–c

Displays statistics for the MVFS caches (see *MVFS Cache Statistics* on page 285).

–r

Displays MVFS remote procedure call (RPC) statistics. These statistics include both counts and real-time waited. Real-time waited may be greater than 100% of a sample period in two cases:

- When an operation takes longer to complete than the sample period; for example, 60 seconds of wait time is recorded in a 30-second sample.
- Multiple processes are waiting at the same time.

In general, real-time percentages are meaningful only when a single process is accessing a VOB.

–v

Displays counts of **vnode** operations.

–V

Displays counts of **vfs** operations.

–h

Displays an RPC histogram. Cleartext fetch RPCs are tallied separately from all other RPCs.

–a

Displays auditing statistics.

–l

Adds more detail to the statistics generated by **–c**, **–r**, **–i**, **–I**, **–v**, and/or **–V**, by providing a breakdown by individual operations. With **–c**, also provides per-cache-entry hit ratios.

–z

(Requires privileged user status.) Resets all running counters to zero.

–A

Displays all statistics.

–d

With **–c** or **–A**, displays additional debugging information for use in diagnosing problems. Use this option only under direction from Rational Customer Support.

–F

Displays statistics for mnode operations and the directory name lookup cache. Use this option only under direction from Rational Customer Support.

Improving VOB Host Performance

16

Nearly all ClearCase operations involve access to data in a VOB. Good VOB host performance ensures that VOB access does not become a bottleneck. This chapter presents techniques for improving the performance of VOB hosts.

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 than can write to it with the same level of performance. For information about the characteristics of a good VOB host, see *VOB Server Host Configuration Guidelines* on page 93.

Minimize Process Overhead

The most effective measures for ensuring good performance from VOB hosts are also the easiest to implement:

- **Keep nonessential processes off the VOB host.** Don't use the VOB host as a server host for another resource-intensive application (for example, a database management system or a Web server) or for a critical network service such as an NIS server on UNIX or a primary domain controller on Windows.
- **Keep ClearCase client processes off the VOB host.** Do not use the VOB server host as an active ClearCase client (a developer's desktop system, for example).
- **Keep view_server processes off the VOB host.** Although you may create shared views on VOB server hosts, the practice can be detrimental to both VOB and view performance. Avoid it where practical.

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 on UNIX hosts, dedicate two disk spindles per VOB: one for the database and source pools, 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).
- Place VOB storage on a certified NAS device.

Consult your system documentation for details about how to use disk striping and mirroring

Add Memory for Disk Caching on Windows

Windows has 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 buffer cache is written 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 can degrade ClearCase performance significantly and produce symptoms like these:

- Extended periods required for **scrubber** and **vob_scrubber** execution
- Very slow **omake** or **clearmake** builds
- Clients experience RPC time-out errors

For more information about the relationship of main memory to VOB size, see *VOB Server Host Configuration Guidelines* on page 93.

Tune Block Buffer Caches on UNIX

Most of the UNIX platforms that ClearCase supports have a dynamic block buffer cache. As much main memory as possible is used to cache file blocks that have been updated by user processes. Periodically, this cache is written to disk. On most UNIX variants, the cache size increases when you add more main memory to the host. For information on architecture-dependent block buffer cache operation, see the online *Platforms Guide*.

Block Buffer Cache Statistics

The UNIX **sar** 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 online *Platforms Guide*.

Note: Interactive performance suffers temporarily when the block buffer cache is written (flushed) to disk. Some UNIX platforms provide user-level control over this event (for example, the HP-UX **syncer** utility), which you may be able to use to minimize the effects of buffer cache writes on interactive performance. The larger the block buffer cache, the less frequently it should be flushed.

Modify Lock Manager Startup Options

Every VOB server host runs a single lock manager process, which controls database access for every VOB on the host. The lock manager runs with default startup options, which are intended to deliver good performance under a wide range of loads. However, you may want to experiment with changing certain lock manager startup options on VOB server hosts that support many VOBs or many users.

Lock Manager Implementations

There are two implementations of the lock manager: one reads and writes data by using a socket; the other, which is available on certain UNIX platforms, uses shared memory, which results in lower latency for each request. The range of values accepted by certain lock manager startup options is implementation dependent. You can tell which type of lock manager is on a system by examining its startup message in the lockmgr_log file. The conventional lock manager prints the following lines:

```
db_VISTA Version 3.20
Database Lock Manager for UNIX BSD
```

The shared memory lock manager prints the following lines:

```
db_VISTA Version 3.20
Database Lock Manager for UNIX System V shared memory
```

To Change Lock Manager Startup Options

Use the following procedures to change lock manager startup options on Windows or UNIX hosts.

Note: After you change any lock manager startup option, you must stop and restart ClearCase on the host before the change takes effect.

On Windows Hosts

On Windows hosts, lock manager startup options are defined internally. To change these defaults, create the following Windows registry key as a **REG_SZ** value and supply the appropriate *num* values as described in *lockmgr Reference Information* on page 293:

HKEY_LOCAL_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\LockMgrCmdLine: REG_SZ : –a almd –f *num* **–u** *num* **–q** *num*

This registry key is preserved when a new version of ClearCase is installed.

On UNIX Hosts

On UNIX hosts, default lock manager startup options are defined in the startup script *ccase-home-dir*/etc/clearcase. To change these values, create a file named /var/adm/rational/clearcase/config/lockmgr.conf that includes a line in the following format:

LOCKMGR_OPTS = –f *num* **–u** *num* **–q** *num*

Supply the appropriate *num* values as described in *lockmgr Reference Information* on page 293. We recommend that this file be owned by **root** and have read-only access.

This file is preserved when a new version of ClearCase is installed. When the file exists, ClearCase starts the lock manager with the options defined in the file. Otherwise, it uses the default values for lock manager startup options.

The following syntactic rules apply to /var/adm/rational/clearcase/config/lockmgr.conf:

- A comment begins with a **#** character and extends to the end of the line.
- White space may precede the **LOCKMGR_OPTS** label and may precede and/or follow the **=** character.
- White space must precede each option (**-f**, **–u**, **–q**) and may separate the option from the specified *num* value.

- Options that are not specified revert to default values. Illegal options are ignored.

lockmgr Reference Information

The **lockmgr** arbitrates all access to each VOB database on a VOB server host. It is started automatically when ClearCase starts on any VOB host.

Synopsis

lockmgr –a almd –f *num* **–u** *num* **–q** *num*

Options and Arguments

SPECIFYING THE SOCKET OR SHARED MEMORY FILE NAME. Default: almd. (You must not change this default.) On UNIX systems, the **lockmgr** communicates with other processes through /var/adm/rational/clearcase/almd, which is a shared-memory file on some platforms and a socket on others (see *Lock Manager Implementations* on page 291).

–a almd

Specifies almd as the leaf name of the socket or shared memory file created by the **lockmgr**. Using any other name is not supported.

Note: To reduce chances of accidental deletion, /var/adm/rational/clearcase/almd is owned by **root**. On platforms that support the shared memory lock manager, this file is created with mode 777 (world writable). If you are concerned that a user may accidentally modify this file, you may restrict write permission to **root** and the owners of all VOBs on the host. Any VOB owned by a user without write permission to /var/adm/rational/clearcase/almd will be inaccessible.

SPECIFYING THE NUMBER OF DATABASE FILES. Default on Windows: 128. Default on UNIX: 256

–f *num*

Specifies the number of database files that can be open concurrently. Each VOB database consists of seven database files. The default startup values allow the lock manager to handle a maximum of 36 VOBs on a UNIX host and 18 on a Windows host. If there are more VOBs on the host, the lock manager will not be able to service requests for all of the VOBs concurrently. User response times will degrade, and the db_server_log or vobrpc_server_log files will include messages of the form:

```
Error: Too many open databases on host (try increasing the -f
argument on lockmgr command line).
```

When this happens, you can either move some of the VOBs to another host, or increase the value of the **–f** option to 7**V* where *V* represents the number of VOBs on the host.

SPECIFYING THE NUMBER OF USERS. Default on Windows: 128. Default on UNIX: 256

–u *num*

Specifies the number of **db_server** or **vobrpc_server** processes that can concurrently request locks. Each active view requires one **vobrpc_server** process for each VOB that the view accesses. In addition, various operations that change VOB metadata cause a **db_server** process to access the VOB through the lock manager. Poor user response time and messages of the form:

```
db_VISTA database -922: lockmgr is busy
```

in the lockmgr_log file may indicate that the value of the **–u** option should be raised.

On hosts that do not support the shared-memory lock manager, the **–u** option cannot be set higher than 1018. On hosts that support the shared-memory lock manager, the **–u** option is bounded only by available virtual memory.

You can compute a very approximate worst case value for **–u** by using the formula:

$V*(N/4 + 5)$

where *V* is the number of VOBs on the host, and *N* is the number of users who access those VOBs. For a more realistic value—one that does not cause the lock manager to consume unnecessary virtual memory on the VOB server host—monitor the total number of **vob_server** and **vobrpc_server** processes running on the VOB server host for an extended period of typical use (perhaps a week or two). Then multiply the peak value by a factor that will accommodate growth (two, or perhaps a little more).

SPECIFYING THE SIZE OF THE REQUEST QUEUE. Default on Windows: 128. Default on UNIX: 1024.

–q *num*

Specifies the number of lock requests for locks to be queued. The lock manager delays queuing lock requests in excess of this value. Poor user response time and messages of the form

```
db_VISTA database -922: lockmgr is busy
```

in the **db_server_log** or **vobrpc_server_log** files (and, often, concurrently in a **view_log** file) may indicate that the value of the **–q** option should be raised. As a rule, this value should be no greater that five times the value of the **–u** option.

Server Performance and NAS Devices

When a VOB or view server uses a NAS device for some or all of its VOB or view storage (see *ClearCase and Network-Attached Storage Devices* on page 15), we recommend that you eliminate or minimize any other uses of the device.

If the NAS device is used by other applications or provides storage for files and directories that are not part of VOB or view storage, the additional load these uses create on the device's I/O subsystem can have a significant negative impact on ClearCase operations that access the VOB and view storage directories. This impact is especially severe when the whole VOB (database and pools) is stored on the NAS device.

If a VOB server must share a NAS device with other uses, we recommend keeping the VOB database on the server itself and locating just the pools on the NAS device. (This configuration is supported only for hosts running UNIX.)

If you experience significant performance degradation when accessing VOBs whose databases are stored on a NAS device, we recommend that you move the VOB database to the VOB server host.

Note: Because of the additional network traffic required by the CIFS (SMB) protocol, VOB server hosts running Windows are the most likely to suffer degraded performance when they locate the VOB database on a NAS device.

Configuring Cross-Platform File System Access

A

Rational supports the use of several third-party products with ClearCase to enable Windows computers to access the file systems of UNIX computers. This capability is required when dynamic views on Windows access VOBs or dynamic views on UNIX. For more information, see *Cross-Platform File System Access* on page 57. Two types of products are supported:

- NFS client products, which run on Windows clients and use the NFS protocol to access UNIX file systems.
- SMB server products, which run on UNIX servers and use the SMB (or CIFS) protocol to provide Windows clients with access to the server's file system.

This appendix explains how to configure currently supported third-party file system access products. For the most recent information on supported products and versions, see the Rational customer Web site.

Note: This appendix does not apply to ClearCase LT, which does not need cross-platform file system access because it does not support dynamic views.

NFS Client Products

Rational supports using these NFS client products to provide Windows computers with access to VOBs and dynamic views on UNIX:

- Shaffer Solutions DiskAccess
- Hummingbird NFS Maestro

If you use an NFS client product, you must:

- Install it on each Windows client that will access VOBs or views on UNIX servers.
- Configure each installation to meet the special requirements of ClearCase.

The rest of this section describes configuration procedures specific to using an NFS client product with ClearCase.

Note: An NFS client product supports access to UNIX file systems by a single Windows host. A Windows computer that uses an NFS client product to access VOBs on UNIX cannot provide other Windows computers with access to those VOBs.

Disabling Automatic Case Conversion

Some NFS client products change the capitalization of file names by default, typically by converting to lowercase. This can cause problems for the MVFS and **cleartool** commands (see *Case-Sensitivity* on page 61), so you must disable case conversion, as described in this section.

Note: Typically, you can use a command-line option to disable case conversion for a particular NFS mount. However, ClearCase automatically mounts remote storage directories. For correct behavior on these mounts, configure NFS to disable case conversion for all mounts.

Shaffer Solutions DiskAccess

To disable automatic case conversion:

1. Click **Start > Settings > Control Panel**. Start the DiskAccess program.
2. On the **Filenames** tab, click **Preserve Case (no conversion)**.

Hummingbird NFS Maestro

To disable automatic case conversion in version 8.0:

1. Click **Start > Settings > Control Panel**. Start the Network program.
2. On the **Services** tab, select **Hummingbird NFS Maestro Client**.
3. Click **Properties** to open the client configuration dialog box.
4. On the **File Access** tab, Under **Filename Case**, select **Preserve Case**.

To disable automatic case conversion in earlier versions:

1. Click **Start > Settings > Control Panel**. Start 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 view on a UNIX host, configure your NFS client with a default protection that grants group write access. Without this permission, other developers cannot modify view-private files created by you.

Shaffer Solutions DiskAccess

To set the default protection:

1. Click **Start > Settings > Control Panel**. Start the DiskAccess program.
2. On the **File Access** tab, verify that the access specified for **User** is `RWX`, **Group** is `RWX`, and **Other** is `RX`.

Hummingbird NFS Maestro

To set the default protection in version 8.0:

1. Click **Start > Settings > Control Panel**. Start the Network program.
2. On the **Services** tab, select **Hummingbird NFS Maestro Client**.
3. Click **Properties** to open the client configuration dialog box.
4. On the **File Access** tab, for both **Default File Permissions** and **Default Directory Permissions**, set the permissions as follows:

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

To set the default protection in earlier versions:

1. Click **Start > Settings > Control Panel**. Start 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**, set 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, you must configure the NFS client to log on using UNIX user and group names that match your Windows user and primary group names. To verify that these matching names exist, pass the name of a UNIX NFS server to *ccase-home-dir*\etc\utils\credmap. In this example, user **NT_WEST\akp** who is a member of the domain group **clearusers** has matching user (**akp**) and group (**clearusers**) accounts on the UNIX Host **saturn**:

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

```
Identity on local system:
       User: NT_WEST\akp
(NT:S-1-3-21-108034363-97353062-1565875335-1402)
       Primary group: NT_WEST\clearusers
(NT:S-1-3-21-108034363-981762062-1565875335-1044)
.
.
.
Identity on host "saturn":
       User SID: UNIX:UID-2270
       Primary group SID: UNIX:GID-88
.
.
.
```

If the user or group ID returned by **credmap** has a negative value, the target UNIX host cannot find a matching UNIX user or group name for the Windows user who is currently logged on. For more information, see *Common User and Group Names* on page 55.

Shaffer Solutions DiskAccess

To set your logon user name:

1. Click **Start > Settings > Control Panel**. Start the DiskAccess 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 dynamic views. When using Hummingbird NFS Maestro to access a dynamic view on UNIX, you must disable this mode. If you do not, you may encounter errors when attempting to open MVFS files. For example:

```
ZwOpenFile returned status 0xc0000043
```

This error indicates a sharing violation.

To disable DOS sharing in version 8.0:

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

2 On the **Services** tab, select **Hummingbird NFS Maestro Client**.

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

4 On the **File Access** tab, Under **File Sharing**, select **None**.

To disable DOS Sharing in earlier versions:

1 Click **Start > Settings > Control Panel**. Start 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 Links**, clear the **DOS-Style Sharing** check box.

Automounting and NFS Client Software

When you mount a VOB or start a dynamic view on UNIX, ClearCase must access the VOB or view storage directory on a UNIX file system. 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.

We recommend that you disable automounting when using any supported NFS client product. This allows the NFS client to use UNC names to access VOB and view storage on UNIX. If you install DiskAccess before you install ClearCase, the ClearCase installation procedure disables automounting for you.

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

1 Click **Start > Settings > Control Panel**. Start the ClearCase program.

2 On the **Options** tab, clear the **Enable automatic mounting of NFS storage directories** check box.

3 Click **OK**.

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.

DiskAccess: Supporting the ClearCase Server Process User

DiskAccess must be configured to allow logins by the ClearCase server process user. After you install DiskAccess on a computer and establish a DiskAccess log-on for yourself, use the following procedure to configure an additional DiskAccess log-on for the ClearCase server process user account on UNIX.

Note: This procedure requires a UNIX account for the ClearCase server process user. See *Common User and Group Names* on page 55 for more information.

1. Log on to the Windows host as the ClearCase server process user.
2. Click **Start > Settings > Control Panel**. Start the DiskAccess program.
3. On the **Authentication** tab, specify the UNIX user name and password for the ClearCase server process user (the name and password that you used in Step 1 of this procedure).
4. Click **OK**.
5. When you exit the program, the confirmation dialog box should show that you have been authenticated on UNIX as the ClearCase server process user. If there is an error or if the confirmation dialog box shows a UID of -1 and GID of -2, you did not supply a valid name and password in Step 3.

Saving the User Identity in a Windows Registry File

After you have established a DiskAccess log-on for the ClearCase server process user, you can generate a Windows registry key file that allows other users to add this support without having to log on as the ClearCase server process user.

1. Run the **creds** utility to display the user SID of the ClearCase server process user (**clearcase_albd** in this example). You will need this SID in Step 3.

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

   ```
   Windows NT user info (on local system):
   .
   .
   SID S-1-5-21-103034363-981818062-1465874335-1064
   .
   ```

2. In a Windows Registry Editor, navigate to the key:

   HKEY_LOCAL_MACHINE\SOFTWARE**Intergraph\DiskAccess\CurrentVersion\Users**

3 Select the subkey that matches the SID of the **clearcase_albd** user as returned by **creds**.

4 Save the key to a registry file by clicking **Registry > Export Registry File** in the Registry Editor. When this file is executed on any Windows computer that has DiskAccess installed, it creates the Windows registry key that enables DiskAccess log-ons for the ClearCase server process user on UNIX.

Windows Tags for UNIX VOBs with Remote Storage Pools

Note: This section applies only when Windows hosts use an NFS Client product to access VOBs with symbolically linked storage pools; if you use a supported SMB server product to provide Windows clients with access to VOBs on UNIX, skip this section.

If a VOB on a UNIX host includes one or more remote storage pools (see *Creating Remote Storage Pools on UNIX Hosts* on page 113), Windows hosts that use an NFS client product cannot access the VOB unless it has a tag in the Windows region that includes a split pool map, which provides a network path to each of the remote pools.

To check a VOB for remote pools, type a command similar to this one on a UNIX computer:

cleartool lspool –long –invob /vobs/libvob

```
 ...
pool "libvob"
 ...
  pool storage link target pathname "/net/gamma/pools/libvob.1"
  pool storage global pathname "/net/io/vb_store/libvob/s/sdft"
 ...
```

If the output includes one or more `pool storage link ...` lines, use one of the following procedures when you create a tag for it in a region established for Windows hosts that use an NFS client product to access VOBs on UNIX.

To use 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 defines two remote pools:

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

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

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

To use the **mktag** command, specify the **–poolmap** option as shown in this example:

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

This example illustrates several important rules for composing the command line:

- A **poolmap** string commonly specifies multiple pools. Use vertical bars (**|**) to separate individual pool specifications. Precede 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.

Poolmap Syntax

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

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

Where:

- *symlink-source* specifies the symbolic link to the remote pool, relative to the VOB storage directory.
- *symlink-target* specifies the full pathname, in UNC format, of the linked pool. The pool must reside in a UNIX directory that has been mounted by an NFS client product running on the local Windows computer. The pathname must be valid on all computers in the region.

Mapping Storage Pools in an Existing VOB Tag

If you inadvertently create a tag in a Windows region for a VOB with remote pools but neglect to specify the poolmap, Windows hosts cannot access the VOB until you modify or replace the tag.

The easiest way to do this is to remove the tag using the ClearCase Administration Console or the **rmtag** command, and import it again as described in *To Create Tags in a New Registry Region* on page 47.

Alternatively, you can use the ClearCase Administration Console to modify or replace the tag with one that includes a split pool map:

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 administrators 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 *Poolmap Syntax* on page 304.

You can also use the **cleartool rmtag** and **mktag** commands to remove the old tag and create a new one that has the same name and a split pool map.

SMB Server Products

Rational supports two SMB server products that make UNIX file systems accessible from Windows computers: Samba, available from www.samba.org, and Syntax TotalNET Advanced Server (TAS) from LSI Logic Storage Systems Inc. This section describes how to install and configure them.

Some versions of Samba and TAS are not supported for use with ClearCase. The *Release Notes* for Rational ClearCase include the most up-to-date information about SMB server products, including information on supported versions and platforms.

Installing and Configuring Samba

Samba can be downloaded from www.samba.org. Download it and follow the installation instructions for the operating system on which you are installing it. Samba must be installed and configured on each UNIX VOB and view server that you want to access from Windows.

To configure Samba for use by ClearCase:

1 Create a Samba username map for the ClearCase server process user.

2 Configure Samba globals.

3 Create shares for VOB and view storage.

4 Start Samba services.

Mapping the ClearCase Server Process User

Samba requires a username map that associates the Windows account for the ClearCase server process user with an appropriate UNIX user account. (For more information about this account, see *Common User and Group Names* on page 55.) The examples in this section assume that the name of the ClearCase server process user account on Windows is **clearcase_albd**.

To create the Samba username map, use any text editor to create a file named username.map on the host where Samba is installed. We recommend that you create the file in the same directory where other Samba configuration files (such as smb.conf) are installed.

The file must contain a line of the form

```
account = clearcase_albd
```

where `account` is the name of an existing UNIX user account that meets the criteria listed in *UNIX Credentials for the ClearCase Server Process User* on page 56. For more information about the username.map file, see the Samba documentation.

Using the Samba Web Administration Tool (SWAT)

The examples in this section describe the configuration of Samba through the use of the Samba Web Administration Tool (SWAT), which is included in the Samba download. Instructions included with the download explain how to enable this tool.

To access the SWAT interface:

1. In a Web browser, enter a URL of this format:

   http://*computer:port#*

   where *computer* is the host name of a UNIX VOB or view host on which you have installed Samba and *port#* represents the SWAT port number. (The default value is 901.)

2. Log on as **root.** The SWAT interface appears in your browser.

Configuring Samba Globals for ClearCase

Click the **GLOBALS** icon at the top of the SWAT interface's home page. Then click **Advanced View**. Set the global options as described in Table 14.

Table 14 Samba Global Settings for ClearCase

Base options	
workgroup	Set to the name of the Windows domain to which ClearCase hosts accessing this server belong
netbios name	Set to the host name of this computer
Security options	
security	**DOMAIN** (recommended) or **USER**. See the Samba documentation for **smbpasswd** for more information about Samba security options.
encrypt passwords	**Yes**
create mask	**0775**
directory mask	**0775**
username map	Set to the local pathname of the username.map file
Locking options	
oplocks	**No**
kernel oplocks	**No**
File-name handling options	
case sensitive	**No**
preserve case	**Yes**

ClearCase has no special requirements for other Samba globals; you may configure them in any way that is appropriate for your site.

Creating Shares for VOB and View Storage

You must create one or more Samba shares to hold server storage locations or individual VOB or view storage directories. To create a Samba share:

1. Click the **SHARES** icon at the top of the SWAT interface's home page.
2. Enter a name for the share in the text box to the right of the **Create Share** button. To simplify administration, we recommend that the share name be similar to that of the UNIX directory whose name you enter in Step 4.

3 Click **Create Share**.

4 Edit the path option under **Base Options**. Set its value to be a directory under which the VOB or view storage areas reside. The VOB or view storage areas do not need to be in the directory specified, but they must be somewhere below the specified directory.

5 Click **Commit Changes**.

Starting Samba Services

The Samba **smbd** and **nmbd** services must be running before Windows computers can access files using Samba. We recommend that you configure your UNIX host to start the **smbd** and **nmbd** services at boot time. Platform-specific instructions for configuring automatic service startup are included in the Samba documentation.

You can also start Samba services manually from the SWAT interface:

1 Click the **STATUS** icon at the top of the SWAT interface's home page.

2 Click **Start smbd**. The page refreshes and should display the **smbd** status as `running`.

3 Click **Start nmbd**. The page refreshes and should display the **nmbd** status as `running`.

Configuring ClearCase to Support Samba

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

1 Click **Start > Settings > Control Panel**. Start the ClearCase program.

2 On the **MVFS Performance** tab:

- Select **Override** for both **Maximum number of mnodes to keep on the VOB free list** and **Maximum number of mnodes to keep for cleartext free list**.
- If you are using a 64-bit build of Samba 2.2.6 or later, set the value for both to 800.
- If you are using any other version of Samba, set the value for both to 200.

3 Click **OK** to apply the changes and close the dialog box.

4 Shut down and restart Windows.

Testing the Samba Configuration on Non-ClearCase Files

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

1. Create a directory on your Samba server (for example, /testshare/testdir) and a test file in that directory (for example, /testshare/testdir/testfile).
2. Create a Samba share using **testshare** as the share name and /testshare as the path name for the share.
3. On a Windows client, create a file in the Samba share. Then verify that the UNIX user and group ownership and permissions for that file are correct.
4. Verify that all Windows clients can access the Samba share, including testing permission and access restrictions.

Testing the Samba Configuration with ClearCase

To verify that ClearCase and Samba are working together properly:

1. On a UNIX VOB or view server, install and configure Samba as described in this chapter, creating shares for VOB and/or view storage.
2. Verify that your ClearCase user and group assignments are appropriate, as described in *Fundamentals of VOB and View Access Control* on page 69.
3. Verify that you can access VOBs and views on the server from a UNIX client.
4. Log on to a ClearCase client on Windows. Use the Region Synchronizer to import VOB tags and view tags for VOBs and views hosted on the UNIX server into the Windows region.
5. Verify that you can use these views and VOBs by performing some basic ClearCase operations (for example, **mkelem**, **checkin**, and **checkout**) in them.

Installing and Configuring TAS

ClearCase supports the TotalNET Advanced Server (TAS) SMB server product from LSI Logic Storage Systems Inc. to provide Windows computers using dynamic views with access to VOBs and views on several UNIX platforms.

This section describes how to install TAS and configure TAS and ClearCase to support cross-platform file access. If you use TAS, you must install and configure it on each UNIX VOB and view server that you want to access from a Windows client.

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 that requires access from Windows.

Note: Rational supports both TAS 6.*x* and 7.*x* for use with ClearCase. TAS 7.*x* includes a simplified configuration GUI, described in *Configuring TAS 7.x to Support ClearCase* on page 314.

Enabling the Multiuser Kernel Driver on AIX

If you are installing TotalNET Advanced Server on an AIX platform, you must enable the multiuser kernel driver after installing TAS. This step provides support for the TAS SMB multiplexor, which is required when using ClearCase with TAS on AIX.

To enable the multiuser kernel driver, use the TAS **smbmxenable** command. To disable the multiuser kernel driver, use the TAS **smbmxdisable** command.

Note: You cannot enable or disable the multiuser support from the Framework interface. You must use the command line. For details about multiuser support on AIX platforms, see the *TAS Administration Manual*.

Accessing the Syntax Administration Framework

Use the Syntax Administration Framework Web interface to configure and administer TAS. To access the Framework:

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

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

   where

   - *computer* is the host name of a UNIX VOB or view host on which you have installed TAS
   - *port#* represents the Framework port number (the default is 7777)

   The Syntax Enterprise Services page appears.
2. Click **Syntax Administration Framework**; a Framework logon program appears.
3. Log on as **root**. The Framework interface now appears in your browser.
4. Click **TAS Configuration and Administration** in the sphere frame. The TAS configuration and administration menu appears in the menu frame.

Performing Initial Setup of TAS

Note: If you are upgrading an existing installation of TAS, the upgrade procedures preserve the previous configuration, including existing TAS volumes and file services that support ClearCase, so you can skip the remaining sections of this chapter. After you have upgraded, verify that opportunistic locks are disabled for each TAS volume that contains ClearCase storage. (The **Support opportunistic locks** check box in the volume definition must not be selected.) For details, see the *TAS Administration Manual*.

The first time you install TAS on a server, you must perform an initial setup on that TAS installation, as described in the *TAS Administration Manual*. Click **Initial Setup** in the menu frame of the Framework Web interface, and follow the instructions in the TAS documentation, subject to the changes noted in these sections that are specific to use of TAS with ClearCase.

For more information about any of the topics related to configuring TAS, see the *TAS Administration Manual.*

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 (TAS 6.*x*)

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

Configure the CIFS realm as follows:

- **Server name** — Type the name of the server, if it is not already the default.
- **Workgroup** — Type the name of the Windows domain to which your ClearCase hosts and users belong.
- **Transports** — Select the protocols appropriate for your site.
- **Device for NetBEUI** — Accept the default.
- **WINS Server(s)** — If you use proxy server authentication mode for CIFS file services, you may have to specify the IP addresses of the WINS servers for the network on which the authentication proxy server resides.

Mapping the ClearCase Server Process User (TAS 6.*x*)

TAS requires a username map that associates the Windows account for the ClearCase server process user with an appropriate UNIX user account. (See *Common User and*

Group Names on page 55 for more information about this account.) This section assumes that the ClearCase server process user account on Windows is named **clearcase_albd**.

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 an existing UNIX user account and click **Create**. This account must meet the criteria listed in *UNIX Credentials for the ClearCase Server Process User* on page 56
   - In **List of client accounts**, type **clearcase_albd**.

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

Creating a Volume (TAS 6.*x*)

Create a TAS volume that exports the directory in which 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 ordinary files before using TAS to access VOBs and views.

To create a TAS volume for use by ClearCase:

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 immediate 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.

Configuring the File Service (TAS 6.*x*)

To configure the TAS file service to support ClearCase:

1. Access the file service:
   a. Click **CIFS (NB) 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.

   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 Options**; the **Authentication Options** form appears. Under **User-mode authentication options**, click **Local** or **Remote**.

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

For information about the authentication mode for your site, see your system administrator.

4 If you select **Remote** authentication, configure the authentication as follows:

- **Proxies**—Click **Proxies** and type the name of the proxy servers in this text box, one per line.

  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 (TAS 6.x)* on page 311.)

- **Use Username map** — Select this check box to ensure that the file service references the **clearcase_albd** username map specified in *Mapping the ClearCase Server Process User (TAS 6.x)* on page 311.

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

- **Use Secure Passwords** — Select this check box.

  Note: If you select **Local authentication**, you must enter every user who will access TAS file services in a local password encryption database on the server that supports 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.

- **Use Username map** — Select this check box to ensure that the file service references the **clearcase_albd** username map specified in *Mapping the ClearCase Server Process User (TAS 6.x)* on page 311.

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

Configuring TAS 7.*x* to Support ClearCase

TAS 7.*x* includes a Framework interface devoted to the configuration of a ClearCase service. The **Configure ClearCase Service** page presents all TAS configuration steps that are specific to ClearCase on a single screen and establishes values for the following TAS service attributes:

- A TAS service (new or existing) to be used by ClearCase
- UNIX and Windows users with permissions to update ClearCase information
- A volume for the storage of VOB data
- A volume for the storage of view data (optional)
- An authentication method for the service

After these values are submitted through the **Configure ClearCase Service** screen, a new TAS service is created or an existing service is modified as specified. TAS volumes for VOB and, optionally, view storage are created and at the paths specified in the **VOB Storage Area Path** and **View Storage Area Path** attributes. These volumes are made available as shares with names derived from the terminal leaf of the path. For example, a volume created in /usr2/clearcase/vobdata would be shared as vobdata. If the terminal leaf is not a legal name for a Windows share, the share is named VOB if it is for VOB storage and VIEW if it is for view storage.

In addition to the attributes specified on the **Configure ClearCase Service** page, TAS configures the service with the following attributes:

- **preserve-whitespace** is set to **on**.
- **Browse master**, **null password login**, and **Windows 95** capabilities are disabled.
- The **freespace report** method is set to **root**.
- Opportunistic locks are disabled.
- Filename case is set to **preserve**.
- umask value is set to **002** (as opposed to the file service default umask **007**).
- Username maps are enabled.

Note: If an existing service is used for the ClearCase service, the attribute values listed above override the previously configured values of the existing service.

To configure a TAS service for ClearCase, select **CIFS Realm** from the **TAS Sphere** menu; then click **Configure ClearCase Service** and complete the following steps:

1 Type or select a value for the following attributes, as necessary:

   - **Service Name** — Type the name of the new service or select an existing service to reconfigure from the list.
   - **Unix ClearCase Admin Account** — Type the name of the UNIX user account to which the Windows ClearCase server process user should be mapped. For more information, see *UNIX Credentials for the ClearCase Server Process User* on page 56.
   - **NT ClearCase Admin Account** — Type the name of the Windows ClearCase server process user account. (This manual uses the name **clearcase_albd** for this account).
   - **VOB Storage Area Path** — specify the path of the UNIX directory where VOBs will be (or have been) created. To browse for this location, click **Browse**.
   - **View Storage Area Path** — specify the path of the UNIX directory where views will be (or have been) created. To browse for this location, select the **Browse** button.

     Note: This field is optional. However, a path for a view volume must be specified if shared dynamic views are hosted on this server.

- **Authentication** — Specify the method of user authentication to the service by selecting the **Local**, **Gateway**, or **Proxy** option. If **Proxy** is selected, the proxy authentication server must be specified in the text field next to the option.

  Note: We recommend that you select proxy authentication when using TAS with ClearCase. If you are reconfiguring an existing service, the authentication method of that service appears for the **Authentication** attribute. In addition, if the existing service was configured for proxy authentication, the previously configured proxy servers appear in the text field. If the existing service was configured for Active Directory authentication, the **Local** option is presented as the default on the **Configure ClearCase Service** screen.

2 Click **Submit** to create to reconfigure the TAS service for ClearCase.

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.

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

Configuring ClearCase to Support TAS

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

1 Click **Start > Settings > Control Panel**. Start the ClearCase program.

2 On the **MVFS Performance** tab:

- Select **Override** for both **Maximum number of mnodes to keep on the VOB 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 Shut down and restart Windows.

Testing the TAS Configuration on Ordinary 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 On a Windows 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 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 verify that ClearCase and TAS are working together properly:

1 On a VOB or view server running UNIX, install and configure TAS as described in this chapter, creating volumes containing VOB and/or view storage.

2 Verify that your UNIX user and group name are valid on Windows hosts. (For more information, see *Common User and Group Names* on page 55.)

3 Verify that you can access VOBs and views on the UNIX server from a UNIX client.

4 Log on to a ClearCase client on Windows. Use the Region Synchronizer to import VOB tags and view tags for VOBs and views hosted on the UNIX server into the Windows region.

5 Ensure that you can use these views and VOBs by performing some basic ClearCase operations (for example, **mkelem**, **checkin**, and **checkout**) in them.

ClearCase and Windows Domains

B

Rational ClearCase relies on the operating system to provide the details of user identity and group membership that determine a user's rights to execute operations on versioned artifacts in VOBs and views. On Windows computers, user and group identity are established when the user logs on to a Windows NT or Active Directory domain. This chapter describes what a domain administrator needs to know about the user, group, and resource accounts that ClearCase requires and about the effect of domain trust relationships on ClearCase.

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

Domain Configurations Compatible with ClearCase

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

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

What ClearCase Requires from Any Domain

Both ClearCase and ClearCase LT impose several requirements on any domain environment in which they operate:

- All users who access a common set of VOBs and views must have domain accounts and have at least one group membership in common.

- All client and server hosts must be members of a domain. If the hosts are not members of the same domain in which the user and group accounts are created, they must be members of a domain that trusts that domain.
- Additional steps must be taken to enable users from multiple domains to access a common set of VOBs and views.

ClearCase (but not ClearCase LT) also requires a special domain account that provides a user identity for the **albd_server** process.

Note: If members of a ClearCase or ClearCase LT community access VOBs and views on UNIX hosts, user and group names for their domain accounts must be identical to those in UNIX account database. For more information, see *Common User and Group Names* on page 55.

ClearCase on Nondomain Hosts

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

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

Nondomain systems can function as ClearCase clients in networks in which 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.

Domain User and Group Accounts

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

The ClearCase users group must have the following characteristics:

- It must be a domain global group or an Active Directory universal group. We recommend that you use a domain global group unless the group needs to include other groups.

- If members of the group must access VOBs or views on a UNIX host, the group name must be the same as the name of the ClearCase users group on UNIX. For more information, see *Common User and Group Names* on page 55.

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

In addition to the ClearCase users group, a ClearCase community (but not a ClearCase LT community) requires two additional domain accounts:

- A ClearCase server process user. The **albd_server** program runs with this identity. In examples throughout this document, the ClearCase server process user is named **clearcase_albd**. The ClearCase server process user must be a member of the ClearCase administrators group.
- A ClearCase administrators group. Members of this group can perform all ClearCase operations, including those that permanently destroy data. Membership in this group should be restricted to the ClearCase server process user and a few ClearCase administrators.

Note: The ClearCase server process user and ClearCase administrators group are not required by ClearCase LT, which uses the built-in **LocalSystem** account for server processes and grants any administrator of the ClearCase LT server ClearCase administrative privileges.

The ClearCase server process user and ClearCase administrators group are normally created by the site preparation process if they do not already exist. They can also be created manually, if necessary. See *Defining Required Domain Accounts Manually* on page 322.

Setting the ClearCase Primary Group

Although you can designate a user's primary group by using various Windows domain account maintenance tools, that group name is not always returned when an application requests the name of a user's primary group (unless the user is a member of only one domain group).

ClearCase users who are members of multiple domain groups must set the user environment variable CLEARCASE_PRIMARY_GROUP to the domain-qualified name of the ClearCase users group (for example, **NT_WEST\clearusers**). This guarantees an unambiguous definition of the group that ClearCase considers the user's primary group. ClearCase users who are members of a single domain group do not have to take this step.

Note: CLEARCASE_PRIMARY_GROUP must be a user environment variable. It cannot be a system environment variable.

The value of CLEARCASE_PRIMARY_GROUP is used only when evaluating a user's rights to access a VOB. It has no other security or access-control implications. Users who are members of multiple groups but have not set CLEARCASE_PRIMARY_GROUP correctly are likely to have problems creating elements or otherwise accessing VOBs, especially in complex domain configurations.

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

To verify that CLEARCASE_PRIMARY_GROUP has been properly set:

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

Defining Required Domain Accounts Manually

Note: This section does not apply to ClearCase LT.

The ClearCase Site Preparation Wizard attempts to create domain accounts for the ClearCase administrators group and ClearCase server process user if they do not already exist. The account and group names specified during site preparation are presented as the defaults when users run the ClearCase installation program on individual hosts.

If the user running the ClearCase Site Preparation Wizard does not have Domain Administrator privileges, the wizard cannot create these accounts. A domain administrator must create them by using the following procedure:

1. Log on as a domain administrator.
2. Using the appropriate domain user and group account management tool, 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 ClearCase server processes and administrators.
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

- On each ClearCase host that is configured to support local VOBs and views, give the **clearcase_albd** user the right to **Log on as a service**.

Multiple User Account Domain Support

If your existing domain configuration requires it, you can enable ClearCase access for users and computers in multiple domains.

Using Active Directory Universal Groups

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

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

1. Verify that the Active Directory environment is operating in native mode. (Universal groups cannot be created in an Active Directory domain that is operating in mixed mode.)
2. Create the ClearCase users group as an Active Directory universal group.
3. Make each domain global group whose members are part of the ClearCase community a member of the ClearCase users group. We do not recommend adding individual user accounts to a universal group. Instead, group the users from each Active Directory domain into a domain global group defined in that domain, and make each of those groups a member of the universal group.
4. Require ClearCase users in to set CLEARCASE_PRIMARY_GROUP to the domain-qualified name of the (universal) ClearCase users group. (You cannot use Active Directory account management tools to specify a universal group as a user's primary group.)

Note: If you are upgrading a multimaster Windows NT domain environment to Active Directory, use the procedure in *Converting Proxy Groups* on page 330 to convert the proxy groups to members of an Active Directory universal group.

Using Proxy Groups and Domain Mapping in Windows NT Domains

When a ClearCase community includes users from multiple Windows NT domains, you must enable the ClearCase domain mapping feature as described in this section to provide all users with access to a common set of VOBs. Because this configuration can be complicated to set up and administer, we recommend that you avoid using it unless organizational or security concerns require you to do so.

Note: When users in proxy groups share a dynamic view on Windows, all directory elements accessed in the view must have mode 777 (write permission for all users).

Suppose that ClearCase users have accounts in domains named ATLANTA, BOSTON, and CHICAGO, 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 domain mapping as follows:

1. Ensure that each ClearCase host is a member of a resource domain that trusts the ATLANTA, BOSTON, and CHICAGO domains.
2. Create the ClearCase users group in one of the user account domains. In this example, the domain is ATLANTA and the group is ATLANTA**clearusers**. VOBs to be shared by users taking advantage of domain mapping must be owned by the ATLANTA**clearusers** group.
3. Configure the **albd_server** on every ClearCase host in each of these domains to log on 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 domains:
   - In the BOSTON domain, create the group BOSTON**clearusers_Boston**.
   - In the CHICAGO domain, create the group CHICAGO**clearusers_Chicago**.

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

   ```
   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** has 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 domain 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 CHICAGO members of **CHICAGO\clearusers_Chicago**.

6 Enable domain mapping on each ClearCase host. To do so, edit the Windows registry on that host to make the following changes:

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

   b Click **Edit > Add Value**.

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

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

   e In the DWORD editor, enter **1** (hex) in the **Data** box.

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

7 Require each ClearCase user to set the user environment variable CLEARCASE_PRIMARY_GROUP to the value **ATLANTA\clearusers**. See *Setting the ClearCase Primary Group* on page 321.

Setting VOB Element Permissions

All elements in any VOB that are accessed by users who are members of proxy groups must allow **Read** rights for **Other**. Newly created elements grant this right by default. Use **cleartool describe** to examine an element's protection. Use **cleartool protect** to change an element's protection. You can also use GUIs like the ClearCase Explorer to examine and change protections of elements.

Setting VOB Storage ACLs

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

Conversion to Active Directory

Like any enterprise-scale application in a Windows network, ClearCase is affected when the network is converted from Windows NT domains to Active Directory domains. The remainder of this appendix describes how this conversion affects ClearCase and ClearCase LT, and how to manage users, groups, hosts, and data during and after the conversion.

Note: If you use ClearCase or ClearCase LT in an environment that is already running Active Directory, you can ignore this information.

Understanding Active Directory

Microsoft provides tools and documentation to facilitate conversion of a Windows network from Windows NT domains to Active Directory. This section assumes you have read the applicable documents from Microsoft and are familiar with the terminology they use and the procedures they describe. In particular, it assumes you have read the Microsoft white paper *Planning Migration from Microsoft Windows NT to Microsoft Windows 2000.* (It is distributed as part of the Windows 2000 Support Tools and is also available on Microsoft's Web site.) That document and related documents introduce several key concepts— including *native mode, mixed mode, domain upgrade, domain migration, SID history,* and *cloning* of principals—that are used throughout this section.

How Active Directory Affects ClearCase

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

Conversion to Active Directory affects ClearCase in several ways:

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

represent owners of objects), VOBs must be updated with new SIDs in these scenarios.

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

Planning Your Active Directory Upgrade or Migration Strategy

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

- An upgrade (often referred to as an in-place upgrade), in which a Windows NT domain controller is converted to an Active Directory domain controller operating in mixed or native mode. After an upgrade, all users, groups, and resources have the same SIDs as they had in their original Windows NT domain.
- A migration, in which user, group, and resource accounts migrate (using a process referred to as cloning) from a Windows NT domain to an Active Directory domain. After a migration is complete, all users, groups, and resources have new SIDs. Because a native mode Active Directory maintains information about each principal's current and former SIDs (which Microsoft refers to as the principal's SID history), both types of domains can be used together for as long as needed.

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

Caution: Microsoft supplies tools for converting the SIDs stored in NTFS ACLs. Never use these tools (or any tools that change native file system protection information) on a VOB or view storage directory. Use only ClearCase utilities to convert SIDs in VOB or view storage directories. See *Using vob_sidwalk to Change or Update VOB Users and Groups* on page 335 for details.

Preparing ClearCase Hosts

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

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

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

Note: If you are upgrading a host from Rational ClearCase version 4.2, each view on the host must be reformatted. A view is reformatted automatically the first time it is started on a host that has been upgraded from ClearCase 4.2 to a more recent release. You can also reformat a view manually by using the **reformatview** command.

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

Domain Upgrade Scenarios

This section describes several scenarios in which one or more Windows NT domains are upgraded to Active Directory domains by using the in-place upgrade procedure defined by Microsoft. If your site is not using this procedure, see *Domain Migration Scenarios* on page 330.

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

Upgrading a Single Domain

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

1. Prepare ClearCase hosts as described in *Preparing ClearCase Hosts* on page 327.
2. If you do not have a backup domain controller online during the upgrade, stop ClearCase on all ClearCase server hosts to prevent ClearCase operations during the upgrade process.
3. Use the procedures defined by Microsoft to perform an in-place upgrade of the Windows NT domain to an active directory domain.
4. After the upgrade of the primary domain controller is complete, shut down and restart all ClearCase hosts.

Upgrading a Master Domain and Its Resource Domains

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

To upgrade a master domain and its resource domains:

1. Upgrade the master domain, as described in *Upgrading a Single Domain* on page 328.
2. Upgrade each resource domain, as described in *Upgrading a Single Domain* on page 328. Configure the upgraded resource domain as a child of the upgraded master domain.
3. After the upgrade of a resource domain is complete, shut down and restart all ClearCase hosts in that resource domain.

Upgrading Multiple Master and Resource Domains

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

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

To upgrade multiple master and resource domains:

1. Prepare ClearCase hosts, as described in *Preparing ClearCase Hosts* on page 327.
2. If you do not have a backup domain controller online for each of the master and resource domains during the upgrade, stop ClearCase on all server hosts to prevent ClearCase operations during the upgrade process.
3. Use the procedures defined by Microsoft to perform an in-place upgrade of the first Windows NT master domain to an Active Directory domain. Upgrade the domain in which the ClearCase users group is defined before you upgrade the domains in which the proxy groups are defined.
4. Upgrade the remaining master domains.
5. After the master domains have been upgraded, you can begin to upgrade the resource domains. As long as you do not alter existing trust relationships between domains that have been upgraded and those that have not, you can upgrade the resource domains in any order and on any schedule that is appropriate for your organization. We recommend that you make all upgraded domains members of the same forest, which allows you to use Active Directory universal groups, as described in *Using Active Directory Universal Groups* on page 323.
6. After the upgrade of a resource domain is complete, shut down and restart all ClearCase hosts in that resource domain.

7 After all domains have been upgraded and the Active Directory domain has been converted to native mode, follow the procedure in *Converting Proxy Groups* on page 330 to eliminate proxy groups and domain mapping.

Converting Proxy Groups

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

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

1 Use Active Directory management tools to rename the ClearCase users group in the primary ClearCase domain. For example, the **ATLANTA\clearusers** group created in the procedure in *Using Proxy Groups and Domain Mapping in Windows NT Domains* on page 324 could be renamed to **ATLANTA\clearusers_Atlanta**.

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

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

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

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

Domain Migration Scenarios

Sites for which a domain upgrade is impossible can use a domain migration process. This process uses Microsoft tools to populate a native-mode Active Directory domain

with user, group, and resource accounts that have been cloned from existing accounts in a Windows NT domain. Both types of domain can operate in parallel and, if the appropriate trust relationships exist between the Windows NT and Active Directory domains, users and groups in either type of domain have equivalent ClearCase access rights.

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

- After a ClearCase host has become a member of an Active Directory domain, you must reconfigure the host's **albd_server** to log as the ClearCase server process user account that exists in the Active Directory domain. The name of the account may still be the same, but the user and group SIDs have changed.
- VOB storage directories must be reprotected so that they include the new SIDs of the VOB owner in the directory ACL.
- VOB databases must be updated with the new SIDs. The ClearCase utility program **vob_sidwalk** replaces old SIDs with new ones. For more information, see *Using vob_sidwalk to Change or Update VOB Users and Groups* on page 335 and the **vob_sidwalk** reference page.

Detailed instructions for performing these tasks are included in this section.

Migrating Multiple Domains

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

Note: Before you begin any migration process, prepare all ClearCase hosts as described in *Preparing ClearCase Hosts* on page 327. Even though the hosts may not be migrating right away, they will not be accessible by migrated users and groups until you have taken this step.

Migrating Users and Groups

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

1. Use the procedures defined by Microsoft to migrate users and groups from the Windows NT domains to the Active Directory domains. Be sure to include the ClearCase users group and, if they exist, the ClearCase administrators group and **clearcase_albd** account in the migration.
2. In many migration scenarios, there is a period when users logged on to the Active Directory domain access the same VOBs as users logged on to a Windows NT domain. To ensure access to VOBs by users in either type of domain, do one of the following:
   - Add the domain-qualified name of the ClearCase users group that has been migrated to the Active Directory domain to the VOB's supplemental group list. For example, you can use the **cleartool protectvob** command as shown here to add the **clearusers** group in the Active Directory domain AD-DOMAIN to the group list of the VOB with storage on the VOB server host at **C:\vobstg\srcs.vbs**:

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

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

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

     .
     .
     .

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

     In this case, the user should set CLEARCASE_PRIMARY_GROUP to the value **NT:S-1-5-21-103034363-981818062-1465874335-1064**
3. Because migrated accounts include SID history, user accounts in the Active Directory domain include twice as many group memberships as they had in the Windows NT domain. (Each user's group list includes groups from both domains.) Users who are members of multiple groups in a Windows NT domain and find that their group list includes more than 32 groups after migration should set their CLEARCASE_GROUPS environment variable to include the SID string that represents the ClearCase users group in the Windows NT domain, as well as the name of the ClearCase users group in the Active Directory domain. For example:

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

Step 2 explains how to use **creds** to obtain the SID string. For more information about the CLEARCASE_GROUPS environment variable, see *Limitations When a User Belongs to More Than 32 Groups* on page 70.

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

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

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

Migrating Individual Hosts

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

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

To migrate an individual ClearCase host:

1. Stop ClearCase on the host.
2. Use the procedures defined by Microsoft to migrate the host to the Active Directory domain.
3. For ClearCase hosts only (does apply to ClearCase LT):
   - After you migrate the host, reconfigure its **albd_server** to log on as the (migrated) ClearCase server process user in the Active Directory domain. You can do this by reinstalling ClearCase on the host and specifying the new account, or you can do it manually, as described in *Defining Required Domain Accounts Manually* on page 322.
   - If the host is a VOB server that will be accessed by UNIX clients also, reset the credentials mapping domain in the **Options** tab of the ClearCase program in

Control Panel. The credentials mapping domain must be one in which user and group account names match those of UNIX users that will access VOBs on this server.

4 Restart ClearCase on the host.

5 Reprotect any VOB and view storage on the host by running the **fix_prot** utility on each VOB or view storage directory as shown in this example:

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

where *storage-dir-pname* is the pathname to the VOB or view storage directory.

6 Update VOB databases with the new SIDs that represent the cloned user and group accounts.

a Log on to the VOB server host as the VOB owner or privileged user.

b Lock the VOB for all users except yourself (**–nusers** *you*).

c Replace the old SIDs with the new ones. Run **vob_sidwalk**. In this example, *vob-tag* is the VOB tag of the VOB you are updating and *SIDfile-path* is the name of a file where **vob_sidwalk** will log the changes it makes:

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

For additional details about how **vob_sidwalk** remaps SIDs, see *Using vob_sidwalk to Change or Update VOB Users and Groups* on page 335.

d Unlock the VOB.

Note: If you had been using proxy groups to enable users from multiple domains to access a common set of ClearCase artifacts, we recommend that you use an Active Directory universal group to eliminate the need for proxy groups and domain mapping. Follow the procedure in *Converting Proxy Groups* on page 330

If VOB Servers Cannot Migrate When Clients Do

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

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

2 Lock the VOB for all users except yourself (**–nusers** *you*).

3 Run **vob_siddump** to generate a map file. (You must use **vob_siddump** because you cannot run **vob_sidwalk** from a remote host.) In this example, *vob-tag* is the VOB tag of a VOB on a server that is still in the Windows NT resource domain, and

SIDfile-path is the pathname to the map file that **vob_siddump** generates. (If *SIDfile_path* cannot be created on a drive that is accessible to the VOB server host, you must copy it to the VOB server host before you perform Step 4.)

vob_siddump –sidhistory *vob-tag SIDfile-path*

4 Log on to the VOB server that hosts the VOB whose tag you used in Step 3. While the VOB still locked for all users except yourself, run **vob_sidwalk** to update the SID information stored in the VOB

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

In this example, *mapfile-path* is the map file you generated in Step 3 of this procedure and *SIDfile-path* is the name of a file in which **vob_sidwalk** logs the changes it makes. For more information, see *Using vob_sidwalk to Change or Update VOB Users and Groups* on page 335.

5 Unlock the VOB.

Note: Unless the VOB remains locked from the time you begin Step 3 until the time you complete Step 4, users can create new objects in the VOB between the steps. If they do, you must perform both steps again.

Using vob_sidwalk to Change or Update VOB Users and Groups

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

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

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

This section provides several examples of procedures that use **vob_sidwalk** and **vob_siddump**. For additional examples of procedures that use **vob_sidwalk**, see

Chapter 8. The **vob_sidwalk** reference page provides complete information on all **vob_sidwalk** and **vob_siddump** options.

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

Remapping Historical SIDs After Domain Migration

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

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

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

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

1. Look up a SID to find the account name.
2. Look up the account name found in Step 1 to find its SID.
3. If the SID returned in Step 2 is different from the SID used in Step 1, the SID used in Step 1 is assumed to be a historical SID, and the SID returned in Step 2 is written to the new-SID field of the current line of *SIDfile-path*.

Remapping Current SIDs When Moving a VOB to a New Domain

When you move a VOB to another domain, you must use **vob_sidwalk** to retrieve and store (in the VOB database) the new SIDs for all security principals who own objects in the VOB. The procedure is essentially the same whether you are moving the VOB to a host in another domain or moving a VOB server host to another domain and leaving the VOB on the host. *Moving a VOB to a Different Domain* on page 148 describes the procedure for remapping SIDs as part of a VOB move.

Reassigning Ownership to the VOB Owner

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

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

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

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

Resetting VOB Storage Directory Protections

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

vob_sidwalk –recover_filesystem *vob-tag SIDfile-path*

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

Using -delete_groups with Replicas That Preserve Identities and Permissions

Rational ClearCase MultiSite customers who use identities- and permissions-preserving replicas (created with **mkreplica –preserve**) must take several additional steps when they migrate those replicas' hosts from Windows NT domains to Active Directory.

Because the changes in SIDs made by **vob_sidwalk** are not propagated by replication, you must run **vob_sidwalk** on each identities- and permissions-preserving replica in a replica family when the server that hosts the replica is migrated to Active Directory. When run on such a replica, **vob_sidwalk** preserves the original SIDs on the VOB's group list, so that operations that require container creation continue to succeed whether or not all such replicas in a family have been updated. After all such members of a replica family have been updated, the administrator must run **vob_sidwalk** again using the **–delete_groups** option to remove these historical group SIDs. We recommend removing historical SIDs because a VOB has a limit of 32 groups on its group list. Keeping unused historical SIDs on the list may cause the list to overflow as new groups are added.

Note: This procedure assumes that you have migrated user and group accounts for all users of all replicas to Active Directory and that all users have set their CLEARCASE_PRIMARY_GROUP environment variable to the name of the ClearCase users group in the Active Directory domain.

1. Synchronize all replicas in the family to ensure that each replica includes the same set of user and group SIDs.
2. Follow the procedure in *Migrating Individual Hosts* on page 333 to migrate hosts. All identities- and permissions-preserving replicas in a family must be processed using the same **vob_sidwalk** options. If the **–map** option is used, you can save time by generating one mapping file and using it on all identities- and permissions-preserving replicas in a family.
3. After the replica has been synchronized again with other replicas whose SIDs have been updated, as described in Step 2 of this procedure, run this command

   vob_sidwalk –sid_history *vob-tag SIDfile-path*

   Then examine the resulting SID file to see whether any new SID mappings are needed (because new user or group identities have been added to the replica). If new SID mappings are required, run **vob_sidwalk** again using the options you used in Step 2.
4. After all identities- and permissions-preserving replicas have been updated (Step 2) and the SID file generated (Step 3) shows that no new SID mappings are needed, run **vob_sidwalk –execute –delete_groups** on each replica. This command deletes historical group SIDs from the VOB's group list.

Configuring the Rational Web Platform

C

The Rational Web Platform (RWP) provides server side support for Web interfaces to various Rational products, including Rational ClearCase. RWP is installed with a default configuration, which is suitable for most sites. Some sites may need to modify the RWP configuration after installation to accommodate various host- or site-specific requirements. For example:

- To make RWP use a different HTTP port number
- To change RWP logging defaults
- To configure access to RWP from another Web server acting as a proxy
- To configure RWP to use secure sockets

This appendix explains how to edit the RWP configuration files to make some of the more common changes in the default configuration. It also includes information on configuring the ClearCase Web interface.

The Rational Web Platform includes a Web server based on the Apache HTTP Server version 2.*x* and a servlet engine based on the Tomcat servlet container version 4.*x*. Additional information about the Apache HTTP Server is available at www.apache.org. Additional information about the Tomcat servlet container is available at jakarta.apache.org.

On UNIX, RWP always runs a single instance of the RWP servlet engine. On Windows, RWP creates a second instance of the servlet engine for use if needed.

Note: The Rational Web Platform supports only the Web interfaces to Rational products. Using it to serve other Web applications or content is not supported.

RWP Installation Directory

RWP is normally installed in one of the following directories:

- On Windows, C:\Program Files\Rational\common\rwp
- On UNIX, /opt/rational/common/rwp. If RWP is installed in another location, the installer creates the symbolic link /opt/rational/common/rwp, which points to the RWP installation directory.

The default RWP installation directory can be changed at installation time by supplying a different path when prompted by the install program.

RWP Configuration Files

RWP configuration is specified in several files. The following files are normally installed in the conf subdirectory of the RWP installation directory:

- rwp.conf specifies configuration parameters for the RWP server.
- ssl.conf specifies configuration parameters for secure sockets if they are used by the RWP server.
- server.xml specifies configuration parameters for the RWP servlet engine.
- On Windows, server2.xml specifies configuration parameters for the RWP ReqWeb servlet engine.
- workers.properties specifies configuration parameters for the connections between RWP and RWP servlet engine(s).

On Windows, the following files are normally installed in the bin subdirectory of the RWP installation directory:

- jk_service.properties controls how the RWP servlet engine runs as a Windows service.
- jk_service2.properties controls how the RWP ReqWeb servlet engine runs as a Windows service.

You can edit these files with any text editor. They include explanations of all configuration parameters. This section describes a few of the parameters that you may need to change.

Note: After changing any configuration parameter in any of these files, you must stop and restart RWP before the change takes effect. See *To Stop and Restart RWP* on page 345.

Configuration File Reference Versions

The RWP installation directory includes reference versions of all configuration files.

- On UNIX:
  - rwp.conf.template
  - server.xml.template
- On Windows:

- rwp.in.conf
- server.in.xml
- server2.in.xml

The install program uses these reference versions to determine whether configuration files have been customized. Do not make any changes to them.

To Change the Default RWP HTTP Port

The port on which RWP listens for HTTP requests is defined by the **Listen** parameter in rwp.conf. For example,

```
Listen 80
```

tells RWP to listen on port 80 (the default for HTTP). You may change this to specify any available port number. For example:

```
Listen 8000
```

tells RWP to listen on port 8000.

Note: If you change the RWP HTTP port number to anything other than 80, all URLs that reference RWP must include the port number. For example:

```
http://RWP_host.domain:8000/ccweb
```

To Change the Default RWP Servlet Engine Ports

The ports on which the RWP servlet engine communicates with RWP are defined in the server.xml and server2.xml files as well as the workers.properties file. Table 15 lists default port numbers, port uses, and the files in which the port numbers are defined.

Table 15 Default RWP Servlet Engine Ports

Port number	Description	Location
8009 (8010 on HP-UX)	Used for communication between RWP and the RWP servlet engine	server.xml, workers.properties
8010	Used for communication between RWP and the RWP ReqWeb servlet engine (Windows only).	server2.xml, workers.properties
8005 (8006 on HP-UX)	RWP servlet engine shutdown port	server.xml

If any of these ports is used by another application on the RWP host, we recommend that you reconfigure that application to use different ports. If you cannot, you must change the ports RWP uses.

The following example, from server.xml, defines port 8009 as the port used for internal communication between RWP and the RWP servlet engine:

```
<Connector className="org.apache.ajp.tomcat4.Ajp13Connector"
       port="8009" minProcessors="5" maxProcessors="75"
       acceptCount="10" debug="0"/>
```

To change either port, change the value of the **port** attribute of the appropriate **Connector** element. For example, the **port="8088"** attribute in the line

```
<Connector className="org.apache.ajp.tomcat4.Ajp13Connector"
       port="8088" minProcessors="5" maxProcessors="75"
       acceptCount="10" debug="0"/>
```

causes internal communication between RWP and the servlet engine to use port 8088.

Note: If you change the port attribute of the **Ajp13Connector** element in server.xml, you must also change the port in this line of the workers.properties file:

```
worker.ajp13.port=8009
```

If you change the port attribute of the **Ajp13Connector** element in server2.xml, you must also change the port in the **worker.ajp13_2.port** line of workers.properties. These files exist only on Windows hosts where RWP supports a second instance of the RWP servlet engine.

To Configure RWP Logging

A number of configuration parameters related to access, error, and event logging in rwp.conf are grouped under the heading **Logging-related directives**. You may want to change any of the following:

- **ErrorLog** specifies the name of the file where errors are logged. For example,

  ```
  ErrorLog logs/error.log
  ```

  specifies that errors will be logged in the file logs/error.log under the RWP installation directory.

  Note: Any RWP log file may be piped to the **rotatelogs** command, as described in *Log Rotation and Log Cleanup* on page 343.

- **LogLevel** specifies the type and severity of errors to be logged. For example,

  ```
  LogLevel warn
  ```

  specifies that errors up to and including warnings will be logged. Table 16 lists the various log levels in order of decreasing severity. Specifying any of these values for **LogLevel** logs events of that severity and all lower severities.

Table 16 RWP Log Levels

LogLevel	Messages logged
emerg	Emergency messages about events that may render the server inoperable (Highest severity.)
alert	Conditions that should be corrected immediately
crit	Critical conditions such as hardware or system errors
error	All other errors
warn	Warning messages
notice	Conditions that may require special handling
info	Informational messages (lowest severity)
debug	Debugging RWP

- **LogFormat** specifies the format in which events are logged. You can choose one of the predefined formats (for example, **common**), or you can define your own format. For more information about format tokens and the rules for constructing log file strings, see the documentation for **mod_log_config** at www.apache.org.
- **CustomLog** specifies the name of the file in which RWP access requests are logged. For example,

  ```
  CustomLog logs/access.log common
  ```

  specifies that access requests will be logged in the file logs/access.log under the RWP installation directory in the **common** log file format.

Log Rotation and Log Cleanup

In the default configuration, most RWP log files are piped to the **rotatelogs** program, which periodically creates a new log file. The following line (which has been split into two lines to fit on the page) uses **rotatelogs** to create a new copy of the access.log file every 86,400 seconds (24 hours).

```
CustomLog "|\"/opt/rational/common/rwp/bin/rotatelogs.exe\"
\"/opt/rational/common/rwp/logs/access.log\" 86400" common
```

The log rotation period begins when RWP is started.

The ClearCase Weekly Log Scrubbing job removes all RWP log files that are more than 30 days old. You can modify this job (as described in *The ClearCase Scheduler* on

page 217) to change the frequency with which the job runs, the age of the log files it removes, or any other aspect of its operation.

Note: If you change any of the default RWP log locations, you must also modify the cleanuplogs script so that it looks for these logs in their new location.

To Change the User Account Used by RWP

On installation, RWP is configured to run with the identity of a built-in user account. You can change this account if necessary by using one of the procedures described in this section.

To Change the RWP User Account on Windows

On Windows computers, RWP is started at boot time by the Windows Service Control Manager and runs with the identity of the built-in **LocalSystem** account (**NT AUTHORITY\SYSTEM**).

To change the identity under which RWP runs on Windows:

1. Run the Services application (in **Control Panel > Services** or **Control Panel > Administrative Tools > Services**). RWP includes the following services:
   - Rational Web Platform, HTTP Server
   - Rational Web Platform, servlet engine
   - Rational Web Platform, ReqWeb servlet engine

Caution: The HTTP Server and servlet engine must run as **LocalSystem** on a Windows RWP host that is supporting the ClearCase Web interface. If these services run as any other user, the ClearCase Web interface will fail. The ReqWeb servlet engine can run as another account if necessary.

2. Edit each service's **Log On** properties to specify either a local or domain account.
3. Run **rwp_restart** to stop and restart RWP (see *To Stop and Restart RWP* on page 345).

To Change the RWP User Account on UNIX

On UNIX computers, RWP initially runs as **root** to obtain access to the required ports. It then changes its identity to that of a user with minimal privileges. The name and group of this user vary from platform to platform (for example, on Solaris it is typically **nobody.nobody** and on HP-UX it is **www.other**). To change this account on a UNIX computer:

1 Edit the **User** and **Group** lines in rwp.conf. The following entries configure RWP to run as the user **rwpuser.other**:

User rwpuser
Group other

2 Edit the **su** command line in the RWP startup script rwp_startup (located in the RWP bin directory). Change the specified user (the first parameter to the **su** command) to match the account you used in Step 1. Do not change anything else on the **su** command line:

su rwpuser –c ...

3 Stop and restart RWP.

To Stop and Restart RWP

RWP is normally started at boot time. If you need to stop or restart RWP (for example, to force it to re-read a changed configuration file), use one of the following commands, which are normally installed in the RWP bin directory:

- **rwp_startup** starts RWP if it is not already running.
- **rwp_shutdown** stops RWP and any associated servlet engine processes.
- **rwp_restart** runs the **rwp_shutdown** and **rwp_startup** commands, in that order, to restart RWP.

Note: Commands that stop and start ClearCase do not affect RWP.

To Configure Access to RWP from Another Web Server

Some sites may need to access RWP by proxy or redirection from another Web server. In this configuration, the other Web server redirects specific URLs to an RWP process running on the same host but using a different port, or running on a separate host. Two common use cases require this type of configuration.

- **RWP and another Web server must run on the same host.** We recommend that you install RWP on a host that does not have to run any other Web servers. If this is impossible, we recommend that you configure the other Web server to use ports that are not being used by RWP. If you cannot do this, you must configure RWP to use ports not used by the other Web server (see *To Change the Default RWP HTTP Port* on page 341) and optionally configure the other Web server to redirect URLs for Rational Web clients to RWP.

- **RWP must run behind a firewall.** To restrict access to RWP, a Web server running on the public side of a firewall can be configured to pass specific URLs to an RWP instance running on the other side of the firewall.

Follow the instructions in this section to enable a proxied or redirected configuration that provides access to RWP from either of the following Web servers:

- Apache HTTP Server
- Microsoft Internet Information Server (IIS)

Note: Instructions for configuring proxied or redirected access to a Rational Web application are specific to the application and the Web server acting as the proxy. Only the Web servers and Rational Software products that are specifically cited in this section can be supported in a proxied or redirected configuration.

Configuring mod_proxy Support for Apache

To configure an instance of Apache HTTP Server to support proxy access to RWP, you must configure the Apache HTTP Server with proxy support supplied by the Apache **mod_proxy** module. Detailed information about how to do this is available at www.apache.org. The following is a summary of the steps you will probably need to take:

1 Configure the Apache HTTP Server to load the mod_proxy module and the other modules on which it depends. This typically requires you to uncomment various **LoadModule** directives related to mod_proxy support in the Apache httpd.conf file. For example

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

You also need to uncomment the `ProxyRequests On` directive in the **<IfModule mod_Proxy.c>** block in httpd.conf:

```
<IfModule mod_Proxy.c>
ProxyRequests On
</IfModule>
```

2 Add the appropriate **ProxyPass** and **ProxyPassReverse** directives within the **<IfModule mod_proxy.c>** block in httpd.conf. **ProxyPass** and **ProxyPassReverse** directives are application specific:

For the ClearCase Web interface, add these **ProxyPass** and **ProxyPassReverse** directives:

ProxyPass /ccweb http://*hostname***[:***port***]/ccweb**
ProxyPassReverse /ccweb http://*hostname***[:***port***]/ccweb**

ProxyPass /Java_Plugins http://*hostname***[:***port***]/Java_Plugins**
ProxyPassReverse /Java_Plugins http://*hostname***[:***port***]/Java_Plugins**

Where *hostname* is the name of the RWP server host and *port* is an optional port number, which you must specify if you have changed the default port on which RWP listens for HTTP requests (see *To Change the Default RWP HTTP Port* on page 341). For example, the following directives would configure the proxy server to support access by the ClearCase Web interface to an RWP process listening on port 81 of a host named **RWP_host**.

```
ProxyPass        /ccweb http://RWP_host:81/ccweb
ProxyPassReverse /ccweb http://RWP_host:81/ccweb
ProxyPass        /Java_Plugins http://RWP_host:81/Java_Plugins
ProxyPassReverse /Java_Plugins http://RWP_host:81/Java_Plugins
```

Note: The URLs specified in this example must be written in the httpd.conf file exactly as specified here, with the exception of the host name and optional port number.

Configuring URL Redirection for Internet Information Server

If RWP must co-exist on a host with an instance of Microsoft Internet Information Server (IIS) that listens for HTTP requests on port 80, you must reconfigure RWP to listen for HTTP requests on a different port (see *To Change the Default RWP HTTP Port* on page 341) and then do one of the following:

- Include a port specifier (for example http://*hostname*:81/ccweb/) in the URLs used by Rational Web interfaces served by this instance of RWP.
- Use the IIS redirection facility to force Rational Web interface URLs directed to port 80 (and received by IIS) to be redirected to RWP.

To configure IIS to use redirection:

1. Run the IIS configuration utility (Internet Services Manager).
2. Create a new virtual directory in the IIS Default Web Site folder:
   - For the **Virtual Directory Alias**, pick a name that reflects the name of the Web client that will use the virtual folder (for example, **ccweb**)
   - For the **Web Site Content Directory**, you must specify a physical directory on the Web server host. Although this directory must exist on the host, it will not be used to hold any Web site content after you configure redirection in Step 4. We recommend that you create a new directory for this purpose and apply protections to it that reduce the chances of its being accidentally deleted.

     Note: If you create this directory as a subdirectory of the RWP installation directory, it will be deleted if RWP is reinstalled on the host.

- For **Access Permissions**, specify **Read** and **Run scripts**.

3 Right-click the virtual directory you created in Step 2 and open its **Properties** dialog box.

4 In the **When connecting to this resource, the content should come from** section of the **Virtual Directory** tab, select **A redirection to a URL**.

5 In the **Redirect to:** box, type the URL used by the Rational Web interface that you are redirecting to RWP. For example, to redirect the ClearCase Web interface (**ccweb**) to use an instance of RWP listening on port 81, type

http://*hostname***:81/ccweb/**

where *hostname* is the name of the host running RWP and IIS.

6 In the **The client will be sent to** section, select **The exact URL entered above**.

7 Verify that browsing to the URL http://*hostname*/ccweb redirects you to the ClearCase Web interface at the URL specified in Step 5.

Configuring RWP To Use Secure Sockets

To provide secure communications between Rational Web clients and RWP, you can configure RWP to support the Secure Sockets Layer (SSL) protocol. To do this, you need to take the following steps:

1 Edit the RWP configuration files to enable SSL support.

2 Modify client URLs as needed to specify the HTTPS protocol.

3 Stop and restart RWP.

Procedures for configuring RWP to support SSL are the same as those for configuring any Apache HTTPD that uses the **mod_ssl** module. These procedures are fully documented at the mod_ssl.org Web site; a summary of the configuration steps is presented here.

The first step in using SSL is to obtain a certificate from a certification authority (CA). RWP includes the **openssl** program (installed in the RWP bin directory), which you can use to generate a self-signed certificate for testing purposes and also obtain a certificate and key from a CA. For more information on **openssl**, see the openssl.org Web site.

You can run **openssl** in the RWP bin directory to generate a self-signed certificate and a matching private key and then install them in the locations specified in Step 2 and Step 3 of the following procedure.

- On UNIX, use this **openssl** command line:

```
OpenSSL> req –new –x509 –nodes –keyout /opt/rational/common/rwp/conf/server.key \
–out /opt/rational/common/rwp/conf/server.crt
```

- On Windows, use this **openssl** command line:

```
OpenSSL> req –new –x509 –nodes –keyout ../conf/server.key –out ../conf/server.crt
```

Note: If you do not have the right to create files in the specified **–out** directory, this command will fail.

To configure RWP to accept SSL connections:

1. Configure RWP to include the ssl.conf configuration file. Uncomment this directive in rwp.conf:

   ```
   Include conf/ssl.conf
   ```

   Note: The ssl.conf file includes a **Listen** directive that specifies the port on which RWP will listen for HTTPS requests. The default is port 443. You can change this in the same way that you change the default HTTP port. If you want RWP to listen only for HTTPS requests, comment out the **Listen** directive in the rwp.conf file.

2. Install the certificate. The default location of the certificate file is specified in this directive in the ssl.conf file:

   `SSLCertificateFile` *rwp-root-dir*`/conf/server.crt`

   where *rwp-root-dir* represents the directory in which RWP is installed on the host. If you install the certificate file in a different location, make sure that this line in ssl.conf references that location.

3. Install the key. The default location of the key file is specified in this directive in the ssl.conf file:

   `SSLCertificateKeyFile` *rwp-root-dir*`/conf/server.key`

   If you install the key file in a different location, verify that this directive references that location.

4. Stop and restart RWP.

Note: To configure a Web application to use SSL, specify the **https** protocol in the application URL. For example:

```
https://RWP_host.domain/ccweb
```

Configuring Secure Access to RWP

There are a number of ways to configure RWP to use SSL to provide secure communications with Rational Web clients such as **ccweb**. Figure 9 shows a typical configuration that allows **ccweb** clients on the public Internet to access RWP through a firewall. Communications between client Web browsers and the RWP host

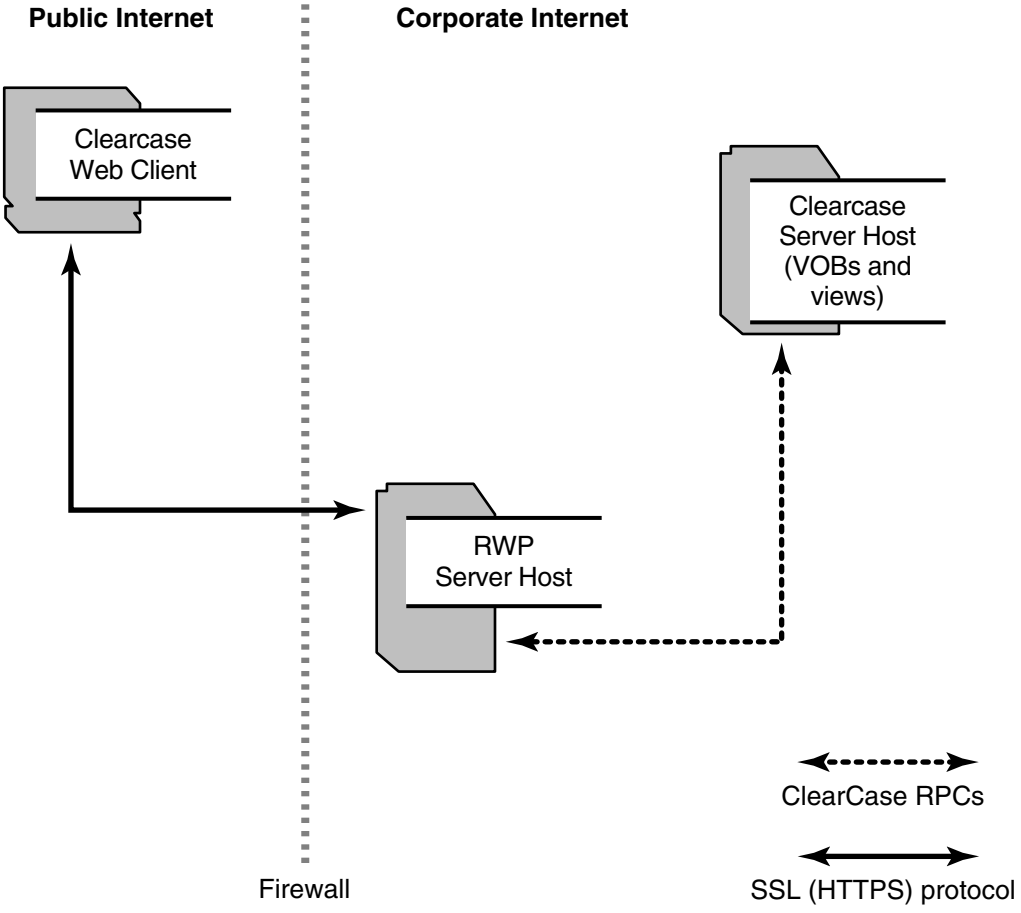
supporting **ccweb** use the HTTPS protocol and are secured by SSL. Communications between RWP and the ClearCase servers on the corporate intranet use ordinary ClearCase remote procedure calls (RPCs) and are not secure. ClearCase RPCs cannot communicate through a firewall, so you cannot place a firewall between RWP and the ClearCase servers used by **ccweb**.

In a configuration like that shown in Figure 9, the **ccweb** URL would be:

https://*hostname*/ccweb

where *hostname* is the name of the RWP server host. HTTPS communications between the ClearCase Web client and RWP would use port 443, the default for HTTPS.

Figure 9 Secure Communications Between ccweb and RWP



Other Modifications to RWP

We do not recommend that you modify any RWP configuration files other than those described in this appendix. Some of the configuration options cannot be changed without adversely affecting the operation of RWP. Any configuration change not recommended in this appendix should be carefully evaulated before introducing them into a production environment.

Configuring the ClearCase Web Interface

For many sites, the ClearCase Web interface requires no special configuration; any host on which RWP is installed can support this interface at the URL

http://*hostname*[:*port*]/ccweb

where *hostname* is the name of the RWP host and *port* is an optional port number if RWP has been configured to use a port other than 80 for HTTP (see *To Change the Default RWP HTTP Port* on page 341).

Note: When RWP runs on a Windows computer, ClearCase Web interface users must be given permission to **Log On Locally** to the RWP host. Windows does not grant this permission by default.

The remainder of this section explains how you can edit the ClearCase Web interface configuration file

ccase-home-dir/config/ccweb/ccweb.conf

if you need to modify the default configuration of the ClearCase Web interface.

Note: Do not confuse this file with the ClearCase Web interface support file for RWP. This support file is installed as C:\Program Files\Rational\common\rwp\ccweb.conf on Windows and /opt/rational/common/rwp/conf/include/ccweb.conf on UNIX. It should not be modified.

Specifying the ClearCase Primary Group

At sites where ClearCase users on Windows set the CLEARCASE_PRIMARY_GROUP environment variable (see *Setting the ClearCase Primary Group* on page 321), you must also specify that primary group for ClearCase Web interface users by modifying this line in the ccweb.conf file:

–primary_group *group-name*

where *group-name* is the name of a domain group that will be used as the CLEARCASE_PRIMARY_GROUP for all users who access the ClearCase Web interface on this host.

Note: When users from multiple domains access the ClearCase Web interface, you must enable domain (described in *Using Proxy Groups and Domain Mapping in Windows NT Domains* on page 324) on the RWP host and specify an appropriate value for **–primary_group** in ccweb.conf.

Web View Storage

The ClearCase Web interface normally creates Web view directories on the RWP server host. These directories are used for temporary storage of files that are checked out or ready to be created as elements. Therefore, they must be on a disk volume that has enough space for the number of Web views to be supported. We suggest allocating 0.5 MB to 1 MB of disk space for each Web view.

On UNIX, these directories are normally created under /var/adm/rational/clearcase/ccweb.

On Windows, these directories are normally created under C:\Program Files\Rational\ClearCase\var\ccweb

If the default Web view storage directory is not appropriate, you can select a different area by modifying ccweb.conf. Add the line

–view_storage *pathname*

where *pathname* is the directory in which you want the Web view directories to be created. This directory must be local to the RWP host.

Note: Although the ClearCase Web interface may create a separate directory for Web view storage, it continues to store executable and administrative files in *ccase-home-dir*/ccweb.

Limiting Upload Size

To better manage Web view storage or reduce the possibility of denial-of-service attacks, you may want to limit the size of files that can be uploaded to the RWP host. To do so, modify the following line in the ccweb.conf file:

–upload_limit *size*

where *size* is the approximate desired size limit in bytes. An attempt to upload a file that is too large results in an error message in the **Client Upload** output window.

Specifying a Session Timeout

You can configure the session timeout interval, which controls how long a user login remains valid. The default value is 14400 seconds (four hours). You can change this default by modifying the line

–session_timeout *seconds*

in ccweb.conf, where *seconds* is an integer number of seconds between 600 (10 minutes) and 2147483647 (about 68 years). Values less than 600 are interpreted as 600.

Specifying a Directory for Temporary Storage

You can designate a directory where the ClearCase Web interface stores temporary files by adding a line of the form

–tmpdir *directory-name*

to ccweb.conf, where *directory-name* is a directory on the RWP host in which the ClearCase Web interface has permission to create and delete files. If this line is not present in ccweb.conf, the ClearCase Web interface uses the value of the TMP or TEMP environment variables, if they exist.

Permission to Download Applets on Windows

Internet Explorer typically requires a user to have local administrator privileges to download applets, including those used by the ClearCase Web interface. After these applets have been downloaded (the first time the Web interface is used), they do not have to be downloaded again unless a ClearCase patch or new ClearCase release changes them.

Configuring Non-ClearCase Access on UNIX

D

Some ClearCase communities need to provide access to VOBs and dynamic views to computers that cannot run Rational ClearCase. This appendix describes how to configure a ClearCase host running UNIX to provide this support.

Note: Non-ClearCase access is not supported on UNIX hosts running ClearCase LT or on any ClearCase host running Windows.

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 ...
  ```

Restrictions on Use

Non-ClearCase access carries several restrictions:

- **VOB access.** Users on the non-ClearCase host can read, but cannot modify, data from VOBs on UNIX VOB server hosts configured for non-ClearCase access. 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 accessed through an export view, they can write to view-private storage. Users can modify these view-private files with an editor and build them, 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 information, see *Building Software*.)

- Because **clearmake** and other build-management tools do not run on the non-ClearCase host, configuration lookup and DO sharing are not available.

Setting Up an Export View for Non-ClearCase Access

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 by 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. An export view must meet two requirements:
   - It must be marked for export by using the **–ncaexported** option to **mkview** or **mktag**.
   - If the view has tags in multiple registry regions, the exported tag must be in the view server host's region.
3. The host uses an /etc/exports.mvfs 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 put each VOB and its export view on the same host, developers working on a non-ClearCase host are likely to access several export views at the same time. For example, to provide non-ClearCase access to three VOBs located on three different hosts, each of which had its own export view, you would need to create NFS mount table entries like these on the non-ClearCase host:

```
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. For each VOB/view pair, ClearCase operations as **checkout** and **checkin** must be made in the appropriate export view. Other operations, such as building and debugging, do not need to reference any view.

Export Configurations

Accessing data from a non-ClearCase host can involve three hosts:

- The host on which the VOB resides
- The host on which the export view resides
- The non-ClearCase host

A multihop configuration, in which the VOB and its export view reside on different hosts, requires MVFS-level communication, which is slower than NFS communication, between the two ClearCase hosts. In addition, 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 may result in time-outs (if VOBs are soft-mounted) or deadlocks (if VOBs are hard-mounted).

We recommend avoiding multihop configurations when using non-ClearCase access.

Avoiding Multihop Configurations

You can avoid multihop configurations by adhering to the following guidelines:

- Locate the export view on the same host as VOB it exports. Although we normally discourage running a **view_server** and **vob_server** on the same host (see *Minimize Process Overhead* on page 289), doing so may be preferable to using a multihop configuration.
- 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 (the .s directory) must be local to the host.

Working with Multihop Configurations

If you decide to use a multihop configuration, ensure that the VOB host (and its pool hosts, if any) never request network services from the view host. This guarantees that no process on the VOB and pool hosts creates an access cycle with the view host. Do not request, either directly or through some other host, ClearCase services or any other network services from the export view host.

One way to help achieve this goal would be to prohibit users from running processes, either directly or indirectly, on the export view host. Such a prohibition would exclude the following specific cases:

- Creation of home directories
- Remote log-ons from a ClearCase host
- Remote backups of the view server hosts by the VOB server or pool hosts

It would also exclude other cases that resulted in a similar pattern of access.

Restricting Exports to Particular Hosts

In a multihop configuration, we recommend using an **–access** option in each entry in /etc/exports.mvfs. This restricts the export to specified non-ClearCase hosts and/or netgroups, and 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.

Using automount with Non-ClearCase Access

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

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

Note: Listing the directory /remote/*viewname* usually shows active mounts only, 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, try running an explicit **mount** command; if it works properly, the problem may lie in the client automounter. Refer to your platforms's documentation for full details on automounter map syntax.

Note: Using the **–hosts** map for automount access does not work properly on hosts that also export their root directory. 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 cannot find a directory on which to mount the view/VOB pair.

For more information about ClearCase and the automounter, see *ClearCase and the NFS Automounter* on page 18.

NFS Problems with Non-ClearCase Access

Depending on the NFS implementation on the non-ClearCase host, you may encounter various problems when using non-ClearCase access.

Problems with 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 data in the view or VOB 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, because it 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 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 may be able to 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. (On some platforms, this method is effective only if you do it on the remote host.)
- 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 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**, unmount the file system and then remount 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. Use label-based rules rather than the **/main/LATEST** rule.

Problems with NFS Locking

Because non-ClearCase access does not support NFS file locking for its files, applications that 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 platforms, it may not work for VOB files. An application can hang if it retries lock requests until it can obtain a lock. 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 to access VOB data from hosts that do not run ClearCase.

Integrations with Microsoft Web Authoring Tools

E

This chapter explains how to configure ClearCase integrations with Microsoft Web authoring tools.

Overview of the Integrations

On Windows platforms, Rational ClearCase and Rational ClearCase LT support integrations with these Microsoft products:

- FrontPage 98, 2000, and later
- Visual InterDev 6.0
- Office 2000 Web Folders functionality for Word, Excel, and PowerPoint
- Web Folders in Internet Explorer 5.5 and later

These integrations provide ClearCase configuration management capability for Web content and Web applications created with these tools. There are two installations, one on the server and one on the client.

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

The client is a Windows computer that runs a supported Web authoring application. Clients open Webs on the server by using a URL and perform checkouts and checkins using a shared snapshot view that resides on the server. Each IIS instance is associated with a single Web content VOB by using the default IIS alias. Multiple IIS instances can be configured to share the same Web content VOB.

Server Setup Overview

A summary of the server installation follows. Detailed instructions for each step are in *Server Set-Up Details* on page 383.

1 Install IIS 4.0 (from the 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 or later. The IIS

version number is stored in the Windows registry key **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\MajorVersion.**

2 Install FrontPage Server Extensions 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.

4 Run the Web Authoring Integration Configuration Wizard and follow the instructions.

Client Setup Overview

A summary of the client installation follows. For detailed instructions, see *Client Setup Procedure* on page 366.

1 Install one or more supported authoring tools.

2 Create and add Webs to source control by using FrontPage 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 from FrontPage or Visual InterDev 6.0.

5 Optionally, enable local mode support for tools that support it.

Server Setup Procedure

The supported server platforms and required software versions for the Web server are listed in Table 17.

Table 17 Supported Integration Platforms for Microsoft Web Authoring Tools

Windows version	Web server software	FrontPage server extensions or 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 17 Supported Integration Platforms for Microsoft Web Authoring Tools

Windows version	Web server software	FrontPage server extensions or 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 or IIS 5.1	4.0 or later

Note: If you use the MS FrontPage Server Extensions 2002, you must obtain the file iisfix.bat from Rational Customer Support.

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: If you must install the Rational Web Platform (RWP) and IIS on the same host, you must configure one or both Web servers to prevent them from interfering with each other. For more information, see Appendix C.

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 are using Windows NT 4.0 server, install IIS 4.0 and either FrontPage Server Extensions (FPSE) or Office Server Extensions (OSE).

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, the Web publishing home directory that you specify is used by the Web Authoring Integration Configuration Wizard to construct a VOB tag and view tag as shown in this example:

drive-letter:*view-tag**VOB-tag*

If you specify C:\webpub\web_content as the Web publishing home directory, the Web Authoring Integration Configuration Wizard creates the following view and VOB for use by the integration:

- A view with the view tag **webpub**
- A VOB with the VOB tag **\web_content**

Caution: The IIS installation provides a default pathname, \Inetpub\wwwroot, for the Web publishing home directory. We strongly recommend that you not use this default pathname. Instead, supply a pathname that will not generate a view or VOB tag that conflicts with an existing one.

After IIS is installed, you may need to run Internet Service Manager to add IIS operators and enable Basic Authentication. If the Web content VOB resides on the IIS host, 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 you installed FPSE in Step 1, it is not necessary to reinstall. If you installed FPSE version 3 during a previous Option Pack install, you can upgrade to OSE, but it is not necessary.

If you did not install FPSE in Step 1, install FPSE version 3.0 (from Windows NT 4.0 Option Pack) or version 4.0 (from Office 2000 CD3, Office Server Extensions, or Windows 2000).

Step 3: Install ClearCase

Install ClearCase or ClearCase LT.

- The ClearCase installation must include local VOB and view support, but does not need to include the MVFS.
- The default ClearCase LT server configuration is appropriate for hosting a Web Authoring Integration.

  Note: The default ClearCase LT server installation includes RWP. To run RWP and IIS on the ClearCase LT server, you must configure one or both Web servers to prevent them from interfering with each other. For more information, see Appendix C.

The Web Authoring Integration Configuration Wizard requires a server storage location. Most ClearCase LT installations create a server storage location as part of the server configuration process. For more information about creating server storage locations, see *Creating Server Storage Locations for VOBs* on page 95.

We recommend that you make a server storage location available on the Web server host. Creating the Web content VOB in this location improves performance and avoids the use of basic authentication.

Step 4: Run the Web Authoring Integration Configuration Wizard

Run the Web Authoring Integration Configuration Wizard and follow the instructions. Help is available for each page of the wizard. 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 opens a **Replace SourceSafe** display. To use the integration, you must click **Yes**. Doing so does not disable any part of SourceSafe except 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 Web content VOB. You can accept the default VOB storage location or use the wizard to select a different location. You can also select an existing VOB.

Several restrictions apply to any VOB accessed through the Web Authoring Integration:

- All groups with permission to create Web content must be explicitly added to the VOB's group list. (The CLEARCASE_PRIMARY_GROUP environment variable is not considered by the server.) For more information, see the **protectvob** reference page.
- Interactive triggers cannot be used.
- VOB symbolic links cannot be used in Webs created with the integration.

After the Web content VOB has been specified, the wizard prompts you to create a Web content view. 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 from 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 views.

The Web Authoring Integration Configuration Wizard maintains a log file at *ccase-home-dir*\var\log\webintegration_log. Review this log if you encounter errors or unexpected behavior.

Client Setup Procedure

This section explains how to set up the Web Authoring Integration on a ClearCase client host.

Note: Every time a Web Authoring Integration user opens a source-control-enabled Web, that user takes a ClearCase license.

Step 1: Install the Client Application

Install one or more of the supported client applications on your client computers:

Step 2: Add Web to Source Control

Using FrontPage 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://***WebServer***/***WebName*.

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 or later

1. Run the Web Authoring Integration Configuration Wizard. From the Web Authoring Integration Configuration Wizard, select **Local Mode Configuration**.

2 In FrontPage, create a new Web. Click **File > New > Web** and complete the **New** dialog box.

3 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 not installed on the client, Help for the Visual InterDev integration is disabled.

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 or Visual InterDev, look for the source control icons that indicate successful completion. In FrontPage, these icons are green dots; in Visual InterDev, they are locks.

If no source control icons appear, or if there are errors or other unexpected results during these procedures, an installation error may have occurred. The log file *ccase-home-dir*\var\log\ssapi_log may have more information about 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 import the files 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** 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** 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.

Local Mode Client Setup for FrontPage 2000 or later

The integration supports two modes: server mode and local mode.

- Server mode operation is available on all the supported authoring tools and provides access to a limited set of ClearCase features. In server mode, the authoring tool runs locally, and all ClearCase operations run remotely on the IIS host. All users share a single snapshot view on the 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 access to Help, as well as a limited **ClearCase** menu in FrontPage 2000 and a ClearCase toolbar in Visual InterDev.
- Local mode operation is supported only for FrontPage 2000 or later; ClearCase must be installed on the same computer that is running FrontPage. Local mode operation provides extra functionality, including a **ClearCase** menu that is available on the FrontPage 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 when using local mode. Checkouts performed in local mode are unreserved by default.

To configure local mode for FrontPage:

1 Install FrontPage 2000 or later.

2 Install ClearCase.

3 Run Web Authoring Integration Configuration Wizard. From the Web Authoring Integration Configuration Wizard, select **Local Mode Configuration**.

4 In FrontPage, click **ClearCase > Create Snapshot View** and complete the View Creation Wizard.

If you access Webs with FrontPage borders, you must reapply these borders in a local-mode view. We recommend not using FrontPage themes unless only server-mode access is used. Theme binding information is not added to 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.

Web Folders Support

The server-mode integration option also enables you to configure your IIS Web server for the ClearCase integration with Office 2000 and the Internet Explorer Web Folders.

If you are using any of the Office 2000 applications or Internet Explorer 5.0 or later, 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.

Updating the Shared View on the Web Server

From time to time, users may want to update the shared Web content view. A script on the Web server performs this function (*ccase-home-dir*\etc\iisfix.bat). Users can access this function from their client systems by opening the URL http://*WebServerName*/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.

Data Migration and Conversion

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 import data from another source control product.

- **Migration**. The integration expects Webs under source control to be rooted in a subdirectory of the Web content VOB root. If you use **cleartool relocate** to relocate Webs from existing VOBs to the Web content VOB, relocate them to a VOB root subdirectory.
- **Importing.** To import a Web under Visual SourceSafe control to ClearCase, run **clearexport_ssafe** on the Visual SourceSafe library where the Web content files reside. Then use **clearimport**, specifying a target directory in the Web content VOB. For more information about importing data into ClearCase, see Chapter 13.

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 that enable the integration to work.

Note: If you are using the FPSE 2002, you must obtain a new copy of the iisfix.bat script from Rational Customer Support.

After this has been done, the source control icons appear when the Web is opened in FrontPage and Visual InterDev.

Accessing Help for the Integration

When ClearCase is installed on client systems, 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 or later. This Help is not available on hosts that do not have ClearCase installed.

Estimating VOB Size

F

If you can estimate the number of elements, versions, and derived objects that a VOB will ultimately contain, you can compute a very rough estimate of the size of a VOB that would accommodate them. VOB storage includes a database and pools. On Windows, the database and pools must reside on a single disk partition. On UNIX, the pools can be located on a different partition or host, using symbolic links, as described in *Creating Remote Storage Pools on UNIX Hosts* on page 113.

Estimating Database Capacity

Keeping a VOB database's working set in memory greatly reduces the time it takes to access VOB metadata, which is why the *VOB Server Host Configuration Guidelines* on page 93 include a recommendation that the VOB server be configured with enough physical memory to hold half of each VOB database it hosts.

Table 18 provides information about the sizes, in bytes, of the four most significant contributors to VOB database space consumption.

Table 18 Sizes of Common VOB Data Types

Data type	Schema version 53	Schema version 54
Element with one version	2770	3970
Each additional version of an element	446	642
Unshared DO (excluding config rec)	1356	1924
Shared DO (excluding config rec)	60	118

Note: Every DO includes a configuration record (config rec). Config rec size is determined by an element's contents and how the DO was built (build script and compiler options). For purposes of estimation, you can assume that a config rec adds between two and ten percent to the size of an unshared DO, and fifty to one hundred percent to the size of a shared DO.

It is more difficult to estimate how many of these objects users will create in a VOB. Estimating the number of elements is usually easy, especially if you are populating the VOB with the contents of another source code control system (see Chapter 13). Estimating the number of versions is harder. Some elements accumulate many more versions than others. You may be able to use the number of versions of some representative elements in other VOBs as an indicator of how your software development methods contribute to version creation. The number of derived objects you create is a function of the number of build targets and number of elements in each target. If you use ClearCase LT, or use ClearCase but do not use dynamic views and **clearmake** or **omake**, you do not create derived objects.

Given an estimate of the number of elements a VOB will contain, you could use the information provided in Table 18 to compute a rough approximation of the VOB's database size by using the formula:

$S = N\ (E + (e * (V - 1)) + D * d)$

where:

- S is the database size in bytes.
- N is the number of elements in the VOB.
- E is the size of an element with one version.
- e is the size of each additional version of an element.
- V is the average number of versions per element
- D is the number of unshared derived objects per element
- d is the size of an unshared derived object.

The following example uses this formula to estimate the size of a VOB formatted with schema version 53 that contains 500 elements with an average of 20 versions and 5 derived objects per element:

500 * (2770 + (446 * (20 - 1)) + 5 * 1356) = 901,200 bytes

These estimates do not account for the sizes of other VOB metadata such as branch types and instances, attribute types and instances, and so on. These data types are generally small, but they can be numerous in some VOBs, especially when UCM is in use.

Estimating Pool Capacity

In addition to the space it consumes in a VOB database, each element requires space in the VOB's source pool, and usually other pools as well.

Every version of every element in the VOB is stored in a container file in a source pool. Different element types have different type managers. Most type managers compress

source data so that each version consumes less storage in the source pool than it does in cleartext form. Table 19 lists compression ratios for commonly used built-in type managers.

Table 19 Type Manager Compression Ratios

Type manager	Type description	Compression ratio
whole_copy	Whole copy of object	1:1
text_file_delta	Interleaved deltas of printable text	50:1
z_whole_copy	Compressed whole-copy	2.5:1
z_text_file_delta	Compressed text_file_delta	200:1

With the information provided in Table 19, you could compute a rough approximation of VOB source pool size by using the formula:

$P = N * V * S / C$

where:

- *P* is the source pool size in bytes.
- *N* is the number of elements in the VOB.
- *V* is the average number of versions per element
- *S* is the average file size of each cleartext element version.
- *C* is the compression ratio of the element's type manager.

The following example uses this formula to estimate the size of the source pool of a VOB that contains 500 elements of type **text_file_delta**, with an average of 20 versions per element and an average version size of 50000 bytes:

500 * 20 * 50000 / 50 = 10,000,000 bytes

For VOBs that contain elements of different types, you must repeat this calculation for each element type.

Elements also require space in the cleartext and DO pools, which are caches that are periodically scrubbed to control their growth. Because space requirements of cleartext and DO pools are determined by site-specific patterns of use, we cannot provide a general rule for estimating their size based on the number of elements in a VOB.

For more information about VOB storage pools, see *VOB Storage Pools* on page 106.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W